VirtuE: Virtual Enterprises for Information Markets *

Alessandro D'Atri
CeRSI
LUISS "Guido Carli" University
Rome, Italy
datri@luiss.it

Amihai Motro
Information & Software Eng. Dept.
George Mason University
Fairfax, VA, 22030, USA
ami@gmu.edu

1 June 2001

ISE-TR-01-02

Abstract

An essential part of a modern economy is an information market. In this market, information products are being traded in countless ways. Information is bought, modified, integrated, incorporated into other products and then sold again. Usually, the manufacturing of an information product requires the collaboration of several participants. A virtual enterprise is a community of business entities that collaborate on the manufacturing of new products. This collaboration is often *ad hoc*, for a specific product only, after which the virtual enterprise may dismantle. The virtual enterprise paradigm is particularly appealing for modeling collaborations for manufacturing information products, and in this paper we present a new model, called VirtuE, for modeling such activities.

1 Information Markets and Virtual Enterprises

An essential part of a modern economy is an information market. In this market, information products are being traded in countless ways. Information is bought, modified, integrated, incorporated into other products and then sold again.

Some information products are *elementary*. An elementary information product is created out of nothing; for example, a reading off an instrument, a photograph taken by a camera, or a

*This work was partially supported by the European Commission under the FAIRWIS project no. IST-1999-12641 within the Key Action "New Methods of Work and Electronic Commerce" of the Information Society Technologies Program.

new customer record added to a database. Often, however, information products are *derived* from other products. A weather report is assembled from multiple instrument readings, a news report requires the analysis of several sources, and even a photograph may be an enhancement of another photograph.

It is possible that an information product is manufactured in its entirely by the same individual or business entity. More often, however, the production of an information product requires collaboration among several different participants.

A particular form of business cooperation that has attracted attention recently is that of a *virtual enterprise*. A virtual enterprise is a community of business entities that collaborate on the manufacturing of new products. The collaboration is often *ad hoc*, for a specific product only, after which the virtual enterprise may dismantle (indeed, former collaborators may become competitors). The members of a virtual enterprise often possess complementary skills and technologies whose combination is deemed necessary for the target product at hand.

The virtual enterprise paradigm appears to be suitable for collaborative productions of information products. In this paper we describe the VirtuE model for virtual enterprises whose core business is information products.

Such a virtual enterprise may be set up to produce an *electronic publication*, with individual members providing services such as photo archives, news, layout, and proofing. A *market research* enterprise may be a collaboration among a credit bureau, a mailing list consolidator, an archive of retail transactions, and a data-mining service. An *on-line library* may be a collaboration among many different information archives and a variety of information processors, providing specialized services such as document indexing, document retrieval and document ranking. In these examples, each of the members could, in turn, enlist the services of other members; for example, the mailing list consolidator could procure individual lists and then enlist the help of another service to integrate the lists while removing replications and resolving inconsistencies.

The main features of VirtuE fall into four areas:

1. **Products**: VirtuE features two types of information products: *content*, which is an item of information, and *process*, which is an operation that modifies existing contents to produce new content. Enterprise members may offer both types of service. A simple global resource, called *dictionary*, is defined for coordinating knowledge about products among the participants.

2. **Manufacturing**. The set of products used or created by each enterprise member is stored in a local resource called *inventory*. For complex products (products that are created from more elementary products), *production plans* must be provided. A production plan specifies how contents and processes are combined to derive the new product.

3. **Transactions**. Since component products are often obtained from other enterprise members, a procurement mechanism is necessary. A catalog exchange process among

the enterprise members establishes the *infrastructure* on which products are exchanged. Exchanges are executed in *two-phase transactions*: an inquiry followed by an order. Since production plans may branch into multiple alternatives, each requiring different transactions to import different components from different members, an *optimization* process selects the optimal plan.

4. **Customization**. VirtuE allows the definition of *performance indicators*, which are formulas that capture various quantitative characterizations of the virtual enterprise; for example, enterprise assets or interdependence level . Another feature of VirtuE are *constitutional rules* which are constraints that express behavior that must be adhered to. Such rules enable the creation of virtual organizations with different style or flavor; for example, an organization in which all participants must be of comparable magnitude (i.e., assets), an organization which is without any competition (similar products are not available from different members), or an organization that resembles a free market.

While we are not aware of any work on virtual enterprises for information markets, our work is at a junction of several active research areas, and we discuss here the most relevant five areas: (1) information marketplaces, (2) information brokering, (3) virtual enterprises, (4) workflow management in virtual enterprises, and (5) federated databases.

The importance of the information market especially in relationship to electronic commerce has been recognized for quite some time [13, 10]. It is within this important marketplace that VirtuE operates, advocating a specific type of collaborative organization to generate new products for this marketplace. VirtuE's goal of modeling information exchange and manufacturing is strongly related to information brokering. Predicting that federations of large number of information system will be cooperating to support information needs of users, [12] proposes an architecture consisting of information providers, information brokers and information consumers.

The concept of virtual enterprise as a cooperative of independent entities that collaborate on the manufacturing of products has been around for decades.[1] The interest of the information technology research community in this area dates only to the mid-1990s, with much of the work focusing on organizational issues, communication processes and information systems support [3, 14, 19]. Recently proposed paradigms to support virtual enterprises include the VEGA software platform [18] and a methodology for fusing the separate business processes of enterprise members [7].

VirtuE's concept of production plans (manufacturing formulas for information products) may be considered a specialized form of workflow management [6, 20, 5]. Within this area, considerable attention had been given recently to the application of workflow management techniques to virtual organizations [9, 8]. Finally, in the area of data modeling, federated database models [11, 17, 16, 15] may be considered as predecessors of VirtuE. A federated database provides an environment in which information may be exchanged among

[1] Often the members of such an enterprise reside in the same geographical area. A notable example is the Saxon region in Germany [4].

autonomous or semi-autonomous participants. Federated databases, however, do not provide features for manufacturing new information products, and do not attempt to model the business aspects of information exchange.

A preliminary version of VirtuE is described in three sections. Section 2 describes the basic structures of the VirtuE model: participants, products, and an infrastructure that participants use to exchange products. This section also defines the global dictionary for coordinating knowledge about products. The manufacturing of new information products from existing products is discussed in Section 3, which introduces inventories and production plans. This section also discusses performance indicators and constitutional rules. Section 4 is devoted to operational issues. It defines two-phase transactions, and discusses the optimization of production plans. Section 5 concludes this paper with a brief summary and a list of subjects currently under investigation.

2 Basic Structures

In this section we define the basic concepts of the VirtuE model. Each virtual enterprise consists of participants, products, and an infrastructure that the participants use for exchanging products. A global resource, called dictionary, is used to coordinate knowledge about products among the participants.

2.1 Participants: Members and Clients

We assume a community of independent members with shared interests. The members are independent in the sense that they remain autonomous and maintain their own assets. These assets include human, equipment or financial resources, as well as business expertise, such as knowledge about their production and delivery processes. Their shared interests are reflected in that they agree to cooperate with each other to produce joint products that are provided to common customers. After the community had been established it could evolve because an existing member departs or a new member joins. This form of evolution provides the virtual enterprise with flexibility and allows it to adapt to new situations.

The set of members will be denoted M. An individual member will be denoted m_i, where i is a unique identifier of the member.

A *client* is an entity outside the virtual enterprise who approaches the virtual enterprise to acquire a product.

2.2 Products: Contents and Processes

In practice, virtual enterprises may produce many different kinds of products. In VirtuE, we shall consider only *information* products, of the type that can be delivered over computer networks. Information products are provided by members of the enterprise to their clients. This provision is the ultimate purpose of an enterprise. Information products are also exchanged among the members of the enterprise in the production phase that precedes the provision of a product to a client.

We distinguish between two basic kinds of information products: *content* and *process*.

2.2.1 Content

Content is an information item. Examples include "a database of customers and the products they purchased in the year 2000", "the codes of all stocks traded in the New York Stock Exchange and their closing values on March 31, 2001","an image of the space shuttle landing in Kennedy Space Center on May 29, 2000", "a stream of news items on the subject of sports", and so on. A *request* for this kind of product describes the information needed; the *response* is information content that answers the description.

To manage the potentially enormous number and variety of contents, we introduce *content types*. Each content is associated with one content type. Examples of content types are *Database*, *Image*, *Document*, and so on.

Each content type C is associated with a sequence of *attributes*, $att(C)$. Each attribute A in $att(C)$ denotes a measurable aspect of all contents of this type, and has an associated *domain* of feasible values $Dom(A)$. Let c be a content of type C, then $c[A]$ denotes the value of the attribute A for this content. $val(c)$ is the sequence of all attribute values of c (in correspondence with the sequence of attributes $att(C)$).

The attributes $att(C)$ can be partitioned into several groups. One particular attribute, *Product_Code*, is associated with every content type. *Product_Code* is a value that is used to identify products within the product offerings of a member. Next, a group of attributes is designated as *specificational*. These attributes are specific to individual content types and provide a description for each particular content of that content type. With these attributes, it should be possible to determine the essence of a product. For example, an *Image* type may have the attributes *Image_Format* and *Resolution* and a *Document* type may have the attributes *Document_Format* and *Author*. This group may also include attributes such as *Subject* or *Title*. The remaining attributes form the *optional* group. Examples include *Quantity*, *Size*, *Timestamp*, *Quality*, *Cost*, *Price* and *Member_Price*. A common optional attribute is *Description*, for describing products in notation such as natural language or keywords. Note that for individual content types, these attributes may be measured differently. We shall use $c[S]$ and $c[O]$ to denote, respectively, the specificational and optional attributes of a content c.

Finally, an attribute is *mandatory* if each content of that type is required to have a valid value for that attribute. For attributes that are not mandatory *null* values are used wherever valid values are either unknown or inapplicable. *Product_code* and the specificational attributes are mandatory.

2.2.2 Process

The second basic kind of information product is process. A process is an operation that modifies given content to produce new content. Examples of processes include (1) aggregating a set of pictures in an *album*, (2) translating a document from one language to another, (3) analysis of financial data to produce stock market recommendations, (4) cleansing data (i.e., removing or correcting errors, resolving inconsistencies, and so on) , (5) filtering data (i.e., separating the data into two parts: wanted and unwanted), (6) ranking a set of data items, (7) merging two lists while removing replications, and so forth. A request for this type of product names the process and provides a set of input contents; the response is the output content. Usually, processing adds value to the original information.

As with contents, we introduce *process types* to classify the different processes. For example, there could be several data compression processes, all belonging to a single process type *Compression*. Each process type P is associated with a sequence of input content types $C_1, \ldots, C_n$ and with an output content type C. When receiving contents $c_1, \ldots, c_n$, where c_i is of type C_i $(i = 1, \ldots, n)$, a process p of type P produces content $p(c_1, \ldots, c_n)$, which is of type C.

Process types also have their attributes. The attribute sequence of a process type P is denoted $att(P)$. Let p be a process of type P, then $p[A]$ denotes the value of the attribute A for this process. $val(p)$ denotes the sequence of all attribute values of p (in correspondence with $att(P)$). The attributes $att(P)$ are divided into the same groups: *Product_Code*, specificational and optional. Examples of specificational attributes include *Maximal_Error_Rate* or *Minimal_Output_Quality*.

To summarize, the basic concepts of information products are: (1) Content, content type and content attribute, and (2) Process, process type and process attribute

A simple analogy that illustrates these concepts would be computer files and file translators. Each file is associated with a specific file type (usually denoted by a suffix to its name), and has specific attributes (such as size, timestamps, and access permissions). Each file translator is associated with two file types: the source type and the target type. An example of a file translator is compression. An attribute for a compression processes would be average compression rate.

The concepts defined here could be cast in an object-oriented framework of objects, classes, methods, inheritance, and so forth. However, for simplicity we shall limit our model to the subset of concepts defined here; i.e., content, content type, content attribute, process, process type and process attribute.

2.2.3 Special Attribute Values

In addition to the values in their associated domains, optional attributes may assume three special kinds of values.

- **Null value.** The *null* value, already mentioned, is used when an attribute value is unavailable or inapplicable. For example, the *Quality* of a particular content may be unknown.
- **Multivalue.** A *multivalue* is a set of possible values. For example, the *Size* of a particular content may be {small, medium, large}. The attribute *Quantity* is often a multivalue. For numerical attributes, multivalues are often specified as *ranges*.
- **Function.** An attribute value may be a function of other attributes: An attribute of a content may be a function of other attributes of this content, and the attribute of a process may be a function of other attributes of this process or of attributes of its input contents. *Price* is often a function of *Quantity*; for example, *if Quantity* < 10 *then 30 else 25*. As another example, *Compression_Rate* may be a function of the *Density* of the input content.[2]

2.3 The Global Dictionary

Of the six aforementioned concepts (content, content type, content attribute, process, process type and process attribute), all but content and process may be considered *intensional* information (content and process are *extensional* information). We assume that all intensional information is available in an enterprise-wide resource called the *global dictionary*. This dictionary assures consistency of naming across the enterprise.

Formally, the global dictionary is a pair $(\mathcal{C}, \mathcal{P})$, where $\mathcal{C}$ is a set of content types and $\mathcal{P}$ is a set of process types. Each content type $C \in \mathcal{C}$ is described by a set of attributes $att(C)$ and each process type $P \in \mathcal{P}$ is associated with a set of attributes $att(P)$. Each attribute $A \in att(C)$ or $A \in att(P)$ is associated with a domain $dom(A)$.

Determining whether two products (content or process) are "the same" is not straightforward, and we define two different levels of identification:

1. **Identical products**: The attribute *Product_Code* uniquely identifies a product within offerings of individual members; hence, the combination *Member_Identifier:Product_Code* identifies a product across the virtual enterprise.
2. **Comparable products**: Two products (possibly from two different members) are *comparable* if they are of the same type and have the same specification; i.e., products e_1 and e_2 of type C are comparable if $e_1[S] = e_2[S]$. Intuitively, when two products are comparable, one could possibly substitute for another.

[2] A concept related to functional values, *derived attributes*, will be introduced in Section 3.3.3.

3 Manufacturing New Products

The basic VirtuE paradigm is that members obtain information components from other members to manufacture new information products. This section explains how manufacturing is accomplished, using two new concepts: inventories and production plans. This section also explains how to create virtual enterprises possessing different characteristics by using constitutional rules and how to monitor their performance with performance indicators.

3.1 Inventories and Catalogs

3.1.1 Inventory

Products (contents or processes) are exchanged by the members of the virtual enterprise. The method of exchange shall be explained later; at this point we note that the products used by each member can be classified according to two parameters.

1. **Native** or **Imported**: A native product is produced locally, whereas an imported product is outsourced from another member of the enterprise.
2. **Internal** or **Exported**: An exported product is provided by this member to others, whereas an internal product is an interim product used only by this member in the manufacturing of other products.

These parameters form four categories of products: native-internal, imported-internal, native-exported, and imported-exported. Figure 1 illustrates these four categories. The box represents the products of an individual member. Product a is native-internal, product b is imported-internal, product c is native-exported, and product d is imported-exported.

Products (contents or processes) can also be classified as *basic* or *complex*. A content is complex if it is derived from other contents by some process; otherwise it is basic. A process is complex if it is a combination of other, more elementary, processes; otherwise it is basic.

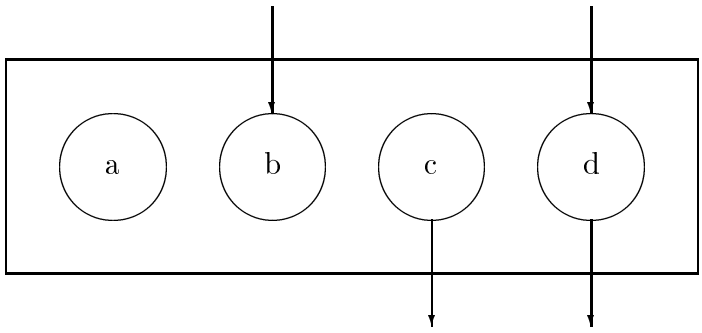

Figure 1: Product classification as native or imported and internal or exported.

Note that imported products are always classified as basic; that is, a complex product is always considered native.

The set products used by a member m are enumerated in an *inventory*, $Inv(m)$. Each inventory entry e is described as follows:

1. **Kind**: an indication whether the product is content or process.
2. **Type**: for a content, the *content type*, for a process, the *process type*. These types are taken from the global dictionary.
3. **Source**: an indication whether the product is native or imported.
4. **Target**: an indication whether the product is internal or exported.
5. **Composition**: an indication whether the product is basic or complex.
6. **Attribute values**: a sequence of values, corresponding to the attributes listed in the global dictionary for this content or process type.[3]

The notation for the first five fields is similar to the notation for attribute values: $e[Kind]$, $e[Type]$, $e[Source]$, $e[Target]$, and $e[Composition]$.

3.1.2 Catalog

A *service* is an offering of an information product by a member of the virtual enterprise for a price. Each member of the virtual enterprise advertises the services that it offers by means of a *catalog*. The catalog is simply the inventory items for which the target indication is "export".

Note that a catalog may include services that require the offering member to obtain assistance from other members of the enterprise. This is referred to as *subcontracting* (outsourcing). Consequently, a change in a catalog (such as a price increase) may propagate to other catalogs.

Each member *distributes* its catalog to a subset of the members of the enterprise, and receives catalogs from other members. A member must either send its catalog to or receive a catalog from at least one other member (otherwise this member would not be able to participate in any activity of the virtual enterprise). This distribution creates the *infrastructure* of the virtual enterprise, as it describes the various channels of procurement. Any change to a catalog, such as a price increase, requires the redistribution of the catalog. The set of members to which a member $m_i \in M$ distributes its catalog is a subset D_i of M. The infrastructure is a subset D of $M \times M$.

[3]These values include $Product\_Code$ (or, for imported product, the combination $Member\_Identifier$: $Product\_Code$).

3.2 Production Plans

For each complex content or process in the member's inventory there must be a *production plan* (a manufacturing formula). These production plans are expressed in terms of other contents and processes. In addition to describing the structure of complex products, production plans also assign their output the appropriate attribute values.

In describing production plans, we use the following notation. c and p identify a content and a process, respectively. If the product is native, then c and p are product codes from this member's inventory. If the product is imported, then c and p are *external* product codes; i.e., each is a combination of a member identifier and a product code from that member's catalog.

3.2.1 Content Manufacturing

Production plans must be provided for each *native complex* content. Native basic contents and imported contents are simply referenced by their product codes. The production plan for native complex content c is a combination of a process p and attribute assignments ϕ_A, as follows.

$$\begin{aligned} c &\leftarrow p(c_1, \ldots, c_n) \\ c[A] &\leftarrow \phi_A(val(p), val(c_1), \ldots, val(c_n)) \text{ for every attribute } A \text{ in } att(C) \end{aligned}$$

The meaning of the first line is that applying the process p to a sequence of contents $c_1, \ldots, c_n$ produces the content c. The second line describes a set of assignments ϕ_A, one for each attribute A of c. Each ϕ_A assigns a value for the attribute A based on the attributes of the input contents $c_1, \ldots, c_n$ and the process p.

Note that each input content c_i may be either native or imported. If an input content c_i is native and complex, then another production plan must be provided for c_i. Hence, the production plan for c may recursively involve other production plans (it must be, of course, free of cycles).

Note also that a content c may have *several* production plans, allowing for alternative manufacturing processes. There is a special production plan called *substitution* that involves no processing:

$$c \leftarrow c'$$

Substitution allows one content to substitute for another. For example, using different substitution formulas with the same left-hand-side, one may specify alternative imports for the same content.

Finally, note also that the process p may be native or imported. When p is imported from member m, then after $c_1, \ldots, c_n$ are materialized, they are sent to m, who subsequently sends back the content c. Note that attribute values are included with the input and output contents.

3.2.2 Process Manufacturing

As with contents, production plans must be provided for each *native complex* process. Native basic processes and imported processes are simply referenced by their product codes. The production plan for a native complex process p is specified as follows.

$$\begin{aligned} p(x_1, \ldots, x_n) &\leftarrow p'(p_1(y_{1,1}, \ldots, y_{1,n_1}), \ldots, p_n(y_{n,1}, \ldots, y_{n,n_k})) \\ p[A] &\leftarrow \phi_A(val(p'), val(p_1), \ldots, val(p_n)) \text{ for every attribute } A \text{ in } att(P) \end{aligned}$$

In the first line, each y variable in the right-hand side is taken from the set of x variables in the left-hand side, and each x variable appears at least once among the y variables. The meaning of the first line is that the process p is a combination of n more elementary processes $p_1, \ldots, p_n$, whose intermediate products are combined with a process p'. The second line describes a set of assignments ϕ_A, one for each attribute A of p. Each ϕ_A assigns a value for the attribute A based on the attributes of the component processes $p_1, \ldots, p_n$ and the combining process p'.

As a simple example, consider this production plan in which $n = 1$:

$$p(x) \leftarrow p'(p_1(x))$$

In this case p is simply a "pipe" comprising the processes p_1 and p'.

Again, production plans must be specified for the component processes p_i that are native and complex. Finally, the production plan

$$p \leftarrow p'$$

allows to *substitute* the process p' for the process p.

Using such production plans, it is possible to establish the *cost* of every content or process. It is the *price* paid for externally procured contents and processes, plus the *cost* of internally procured products and processes. The price of this content (as advertised in this member's catalog) would usually be higher. The difference is *profit*. The new cost would be determined by the assignment ϕ_{Cost}.

Consider a process $c \leftarrow p(c_1, \ldots, c_n)$ and assume that p is a native process. In this case, the assignment ϕ_{Cost} would be

$$c[Cost] = p[Cost] + \sum_{\substack{c_i \\ c_i[Source]='native'}} c_i[Cost] + \sum_{\substack{c_i \\ c_i[Source]='import'}} c_i[Price]$$

3.3 Performance Indicators and Constitutional Rules

Using the notation that has been developed in earlier sections we may express various indicators and rules. The indicators are quantitative characterizations of the enterprise, whereas the rules express behavior that must be adhered at any point in time.[4]

[4] In the following we use generic mathematical notation to express indicators and rules. We intend to develop a more specialized language in which these will be defined.

3.3.1 Performance Indicators

Performance indicators may be structured in a three-level hierarchy:

1. Product-specific
2. Member-specific
3. Enterprise-wide

Product-specific indicators characterize individual products. Member-specific indicators characterize the performance of a member; often, they are defined by summarizing product specific-indicators. Similarly, enterprise-wide indicators characterize the performance of the entire enterprise; often, they are defined by summarizing member-specific indicators.

Product-specific indicators could be considered *derived attributes*. A simple example is profit, the difference between the price and the cost of a product:

$$Profit(e) \quad = \quad e[Price] - e[Cost]$$

Another example is *exclusivity*: the number of comparable products available throughout the enterprise. Let $e \in Inv(m), e[Target] =' export'$

$$Exclusivity(e) \quad = \quad \mid \{e' \mid e' \in Inv(m') \wedge m' \neq m \wedge e'[Target] =' export' \wedge e'[S] = e[S]\} \mid$$

A product e is *exclusive* if $Exclusivity(e) = 0$. The purpose of the restriction to export products is to avoid interim products, which are unavailable to outsiders. A third example is *external-dependence* (import-dependence), which measures the ratio of the cost of imports used in the manufacturing of a product to the price of the product. Let $Imp(e, q)$ be the set of imported products used in a production plan q for product e. Then

$$Dependence(e) \quad = \quad \min_q \frac{\sum_{e' \in Imp(e,q)} e'[Cost]}{e[Price]}$$

Additional product-specific indicators could be defined similarly: The *robustness* of production could be measured by the number of alternative production plans for a product, the *complexity* of a product could be measured by the (average) depth of its production plans or the (average) number of components used, and so on.

An example of a member-specific indicator is the *breadth* of a member, which measures the number of (export) products in its inventory (i.e., the size of its catalog):

$$Breadth(m) \quad = \quad \mid \{e \mid e \in Inv(m) \wedge Target[e] =' export'\} \mid$$

Using product exclusivity, one could measure *member exclusivity* as the average exclusivity indicator of this member's products. The *assets* of a member can be defined as the total

price of all the items in its inventory marked "export" less the total cost of the items marked "import". The difference represents the *total value added* by this member.

$$\begin{aligned} Import(m) &= \sum_{\substack{e \in Inv(m) \\ e[Source]='import'}} e[Cost] \\ Export(m) &= \sum_{\substack{e \in Inv(m) \\ e[Target]='export'}} e[Price] \\ Assets(m) &= Export(m) - Import(m) \end{aligned}$$

An important indicator is the *level of interdependence* (cooperation) among the members of a virtual enterprise. This may be measured as the ratio of imports to assets: the higher the ratio the more dependent is the member on other members:

$$Depend(m) = \frac{Import(m)}{Assets(m)}$$

An enterprise-wide indication of interdependence may be obtained by averaging the individual interdependence indicators:

$$Depend = \frac{1}{n} \sum_{m \in M} Depend(m)$$

If this indicator falls below a threshold, it may be advisable to reorganize the enterprise or possibly dismantle it altogether.

3.3.2 Constitutional Rules

The global behavior of a virtual enterprise is governed by a set of enterprise rules. These rules reflect the *constitution* of the enterprise and must be satisfied at any time. This constitution gives specific enterprises their individual characteristics. Some examples that illustrate this concept follow.

We already mentioned that each member of the virtual enterprise must be involved in at least one catalog exchange. This rule may be specified as follows:

$$\begin{aligned} &\forall m \in M \\ &\mid \{m' \mid m' \in M \wedge ((m, m') \in D \ \vee \ (m', m) \in D)\} \mid \geq 1 \end{aligned}$$

There may be a rule that requires all members to give fellow members preferred treatment over clients; that is, a member cannot offer the same products to clients at prices lower than those it offers to members:

$$\begin{aligned} &\forall m \in M \ \ \forall e \in Inv(m) \\ &e[Price] \geq e[Member\_Price] \end{aligned}$$

A rule may be defined to disallow "dumping", the practice of selling items below their cost; that is, the price charged for must exceed the cost incurred in producing the item (an example of cost calculation was given earlier).

As a final example, consider a rule that establishes *product exclusivity*; i.e., there are no comparable products in the virtual enterprise.

$$\forall m_1, m_2 \in M \; \forall e_1 \in Inv(m_1) \; \forall e_2 \in Inv(m_2) : \\ e_1[S] = e_2[S] \Longrightarrow m_1 = m_2 \; \wedge \; e_1[Product\_Code] = e_2[Product\_Code]$$

Recall that identical products are comparable; in effect, this rule states that comparable products are identical.

4 Transactions

The infrastructure defined by the distribution of catalogs supports the operations of a virtual enterprise. The basic unit of operation in a virtual enterprise is a *transaction*. A transaction begins when a request for an advertised service (content or process) is sent from one participant to another, and terminates when the request is satisfied.

There are two types of transactions in a virtual enterprise.

- **External transaction**. An external transaction is a request for a service which is submitted from a client to one of the members of the virtual enterprise. The member processes the request and provides a solution. A member of the virtual enterprise who processes an external transaction acts in a role of a *service provider*.
- **Internal transaction**. To satisfy an external transaction, a service provider may decide to purchase a service from another member. Such transactions are called *internal transactions* or *subcontracts*. A member of the virtual enterprise who processes an internal transaction acts in a role of a *subcontractor*. Sub-contracting is related to information brokering.

The execution of external transactions is the ultimate purpose of the virtual enterprise. Each member of a virtual enterprise may act as a service provider on some transactions and as a subcontractor on other transactions.

To initiate a transaction, the requester must provide the exact specifications of the service required. Because products may have attributes that are multivalues or functions, a preliminary exchange of information may be necessary. Such exchanges are called *inquiries*. Actual requests for service are called *orders*. Hence, in general, products are traded with *two-phase* transactions.

4.1 Inquiry

An inquiry is a request for additional information, and is necessary when some of the attributes of the product are multivalued and other attributes are functionally dependent on

them. For example, the *Price* of a product may depend on a multivalued attribute *Quantity*. As another example, *Quality* may depend on *Size*, e.g., low quality for large size, or high quality for small size, with the same *Price* for either combination.

In an inquiry, the requester specifies values for some attributes. In response, the service provider specifies values for all other attributes. There are two types of inquiries.

- **Simple.** In a simple inquiry, the inquirer provides specific values for each of the multivalued attributes. For numerical attributes, a fixed value may also be *min* or *max*, indicating the lowest or highest in the set. In response, the provider returns the corresponding values for the functional attributes.
- **Optimum.** In an optimum inquiry, the inquirer selects one attribute as a *target* (it may be a multivalued attribute or a functional attribute, but its domain must be totally ordered) and provides an optimization instruction (*min* or *max*) for this attribute. For each of the other multivalued or functional attributes the inquirer has the option of either (1) leaving it unchanged (i.e., a *don't-care* declaration), (2) restricting it (e.g., an attribute originally in the range $(10, 100)$ may be restricted to the narrower range $(40, 60)$), or (3) fixing it with a single value. In response, the provider returns the values for the multivalued attributes (and the corresponding values for the functional attributes) that optimize the target attribute.

Note that to determine its responses, it may be necessary for the provider to initiate inquiries of its own to its subcontractors.

Assume a product whose *Price* depends on the multivalued attributes *Quantity* and *Quality*. Specifically, assume that *Quantity* is in the range $[1, 100]$ and *Quality* is in the range $[1, 3]$ and the dependency of *Price* on *Quantity* and *Quality* is according to this table:

Quality	Quantity	Price
1	1–10	50
1	11–50	45
1	51–100	40
2	1–50	80
2	51–100	70
3	1–100	120

In a *simple* inquiry the inquirer specifies the quantity and quality and the provider returns the price. For example, for $Quality = 2$ and $Quantity = 40$ the provider would return $Price = 80$, and for $Quality = max$ and $Quantity = 10$ the provider would return $Price = 120$. In an *optimum* inquiry, the inquirer can restrict the quality or the quantity to narrower ranges and ask for the best price. For example, for $Quality \geq 2$, $Quantity \leq 80$ and $Price = min$, the provider would return $Quality = 2$, $51 \leq Quantity \leq 80$ and $Price = 70$. Alternatively, the inquirer may ask for the best quality that may be obtained below a particular price. For example, for $Quality = max$, $Quantity \leq 60$ and $Price \leq 100$, the provider would return $Quality = 2$, $51 \leq Quantity \leq 60$ and $Price = 70$.

4.2 Order

After making the necessary inquiries, a participant may issue an order. Each order must include these parameters:

1. The *code* of the product ordered.
2. Specific *values* for its multivalued attributes, if any.
3. The *input contents* (and their attributes), if the product is a process.

Once a member receives an order (either external or internal), it must put together a plan that will deliver the product as specified.

4.3 Optimization

If every product (content or process) had only one set of specifications (i.e., without multivalued attributes such as *Quality* or *Size*), and if every product had a unique production plan, then the fulfillment of orders would be a straightforward process. By allowing a product to have different specifications and alternative production plans (including multiple options for importing component products), the fulfillment of orders becomes a process that requires *optimization*.

Note that we assume that when a service provider advertises a product with multivalued attributes, there must be appropriate production plans to manufacture products that will meet *any* of the advertised attributes. Some issues involved in such optimization are discussed below.

The resources necessary for putting together an execution plan may be available from multiple sources, thus suggesting various alternatives that must be considered. We illustrate this with two examples.

- Content may be available from multiple sources with different attributes. For example, a member m who needs content c in quantity q may have the options of
  1. Buying c with in quantity q at price p.
  2. Buying c in higher than needed quantity q' but at a lower price p'.
- Content may be derived in different ways. For example, a member m who needs content c may have the options of
  1. Buying c from member m_1.
  2. Buying content c' from member m_2 and using the services of member m_3 to transform c' to c.

In general, each service provider adopts an attribute (or a weighted combination of attributes) that would serve as its *optimization target*. Given an inquiry or an order, the service provider would derive all the possible production plans, calculate the value of the optimization target for each plan, and choose the optimal plan. While optimization is defined through an exhaustive process that considers every possible plan, in practice, when this approach is infeasible due to a large number of plans, heuristic optimization methods should be developed.

Typically, a service provider would attempt to optimize the *cost* of production, while satisfying all the product specifications. That is, among the production plans that answer the specifications, the provider would choose the plan whose cost is minimal.

5 Conclusion

We presented a preliminary version of VirtuE, a model for virtual enterprises that collaborate on the manufacturing of information products. Some of the features of VirtuE that make it suitable for this purpose are

1. Two types of information products, *content* and *process*, and a *global dictionary* for knowledge coordination.
2. *Inventories* and *production plans* for expressing the manufacturing processes of information products.
3. *Catalog exchanges* and *two-phase transactions* (inquiry and order) for enabling the exchange of information products.
4. *Constitutional rules* and *performance indicators* for creating virtual enterprises with different characteristics and for monitoring their behavior.

There are many possible research directions to follow up the preliminary results presented in this paper, and we mention here briefly four such directions.

1. **Enduring products and transactions.** The model we described considered "one-time" products and transactions; that is, each transaction traded a single, already available commodity. Many information products are manufactured *continuously* from *streams of information*. Accordingly, transactions are contracts for the continuous supply of services. The proper extension of VirtuE to model continuous services is currently under investigation.
2. **Tracking performance over time.** The current model does not track the actual performance of a virtual enterprise over time. Such tracking would permit new performance indicators, such as *demand* or *profit*.

3. **Dynamic reorganization.** An important advantage of virtual enterprises is their ability to adapt quickly to changing markets. By monitoring the actual performance of an enterprise, including changes in the types or quantities of products ordered, it would be possible to prescribe changes in membership, production plans and infrastructure that would improve the overall performance of the enterprise. A simple analogy from the area of databases is the reorganization of distributed databases, to accommodate dynamically changing patterns of transactions.

4. **Extended cost modeling.** Through the use of performance indicators and constitutional rules, VirtuE attempts to model the business aspects of information exchange and manufacturing. A possible extension is to consider cost models such as [1, 2].

Once the VirtuE model has been finalized, the next steps would be to develop a functional specification for a software environment based on this model that will support the activities of a virtual enterprise for information markets, and subsequently to develop a prototype system in accordance with these specifications.

References

[1] W. Appel and R. Behr. Towards the theory of virtual organisations: A description of their formation and figure. *Virtual-Organization.Net Newsletter*, 2(2):15–36, June 1998.

[2] B. Belgradek, K. Kalpakis, and Y. Yesha. Maximizing seller's profits for electronic commerce. In *Proceedings of International IFIP/GI Working Conference on Trends in Distributed Systems for Electronic Commerce (W. Lamersdorf and M. Mertz, Editors)*, Lecture Notes in Computer Science No. 1402, pages 26–38. Springer-Verlag, Berlin, Germany, 1998.

[3] A. Mowshowitz (Editor). Special section on virtual organizations. *Communications of the ACM*, 40(9):30–64, September 1997.

[4] K. Erben and K. Gersten. Cooperation networks towards virtual enterprises. *Virtual-Organization.Net Newsletter*, 1(5):12–22, December 1997.

[5] A. Gal and D. Montesi. Inter-enterprise workflow management systems. In *Proceedings of the 10th International Workshop on Database and Expert Systems Applications* (Florence, Italy, 1–3 September), pages 623–627, 1999.

[6] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: from process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, April 1995.

[7] D. Georgakopoulos, H. Schuster, A. Cichocki, and D. Baker. Managing process and service fusion in virtual enterprises. *Information Systems: Special Issue on Information Systems Support for Electronic Commerce*, 24(6):429–456, September 1999.

[8] C. Godart, O. Perrin, and H. Skaf. Coo: a workflow operator to improve cooperation modeling in virtual processes. In *Proceedings of the Ninth International Workshop on Research Issues on Data Engineering* (Sydney, Australia, March 23-24), pages 126–131, 1999.

[9] P. Grefen and Y. Hoffner. Crossflow: Cross-organizational workflow support for virtual organizations. In *Proceedings of the Ninth International Workshop on Research Issues on Data Engineering* (Sydney, Australia, March 23-24), pages 90–91, 1999.

[10] V. Grover and T. C. Teng. E-commerce and the information market. *Communications of the ACM*, 44(4):79–86, April 2001.

[11] D. Heimbigner and D. McLeod. A federated architecture for information management. *ACM Transactions on Office Information Systems*, 3(3):253–278, July 1985.

[12] V. Kashyap and A. P. Sheth. Semantics-based information brokering. In *Proceedings of the Third International Conference on Information and Knowledge Management*, pages 363–370, 1994.

[13] S. Laufmann. The information marketplace: The challenge of information commerce. In *Proceedings of the Second International Conference on Cooperative Information Systems (M. L. Brodie, M. Jarke and M. P. Papazoglou, Editors)*, pages 147–157, 1994.

[14] P. Monge and G. DeSanctis (Editors). Special issue on virtual organizations. *Organization Science*, 10(6), November-December 1999.

[15] S. Prabhakar, J. Huang, J. Richardson, J. Srivastava, L. EePeng, S.-Y. Hwang, S. B. Navathe, A. Savasere, and M. Foresti. Federated autonomous databases: Project overview. In *Proceedings of the Third International Workshop on Research Issues on Data Engineering: Interoperability in Multidatabase Systems (H.-J. Schek, A. P. Sheth, B. D. Czejdo, Editors)*, pages 216–219, 1993.

[16] L. J. Seligman and L. Kerschberg. Knowledge-base/database consistency in a federated multidatabase environment. In *Proceedings of the Third International Workshop on Research Issues on Data Engineering: Interoperability in Multidatabase Systems (H.-J. Schek, A. P. Sheth, B. D. Czejdo, Editors)*, pages 18–25, 1993.

[17] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous and autonomous databases. *Computing Surveys*, 22(3):183–236, September 1990.

[18] B. Suter. A cooperation platform for virtual enterprises. In *Proceedings of the VoNet Workshop on Organizational Virtualness (P. Sieber and J. Griese, Editors)*, pages 155–164, 1998.

[19] *Virtual Organization Net.* Electronic Journal of Organizational Virtualness (ISSN 1422-9331). http://www.virtual-organization.net.

[20] D. Worah and A. P. Sheth. Transactions in transactional workflows. In *Advanced Transaction Models and Architectures (S. Jajodia and L. Kerschberg, Editors)*, pages 3–34. Kluwer Academic Publishers, 1997.