J Intell Inf Syst (2008) 30:33–53
DOI 10.1007/s10844-006-0013-0

VirtuE: a formal model of virtual enterprises for information markets

Alessandro D'Atri · Amihai Motro

Received: 1 June 2005 / Revised: 1 June 2005 /
Accepted: 10 February 2006 / Published online: 1 February 2007

Abstract A vital 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. Often, 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 complex 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. VirtuE has three principal components. First, it defines a distributed *infrastructure* with concepts such as members, products, inventories, and production plans. Second, it defines *transactions* among members, to enable collaborative production of complex products. Finally, it provides means for the *instrumentation* of enterprises, to measure their performance and to govern their behavior.

Keywords Virtual enterprise · Information markets · Information manufacturing

A. D'Atri
CeRSI, Luiss "Guido Carli" University, Rome, Italy
e-mail: datri@luiss.it

A. Motro (✉)
Information & Software Engineering Department, George Mason University,
Fairfax, VA, USA
e-mail: ami@gmu.edu

1 Introduction: information markets and virtual enterprises

A vital 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 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.

As these examples illustrate, there are various forms of derivation. An information product may be an *aggregation* of other information products; for example, a collection of news items and photographs on a particular subject, or a union of individual mailing lists. An information product may be a *translation* of another product, such as a translation of a news item into another language, or the conversion of a photograph from one format to another. A more demanding form of derivation is *refinement*; for example, the analysis of raw financial data to produce stock market recommendations, or the cleansing of data to remove errors and resolve inconsistencies. As with the manufacturing of physical objects, the road from raw materials to finished goods is sometimes long and arduous. Along the way, the product increases in value.

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 coalition of business entities, chosen from a larger community of available business entities called the *marketplace*, that collaborate on the manufacturing of complex 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. 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.

In this paper we describe the VirtuE model for virtual enterprises whose core business is information products. The purpose of such a model is threefold. The concept of virtual enterprises has been previously articulated by others (see the discussion in Section 2); yet, at the present, the literature does not show unanimity

with respect to essential principles, precise terminology, or formal definitions. VirtuE is therefore an attempt to establish a uniform platform for virtual enterprises, in which existing concepts can be formalized and standardized. Second, such formal treatment enables deeper (often quantitative) investigation of additional aspects of virtual enterprises, thus advancing the area even further. Finally, a formal model is an essential step before undertaking an implementation of a software system that supports the activities of virtual enterprises.[1]

Essentially, the VirtuE model defines a distributed environment for virtual enterprises. The model has three principal components: (1) infrastructure, (2) procurement, and (3) instrumentation. We review them briefly.

Infrastructure. The infrastructure of a virtual enterprise is defined with these six fundamental concepts.

1. *Members.* Each virtual enterprise is a coalition of individual members. These members are chosen from a larger community of potential business entities. The members may be modeled as nodes in a network.
2. *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 product.
3. *Dictionary.* The dictionary is an enterprise-wide resource that maintains and shares knowledge about the products that are exchanged among the members. It is intended to assure consistency of interpretation across the enterprise.
4. *Inventories.* The products either used or created by each enterprise member are described in a local resource called inventory. Items in the inventory may be regarded as *instances* of product types described in the dictionary.
5. *Catalogs.* The subset of the inventory that is offered to other members of the enterprise is called catalog. Members distribute their catalogs to other members of their choice. This defines a directed graph on the member nodes; the graph models the infrastructure on which products are exchanged.
6. *Production Plans.* For complex products (products that are created from products that are more elementary), production plans must be provided. A production plan specifies how contents and processes are combined to derive the new product. In particular, it specifies the dependence of a product on products that must be procured from other members.

Procurement. Since component products are often obtained from other enterprise members, a procurement mechanism is necessary. Procurement is executed in *transactions*. Since production plans may branch into multiple alternatives, each requiring different transactions to import different components from different members, each member must designate a primary production plan for each of its products. Typically, the primary plan is chosen to optimize the interests of the producing member.

Instrumentation. 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

[1] A concise, preliminary description of VirtuE was given in D'Atri and Motro (2002).

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.

The VirtuE model is described in four sections. Section 3 describes the basic structures of the model: members, products, and a dictionary for coordinating knowledge about products. The manufacturing of new information products from existing products is discussed in Section 4, which introduces inventories, catalogs and production plans. Section 5 is devoted to procurement. It defines transactions and describes the process of fulfillment. Section 6 discusses instrumentation by means of performance indicators and constitutional rules. Section 7 concludes this paper with a brief summary and a list of subjects for further investigation. We begin with a review of research related to our work.

2 Related work

Our work is at a junction of several research areas, and we discuss here the most relevant five areas: (1) information marketplaces, (2) virtual enterprises, (3) workflow management in virtual enterprises, (4) information brokering, and (5) federated databases.

A marketplace is a commercial sphere where buying and selling takes place. In modern economies, marketplaces are usually dedicated to specific types of goods or services. Thus, an *information marketplace* is a setting in which information products and services are being traded. The importance of information marketplaces, especially in relationship to electronic commerce, has been recognized for quite some time (Grover & Teng, 2001; Laufmann, 1994). It is within such a marketplace that VirtuE operates, advocating a specific type of collaborative organization to generate new products for this marketplace. In this paper, we also use the term "marketplace" in a more restricted sense, to refer to the community of business entities that are potential participants in collaborative ventures.

Cooperatives of independent entities that collaborate on the manufacturing of goods have been around for decades. Often the members of such cooperatives reside in the same industrial district.[2] This geographical proximity provides advantages of common culture and mutual trust (Brusco, 1992). The collaborating entities are often of small and medium size, and their strategic approach is to focus on their core business (i.e., excel in a limited section of the "value chain"), and to seek collaborations with neighboring entities to perform the other requisite activities in the value chain.

Essentially, virtual enterprises (also referred to as virtual organizations or corporations) are modern versions of these cooperatives, from which geographical constraints have been removed. By means of communications and information technology the entities participating in an alliance need not be confined to a particular location. Virtual enterprises are often characterized as agile, flexible, dynamic,

[2] A notable example is the Saxon region in Germany (Erben & Gersten, 1997).

proactive, and unconstrained by predefined structures. The essential principles of virtual enterprises may be summarized thus (Barbini & Datri, 2005; Camarinha-Matos, 2003; Davidow & Malone, 1992; Goldman, Nagel & Preiss, 1995):

- *Market-driven cooperation.* Virtual enterprises are set-up to exploit specific business opportunities, and are therefore intensely result-oriented.
- *Complementariness of skills.* The members of each virtual enterprise are chosen to complement each other's competencies.
- *Dynamic participation.* Members can join or withdraw from an enterprise, according to their own self-interests.
- *Coalition of peers.* A virtual enterprise is not dominated by individual members; rather, it is a coalition of peers.
- *Controlled sharing.* Members work together, integrating their processes and sharing their resources; yet, sharing is not boundless, and members may protect certain assets from their peers.
- *Limited duration.* Virtual enterprises are not intended to be permanent, or even long-term organizations; rather, they are aimed at achieving short or medium term goals.

The interest of the information technology research community in the area of virtual enterprises dates to the mid-1990s, with much of the work focusing on organizational issues, communication processes and information systems support (Monge & DeSanctis, 1999; Mowshowitz, 1997; Virtual Organization Net, 2005). Proposed paradigms to support virtual enterprises include the VEGA software platform (Suter, 1998) and a methodology for fusing the separate business processes of enterprise members (Georgakopoulos, Schuster, Cichocki & Baker, 1999). An overview of current approaches towards the establishment of infrastructures for virtual enterprises is given in Camarinha-Matos and Afsarmanesh (2004). The focus of VirtuE are information markets. Two segments of the information market that have been considered suitable for virtual enterprises are tourism (Afsarmanesh & Camarinha-Matos, 2000) and learning (D'Atri & Pauselli, 2004).

Workflow models have become the principal tool for modeling business processes and interactions among organizations, and workflow management systems provide powerful support for automating these activities (Gal & Montesi, 1999; Georgakopoulos, Hornick & Sheth, 1995). These concepts have also been used as a framework for the virtual enterprise paradigm (Tagg, 2001). For example, Grefen and Hoffner (1999) and Godart, Perrin and Skaf (1999) propose workflow management systems to support loosely coupled interorganizational cooperations and to automate the activities of virtual enterprises. Workflow management is therefore related to VirtuE's concept of production plans (manufacturing formulas for information products).

A VirtuE member who is contracted by a external client to provide needed information, and who then seeks the cooperation of other enterprise members in satisfying this request, resembles in his behavior an *information broker*. Essentially, an information broker is a software tool that receives a request for information, then searches many different repositories for information relevant to the request, and finally assembles its findings in an answer to the request. Such an architecture,

consisting of information providers, information brokers and information consumers is described in Kashyap and Sheth (1994). In a way, each VirtuE member, is an information broker of sorts, but with three important differences. First, all VirtuE members can act as "brokers," "providers" or "consumers" of information; second, VirtuE's members do not search for relevant information; rather, information is advertised and brought to their attention. Finally, VirtuE assumes that the semantics of the information are commonly understood, whereas Kashyap and Sheth (1994) assumes that relevance has to be discovered. Altogether, VirtuE's is a more structured and preplanned environment. It should be noted that information brokers share important features with information mediators (Weiderhold, 1992).

Finally, in the area of distributed database architectures, *federated database* models (Heimbigner & McLeod, 1985; Prabhakar et al., 1993; Seligman & Kerschberg, 1993; Sheth & Larson, 1990) may be considered predecessors of VirtuE. The term federated database has been used for a variety of architectures, but usually it refers to a distributed collection of participating database systems that exhibits three principles: (1) *Autonomy*: The participating systems maintain a high degree of autonomy, and each can function independently; (2) *Sharing*: These systems participate in a larger organization that provides mechanisms for information coordination and exchange; and (3) *Heterogeneity*: The participants may support different data models and languages and could be implemented in different software systems. Federated databases, however, do not provide specific features for manufacturing new information products, and do not attempt to model the business aspects of information exchange.

3 Basic structures

In this section we define the basic concepts of the VirtuE model. Each virtual enterprise consists of members and products. A global resource, called dictionary, is used to coordinate knowledge about products among the members.

3.1 Members

A *marketplace* is a community of business entities that are potential participants in business coalitions. Each virtual enterprise chooses its *members* from this marketplace. The members are independent but have shared interests. They 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 clients. After the community had been established, it could evolve because a new member joins or an existing member departs. This form of evolution provides the virtual enterprise with flexibility and allows it to adapt to new market situations.

Note that we do not assume that members are automated systems, and that a virtual enterprise is a fully automated interoperative system. Rather, the work performed by members combines human and computer activities.

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

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

3.2 Products

In practice, virtual enterprises may produce many different kinds of products. In VirtuE, we 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 kinds of information products: *content* and *process*.

3.2.1 Content

Content is an information item. Examples include "a database of customers and the products they purchased in the year 2004," "the codes of all stocks traded in the New York Stock Exchange and their closing values on March 31, 2005," "an image of the launching of the space shuttle Columbia on January 16, 2003," and so on. A *request* for this kind of product describes the information needed; the *response* is information content that satisfies 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 *Document*, *Image*, *Database*, 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) are partitioned into three 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* and *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 use $c[S]$ and $c[O]$ to denote, respectively, the subsequences of specificational and optional attributes of att(c).

Each content is required to have a valid value for the *Product_Code* attribute and for each of the specificational attributes. In the optional attributes, *null* values are permitted wherever valid values are unavailable.[3]

3.2.2 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 into another, (3) analyzing 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 $\mathrm{att}(P)$. Each attribute A in $\mathrm{att}(P)$ is associated with a domain $\mathrm{dom}(A)$. Let p be a process of type P, then $p[A]$ denotes the value of the attribute A for this process. $\mathrm{val}(p)$ denotes the sequence of all attribute values of p, in correspondence with $\mathrm{att}(P)$. The attributes $\mathrm{att}(P)$ are divided into the same three 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 may be classified as either *intensional* or *extensional*. Content types, content attributes and their domains, and process types, process attributes and their domains are intensional concepts (i.e., they describe a schematic view of the information products); whereas content instances and values, and process instances and values are extensional concepts (i.e., they describe individual instances of the information products).

A simple analogy that illustrates these concepts is 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.

[3] Null values are discussed in Section 3.2.3.

3.2.3 Special attribute values

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

Null value. The *null* value, already mentioned, is used whenever a valid attribute value is not available. The reason for its unavailability may be that the value is either inapplicable, unknown or undisclosed. For example, the *Quality* of a particular content may be unknown. Null values are allowed only in the optional attributes.

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* (the number of product units that are packaged together) is often a multivalue. For numerical attributes, multivalues are often specified as *ranges*. Hence, the semantics of multivalues is that of disjunction. The advantage of multivalues is that they summarize multiple products in a single compact description. Multivalues are not permitted when products must be fully specified.[4]

Functional value. 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 an attribute of a process may be a function of other attributes of this process or of attributes of its input contents. Functional values are often used in conjunction with multivalues. Thus, *Price* may be a function of *Quantity*; for example, *if Quantity* < 10 *then 30 else 25*. Note that an attribute of a process may be a function of the attributes of its input contents; for example, *Compression_Rate* may be a function of the *Density* of the input content.[5]

3.3 Dictionary

We assume that all intensional information (i.e., types, their attributes, and the attribute domains) is maintained in an enterprise-wide resource called the *dictionary*. This global knowledge resource is available to every member of the enterprise (as well as to the entities of the encompassing marketplace and to external clients). Every product in the virtual enterprise is an instance of a type described in the dictionary.

The purpose of the dictionary is to assure consistency of terminology across the enterprise. It is thus similar in purpose to multidatabase schemas (Motro, 1999) or XML schemas (Lee & Chu, 2000).

Specifically, the dictionary is intended to assure that when members or clients exchange information, requests and responses are interpreted correctly by the parties involved. Thus, using the dictionary, members of the enterprise would be able to determine things such as (1) the meaning of content c; (2) the difference between content c_1 and content c_2; or (3) whether content c is equal to the result of applying process p to contents $c_1, \ldots, c_n$; i.e., whether $c = p(c_1, \ldots, c_n)$.

[4] For example, when one members sends content to another member for processing, the accompanying specificational attributes must be free of multivalues. This subject is discussed in Section 5.

[5] A concept related to functional values, *derived attributes*, is introduced in Section 6.1.

Formally, the 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 sequence of attributes att(C) and each process type $P \in \mathcal{P}$ is described with a sequence of attributes att(P). In turn, each attribute A in att(C) or in att(P) is associated with a domain dom(A).

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

1. *Identical products.* The attribute *Product_Code* uniquely identifies a product within the 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 E are comparable if $e_1[S] = e_2[S]$. Intuitively, when two products are comparable, one could substitute for another.
3. *Similar products.* A *similarity measure* could be defined between any two product specifications (Frakes & Baeza-Yates, 1992). This measure would reach its maximum value when the specifications are equal, and would decrease in value as the specifications diverge. Intuitively, when a particular specification could not be satisfied, the most similar product available could be substituted.

Note that if two enterprise members offer the same product (for example, two copies of the same computer file), their offerings would not be recognized as identical products, as they would have different product codes. If appropriate information is included in the specificational attributes, then such products could be recognized as comparable products. For example, to recognize that two audio files correspond to the same song, the specificational attributes should include the title, the performer, the year of recording, the format and the length.

4 Manufacturing new products

The fundamental VirtuE paradigm is that members obtain information products from other members to manufacture more complex information products. This section explains how manufacturing is accomplished, using three new concepts: inventories, catalogs, and production plans.

4.1 Inventories

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

1. *Source*. Each product is either *native* or *import*. A native product is produced locally, whereas an import product is obtained from another member of the enterprise
2. *Target*. Each product is either *internal* or *export*. An export 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.

Import products are always internal. If a member wishes to export an import product, the product must first undergo a "change of identity."[6]

Products (contents or processes) are also classified according to their *composition*. Each product is either *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.

Import products are always classified as basic; that is, complex products are always native.

The set of 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 dictionary).
3. *Source.* an indication whether the product is native or import.
4. *Target.* an indication whether the product is internal or export.
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 dictionary for this content or process type.

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]. Note that products in the inventory are identified by their *Product_Code* (or, in the case of import products, by the combination *Member_Identifier.Product_Code*), which is the first component in the attribute value sequence.

4.2 Catalogs

Each member of the virtual enterprise advertises the products that it offers by means of a *catalog*. The catalog is simply the inventory items for which the target indication is "export."

Each member distributes its catalog to a subset of the members of the enterprise, and receives catalogs from other members. This distribution creates the *infrastructure* of the virtual enterprise, as it describes the various channels of procurement. 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.

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 $M \times M$. Graphically, the infrastructure is a directed graph on the set of nodes M, where an edge from member m_i to member m_j indicates a procurement channel that allows member m_j to obtain products from member m_i.

Note that a catalog may include products that require the offering member to obtain assistance from other members of the enterprise. This is referred to as *subcontracting*. Consequently, a member's catalog depends on the catalogs of its subcontractors. Therefore, a change in a catalog, such as the addition or the

[6]This subject is explained in Section 4.3.

withdrawal of a product, or a change in the specification of a product (e.g., a price change), requires its redistribution, and may then propagate to other catalogs. In addition, when a new member joins the enterprise and distributes its catalog, or when an existing member departs from the enterprise and withdraws its catalog, other members may have to update their catalogs.

Thus, catalog update and redistribution can have a "ripple effect." Indeed, it is possible that the process becomes cyclical; for example, two competing members could repeatedly update the specifications of their products (e.g., the price) to outdo each other. To guard against this possibility, it may be desirable to impose a limit on catalog updates; for example, require that catalogs remain in effect for a predefined duration.

4.3 Production plans

For each complex content or process in the member's inventory there must be a *production plan*. 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 an import, 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.

4.3.1 Content manufacturing

Production plans must be provided for each *native complex* content. Native basic contents and import 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.

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

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 specificational attributes of the input contents $c_1, \ldots, c_n$ and the process p.

As an example, assume a process that adds sound to a video clip. The process has two input contents: a video clip and a soundtrack, and its output content is a video clip with synchronized sound. The process also assigns values to the attributes of the output content; for example, *Length*, *Frame_Rate* and *File_Size* (possibly, the first two are copied from the attributes of the video clip).

Note that each input content c_i may be either native or import. 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. However, from the set of alternative production

plans (i.e., the production plans with the same left-hand-side), one plan is designated by the manufacturing member as the *primary* production plan. VirtuE does not prescribe any procedure for choosing the primary plan (though it may be assumed that members choose primary plans to optimize their interests; for example, to minimize the cost of the product or to maximize its quality).

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 plans with the same left-hand-side, one may specify alternative imports for the same content. For example, assume that product p_9 of member m_1 may be imported either from member m_2 (where its product code is p_5) or from member m_3 (where its product code is p_6), and is then offered as an export. p_9 will have two production plans:

$$p_9 \leftarrow m_2.p_5$$
$$p_9 \leftarrow m_3.p_6$$

The primary production plan indicates the preferred source. Note that in m_1's inventory there are three products: $m_2.p_5$ and $m_3.p_6$ have the designations "import", "internal", and "basic", whereas p_9 has the designations "native", "export", and "complex." While the process of substitution does not change the content, it assigns it a new *Product_Code* and may modify some of its other attributes (e.g., price).

Finally, note that the process p may be native or import. 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.

4.3.2 Process manufacturing

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

$$p(x_1, \ldots, x_n) \leftarrow p'(p_1(y_{1,1}, \ldots, y_{1,k_1}), \ldots, p_n(y_{n,1}, \ldots, y_{n,k_n}))$$
$$p[A] \leftarrow \phi_A(p'[S], p_1[S], \ldots, p_n[S]) \text{ for every attribute } A \text{ in att}(P)$$

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 specificational 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'.

As another example, consider a process p that assembles proposals (tenders) that conform to the requirements of calls-for-proposals. It begins with a draft text

document x_1, initial sketches x_2, and preliminary financial figures x_3. It incorporates three separate processes: process p_1 copyedits the draft text document; process p_2 creates professional illustrations from the initial sketches; and process p_3 creates a final budget from the preliminary figures. The contents produced by these three processes are integrated by a final process p_4 into a single package:

$$p(x_1, x_2, x_3) \leftarrow p_4(p_1(x_1), p_2(x_2), p_3(x_3))$$

A typical attribute of p could be *Time* (the turnaround time). Assuming each of the processes p_1–p_4 has a similar attribute, a possible assignment would be

$$p[Time] \leftarrow p_4[Time] + \max\{p_1[Time], p_2[Time], p_3[Time]\}$$

Again, production plans must be specified for the component processes p_i that are native and complex. From the set of alternative production plans, one plan must be designated *primary*. Finally, the production plan

$$p \leftarrow p'$$

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

Using primary 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, allowing for a profit.

Consider a native process $c \leftarrow p(c_1, \ldots, c_n)$ and assume that c has an attribute *Cost*. The assignment ϕ_{Cost} would be

$$c[Cost] \leftarrow p[Cost] + \sum_{\substack{1 \le i \le n \\ c_i[\text{Source}]=''native''}} c_i[Cost] + \sum_{\substack{1 \le i \le n \\ c_i[\text{Source}]=''import''}} c_i[Price]$$

5 Transactions

To exchange products over the infrastructure defined by catalog distribution, we introduce *transactions*. A transaction begins when a request for an advertised product (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 product 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 *provider*.
- *Internal transaction*. To satisfy an external transaction, a provider may decide to purchase products from other members. 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*. Subcontracting is related to information brokering (Kashyap & Sheth, 1994).

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

Before initiating a transaction, the procuring member consults the catalog of the providing member (the subcontractor), where the product is described. This description includes the attribute values of the product. Each of these values may be either a regular value, a null value, a multivalue, or a functional value. The procuring member must choose specific values for each of the multivalues (if any). Once this is done, the functional values (if any) are computed automatically, and the product is then fully specified. Hence, an *order* is a combination of a *Product_Code* and specific values for each of the multivalues. For products of type process, orders must include in addition the *input contents* and their specificational attributes. The latter should not have multivalues.

Ordering may be visualized as completing online computer forms. In these forms, there are four kinds of fields: fields that are already filled in (regular values), fields that are blank (null values), fields that must be filled in by means of drop-down menus (multivalues), and fields that are filled in automatically as the multivalues are chosen (functional values). When the order is for a process, the form also includes fields for the specificational attributes of each of the input contents, which the procuring member must complete (and then upload the contents themselves). The transaction is completed when the procuring member downloads the procured content or the content that is the output of the procured process.

As an example, consider an enterprise member who markets bundles of web design objects. Assume that this content is specified with six attributes: *Product_Code*, *Size*, *Format*, *Resolution*, *Quantity*, and *Price*. Assume that *Product_Code* is 2405, *Format* is "TIFF", and *Size* is left unspecified. Assume further that *Price* is a functional attribute that depends on the multivalued attributes *Resolution* and *Quantity*. Specifically, assume that *Resolution* is either "low", "medium", or "high", *Quantity* is either 10 or 50, and the dependency of *Price* on *Resolution* and *Quantity* is specified in this table:

Resolution	Quantity	Price
low	10	50
low	50	200
medium	10	70
medium	50	280
high	10	100
high	50	400

An example of an order would be

$$Product\_Code = 2405$$

$$Resolution = \text{“medium”}$$

$$Quantity = 50$$

To complete this order, the providing member will deliver a bundle of 50 web design objects of medium resolution for the price of 280.

As another example, consider an enterprise member who offers a list consolidation service. Assume that this process is specified with three attributes: *Product_Code*, *Method*, and *Price*; and it requires two input contents of type *List*, whose specificational attributes are *Size* and *Accuracy*. Assume further that *Product_Code*

is 6138, *Method* is "MagicMerge", and *Price* is a functional attribute whose value depends on the *Size* attributes of the two inputs, as follows $Price = (Size_1 + Size_2) \times 0.02$. An example of an order would be

$$Product\_Code = 6138$$
$$< List_1 > Accuracy = 0.5, Length = 8,032$$
$$< List_2 > Accuracy = 0.7, Length = 7,521$$

The provider will consolidate the given lists using the MagicMerge method, charging the procurer $(8,032 + 7,521) \times 0.02 = 311.06$. The output list will be assigned attribute values such as *Accuracy*= 0.8 and *Length*= 9,332.

5.1 Enduring transactions

The transactions we described procured "one-time" products; that is, each transaction was for a single product and for immediate delivery. Many information products are updated periodically. In such cases, transactions should be contracts for the recurring supply of products. We term such transactions *subscriptions*. Handling subscriptions in VirtuE does not require significant extensions.

Subscriptions (for either contents or processes) require that product descriptions include attributes such as *Begin_Delivery*, *End_Delivery*, and *Delivery_Frequency*. Most probably, the *Price* attribute will have a functional value that will depend on the period of subscription and the frequency. Production plans for products that have to be delivered periodically would rely on subcontracts that are subscriptions as well.

6 Instrumentation

This section explains how to create virtual enterprises possessing different characteristics by using constitutional rules, and how to monitor their performance with performance indicators. Indicators are quantitative characterizations of the enterprise, whereas rules express behavior that must be adhered to at any point in time.

6.1 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 e:

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

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

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

A product e is *exclusive* if $\text{Exclusivity}(e) = 0$. 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 $\text{Imp}(e, q)$ be the set of import products used in a production plan q for product e. Then,

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

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):

$$\text{Breadth}(m) = |\ \{e \mid e \in \text{Inv}(m) \wedge \text{Target}[e] = \text{“export”}\}\ |$$

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:

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

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:

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

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

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

where n is the number of members in the enterprise. High levels of *Depend* indicate a virtual enterprise which is highly collaborative.

6.2 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 is specified as follows:

$$\forall m \in M$$
$$|\{m' \mid m' \in M \wedge ((m, m') \in D \ \vee \ (m', m) \in D)\}| \geq 1$$

A stronger rule is *connectivity*, which corresponds to a requirement that the (undirected) infrastructure graph is connected. Formally,

$$\forall m, m' \in M \ \exists m_0, m_1, \ldots, m_n \in M$$
$$m_0 = m \ \wedge \ m_n = m' \ \wedge \ \forall\, 0 \leq i < n$$
$$(m_i, m_{i+1}) \in D \ \vee \ (m_{i+1}, m_i) \in D$$

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:

$$\forall m \in M \ \ \forall e \in \mathrm{Inv}(m)$$
$$e[Price] \geq e[Member\_Price]$$

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

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

$$\forall m_1, m_2 \in M \ \forall e_1 \in \mathrm{Inv}(m_1) \ \forall e_2 \in \mathrm{Inv}(m_2)$$
$$e_1[S] = e_2[S] \implies m_1 = m_2 \ \wedge \ e_1[Product\_Code] = e_2[Product\_Code]$$

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

There may be a rule that establishes that all members of the enterprise should have "assets" of comparable size; for example, the assets of the largest member would not be more than twice the assets of the smallest member. This would assure that the operations of the enterprise are not dominated by a single member.

There may be a rule that specifies a threshold level of cooperation; i.e., a threshold value for the indicator *Depend*, defined earlier. If this level is not reached, it may be advisable to reorganize the enterprise or possibly dismantle it altogether.

7 Conclusion

Virtual enterprises have gained support among many as the preferred structure for corporations in the 21st century. While the essential principles of virtual enterprises are mostly agreed upon, a formal model of virtual enterprise has been curiously

missing. Such a model fulfills important functions, not the least of which is that it promotes standardization of concepts, definitions and terminology.

We presented VirtuE, a formal model of virtual enterprises whose core business are information products and processes. 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 describing the manufacturing processes of information products.
3. *Catalog* distribution and *transactions* for enabling the exchange of information products, to support production plans that incorporate products from other members.
4. *Constitutional rules* and *performance indicators* for creating virtual enterprises with different characteristics and for monitoring their behavior.

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

Plan Selection. In VirtuE the resources necessary for putting together a production plan for a given product may be available from multiple sources. This results in the possibility of alternatives production plans for the same product.

As an example, 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 in quantity q at price p, or (2) buying c in higher than needed quantity q' but at a lower price p'.

As another example, 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, or (2) buying content c' from member m_2 and using the services of member m_3 to transform c' into c.

In such situations, the service provider should adopt an attribute (or a weighted combination of attributes) that would serve as its *optimization target*, and, given an order, should select the production plan that optimizes the target. In VirtuE we assumed simply that the plan to be used is determined ahead of time (it is the *primary* production plan). The optimal selection of production plans (possibly in real-time) is a challenging research issue.

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. An analogy from the area of databases is the reorganization of distributed databases, to accommodate dynamically changing patterns of transactions.

Optimal production plans. Assume a product that is assembled by a *hierarchical* process: Members at the bottom of this hierarchy assemble elementary products into more complex products that they send to members at the next level; these members, in turn, assemble these complex products into yet more complex products that they

send to members at the next level; until, finally, a "root" member assembles the final product. An interesting issue is the optimal number of levels in such a hierarchy. Initial results are reported in Motro, D'Atri and Gafni (2006).

Tracking performance over time. The instrumentation of VirtuE (both the performance indicators and the constitutional rules) only monitors current inventories and production plans, and cannot track the performance of a virtual enterprise over time. For example, it does not monitor the execution of transactions, changes in inventories, shifts in demand, and so on. Such tracking would permit new performance indicators, such as *overall profit*, and new constitutional rules, such as rules that set limits on certain kinds of transactions.

Acknowledgement This work was performed within *Interop: Interoperability Research for Networked Enterprises Applications and Software*, European Network of Excellence IST-508011 (http://interop-noe.org/).

References

Afsarmanesh, H., & Camarinha-Matos, L. M. (2000). Future smart-organizations: a virtual tourism enterprise. In *WISE 2000, Proceedings of the First International Conference on Web Information Systems Engineering, Volume I (Main Program)*(pp. 456–461). Washington, D.C.:IEEE Computer Society.

Barbini, F. M. & D'Atri, A. (2005). How innovative are virtual enterprises? In *Proceedings of ECIS 05, The 13th European Conference on Information Systems*. Massachussetts: Kluwer.

Brusco, S. (1992). The idea of industrial districts: its genesis. In F. Pyke, G. Becattini et al. (Eds.), *Industrial Districts and Inter-firm cooperation in Italy*, International Institute of Labour Studies (pp. 10–19). Massachussetts: Kluwer.

Camarinha-Matos, L. M. (2003). New collaborative organizations and their research needs. In L. M. Camarinha-Matos & H. Afsarmanesh, (Eds.), *Processes and Foundations for Virtual Organizations, Proceedings of the Fourth Working Conference on Virtual Enterprises* (pp. 3–14). Massachussets: Kluwer.

Camarinha-Matos, L. M., & Afsarmanesh, H. (2004). Elements of base VE infrastructure. *Journal of Computers in Industry*, *51*(2), 139–163.

D'Atri, A., & Motro, A. (2002). VirtuE: virtual enterprises for information markets. In *Proceedings of ECIS 02, 10th European Conference on Information Systems; Research Track: Digital Economy-Models for e-Business and m-Business* (pp. 768–777). Massachussetts: Kluwer.

D'Atri, A., & Pauselli, E. (2004). Virtual enterprises to develop learning environments in an e-marketplace. In *Proceedings of The IASTED International Conference on Web-based Education*, (pp. 200–205). Massachussetts: Kluwer.

Davidow, W. H., & Malone, M. S. (1992). *The Virtual corporation: structuring and revitalizing the corporation for the 21st century*. New York: Harper Collins.

Erben, K., & Gersten, K. (1997). Cooperation networks towards virtual enterprises. *Virtual-Organization.Net Newsletter*, *1*(5), 12–22.

Frakes, W. B., & Baeza-Yates, R. (1992). *Information retrieval: data structures and algorithms*. Englewood Cliffs, New Jersey: Prentice Hall.

Gal, A., & Montesi, D. (1999). Inter-enterprise workflow management systems. In *Proceedings of the 10th International Workshop on Database and Expert Systems Applications* (pp. 623–627). Washington, D.C.: IEEE Computer Society.

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

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

Godart, C., Perrin, O., & Skaf, H. (1999). Coo: a workflow operator to improve cooperation modeling in virtual processes. In *Proceedings of the Ninth International Workshop on Research Issues on Data Engineering* (pp. 126–131). Washington. D.C.: IEEE Computer Society.

Goldman, S. L., Nagel, R. N., & Preiss, K. (1995). *Agile competitors and virtual organizations: strategies for enriching the customer*. New York: Van Nostrand.

Grefen, P., & Hoffner, Y. (1999). Crossflow: cross-organizational workflow support for virtual organizations. In *Proceedings of the Ninth International Workshop on Research Issues on Data Engineering* (pp. 90–91). Washington, D.C.: IEEE Computer Society.

Grover, V., & Teng, T. C. (2001). E-commerce and the information market. *Commun. ACM*, *44*(4), 79–86.

Heimbigner, D., & McLeod, D. (1985). A federated architecture for information management. *ACM Trans. Off. Inf. Sys.*, *3*(3), 253–278.

Kashyap, V., & Sheth, A. P. (1994). Semantics-based information brokering. In *Proceedings of the Third International Conference on Information and Knowledge Management* (pp. 363–370). New York: ACM.

Laufmann, S. (1994). The information marketplace: The challenge of information commerce. In M. L. Brodie, M. Jarke & M. P. Papazoglou, (Eds.), *Proceedings of the Second International Conference on Cooperative Information Systems* (pp. 147–157) Ontario, Canada.

Lee, D., & Chu, W. W. (2000). Comparative analysis of six XML schema languages. *ACM SIGMOD Record*, *29*(3), 76–87.

Monge, P., & DeSanctis, G. (Eds.), (1999). Special issue on virtual organizations. *Organ. Sci.*, *10*(6), 693–703.

Motro, A. (1999). Multiplex: a formal model for multidatabases and its implementation. In *Proceedings of NGITS 99, Fourth International Workshop on Next Generation Information Technologies and Systems*, Lecture Notes in Computer Science No. 1649 (pp. 138–158). Berlin Heidelberg New York:Springer.

Motro, A., D'Atri, A., & Gafni, E. (2006). How Deep Should It Be? On the Optimality of Hierarchical Architectures. In *Proceedings of NGITS 06, Sixth International Conference on Next Generation Information Technology and Systems*, Lecture Notes in Computer Science No. 4032 (pp. 260–273). Berlin Heidelberg New York: Springer.

Mowshowitz, A. (Ed), (1997). Special section on virtual organizations. *Commun. ACM*, *40*(9), 30–64.

Prabhakar, S., Huang, J., Richardson, J., Srivastava, J., Ee-Peng, L., Hwang, S.-Y., Navathe, S. B., Savasere, A., & Foresti, M. (1993). Federated autonomous databases: project overview. In H.-J. Schek, A. P. Sheth & B. D. Czejdo, (Eds.), *Proceedings of the Third International Workshop on Research Issues on Data Engineering: Interoperability in Multidatabase Systems*, (pp. 216–219). Minneapolis, Minnesota: Honeywell Inc.

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

Sheth, A. P., & Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous and autonomous databases. *Comput. Surv.*, *22*(3), 183–236.

Suter, B. (1998). A cooperation platform for virtual enterprises. In P. Sieber & J. Griese (Eds.), *Proceedings of the VoNet Workshop on Organizational Virtualness* (pp. 155–164). Massachussetts: Kluwer.

Tagg, R. (2001). Workflow in different styles of virtual enterprise. In *Proceedings of ITVE 01, Workshop on Information Technology for Virtual Enterprises* (pp. 21–30). Washington, D.C.: IEEE Computer Society.

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

Weiderhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer Society 25*(3), 38–49.