

# Virtual Enterprise Transactions: A Cost Model

A. D'Atri<sup>1</sup> and A. Motro<sup>2</sup>

**Abstract** A transaction is a bilateral exchange between two parties in which goods are delivered in return for payment. In virtual enterprise environments, transactions are the principal means of collaboration among the enterprise members, as well as the mechanism with which the enterprise provides its products to its external clients. In this paper we examine the concept of transactions in virtual enterprises. We define transactions as recursive processes (similar to supply chains), and we formalize the concept of transaction cost. We then examine transaction failure and transaction risk, which lead us to the concept of transaction expected loss. The overall goal is to allow enterprise members to launch those transactions that minimize their expected losses. We also discuss how initiating redundant transactions can further reduce expected losses.

## Introduction and Background

A transaction is a bilateral exchange between two parties in which goods are delivered in return for payment. In a virtual enterprise environment, transactions are the mechanism with which the enterprise provides products to its external clients. Such transactions are termed *external* transactions. To fulfill an external transaction, the contracted enterprise member may procure necessary products from other members of the enterprise. These exchanges are termed *internal* transactions. The supplier of an internal transaction may, in turn, initiate other internal transactions (termed sub-transactions). Altogether, the execution of an external transaction is a *distributed* effort of a group of enterprise members.

In this paper we examine the concept of transactions in virtual enterprises. We begin by defining transactions as recursive processes (similar to supply chains), and then we formalize the concept of transaction cost. Later on we examine transaction failure and transaction risk, which lead us to the concept of transaction expected loss,

---

<sup>1</sup>Università LUISS Guido Carli, Roma, Italy, datri@luiss.it

<sup>2</sup>George Mason University, VA, USA, ami@gmu.edu

described afterwards. The overall goal is to allow enterprise members to launch those transactions that *minimize their expected losses*. We also discuss how initiating redundant transactions can further reduce expected losses. This effort is within the framework of the VirtuE model for virtual enterprises that we defined in earlier works [3].

The concept of transaction has been discussed extensively in economics and related disciplines (business, banking, etc.); and (with a considerably different interpretation) in computer applications such as database systems [5] or workflow systems [6]. In the area of virtual enterprises [2], the VirtuE model introduced a concept of transaction that combines elements from both distributed database systems and economics. In other words, it used the structures of computer transactions to implement concepts borrowed from economics. The work here continues in this vein, with the introduction of a cost model. This cost model, which borrows concepts from transaction cost theory (for example, search and information cost [7]), enables us to discuss formally concepts such as transaction failure, transaction risk, and transaction expected loss – in virtual enterprise environments.

## Transaction Model

A *transaction* is a bilateral exchange between two parties in which goods are delivered in return of payment. The party initiating the transaction, requesting the goods and providing the payment is the *client*; the party responding to the request, providing the goods and receiving the payment is the *supplier*. Transactions consist of distinct phases, and in this paper we shall assume transactions have three phases: (1) *order* is the request by the client to the supplier that describes the goods needed; (2) *manufacturing* is the phase in which the supplier prepares the goods; and (3) *fulfillment* is the delivery of the goods by the supplier to the client. Thus, orders initiate transactions, and fulfillments conclude them.

In a virtual enterprise environment, transactions are the mechanism in which the enterprise provides products or services to its clients. In the virtual enterprise environment described in VirtuE [1], such transactions are termed *external* transactions. To fulfill an external transaction, the enterprise member may procure necessary products or services from other members of the enterprise. These exchanges are termed *internal* transactions. In an internal transaction both the client and the supplier are members of the enterprise. The supplier of an internal transaction may, in turn, initiate other internal transactions. Altogether, the execution of an external transaction is a *distributed* effort of a group of enterprise members.

As described in VirtuE,<sup>3</sup> when receiving orders (in either internal or external transactions), enterprise members consult two resources to determine how to respond to the transaction. These resources help the member determine *what is needed* to fulfill the order, and *how to obtain it*.

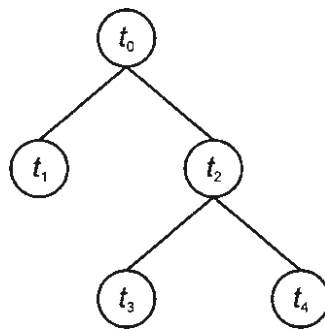
---

<sup>3</sup>VirtuE considers goods that are either products or services. For simplicity, and without loss of generality, we shall assume here goods that are products.

1. *What is needed: production plan.* A production plan is a local resource that enumerates the products that the member must procure from other enterprise members to manufacture a given product.<sup>4</sup> Note that a given product may have multiple (alternative) production plans.
2. *How to obtain it: the product catalog.* The product catalog is an enterprise-wide resource in which members advertise the products that they supply. Each record in this catalog describes a product, a supplier of the product, and additional information, notably the price.

Once the member decides on the production plan and selects the suppliers of the necessary component products, it launches a set of transactions to procure these components. When these transactions have been fulfilled, the member can assemble the product and deliver it to the client. A transaction is therefore a recursive process. We assume that the process *terminates* with transactions that procure *primitive* products: products whose manufacturing does not require importation.

Transactions can be illustrated with tree diagrams, in which nodes represent manufacturing of products by suppliers, and edges indicate initiation of sub-transactions. In a transaction tree the root node models the manufacturing of the ultimate product (the product ordered by the external client), internal nodes model the manufacturing of component products, and leaf nodes model the manufacturing of primitive (import-free) products. Each transaction is assumed to have a unique identifier, which is assigned to the manufacturing node. A simple transaction tree is shown Fig. 1. In this example, an external client submits a transaction  $t_0$  to order product  $p_0$  from member  $m_0$ . To fulfill this order,  $m_0$  submits two sub-transactions:  $t_1$  orders product  $p_1$  from member  $m_1$ , and  $t_2$  orders  $p_2$  from  $m_2$ .  $m_1$  fulfills  $t_1$  locally; but  $m_2$  submits two additional sub-transactions:  $t_3$  orders  $p_3$  from  $m_3$ , and  $t_4$  orders  $p_4$  from  $m_4$ . Both  $m_3$  and  $m_4$  fulfill their orders locally.



**Fig. 1** A transaction tree

<sup>4</sup>VirtuE's production plans consider also component parts that are available to this member locally. Without loss of generality, we shall assume here that all component parts are procured externally.

## Transaction Cost

There are different costs associated with the execution of a transaction, and we adopt a simple cost model, in which each transaction has an associated *price*. This is the amount paid by the client to the supplier in return for the product (this price is advertised in the catalog). We assume that  $price(t)$  is the sum of three cost components. Let  $t_1, \dots, t_n$  be the sub-transactions of  $t$ .

1. *Procurement cost*:  $price(t_1), \dots, price(t_n)$ . This is the amount paid by the manufacturer (the client) to each of its suppliers.
2. *Transaction overhead cost*:  $overhead(t_1), \dots, overhead(t_n)$ . This is the cost associated with the execution of each sub-transaction and borne by the client.
3. *Manufacturing cost*:  $manufacture(t)$ . This is the cost of manufacturing the product from its  $n$  components (it incorporates the manufacturer's profit).

Altogether:  $price(t) = manufacture(t) + \sum_{i=1,n} (price(t_i) + overhead(t_i))$ .

The cost of executing transactions is generally split between the client and the supplier. Here, the client is the member executing  $t$ , and the suppliers are the members executing  $t_1, \dots, t_n$ . We assume that the part borne by each supplier  $i$  has been incorporated into  $price(t_i)$ . Similarly, we could have assumed that the part borne by the client has been incorporated into  $manufacture(t)$ . Yet, for reasons that will become apparent later, we prefer to keep this cost separate, as  $overhead(t_i)$ .

Let  $t_0$  denote the root of a transaction initiated by an external client, then  $price(t_0)$  is the amount the external client pays for the ultimate product. As price is always a recursive summation of manufacturing costs and transaction overhead costs, it is sufficient to record the latter two. Hence, we label each node of a transaction tree with the associated manufacturing cost, and we label each edge with the associated overhead cost.

We illustrate costs in Fig. 2, which shows the example of Fig. 1 with example costs. In this figure, node labels denote the transaction identifiers and the manufac-

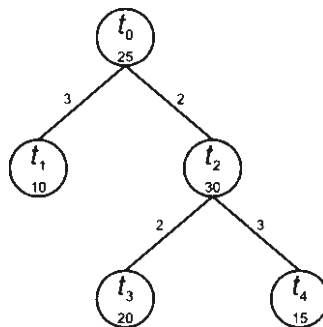


Fig. 2 A transaction tree with its associated costs

turing costs, and edge labels denote the transaction overhead costs. The derived transaction prices are then:  $price(t_1)=10$ ,  $price(t_3)=20$ ,  $price(t_4)=15$ ,  $price(t_2)=70$ , and  $price(t_0)=110$ .

## Transaction Failure

Until now we assumed that all orders are fulfilled, and transactions complete successfully. We consider now the possibility of transaction failure. A transaction fails when an order that has been placed is not fulfilled. Failure could be due to a variety of different reasons: a communication failure, sudden withdrawal from the enterprise, refusal to honor prior commitments, and so on. Failure can be modeled as a disconnection of an edge in the transaction tree.

To analyze the *cost* of transaction failure, we make several assumptions. First, payment is part of fulfillment. Second, all the sub-transactions of a given transaction are placed simultaneously, and it is not possible to *cancel* orders that have already been placed. Finally, transactions cannot be *reversed*; i.e., goods may not be returned for a refund.<sup>5</sup>

Under these assumptions, the client of a failed transaction does not pay for the cost of the product (as fulfillment never took place), and it bears only the cost of the transaction overhead (as this cost was already invested). The client, however, still pays for all the other component products that have been delivered, and for the cost of manufacturing (although manufacturing was not completed). Assuming the supplier of the failed transaction completed all its sub-transactions promptly,<sup>6</sup> it bears the complete cost of the product it committed to produce.

Consider the transaction tree in Fig. 3. For simplicity, assume that all manufacturing costs are 5, and all transaction overhead costs are 1. The overall price is then 107. Assume now two failures: in  $t_7$  and in  $t_{11}$ . The former failure propagates to  $t_2$ , and the latter propagates to  $t_5$  and  $t_1$ , and consequently  $t_0$  fails. The external client does not pay the virtual enterprise price of 107, and this loss is distributed as follows:  $t_{11}$ : 5,  $t_5$ : 12,  $t_1$ : 12,  $t_7$ : 17,  $t_2$ : 24, and  $t_0$ : 37. Thus, only members on a failure path bear the price of failure ( $t_0$ ,  $t_1$ ,  $t_5$  and  $t_{11}$  are on the path of the first failure, and  $t_0$ ,  $t_2$  and  $t_7$  are on the path of the second failure). Essentially, the price they pay is their overhead for the sub-transactions they launched, and the price of products that were delivered but proved unnecessary because the product for which they were intended was not manufactured.

Because the price of the ultimate product is the sum of all the node costs and all the edge costs, and our distribution of the costs assigned each of these costs to exactly one member, *the losses borne by the participating members add up to the price of*

<sup>5</sup>In VirtuE, these assumptions can be expressed by means of *constitutional rules*.

<sup>6</sup>If not, then there had been an earlier failure.

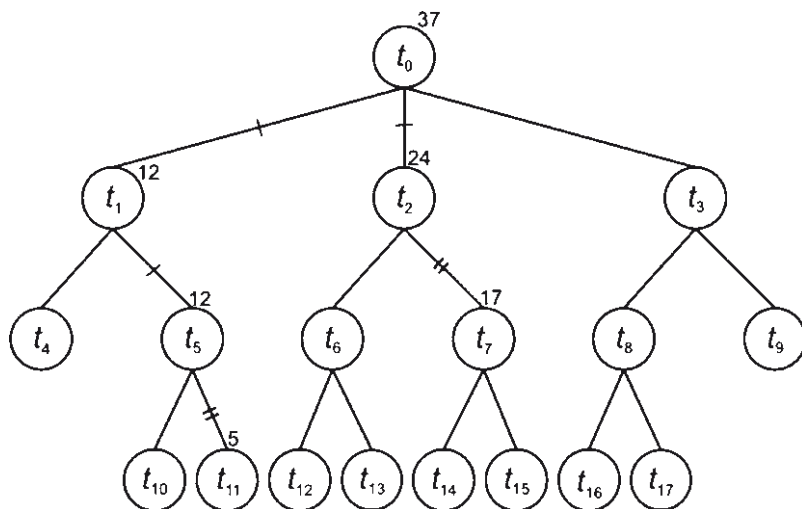


Fig. 3 The distribution of failure costs

the ultimate product. An obvious conclusion is that because the price of products increases along the supply chain, “high level” suppliers tend to pay a higher share of the loss. Although this higher risk would normally be mitigated by higher profit margins, it is important to minimize the risk of failure. This issue is discussed next.

### Transaction Risk

We define *transaction risk* as the probability that a transaction fails; that is, the probability that a supplier will not fulfill an order for a particular product.  $risk(t)$  denotes the risk in transaction  $t$ . We assume that  $risk(t)$  combines two different components: (1) *Supplier risk* measures the risk associated with a particular supplier  $m$ ; it is the proneness of the supplier to fail. (2) *Product risk* measures the risk associated with the manufacturing of the particular product  $p$ . We interpret this risk as the probability that at least one of the transactions to procure components for  $p$  fails. With this definition, product risk increases with the complexity of the product (its number of components): If a product is changed to require an *additional* component, product risk will increase. Altogether, the risk of a transaction  $t$ , in which supplier  $m$  is requested to manufacture a product  $p$ , is the probability that  $m$  fails, or at least one of the sub-transactions to procure components for  $p$  fails.

For a transaction  $t$  that orders a primitive (import-free) product from supplier  $m$ , the risk is simply that of supplier failure. It is convenient to view supplier failure as the probability of *node failure*, and transaction failure as the probability of *edge failure*.

In general, it cannot be assumed that the  $n + 1$  events that define  $risk(t)$  (the failure of  $m$  or of one of the  $n$  sub-transactions it issues) are *mutually exclusive*; i.e., it

may not be assumed that there is at most one failure. In such cases,  $risk(t)$  must be calculated according to De Moivre’s *inclusion-exclusion principle* [4]. In practice, however, unless  $n$  is small, it is normally impossible to calculate  $risk(t)$  in this way, and one must settle for lower and upper bounds, such as those suggested by the Bonferroni inequalities [1]. If we assume that the  $n + 1$  events are *independent* (i.e., the failure of a node and the failure of each incoming edge are unrelated), then  $risk(t)$  may be calculated from supplier risk values only. That is, risk values may be propagated from the leaf nodes to the root of the transaction tree.

As an example, consider the previous transaction tree and assume that the risks of member failure are 0.01 for primitive manufacturers, and 0.03 for the others.<sup>7</sup> Then, in the three phases of propagation (shown in Fig. 4), we derive that the risk of the external transaction is  $risk(t_0) = 0.29197$ . As the example demonstrates, the risk increases with the number of primitive products (in this example, 10), and the number of intermediate suppliers (in this example, 8).

Thus, to calculate transaction risk, our model requires the complete transaction tree and the supplier risk values for each participating member. We may assume that these risk values are calculated by the enterprise (and updated after each completed or aborted transaction), and are available to its members.<sup>8</sup> In terms common in the Internet today, supplier risk corresponds to the supplier’s “reliability rating”.

A feature of the transaction model introduced earlier is the possibility to procure a product from different suppliers. This suggests initiating the sub-transaction with the lowest risk. We showed how transaction risk can be propagated upwards the transaction tree, but it cannot be assumed that a client knows in advance the subtree that will be associated with each of its orders (information necessary to calculate

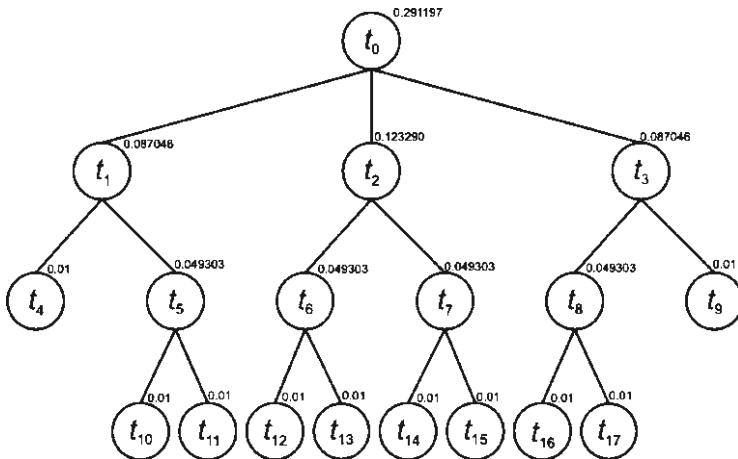


Fig. 4 The propagation of risk

<sup>7</sup>In general, it should not be assumed that manufacturers are divided into these two types.

<sup>8</sup>VirtuE handles these values as *member-specific performance indicators*.

risk). This could be addressed by assuming that each supplier associates with each its products an *estimate* of the transaction risk.

A second flexibility built into our transaction model is the possibility of alternative production plans for a given product (different production plans require different sets of components). The availability of risk values for each sub-transaction will also allow clients to estimate the risks of alternative production plans.

Altogether, we sketched how enterprise members can apply knowledge of transaction risk to either (1) choose among alternative suppliers (for a product they need), or, more ambitiously, (2) choose among alternative production plans (to fulfill an order they received). Yet, choosing the option with the lowest risk may not always be the best choice, as the least-risky transaction may have a much higher price. Considering both price and risk is discussed next.

## Expected Loss and Redundant Ordering

Assume member  $m$  initiates a transaction for product  $p$  for price  $c$  with risk  $r$ . In the previous section we discussed how  $m$  could optimize its operation to minimize the risk  $r$ . Similarly,  $m$  could optimize its operation to minimize the cost  $c$ . Yet, both these approaches may be unattractive: Minimizing risk could come at high cost, and minimizing cost could come at high risk. A more balanced approach is to minimize transaction *expected loss*.

Recall that when a transaction initiated by  $m$  fails,  $m$  bears only the cost of its overhead; yet  $m$  is obligated to pay the cost of the other transactions that were issued in parallel for the same manufacturing task, and completed successfully, and for the cost of manufacturing. Denoting these costs  $l$ , the transaction expected loss is  $r \cdot l$ .

For example, consider a manufacturing task for which  $m$  needs two parts,  $p_1$  and  $p_2$ , and each part has two alternative sources:  $p_1$  can be obtained for cost 10 with risk 0.4 or for cost 20 with risk 0.2, and  $p_2$  can be obtained for cost 30 with risk 0.3 or for cost 60 with risk 0.1. For simplicity, we ignore transaction overhead and the cost of manufacturing. Assume  $m$  chooses the first option for each part. There are four distinct possibilities: The first transaction succeeds and the second fails – the probability is  $(1-0.4) \cdot 0.3=0.18$ , and the loss is 10. The first transaction fails and the second succeeds – the probability is  $0.4 \cdot (1-0.3)=0.28$ , and the loss is 30. There is no loss to  $m$  if both transactions succeed (probability  $(1-0.4) \cdot (1-0.3)=0.42$ ), or if both fail (probability  $0.4 \cdot 0.3=0.12$ ). Hence, the expected loss of this choice of options is  $0.18 \cdot 10+0.28 \cdot 30=10.2$ . Similarly, the expected loss for choosing the first option for  $p_1$  and the second option for  $p_2$  is 22.2, the expected loss of choosing the second option for  $p_1$  and the first option for  $p_2$  is 9, and the expected loss for choosing the second option for each part is 12.4. This suggests preferring high cost with low risk for  $p_1$ , and low cost with high risk for  $p_2$ .

To minimize expected losses, members could find advantage in placing *redundant orders*. That is,  $m$  could procure a product  $p$  from two members: from  $m_1$  with cost  $c_1$  at risk  $r_1$ , and from  $m_2$  with cost  $c_2$  at risk  $r_2$ . The cost increases to  $c_1+c_2$  but

(assuming independence) the risk is reduced to  $r_1 \cdot r_2$ . Expected loss may be reduced as well.

A similar approach could be followed for complete production plans. We can calculate the expected loss of an entire production plan. Members could then choose the production plan that minimizes their expected losses, and they could consider redundant ordering that will reduce their expected losses even further.

## Conclusion

We described a simple yet powerful cost model for transactions in virtual enterprises.

*Simplicity* is maintained with several assumptions, including: (1) Transactions involve only three phases: ordering, manufacturing and fulfillment; (2) all sub-transactions are placed simultaneously (e.g., it is not possible to choose the second sub-transaction after the first has been completed); (3) orders may not be cancelled or reversed (no refunds); (4) transaction cost is the sum of external procurement, transaction overhead, and local manufacturing; (5) transaction risk is the combination of manufacturer risk and product risk, where the latter is the risk in procuring the product components; and (6) the failure of a node (a manufacturer) and any of its incoming edges (the sub-transactions it issued) are independent events, allowing for simple calculation of risk based on probabilities of individual events only.

The *power* of the model is in its ability to represent appropriately many real-world situations and perform several optimizations, including: (1) The freedom to order a part from different sources, and to assemble a part according to different schemes; (2) the just distribution of the cost of a failed transaction among the suppliers on the failure path, with higher costs being borne by suppliers higher on the supply chain; (3) the positive correlation between product complexity and product risk; (4) the propagation of product risks upward the supply chain; (5) the optimization of individual member operations to reduce costs, risks, or expected losses; and (6) the ability to place redundant orders to further improve any of these variables.

Much work remains to be done, with future work to be focused on: (1) Relaxing some the abovementioned model restrictions; (2) allowing members to maintain stock, thus allowing “economics of scale” (for example, transaction overhead would be paid less frequently); and (3) devising efficient algorithms for optimizing members’ operations (in particular, how to deploy redundant ordering efficiently).

## References

1. C.E. Bonferroni. Teoria statistica delle classi e calcolo delle probabilità. *Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8: 1936, 1–62.
2. L.M. Camarinha-Matos and H. Afsarmanesh. The Virtual Enterprise Concept. In *Proceedings of the IFIP TC5 WG5.3 / PRODNET Working Conference on Infrastructures for Virtual Enterprises: Networking Industrial Enterprises*, IFIP Conference Proceedings, Vol. 153, 1999, pages 3–14.

3. A. D'Atri and A. Motro. VirtuE: a Formal Model of Virtual Enterprises for Information Markets. *International Journal of Intelligent Systems*, Vol. 30, No. 1, February 2008, pages 33–53.
4. A. De Moivre. *Doctrine of Chances – A Method for Calculating the Probabilities of Events in Plays*. Pearson, London, 1718.
5. A.K. Elmagamid. *Database Transaction Models for Advanced Applications*. M. Kaufmann Publishers, San Mateo, California, 1992.
6. P. Grefen. Transactional Workflows or Workflow Transactions? In *Proceedings of DEXA 02, 13th International Conference on Database and Expert Systems Applications*, Lecture Notes in Computer Science, Vol. 2453, Springer, 2002, Pages 327–349.
7. G.E. Smith, M.P. Venkatraman , R.R. Dholakia. Diagnosing the Search Cost Effect: Waiting Time and the Moderating Impact of Prior Category Knowledge. *Journal of Economic Psychology*, Vol. 20, 1999, pages 285–314.