Implementation of Large Catalogs for Price Enforcement in B2B E-Commerce

Trung T. Pham
DIcentral Corporation
tpham@DIcentral.com

Simone K. Füchter
Universidade do sul do Brasil
simonekf@terra.com.br

Abstract

This paper presents an efficient architecture to implement a catalog for its application in B2B electronic commerce. The architecture is designed based on a client-server model under the constraint of the unbalanced computing power between the buyer and suppliers. Computational algorithms are designed according to this assumption of limited computer resources at the client side and ample resources at the server side. The application is to provide a complete and workable catalog solution for price enforcement when fulfilling a purchase order to eliminate costs associated with having to make amendments to a past purchase order and having to delay a business transaction.

1. Introduction

A catalog [1-3] is a listing of products with detail descriptions, unique identification codes, classification codes, prices, and physical dimensions. Traditionally, a catalog is used as a static sales tactic. It was published on paper in the form of a thick book to be mailed massively to potential consumers. These potential consumers would browse the content of a catalog, making a purchase decision based on either his personal need or on his personal impulse. The purchasing was conducted through telephone or regular postal mail and the products were shipped directly to the buyer's home.

The rapid adoption of electronic commerce has helped extending the traditional catalog setting into an electronic catalog service [4-6] for business-to-consumers (B2C) electronic commerce environment. In this environment, a catalog is listed on a website allowing potential customers to browse through the listing of products and select what to buy. As the result, a B2C catalog normally retains all the visual appeals of a traditional paper catalog with the objective of luring consumers into making a purchase decision mostly on impulse.

In a business-to-business (B2B) electronic commerce environment, the purchasing behavior is different from that in a B2C environment [7-10]. The major difference in the purchasing behavior is that the pricing scheme is normally negotiated and approved before the actual buying takes place with a specific supplier. Once a pricing is approved, the buying decision is made based on the need of an organization. Therefore, the application of a catalog in a B2B electronic commerce environment is strictly to locate a listed item to retrieve appropriate data to fill out a purchasing order so that a supplier can prepare the invoice, shipment notice, and actual shipment.

One major impediment of the B2B catalog application is the lack of data maintenance [11-12]. This problem often leads to incorrect data on a purchase order, causing the delay in completing a business transaction. There are different incorrect data scenarios, but the most severe one that often stops a purchase order immediately is the incorrect price. The supplier certainly does not want to accept a loss if the submitted price is lower (than the actual price) or a potential penalty such as being removed from the approved vendor list if it was discovered, regardless of whose fault, that they actually charge the buyer a higher price. Furthermore, not fulfilling a purchase order often causes a tangible labor cost of redoing a purchase order as well as an intangible opportunity cost of delaying the whole business process.

It is therefore important to implement an effective catalog service on both the buyer and supplier sides. On the buyer side, a catalog must be maintained with up-to-date data. On the supplier side, an up-to-date catalog must be used to verify the pricing on all purchase orders to ensure efficiency in a business transaction. Normally, the buyer is a large organization with vast computing resources and the supplier is a small organization with limited computing resources. This asymmetric scenario makes the process of efficiently using a catalog an interesting challenge in a B2B electronic commerce.

This paper presents an efficient architecture to implement a catalog service in a client-server architecture. On the server side, business rules are imposed to control the pre-negotiated terms and conditions. On the client side, efficient price checking algorithm is used to detect incorrect data for correction and to trigger automatic update of the catalog on the server side. The system is designed with the objective of minimizing human intervention as much as possible. The intended result is to eliminate costly incorrect data on a purchase order.

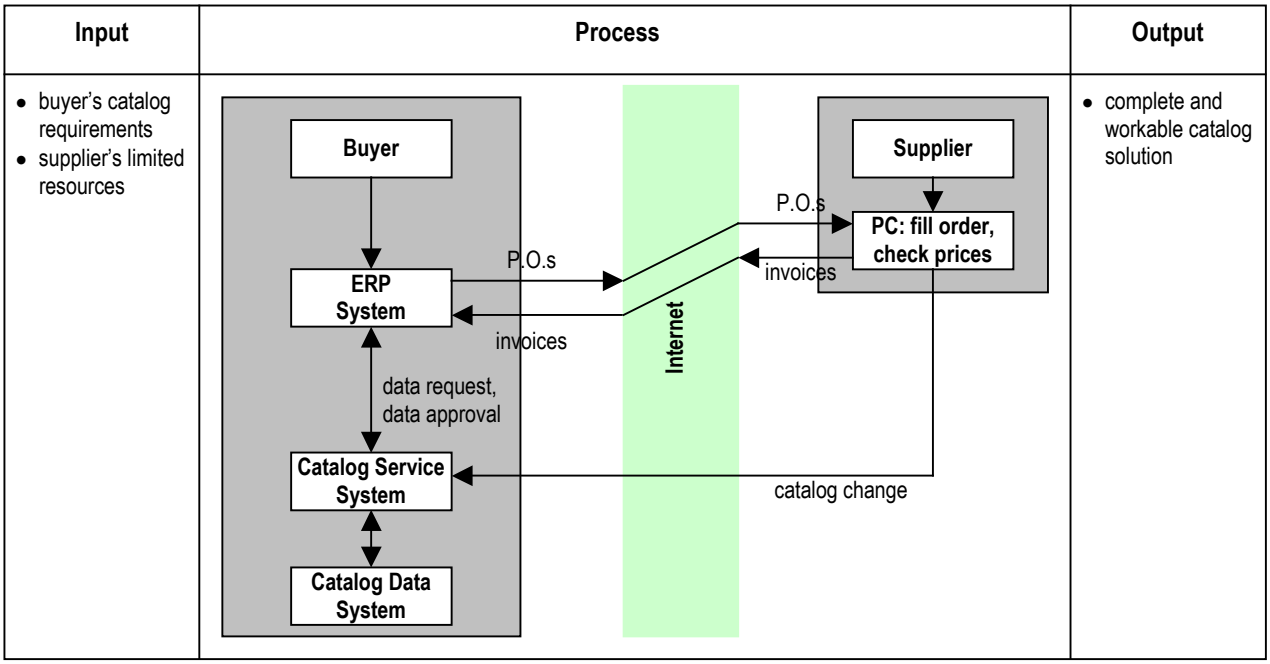

Figure 1. A catalog service can be implemented in a client-server architecture.

2. B2B Electronic Catalog Service

A B2B electronic catalog service is the management of several catalogs in a way that up-to-date data are maintained to efficiently provide accurate information for a buyer to fill out a purchase order. In this setting, a buyer has access to a set of catalogs, each submitted by a supplier, to identify the products it needs, retrieve appropriate information on these products, and fill out a purchase order to send to a supplier. The supplier, upon receiving a purchase order, will perform a data validation to ensure that the products listed on a purchase order are validly identified and the prices correctly listed before checking its inventory system for availability.

The electronic catalog service can be visualized as a shared platform for the supplier to maintain the catalog data and the buyer to identify the product and retrieve appropriate information (Figure 1). Due to the nature of an asymmetric B2B environment [13-14], a single buyer is normally connected to a large number of suppliers, each having a separate catalog listing its products. This single buyer has tremendous influence over the suppliers to dictate the requirements and implementation procedure to fit into its infrastructure [15]. In this setting, a single buyer would see a virtual marketplace connecting to its enterprise resource planning system. Whenever a purchase order is prepared, appropriate data will be retrieved from these catalogs and used to specify the items listing (in the purchase order).

2.1. B2B E-Catalog Functional Requirements

A B2B electronic catalog service must provide three basic functionalities: (i) price checking for the supplier, (ii) product identification for the buyer, and (iii) data synchronization between suppliers and buyer.

A supplier must be able to quickly retrieve pricing information based on some product identification code. The code, normally a sequence of alpha-numeric characters, must be uniquely defined (or assigned) to each product. Considering the limited computing power of suppliers, this data retrieval must be simple and relatively efficient. Furthermore, the procedure must be implemented on a stand-alone software (instead of the traditional database management system with fully efficient search capability).

A buyer must be able to quickly specify what it wants to buy and retrieve a listing of available products from a pool of approved suppliers. Normally, a buyer has an enterprise resource planning (ERP) system that schedules what it wants to buy at certain time intervals to maintain its just-in-time inventory system for manufacturing or reselling. From the ERP system, it is imperative to convert the specifications (or descriptions) of a product into specific product identification codes provided by different suppliers.

A catalog must be kept up-to-date for the buyer to use. The maintenance of the data must be initiated at the supplier side and performed at both the supplier's and buyer's sides. Both sides must agree on the schedule of

data maintenance to avoid having two different versions of data. The schedule of maintenance is the time when a supplier can update the data and send them to the buyer. The buyer must, upon receiving the new data, incorporate these data into its system within some reasonable timeframe. The incorporation process normally includes manual review and approval authorization before the new data become effective.

2.2. B2B Electronic Catalog Contents

A catalog content must contain at least the following information: (i) supplier's product identification code, (ii) product price, (iii) product description, (iii) product physical dimension, (iv) product packing quantity.

Supplier's product identification code is an alpha-numeric number uniquely assigned to a product by a supplier for the purpose of inventory handling and maintenance. A supplier must organize its warehouse with a stock of products in a way that they can be easily retrieved for shipping out to the buyers. The identification code is used for this purpose.

Product description is the details that help human understand what it is to confirm what is listed in the purchase order. This information helps the buyer to minimize cost associating with purchase order errors. The information also helps the supplier to avoid packing the wrong items for shipping to the buyer.

Product physical dimension consists of the measurements of size and weight. This information helps the calculation of shipping cost as well as the handling at the receiving dock.

Product packing quantity specifies how many units are bundled into a single package. This information determines the minimum quantity a buyer must order. It is used to help the buyer planning its just-in-time inventory system.

2.3. B2B Electronic Catalog Data Format

The catalog content described in the previous sub-section must be put into a computer file satisfying specific data format acceptable and usable by the buyer. Traditionally, electronic data interchange follows a standard established by the X12 organization [16]. This standard defines a catalog data format known as the 832 transaction. Recent development in XML introduces similar catalog data format in various business XML standards [17-19].

All standards define the catalog data format in general with three sections: header, body, and trailer. The header consists of the supplier's information such as the supplier's name, address, contact information, etc. The body consists of a repetition structure, with each entry in the repetition structure representing a product item in a catalog. The trailer consists of the summary of the repetition structure in the body.

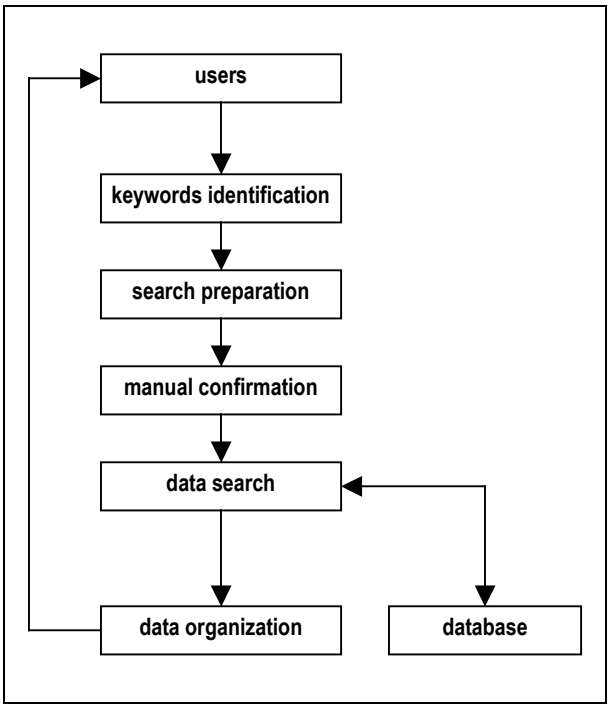

Figure 2. Implementation of Product Identification Procedure.

Depending on the extensive definition of the catalog practice, some standards might provide additional variety of catalog transactions such as catalog change (to adjust only a subsection of a catalog), catalog audit, catalog approval notice, etc. These transactions were designed to accommodate the business practice involving the use of catalog data in a business process specifically setup by individual buying organization. The additional transactions were also designed to minimize data volume involved in the data exchange process.

3. Product Identification Procedure

This section outlines the procedure for a buyer to convert a general description of a product that it wants to buy into some universal product classification code. This universal product classification code will allow an effective data retrieval of a listing of availability from different suppliers. The listing will then provide specific supplier's product identification codes for the purpose of filling out a purchase order to a particular supplier.

3.1. General Implementation

The product identification procedure (Figure 2) consists of the following steps: (i) keywords describing the product, (ii) search preparation, (iii) manual

confirmation (if necessary), (iv) data search, (v) listing preparation.

Step 1 in Figure 2 allows the user to specify the product he wants to buy in some interactive session. The user enters some keywords to describe the product. These keywords are used in Step 2 to search a standardized library of universal product classification code descriptions. Once a match (or a set of matches) can be found, the interactive session continues with the user either confirm that match or narrow down the choice from a listing of matching products (Step 3). The data search in Step 4 will use the universal product code as a guideline to search all available catalogs for entries with that universal product code. Step 5 will organize the results in some order to list the available products according to some customized ranking to help the user selects what to buy and from whom to buy.

It can be seen that the product identification process involves double searching of hierarchal structures. Taking advantage of the knowledge of the hierarchal structures, one can effectively design a search procedure to minimize the searching time involving a large database.

3.2. Universal Product Classification

There are several standards defined for different usages product identification. Most were designed to give a unique product code of each product by a manufacturer, allowing the identification of both a product and its manufacturers. For the purpose of using catalog for e-commerce service, there are a few standards satisfying this requirements, the most popular ones include the United Nations Standard Products and Service Code (UNSPSC) [20-21], the Standard Industrial Classification (SIC) Index, the North America Industry Classification System (NAICS), the National Institute of Government Purchasing (NIGP) Code, etc.

Each of the classification systems assigns some numerical code to define a product and a broader product family that it belongs to. The product family can be structured to some hierarchical structure defining product grouping, manufacturer grouping, industry grouping, etc. There exist several commercial products providing a cross-reference lookup to convert one coding scheme to another.

A catalog should incorporate some product classification to help the buyer identifying the product it wants to buy. This incorporation can either be done by the supplier or the buyer. Most of the time, the buyer leverages its buying power to force the suppliers to implement the coding in their catalogs. Each entry in a catalog will contain additional classification code and code descriptions.

3.3. Search Procedure

The search procedure must be conducted efficiently to cut down the waiting time for the buyer who must monitor and manually fill out a purchase order. This subsection lists an example of a search procedure that satisfies the efficient timing requirement.

At the buyer's site, a database is used to maintain catalog data. Each record represents a unique product by a supplier. The database designer can either set up a single large table containing catalog data from all suppliers or multiple tables, each containing a catalog from a supplier.

Each record in a database will contain a set of product classification codes. Let $P_n = \{ p_{n,1}, p_{n,2}, \ldots, p_{n,M} \}$ be a set of classification codes assigned to a product record in a database. One can normally submit a query retrieving all records with the predefined data fields equal to this data set P_n. This procedure of brute-force query will return the product listing accordingly at some cost of computing time.

For a specific catalog set up with classification codes $P_n = \{ p_{n,1}, p_{n,2}, \ldots, p_{n,M} \}$ for $n = 1, 2, \ldots, N$, a modified multiple-index double search can be used to improve the computing time as follows. The table listing of records is ordered before actual operation. An additional table containing pointers to the table of catalog records is created. This table is significantly smaller than the table of catalog records and will provide a faster search. Once the pointers are identified, a block of data from the table of records is retrieved directly.

The table of records can be ordered according to the codes $P_n = \{ p_{n,1}, p_{n,2}, \ldots, p_{n,M} \}$ using the basic iterative swapping described in the following pseudo-codes:

```
define arrays x dimension N;
loop index i from 2 to N:
{
  loop index j from 1 to i-1:
  {
    if( x[j] > x[i] )
    { // swap x[j] and x[i]:
      z = x[j];
      x[j] = x[i];
      x[i] = z;
    }
  }
}
```

Once the table is order, an index is assigned sequentially for each record. This index is used to construct the pointer table where for each set of codes $P_n = \{ p_{n,1}, p_{n,2}, \ldots, p_{n,M} \}$, a beginning index and an ending index of a block of data containing those codes are listed.

The search of the index table to retrieve the data pointer can be simply conducted with the binary search procedure. In this approach, the index data must also be organized in numerical order with the basic iterative

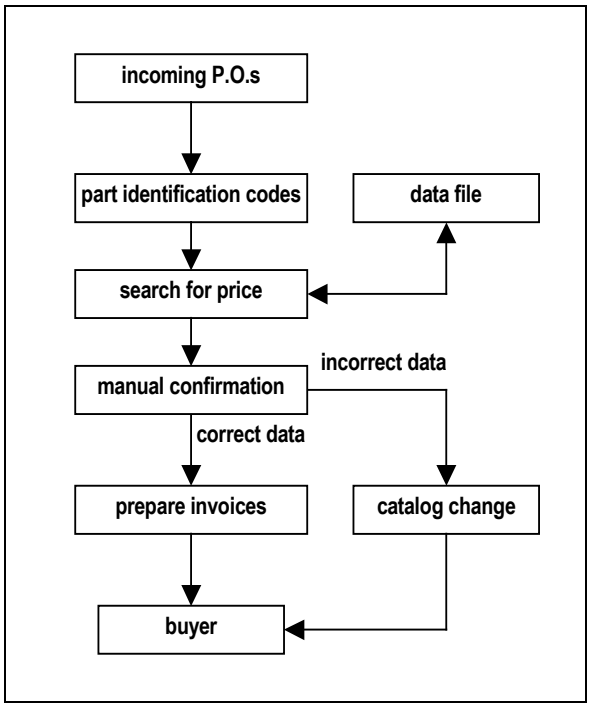

Figure 3. Price Checking Process at the Supplier Side.

swapping presented earlier. The binary search is modified for multiple criteria and described in the following pseudo-codes:

```
define tree structure treeIndex;
load dataIndex with predefined data;
define array code dimension N;
// search index from dataIndex:
define tree structure tmp = treeIndex;
loop index i from 1 to N:
{
  tmp = binarySearch(tmp,code[i]);
}
finalResult = valueOf( tmp );
```

The index table itself can be indexed to further improve the search procedure. This approach is called double index search. The procedure can be repeated with another index table containing pointers to the index table constructed earlier. Then the search for this smaller index table will lead to direct access of the pointers that point directly to the ordered data table for direct retrieval.

4. Price Enforcement Procedure

The supplier, normally with limited computing power, must check the incoming purchasing orders for incorrect data before attempting to fulfill those orders. The most critical data that must be checked is the pricing information. If pricing is incorrect, the supplier normally will have to contact the buyer to request a correction that involves an official cancellation of the incorrect purchase order and a submittal of a new purchase order with correct pricing. Some buyer with sophisticated ERP system can send a modification to an existing order.

The procedure for price checking (illustrated in Figure 3) consists of the following steps: (i) for each line item, retrieve the supplier product identification, (ii) use the supplier product identification to retrieve from a local catalog the associating unit price, (iii) compare the retrieved price with the price used on the purchase order, (iv) if unit price is incorrect, notify the buyer and send a catalog update to the buyer and stop, (v) if unit price is correct, calculate the total price in the line and compare to the total line price on the purchase order, (vi) if total line price is correct, proceed to step (i) for another line until all lines are checked out okay, (vii) if total line is not correct, notify the buyer and stop.

The price checking procedure above will stop at the first mistake. One can easily modify the stopping condition to wait until all line items are checked and if errors are detected, appropriate actions are carried out for all errors instead of for just the first one encountered.

The bottleneck in the above procedure is the retrieval of unit price from an electronic catalog listing at the supplier site. Most of the time, the supplier does not have any database capability and must rely on inexpensive software designed specifically for this purpose. A simple software can be designed for price checking with the following algorithm: (i) load the catalog into the memory using either parallel arrays or array of object structures, (ii) load all the supplier product identifications and their corresponding unit price and total line price into memory arrays, (iii) set up a repetition structure checking one line item at a time performing the basic steps listed in Figure 3. The advantage of this program is to avoid reloading a massive catalog for each query per line item. This advantage is achieved at the cost of using considerable computer memory, which is abundantly inexpensive for personal computer nowadays.

Step (ii) in Figure 3 can be easily carried out with a simple binary search, which requires the data to be ordered according to the supplier product identification codes. The binary search guarantees that the number of search will not exceed the number of search in the worst case scenario of

$$N_{max} = \rho (\log_2(M)),$$

where M is the number of products in a catalog, and the function $\rho(\cdot)$ returns a round up integer of a floating number.

5. Data Synchronization Procedure

Data synchronization is the process of maintaining a catalog at two different sites. This process defines the business practice of a buying organization allowing a

supplier when to update a catalog at the buyer's side. There are three scenarios that cause the update: (i) if pricing data are not current, (ii) if the supplier decides to change catalog pricing data, and (iii) if the supplier provides additionally new products. In the case of data not being current, a catalog modification is often accepted immediately. In the cases of data changing and new products, it is often preferred that the catalog modification is done at some predefined time intervals, e.g., every quarter of the year.

The condition for triggering a catalog modification is a definition of business practice and therefore is logically defined at the business process level. The business rule must be agreed upon by both the buyer and supplier. This rule must be clearly defined for the buyer's ERP system to detect and enforce with minimal human intervention. Most of the time, the buyer must manually go through each catalog change request to review the data and approve it before putting it into an operational database.

In order to perform data synchronization, it is convenient to send to the buyer only affecting data instead of the complete catalog. This synchronization can be done at the business transaction level where a catalog modification transaction must be defined to contain the following basic data: (i) header containing supplier's information, (ii) body containing affected products where each product is listed along with specific instruction for modification, i.e., delete the item, add the new item as new, and modify the existing item, and (iii) trailer containing the summary of the body for parity checking of the transaction. In the case when a buyer allows a supplier to submit a data modification because of incorrect pricing data, additional reference (or reference code) must be defined to distinguish this case to the case of the supplier changing price on its own. This reference code will be listed in the line item of the repetition structure in the body of the transaction.

6. Conclusion

A comprehensive architecture for electronic catalog was defined just for the basic purpose of electronic trading between a buyer and several suppliers. The solution was specifically designed in an efficient manner to minimize the cost demand at the supplier side that only has minimal computing power. This solution will help the buyer to eliminate cost associating with errors on a purchase power as well as the supplier to speed up the process of fulfilling purchase orders.

References

[1] K. Muldoon. *How to profit through catalog marketing*. Lincolnwood, IL: NTC Business Books (1996).

[2] K. Muldoon. *Catalog marketing: the complete guide to profitability in the catalog business*. New York, NY: American Management Association (1988).

[3] J. M. de Bernardi. *The catalog showroom formula*. New York, NY: Chain Store Age Books (1974).

[4] J. Cole, and W. Jones (eds.). *E-serials cataloging: access to continuing and integrating resources via the catalog and the Web*. New York, NY: Haworth Information Press (2002).

[5] A. Birch, P. Gerbert, and D. Schneider. *The age of e-tail : conquering the new world of electronic shopping*. Milford, CT: Capstone (2000).

[6] K. Clay, R. Krishnan, E. Wolff. *Prices and price dispersion on the web : evidence from the online book industry*. Cambridge, MA: National Bureau of Economic Research (2001).

[7] H. Assael. *Consumer behavior and marketing action*. Cincinnati, OH: South-Western College Pub. (1998).

[8] J. Paul Peter, J. C. Olson. *Consumer behavior and marketing strategy*. Chicago, IL: Irwin (1996).

[9] T. V. Bonoma, and G. Zaltman (eds.) *Organizational buying behavior*. Chicago, IL: American Marketing Association (1978).

[10] M. L. Vasu, D. W. Stewart, G. D. Garson. *Organizational behavior and public management*. New York, NY: Marcel Dekker (1998).

[11] J. T. Hackos. Content management for dynamic Web delivery. New York, NY: John Wiley & Sons (2002).

[12] J. Dunham. *Database performance tuning handbook*. New York, NY: McGraw-Hill (1998).

[13] L. D. Fredendall, and E. Hill. *Basics of supply chain management*. Boca Raton, FL: St. Lucie Press (2001).

[14] S. Chopra, and P. Meindl. *Supply chain management: strategy, planning, and operation*. Upper Saddle River, NJ: Prentice Hall (2001).

[15] T. Pham. "Managerial considerations to adopting electronic data interchange." *Proceeding of the IEEE Conference on E-Commerce*, Newport Beach, CA (June 2003).

[16] Data Interchange Standards Association. *Electronic data interchange X12 standards: draft version 4, release 3*. Alexandria, VA: Data Interchange Standards Association (1999).

[17] "PIDX XML standards master." *American Petroleum Institute* (2001).

[18] "cXML user's guide." cXML.org (2003).

[19] "xCBL structure reference." xCBL.org (March 2003).

[20] "Using the UNSPSC: why coding and classifying products is critical to success in electronic commerce." *Granada Research White Paper* (Oct. 2001).

[21] A. Ramakrishnan. "Leveraging the power of UNSPSC for business intelligence." *Satyam White Paper #DWBI006* (no date listed).

[22] NIGP Code Directory. Georgia Department of Administrative Services – State Purchasing Office Website (http://www2.state.ga.us/departments/doas/procure/nigp/nigp_frame.html).

[23] B. Kerns, B. Unbehagen, and F. Simonton. "Aligning the NIGP Code with purchasing card expansion." *NIGP Research Paper* (May 1, 2003).