RC22349 (W0202-067) February 27, 2002
Computer Science

IBM Research Report

Web-based Sell-side Commerce Aggregation

Shiwa S. Fu, Shyh-Kwei Chen, Jih-Shyr Yih, Florian Pinel, Trieu C. Chieu
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Web-based Sell-side Commerce Aggregation

Shiwa S. Fu, Shyh-Kwei Chen, Jih-Shyr Yih, Florian Pinel, and Trieu Chieu

IBM T. J. Watson Research Center

P.O. Box 704, Yorktown Heights, N.Y. 10598

Abstract

Large corporations that sell complete products may consist of many autonomous divisions. To aggregate numerous commerce activities among these divisions and to provide a unified storefront (i.e., a hub) can be convenient for business partners to purchase products. In this paper, we present a 2-tier solution for aggregating sell-side activities. The advantages lie in two folds. For the customer, our solution provides a single entry point allowing easy navigation and one stop shopping. For the large corporation, it provides a centralized control over catalog management and order processing to reduce redundant web investment, and yet allows customers to configure products locally at individual divisions.

Keywords: Electronic Commerce, Commerce Aggregation, Catalog Aggregation, Order Separation, and Internet.

1. Introduction

Large corporations that sell complete products may consist of many autonomous divisions, where each division provides parts for complete products. Efficiently and effectively aggregating numerous commerce activities among these divisions becomes an important issue. Building a unified storefront (i.e., a hub) for all of the divisions can be a convenient approach for business partners to purchase products, since there is only one point of entry. Customers can easily navigate the hub, and locate desired products. They also have options in configuring the products based on different attributes and specifications provided by individual divisions. On the other hand, it is desirable for large corporations to have a centralized control over catalog management and order processing to reduce redundant web investment on individual divisions.

Sell-side commerce aggregation can roughly be divided into three approaches: (1) loosely coupled (e.g., Ariba), where each seller hosts her/his private catalog and order system, while customers invoke punch-out protocol via Ariba Commerce Service Network (ACSN) [1] to reach sellers' catalogs for shopping, (2) tightly coupled, where the entire catalog and order system are aggregated in one centralized hub (e.g., Open Market [2]), and (3) hybrid. The first approach is good for inter-corporation commerce. On one hand, these corporations need a portal (e.g. Ariba) to redirect customers to their sites for shopping. On the other hand, they intend to maintain their private catalog and order systems autonomously. However, a central control point could potentially cause management issues for competitive companies. As for the second approach, it is suitable for small or medium sized companies that have few divisions, but not for large corporations with many divisions. This is because hosting all product information from divisions can easily overload the hub, and may require intensive update and coordination mechanisms to maintain a consistent catalog view when there are changes to the aggregated catalog. It is therefore desirable to pursuit a hybrid approach, where a unified storefront (hub) maintains limited product information from divisions to alleviate the workload and yet interacts closely with divisions that still operate autonomously.

In this paper, we present a 2-tier solution for aggregating sell-side activities. The hub maintains limited yet sufficient product information for customer shopping and transparently retrieves detailed information from divisions on behalf of customers' request. By maintaining sufficient catalog information, the customer can still accomplish shopping on the hub even though the division system is down. In the proposed solution, customers place orders on the hub. As an order may consist of items from different divisions, our order management component includes automatic order separation, so that a single order to the hub can be separated and delivered to the corresponding divisions. Furthermore, product configurations are done at division sites: it is impractical to configure products at the hub, as it may not have enough knowledge about specific products like individual divisions do.

Our solution is different from most of the Web-based e-commerce solutions, such as Ariba Commerce Service Network (ACSN), Amazon.com's Outlet [3], Yahoo Shopping [4], and Dell.com [5]. Ariba's B-to-B solution mainly deals with inter-corporation commerce, whereas our solution focuses on aggregating commerce for a large corporation that has many divisions. Amazon.com's Outlet provides a platform for sellers to join conveniently, but it does not include order separation. Yahoo Shopping is a portal that provides sellers an entry point, but it does not provide configurators. Besides all of Yahoo Shopping's features, our orders are created either online, offline, or by configurator. Dell.com's build-to-order solution basically is a configurator for parts from different companies, while our solution provides full commerce aggregation from divisions in addition to configuration capability. For example, our solution provides off-line order preparation for customers, and off-line catalog preparation for divisions, using popular spreadsheet editors. Additionally, it handles lightweight catalog replication between the hub and divisions automatically.

2. Proposed Solution

We describe the architecture and its major components in Section 2.1, the features of each component in Section 2.2, and catalog punchout in Section 2.3.

2.1. Architecture

The proposed 2-tier architecture, i.e., hub and division, is depicted in Figure 1. It consists of three major components: catalog aggregation/replication subsystem, order subsystem, and remote access subsystem.

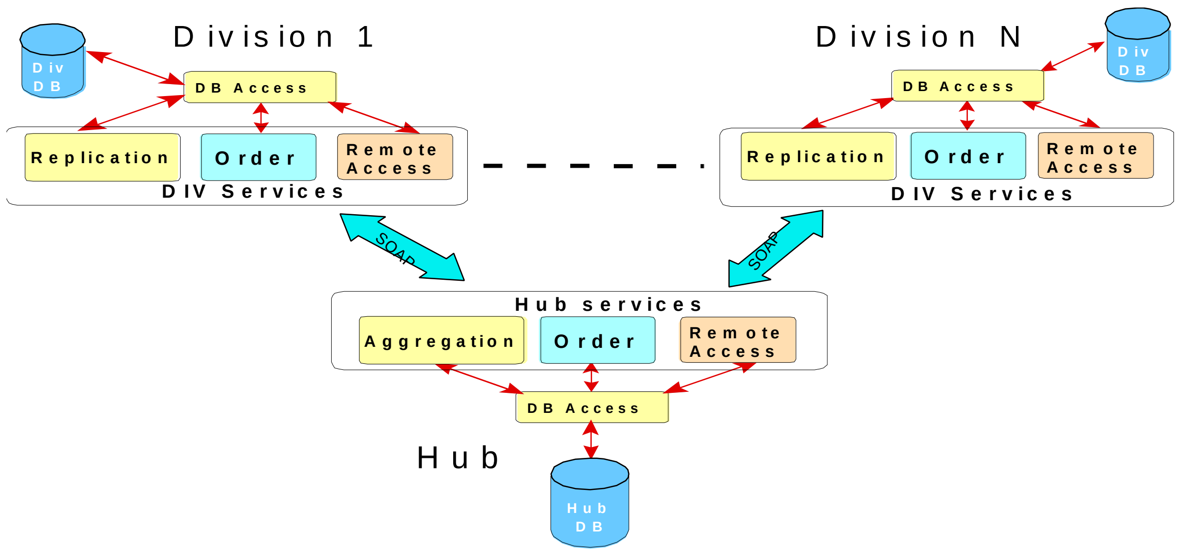

Figure 1: Proposed 2-tier architecture.

The catalog aggregation subsystem handles automatic catalog extraction, replication, aggregation, and product bundling. Its goal is to build a unified storefront by aggregating and integrating products from different divisions into the hub seamlessly. The order subsystem accepts and manages customers' online and offline orders. The hub maintains limited yet sufficient product information. If customers request information that is not located in the hub, the remote access subsystem employs the following three methods to fulfill customers' demands: (1) pure URL redirection to retrieve image or multimedia files, (2) remote program invocation for detailed product or personalized information, and (3) catalog punchout for configurator. Details of these components and their features are described in the following subsection.

2.2. Features

The catalog aggregation subsystem, order subsystem, and remote access subsystem contains many modules to accomplish commerce aggregation as shown in Figures 2 and 3. To provide better flexibility, the hub and division publish all these modules as services in a private UDDI [6, 7].

The hub aggregates products from different divisions. As each division may have a different web-based electronic commerce system, each division most likely has its own data format to present products and catalog. The catalog subsystem in the hub is responsible for resolving and merging division's data format against that of the hub's, integrating divisions' products into the proper category in the hub, and bundling products from different divisions as shown in the catalog subsystem of Figure 2. Before catalog aggregation, the division uploads the DTD of its local catalog and products. The DTDMapper service in Figure 2 maps division's DTD into hub's database and creates an XSL (Extensible Stylesheet Language) [8] mapping file for the Integrator service in the next step (see Figure 2) to handle real data. On the division site, the ReplicationManager service collects any updates on the division catalog and products and periodically sends these updates, in a batch file, in the XML (Extensible Markup Language) [9] format to the Integrator service for uploading, updating, and/or deleting its own division products in the hub. The Integrator service aggregates and integrates products into the hub's database. During the processes of product aggregation, related products, on the fly, are bundled together by the Bundler in Figure 2.

On the customer side, in addition to the online processes, it may be convenient for a customer to prepare orders offline using spreadsheet editors such as Lotus 1-2-3 and Microsoft Excel. The customer uses the Extractor to select and download his/her previously submitted orders or interest list of products in a spreadsheet format; consequently, an order is created based on the interest list or submitted order using spreadsheet editor. The customer then uploads orders to the OrderManager as illustrated in the order subsystem of Figure 2.

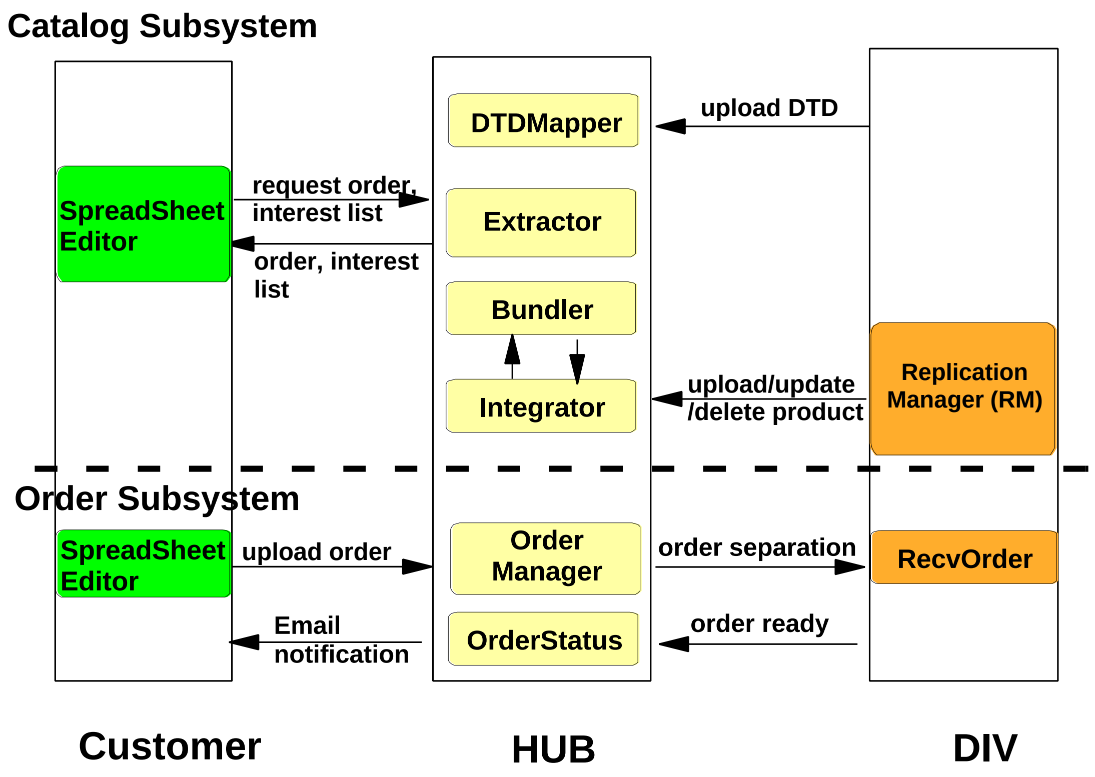

Figure 2: Processes of catalog aggregation and order management.

A customer may buy many items within one purchase order online or offline, and these items may belong to different divisions. Accordingly, the OrderManager service (see Figure 2) in the hub separates the order into many sub-orders and sends them to the corresponding divisions. After receiving a sub-order, the RecvOrder service (see Figure 2) in the division processes and stores it in the local database. After the sub-order could be fulfilled, an order ready message is sent to the OrderStatus service in the hub to update the items status in the parent order. After receiving all order ready messages, the OrderStatus service sends an E-mail notification message to the customer. The customer can always logon to the hub to check the status of individual items in the order.

The remote access subsystem employs three methods to retrieve information located in the division sites: (1) pure URL redirection to retrieve image or multimedia information, (2) remote program invocation via SOAP (Simple Object Access Protocol) [10] for detailed product and personalized information, and (3) catalog punchout for configurator.

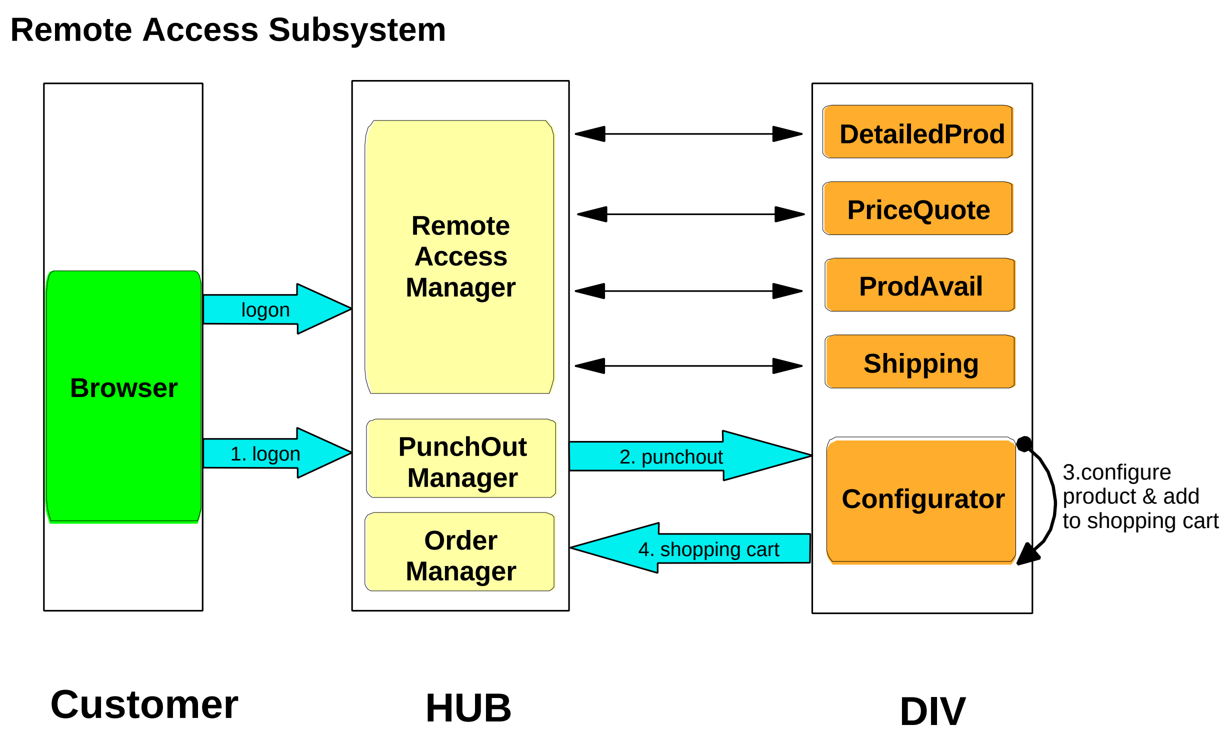

Figure 3: Processes of remote access.

The size of image and multimedia files is usually huge. To avoid overloading the hub, this information is stored and maintained in the divisions. The hub only keeps URL links and uses URL redirection for remote access. Besides image and multimedia files, information such as detailed product description, price quote for a specific customer, product availability, and shipping option are confidential, and their sizes are relatively small. As illustrated in Figure 3, the RemoteAccessManager in the hub invokes the corresponding services (DetailedProd, PriceQuote, ProdAvail, and Shipping) in the division via SOAP to retrieve the desired information. Customers purchase a complete product that may consist of many parts with different configuration options under certain restrictions. A configurator is a tool for customers to compose their own selections. Dell.com's built-to-order tool is a typical example. The division usually hosts the configurator as it has a better knowledge of how to configure a product. Unlike detailed product and personalized information being retrieved in a simple request-response pattern, product configuration typically involves three steps: the customer (1) leaves the hub site and logons to the division site, (2) browses the division local catalog and configures a complete product, and (3) checks out and returns back to the hub. To handle these prolonging processes, we employ catalog punchout as described in the next subsection.

It should be noted that information acquired by the remote access subsystem enhances customer's shopping experience. As the hub maintains sufficient catalog information, customers can still order products (except configurable products) from the hub even though the corresponding division web site is down.

2.3. Catalog Punchout

We describe the Ariba catalog punchout first. It has three major components: Ariba buyer client, Ariba Commerce Service Network (ACSN), and sell-side system. Ariba buyer client contains a browser and the Ariba buyer e-procurement system. Ariba punchout protocol consists of eights steps as shown in Figure 4. To enable a buyer to punch out of his/her local catalog (hosted by The Ariba buyer e-procurement system) to the desired seller's catalog, a seller must first publish its catalog index information to ACSN, and the catalog index must be downloaded into the buyer e-procurement system as illustrated by Step 1 in Figure 4. The catalog index contains, for example, seller ID, seller part ID, and punchout website. When the buyer requests the seller's punchout catalog (Step 2), it triggers ACSN to exchange PunchOutSetupRequest message for PunchOutSetupResponse message with the sell-side system for authentication check (see Step 3). Consequently, the buyer can logon to the sell-side system (Step 4) with the desired seller's catalog displayed on the buyer's Browser. At this end, the buyer is entitled to browse the seller's catalog, selects product items, and adds them to the shopping cart (Step 5). After checking out, a PunchOutOrder message containing the purchased items in the shopping cart is transmitted to the buyer e-procurement system (Step 6). After going through internal approval processes provided by buyer e-procurement system, a purchase order message (OrderRequest) is sent to ACSN and forwarded to the sell-side system (Step 7). Finally, an order response message (OrderResponse) is sent back to the buyer (Step 8) to end the process.

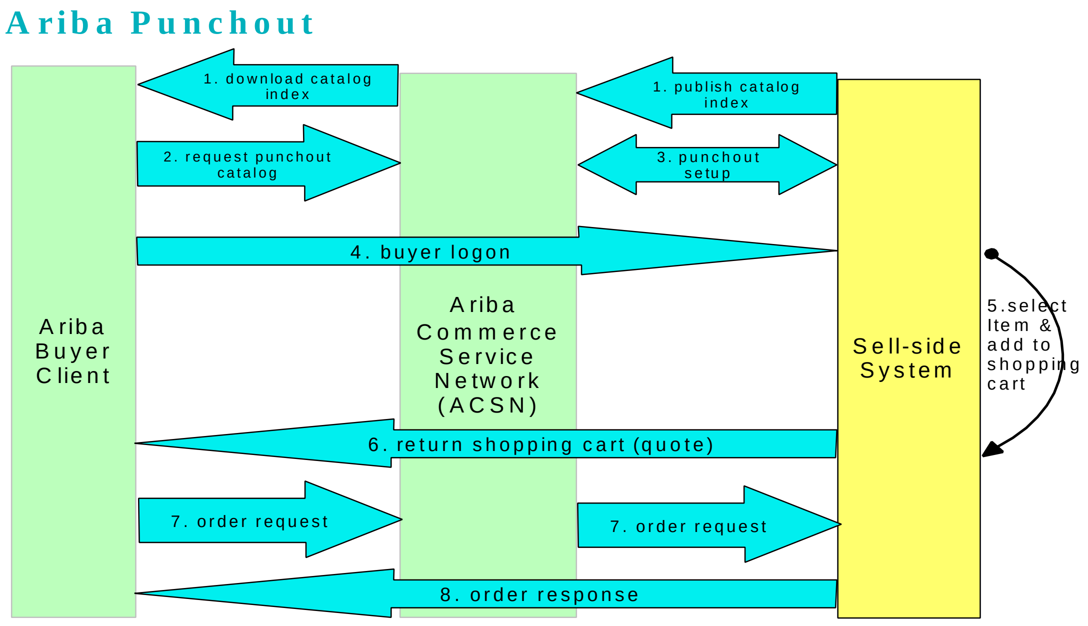
Ariba Punchout
1. download catalog index
1. publish catalog index
2. request punchout catalog
3. punchout setup
4. buyer logon
Ariba Buyer Client
Ariba Commerce Service Network (ACSN)
Sell-side System
5.select Item & add to shopping cart
6. return shopping cart (quote)
7. order request
7. order request
8. order response

Figure 4: Ariba punchout.

The Ariba punchout protocol mainly deals with inter-corporation commerce, where buyer's request for seller's catalog most likely invokes long lasting punchout process. In our proposed 2-tier solution, neither ACSN nor the Ariba buyer e-procurement system is needed, and only the request for configurable products requires catalog punchout as previously illustrated in Figure 3. The punchout links were seeded in the hub's aggregated catalog when the division uploads products requiring configuration. A customer, after logon (see Step 1 in Figure 3), requesting a configurable product punches out to the division site (Step 2 in Figure 3). The proposed solution employs single sign-on [11] to enable the customer logon to the division site transparently. The customer browses the division catalog, configures product items, and adds them to the shopping cart (Step 3). The shopping cart information is collected and transmitted to the OrderManager in the hub (Step 4). On the fly, a purchase order record is created and stored in the hub database.

3. Detailed Flow

We describe the detailed flow of the catalog and order subsystems as follows. The catalog subsystem contains the DTDMapper service to resolve and merge division data format into the hub database, the Integrator service to load and integrate the division products into the proper category in the hub, and the Bundler service to bundle products from different divisions on the fly. To merge products, the division must first upload the DTD of the data format as depicted in Figure 5.

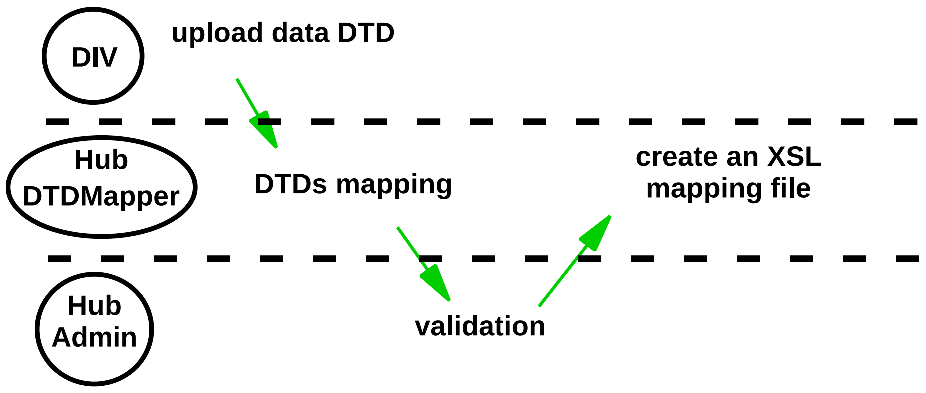

Figure 5: Flow of uploading DTD of data format.

Several works [12, 13] have proposed to map DTD into relational database. These methods treat the content of an XML document as a tree of objects and apply existing object-relational mapping techniques to *create* relational database schema (tables). However, as most E-commerce systems already have database tables to host catalog, products, and their relationships, these methods still need manual mapping processes. We employed DTDMapper service to map the incoming DTD with the existing database schema's DTD. The DTDMapper is a component of IBM WebSphere Catalog Manager [14]. It provides users with a GUI (Graphical User Interface) to compose the mapping. The result of the mapping is an XSL file after hub administrator's approval (see Figure 5). The generated XSL file specifies the rules for transforming the incoming division catalog and products into an XML format that describes the hub database tables.

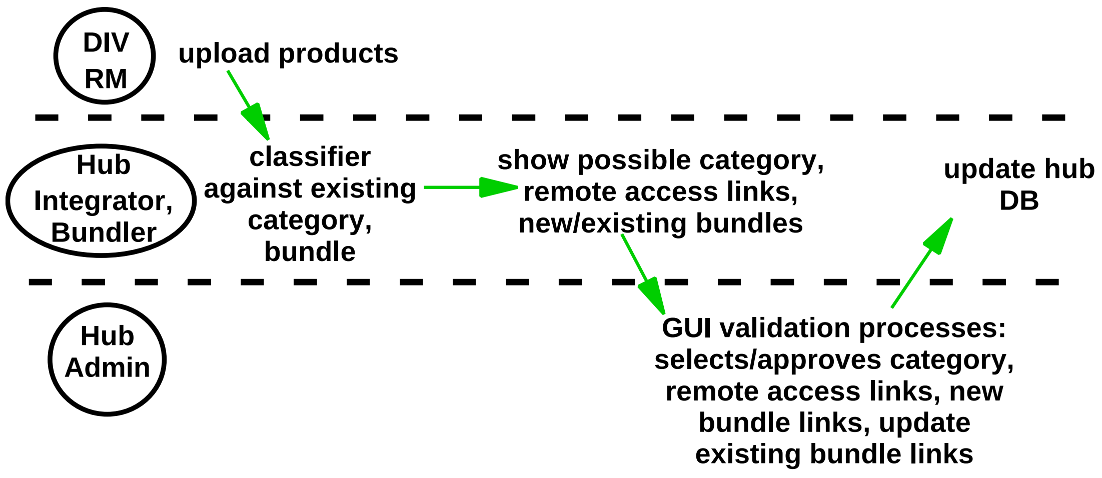
DIV RM
upload products
Hub Integrator, Bundler
classifier against existing category, bundle
show possible category, remote access links, new/existing bundles
update hub DB
Hub Admin
GUI validation processes: selects/approves category, remote access links, new bundle links, update existing bundle links

Figure 6: Flow of uploading product.

The flow of catalog integration and product bundling are shown in Figure 6. The Integrator service uses keywords search to find possible categories to place the uploading products. In addition to catalog integration, the Integrator service calls the Bundler service to find possible bundle and cross-sell relationships among the uploaded products and the products in the hub. The product bundling includes (1) accessory bundling, e.g., certain cartridges bundle with specific printers, (2) compatible bundling, e.g., cartridges of a new brand that are compatible with cartridges of a known brand and known bundling with a printer, (3) peer bundling, e.g., buy printers, list other brands or top sellers of similar products, and (4) cross bundling, e.g., buy printers, list papers. The Integrator service also establishes remote access links for the remote access subsystem to retrieve information from the divisions. The Integrator displays all possible categories that match the uploaded products, possible product bundling, and remote access links for the hub administrator to select and validate as illustrated in Figure 6. Similar flow for updating/deleting products is depicted in Figure 7. The ReplicationManager service is a scheduler. It collects updates to the division catalog and products, and periodically sends updates (delta information) to the hub. It should be noted that updates and deletes might alter existing categories and bundling.

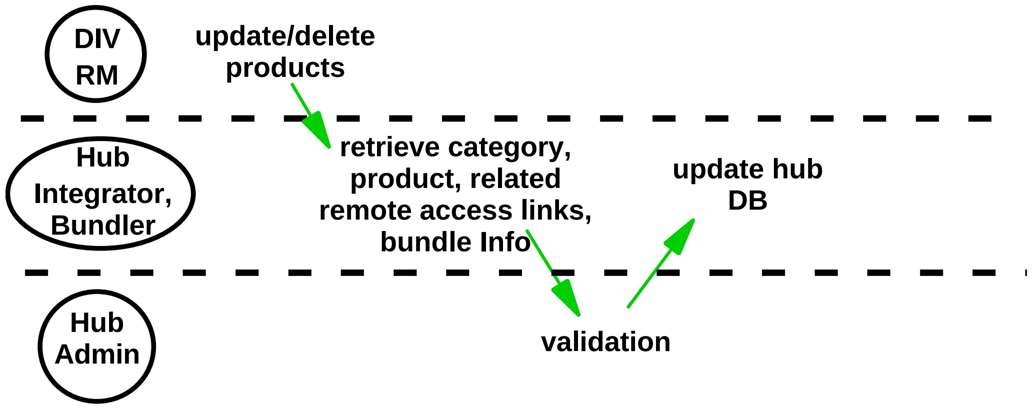

Figure 7: Flow of updating/deleting product.

It may be convenient for customers to prepare an order offline and upload it to the hub as illustrated in Figure 8. The customer first downloads previously submitted orders or an interest list in the spreadsheet format (comma separated values) from the hub, uses a spreadsheet editor to create the order, and uploads it to the hub (see Figure 8(a)). The hub checks with division(s) the price for that specific customer and product availability before creating the final order. As an order may contain items from different divisions, the hub separates the order into sub-orders and sends them to the corresponding divisions. Each division receives and saves the sub-order in its database and replies with an order ready message to the hub once the sub-order can be fulfilled. The hub updates the order status and sends an E-mail notification to the customer when all the sub-orders are ready.

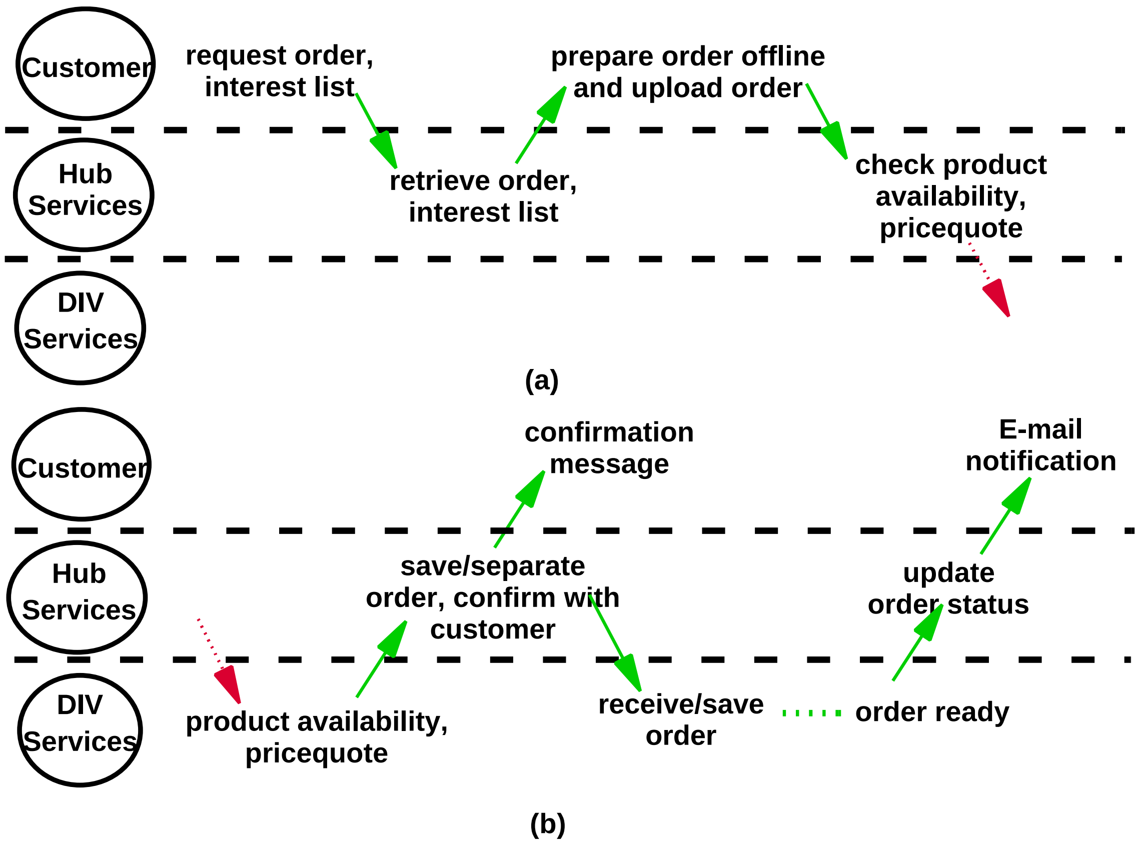
Customer
Hub Services
DIV Services
request order, interest list
retrieve order, interest list
prepare order offline and upload order
check product availability, pricequote
(a)
Customer
Hub Services
DIV Services
product availability, pricequote
save/separate order, confirm with customer
confirmation message
receive/save order
order ready
update order status
E-mail notification
(b)

Figure 8: Flow of order processes.

4. Conclusions

In this paper, we propose a 2-tier solution that aggregates sell-side commerce. The resulting unified storefront helps customers easily navigate the unique consolidated catalog and locate desired products. It also provides large corporations with a centralized control over catalog and order managements to reduce redundant web investment on the individual divisions. A hub storefront was created to aggregate business activities such as detailed product description, price quote, product availability queries, and order submission and separation.

The proposed solution consists of three components: catalog aggregation sub-system, order subsystem, and remote access subsystem. The catalog aggregation subsystem handles automatic catalog extraction, replication, aggregation, and product bundling. The order subsystem accepts and manages customers' online and offline orders. The remote access subsystem combines pure URL redirection, remote program invocation (web services), and catalog punchout to enable interactions between the hub and the divisions.

Acknowledgments

The authors would like to thank David Krieff and Pierre Scemla for their assistance and feedback in developing a prototype of this solution.

References

[1] "cXML User Guide", version 1.2.007, Nov. 2001. http://xml.cxml.org/current/cXMLUsersGuide.pdf.

[2] http://www.openmarket.com.

[3] http://www.amazon.com.

[4] http://shopping.yahoo.com.

[5] http://www.dell.com.

[6] Steve Graham, "The Role of Private UDDI Nodes in Web Services, Part 1: Six Species of UDDI", May 2001. http://www-106.ibm.com/developerworks/webservices/library/ws-rpu1.html.

[7] UDDI Project, "UDDI Technical White Paper", Sept. 2000. http://www.uddi.org.

[8] James Clark, "XSL Transformation (XSLT) 1.0," W3C Recommendation, http://www.w3.org/TR.

[9] T. Bray, et al., "Extensible Markup Language (XML) 1.0 Specification," W3C Recommendation, http://www.w3.org/TR.

[10] E. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer, "Simple Object Access Protocol (SOAP) 1.1", May 2000. http://www.w3.org/TR/SOAP.

[11] A. Volchkov, "Revisiting Single Sign-on: a Pragmatic Approach in a New Context", *IT Professional*, vol. 3, issue 1, pp. 39-45, Jan.-Feb. 2001.

[12] R. Bourret, C. Bornhovd, A. Buchmann, "A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases", *Proceedings of 2nd International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (WECWIS), 2000.*

[13] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, J. Naughton, "*Relational Databases for Querying XML Documents: Limitations and Opportunities*", VLDB, 1999.

[14] "IBM WebSphere Catalog Manager: Workbook and Solutions Guide", version 1.1, Feb 23, 2001. http://www-3.ibm.com/software/webservers/commerce/catalogmanager/downloads.html.