RC 22210 (99179) October 18, 2001 Computer Science

IBM Research Report

Downloadable Service Contracts for Disconnected Transactions

Sidharth Choudhury
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712

Asit Dan
IBM Research Division
T. J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Downloadable Service Contracts for Disconnected Transactions

Sidharth Choudhury
Dept. of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
sidharth@cs.utexas.edu

Asit Dan
IBM T.J. Watson Research Center
Hawthorne, NY 10532
asit@us.ibm.com

Abstract

Client participation in all web-based services, e.g., shopping in online stores, creating service agreements as in buying a car insurance, currently demands browser based online operations with on-the-spot decision making. In contrast, organizational (business-to-business) procurement relies on complex decision making software at each end for comparison shopping and/or satisfying various organization specific goals. Supporting disconnected operations for business-to-consumer scenario can introduce some of the same benefits at the individual consumer level. In this paper, we propose the concept of a downloadable service contract, that specifies unambiguous rules of interaction between an end client and a service provider. The client initially negotiates with the service provider a contract, which is subsequently used not only to guide off-line client operations, but also for validating at the service provider the log of actions, i.e., a purchase order, submitted by the client. A generic application program for browsing a downloaded contract and creating an order, may be made available by the service providers. However, a client may have its own software with many value-added services such as comparison shopping across multiple contracts, matching complex user specific goals and maintenance of interaction histories. We describe a prototype system that extends the current Ariba cXML protocol for MRO procurement.

1 Introduction

End-consumer participation in today's web based services, e.g., buying items from an online storefront, buying a car insurance, etc. requires making many on-the-spot decisions, and/or answering many associated questions. In contrast, a shopper enjoys the comfort of browsing and decision making at his own pace in a paper catalog based buying. First, the catalog based buying provides the consumer an opportunity to consider various personal goals, e.g., meeting a personal budget, prioritization of needs, reconsidering selections and other purchasing decisions, and even recording the decisions by a rare organized shopper. Additionally, it provides the consumer the ability to compare prices and product attributes from various catalogs during the decision making process. Reintroducing the comfort of "at your own pace decision making" in electronic shopping will require downloading a catalog by the end consumer to a local computer, and subsequently, performing the browsing and the decision making process in a disconnected mode. Once, the end consumer is comfortable with her own decisions, she reconnects with the online store to submit a newly created order.

Consider another scenario in purchasing a more complex product, e.g., buying a car or home insurance over the phone, where the client answers many questions in defining the service. Some of these questions define the buyer's profile, while the others define specific requirements of the client. The implications of choosing a particular answer are not always very clear to the client until she has a chance to examine the terms and conditions. Making such a process online involves clients participating in long interactive sessions. At a minimum, such sessions need to be persistent, and need to provide backtracking capability to the client. Again, making the purchasing process comfortable to the end-consumer requires the dialog to be performed in a disconnected mode. This can be achieved by downloading an intelligent catalog with embedded questions to the client machines.

Suppliers face some new challenges in facilitating disconnected mode of client operations. First, suppliers would like to maintain strict control on the valid duration of a downloaded catalog. Second, suppliers may not want to reveal some of their business policies (regarding promotion, customer loyalty, competitive pricing, etc.) to clients. Moreover, suppliers would like the ability to change such policies at will in responding to market conditions. Finally, suppliers need to provide custom software to clients for facilitating off-line operations.

In this paper, we introduce the concept of a downloadable service contract in addressing all the above requirements. A buyer connects to a online store or supplier and provides minimal information to create/customize a contract. The contract captures in an unambiguous manner customized rules of interactions including valid duration and imposed limits. The contract is downloaded to the client machine for facilitating disconnected operations. An application program for browsing a downloaded contract and creating an order, may be made available by the supplier. However, the standard format of a contract allows a generic client code to be used for off-line operations. The final selections of a buyer are conveyed to the supplier at a later point, who checks to see that the contract has not been violated by the client, before fulfilling this order. We note in passing that the organizational (B2B) procurement always benefited from such off-line decision making process where complex decision making software systems are deployed in satisfying various business goals. Both buyers and suppliers not only optimize their objectives, but also cooperate in forecasting their needs and/or maintaining inventory. Typically, such collaborative commerce interactions are based on long-term contractual agreements[1, 2]. Dealing with a large number of short-term consumer contracts introduces many interesting challenges, e.g., dynamic pricing [3], contract and risk management.

The remainder of the paper is organized as follows. In the following section, we provide a detailed description of the essential components of a downloadable service contract. In Section 3, the life cycle of such a contract has been explained. Section 4 describes the design of an efficient client contract management system. Server side contract support have been briefly discussed in Section 5. Details of a prototype that extends Ariba cXML protocol based MRO procurement, are presented in Section 6. Section 7 looks at some related research while Section 8 concludes the paper and presents some future research directions.

2 Downloadable Service Contract

The contract specifies the rules and permissible actions by a client application in disconnected mode. A "service" and associated "service" application are meant to relate to product offerings as well as service offerings. For example, the service contract could be for an on-line store, which includes a set of items that can be ordered, the prices of these items, and the time-frame for which these prices are valid. It could also include constraints such as bounds on the number of items or aggregate purchase price. Additionally, it may include rules for operation sequence on the data such as dependency across ordering multiple items, and/or rules on multiple reconnect to the server for submission of disconnected operations, e.g., cancelling orders, query on state of orders, etc.

In most scenarios, the supplier will first draft such a service contract. However, the contract can be customized to reflect client interest and/or jointly created via negotiation between the supplier and the consumer. The challenges involved in designing an efficient supplier contract management system will be explored later in Section 5.

2.1 *Downloadable Service Contract Language (DSCL)*

The service contract is an XML document described by an XML document type definition (DTD) or XML-Schema. The DTD or Schema defines the tree structure of XML tags which form the contractual building blocks of the contract document. The overall structure of a service contract has been shown in Fig. 1. Various parts of a contract are detailed below.

- **Information about the Contract**
  Overall properties of the service contract like the contract identification code (with respect to the supplier), date of issue of the contract, are defined in the Preamble section. The Roles and Participants section provides information about the buyer and seller. In the current scenario (e.g., grocery shopping), the buyer is typically an individual, and is identified by an appropriate identification/authentication scheme. However, the buyer can also represent an organization as in typical B2B procurements. The seller is typically an organization and is uniquely identified by the combination of its identification type (for e.g. DUNS code) and their identification code. The supplier contact information includes the URL/network address where the final shopping order is to be submitted by the buyer.

- **Catalog**
  The Catalog section carries all the information about the items in the personal catalog created by the buyer from the original supplier catalog. The *ItemList* contains a list of the items specified in the catalog. The items are indexed using their *Item IDs*. A short description about each item combined with other information like the base price of the item, its unit of measure, its classification code type etc. are also included [4]. *ItemGroupList* recursively combines item groups that can be referenced in the pricing section to specify wider variety of offers based on such groupings.

- **Pricing Structure**
  In this section, the contract specifies various special offers satisfying specific purchasing conditions. As can be seen in Fig. 1, the tags in this section build upon the information presented in the Catalog section by adding pricing information about the item relative to the contract. *ItemPriceReduction* specifies the

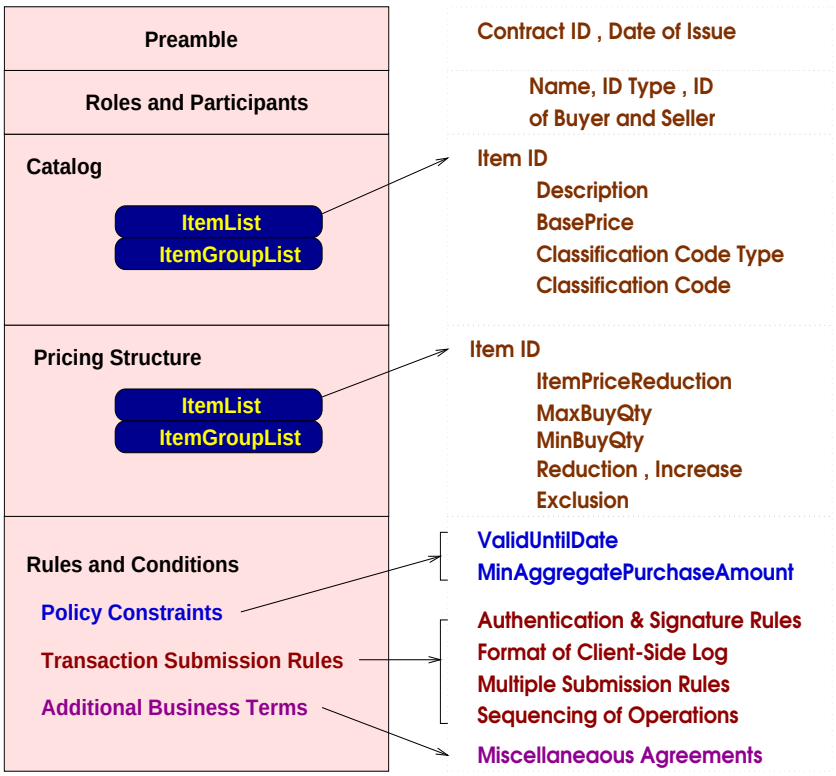

Figure 1. Structure of a Downloadable Service Contract

price reduction on an item upon purchase of a quantity specified by the range *MinBuyQty* and *MaxBuyQty* (Please see the example below). The reduction can also be specified at a group level *GroupPriceReduction*. Some additional restrictions/benefits associated with the purchase of an item or a group of items can also be presented using the *Reduction*, *Exclusion* and *Increase* clauses. The items listed in these clauses must be present in the catalog. They represent either a reduced price on another item(s), excluded items if the current pricing reduction is exercised (i.e., nullification of another reduction clause for these excluded items) or even an increase in the prices of other items (subtracting from their corresponding reductions expressed elsewhere). Note that in most scenarios, the pricing associations amongst items would be simple and expressed through just one of the clauses, i.e., *Reduction*, *Exclusion* or *Increase*. The example below is for illustration purpose only to demonstrate the power of the specification language. A wide variety of pricing schemes for multiple dependent items can be captured by such a simple structure.

In this example (see Table 1), if a customer buys a quantity of item A which lies between 2 and 10 then she gets a reduction of 25% on the base price of the item (as specified in the Catalog section). Furthermore, she gets the option of buying items B & C which have their base prices reduced by 40% and 35% respectively. However, she would not be allowed to exercise any reduction on items X & Y in this case. Additionally, the prices of items P & Q get increased by 10% and 5% respectively.

- **Additional Policy Constraints**
  The Policy Constraints subsection deals with additional machine processable terms and conditions that the buyer has to comply with, in order to make a valid purchase on the supplier catalog. First, a *ValidUntilDate* is specified after which the contract becomes void. The client application program can make use of this field to alert the client about upcoming deadlines. A complex timing rule might be a price for a product that would be applicable if the transaction completed within a particular date, and another price that would be used if not and so on. Additional clauses may include, *MinAggregatePurchaseAmount* i.e., the minimum amount of purchase for this special offer, availability guarantees on specific item(s), etc.

Table 1. Pricing Structure Example

Item ID: A

ItemPriceReduction = 25	MinBuyQty = 2	MaxBuyQty = 10
REDUCTION (Item ID,Price)	EXCLUSION (Item ID)	INCREASE (Item ID,Price)
(B,40) (C,35)	X Y	(P,10) (Q,5)

- **Transaction Submission Rules**
  The contract specifies rules and mechanisms for submission of off-line transactions by the client. This includes security protocol used for document exchange [2], i.e., details of authentication and document signing, URL to which transactions are to be submitted, and format of a purchase order and/or a set of transactions, etc. Furthermore, this specifies rules on multiple submissions and/or sequencing of operations, e.g., cancellation, modification, etc.

- **Additional Business Terms and Conditions**
  This subsection spells out *miscellaneous business agreements* like the return and refund policies, payment and shipping options etc. using plain text.

We have created a version of this specification language which is being used in the current prototype. The language builds on cXML catalog description [4], and the overall contract refers to such a document for catalog description. For lack of space, we omit details of DSCL. We are also exploring extensions to the current policy specification beyond reduction, exclusion and increase associated with a catalog item.

3 Life Cycle of a Downloadable Service Contract : A Three Phase Process

The life cycle of a downloadable service contract can be broadly divided into three phases. The operations have been illustrated in Fig. 2.

1. **Contract Creation and Download**
   In the first phase, the client requests service catalogs from the supplier server while being connected to the server. The client selects one or more services and/or catalogs for browsing, and/or negotiates associated terms and conditions. Both browsing of catalogs and contract negotiation can be performed either by human agents via a web browser or performed via server to server interactions. The client takes into account not only its own needs and policy objectives, but also service contracts offered by the supplier. During this process, the client composes a personal contract to be used during off-line operations. As detailed earlier, the composed contract contains not just the selected catalog items but also various policy constraints.

   The composed contract may need to be signed by both the client and the supplier. Signing the contract not only protects the client from future disputes, but also protects the supplier in managing outstanding copies of catalogs and valid durations on offered prices. Once the contract is downloaded, the client parses it to populate internal tables for easy access.

2. **Client-Side Disconnected Operations**
   The client uses the parsed contract during this phase (c.f. phase II of Fig 2) to perform disconnected mode shopping. As mentioned earlier, the client may download a server-provided customized application program for off-line operations and creating a purchase order acceptable to the server. However, she may have her own personal software that can provide many value-added services, e.g., comparison shopping, satisfying user goals, etc. A detailed discussion of these issues is presented in the next section.

   The benefits of off-line shopping have been already been outlined in Section 1. The client may not submit her purchase order until she is comfortable with her own decisions; the ability to reconsider her own decisions is important. She makes informed selections after taking the contract's item dependency rules into consideration. At each step of the off-line operations the contracts rules are checked by the application program. Finally, when the client decides to submit her selections, they are transformed into a purchase order by the helper software. The purchase order format is preferably standardized, and specified in the service contract.

3. **Transaction Submission to Supplier**
   Upon the client's reconnection to the supplier server, the transformed client log, i.e., purchase order, is uploaded onto the server. The server now verifies the disconnected order based on the contract rules as shown in the phase III of Fig. 2. If any of the selec-

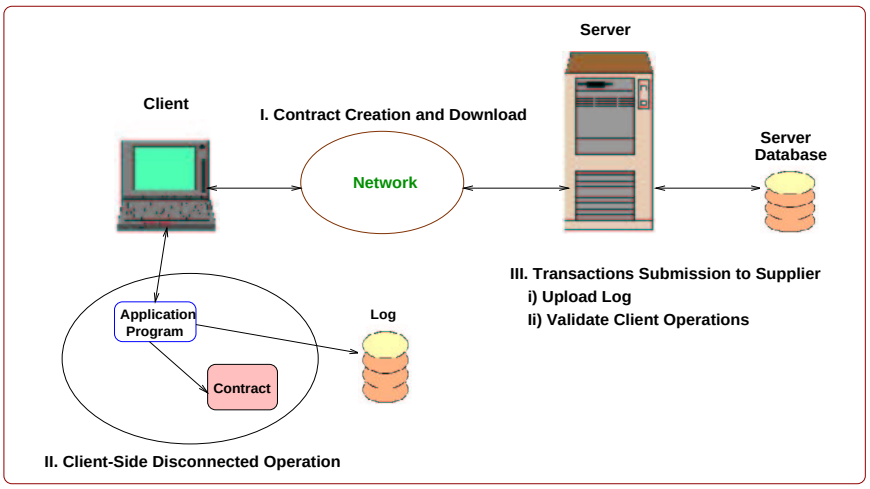

Figure 2. The three phases in the life cycle of a downloadable service client

tions/actions are inconsistent with the rules (e.g., purchase limit, valid duration), the server rejects the order/transaction list and sends a failure notification to the client. Alternatively, if the selections/actions were all consistent with the rules, the submitted order is accepted and a message is sent to the user confirming transaction success.

Handling of rejection of submitted order will depend on specific failure scenarios. Expiry of valid duration may imply restarting the whole process, i.e., the client must start over again by downloading a new contract and by retrying execution, conforming to the rules specified in the contract. On the other hand, failure to adhere to contract rules can be addressed by changing and resubmitting the order.

4 Client Contract Management System (CCMS)

The client may choose to download an application program for browsing the downloaded contract and then creating a purchase order from it, if the supplier supports this kind of an option. The program presents the user interfaces to the client in a manner that is intended in the contract. For example selection of specific items, automatically excludes other items from the price reduction list as specified in the contract. It may also create alerts for upcoming deadlines after which the contract would become invalid. It may provide support on sorting the items in terms of reductions or absolute prices. Furthermore, the application program may facilitate order submission, i.e., creating the final order and submitting to the specified URL. Finally, the program may be invoked to cancel or modify an already submitted order (within a specified time limit), in which case the program follows the action sequence specified in the contract.

Clearly, a more powerful alternative to the use of server-provided application program is the presence of a client specific service contract management system (CCMS). In this case, the client needs to download only the service contract. CCMS can use appropriately specified contract terms to create user interfaces and implement all the client application steps, as specified above. The contract specification language needs to be standardized for such a scheme to work. The service contract specification language (DSCL) presented in the Section 2 can be considered as a first step in that direction.

In this scenario, it is especially important for the server, upon reconnection with the client, to check the submitted order and/or action sequence for proper enforcement.

4.1 CCMS Value-Added Services

In addition to the basic functionalities as sketched above, CCMS may provide various value added services, e.g., comparison shopping across multiple contracts, matching complex user specific goals and maintenance of interaction histories. It can also play a major role in the automated contract negotiation process. The client can setup a user profile as well as current needs in CCMS, enabling CCMS to act on its behalf while corresponding with the suppliers. Let us consider a typical grocery shopping experience. The client may maintain her broad interest list, as well as essential items to be purchased during the next shopping trip. Individual resource planning (IRP) and maintenance of personal inventory are obvious next steps, where purchasing of bulk can be scheduled appropriately. CCMS can automate this contract creation process and saves the client from per-

sonally visiting every supplier's WWW address, and creating a customized catalog to suit her needs.

CCMS can provide sophisticated comparison and query functions. Given a specific set of customer requirements, it can compare against all downloaded contracts, and select appropriate set of items against specific catalogs to optimize on price and/or best match on product attributes. This is similar to bid evaluation functions in B2B procurement [5].

4.2 Relational Database Schema Design

In this section, we look at the detail design of the relational database tables that need to be supported in CCMS. The simple entity-relationship (E-R) diagram shown in Fig. 3 helps us to capture a high-level description of the data involved. We follow the conventions of the E-R model described in [6]. The goal of our database design is to support the kind of functionality needed to store and work with a downloaded service contract (as presented in Section 3.1).

In Fig. 3 we show the relationship set `SUPPLIES`, in which each relationship indicates a contract that is supplied by a supplier. Every supplier is uniquely identified by a combination of two fields, `ID_TYPE` and `ID`. The type(s) of business protocols supported by the supplier server and its network address or URL needs to be specified in `SUPPLIER`. The user's `USER_ID` and `PASSWORD` are also stored along with other information about the supplier. There exists a key constraint that each contract has at most one supplier (multiparty service contracts are not supported at present) and a participation constraint that every contract needs to be associated with a supplier. The thick line connecting `CONTRACT` and `SUPPLIES` indicates the former while the latter constraint is represented by the arrow on it.

The `CONTRACT` entity set stores all the information associated with the contract like the `CONTRACT_ID`, `VALID_UNTIL_DATE` and `MIN_AGG_AMT`. We are currently exploring ways to represent the transaction submission rules (explained in Section 2.1) of a contract in the `CONTRACT` entity set. For now, we store such rules directly in the `RULES` field. The items present in the associated catalog are stored in the weak entity set `ITEM`. It must be noted that each item is present in exactly one contract, the corresponding `CONTRACT_ID` acts as the identifying owner. Items present in groups are handled similarly through `GROUP_ITEM`. Every `CONTRACT` entry needs to be associated with at least an item while it may not contain any groups of items. Hence, `CONTRACT` is connected to `ITEM_REL` and `GROUP_ITEM_REL` by a thick and ordinary line respectively.

From the above ER diagram, we generate a relational database schema using some standard rules. Each of the entity sets are mapped to a relation in a straightforward way. This leads to four tables namely `SUPPLIER`, `CONTRACT`, `ITEM` and `GROUP_ITEM`. Fig. 4 shows three of these tables, the `GROUP_ITEM` table has been left out as it is very much similar to the `ITEM` table. Creation of a separate table for the one-to-many `SUPPLIES` relationship set has been avoided. This is made possible by storing the primary key of `SUPPLIER` ie. (`ID_TYPE,ID`) in the `CONTRACT` table as a foreign key and adding a constraint that these field cannot take null values. Such a scheme captures both the key and the participation constraints associated with the `SUPPLIES` relationship. Tables for the `ITEM_REL` and `GROUP_ITEM_REL` have been similarly avoided.

The above tables have been used in the current prototype, whose associated interaction protocol is detailed in Section 6. The next subsection provides a broad overview of the building blocks of CCMS. Only a subset of these functions (i.e., no server to server automated negotiation) are realized in our current prototype.

4.3 Components and Execution Flow

Fig. 5 illustrates the flow of execution in CCMS and a high level overview of its various building blocks.

The main user interface presented to the user gives him the option of :

- Create Downloadable Contract
- Browse/Submit Downloaded Contract
- Review Submitted Purchase Orders

The above options provided by CCMS enable the user to first create a downloadable contract and then browse through that contract to create her purchase order. At the end of every browsing session, she gets of the option of submitting her selections or storing them locally so that she can come back and submit the order in a later session. We will next look at a possible user interaction scenario with the help of Fig. 5.

Contract Creation and Download (Steps 1 to 10 of Fig. 5)

The user can setup a *User Profile* with CCMS and provide it with *Current Needs*, a description of the items she wants to procure. She then chooses to *Create Downloadable Contract* through the main user interface which hands over the user's request to the *Contract Manager*. The *Contract Manager* is the central controlling unit in CCMS. It helps the user to perform online browser-based negotiation of a downloadable contract with the possible suppliers. However, automated contract negotiation is also possible with the help of the *Contract Negotiator*.

The *Contract Negotiator* starts the negotiation of a service contract by composing an appropriate request addressed to all the possible suppliers. It refers to the *User*

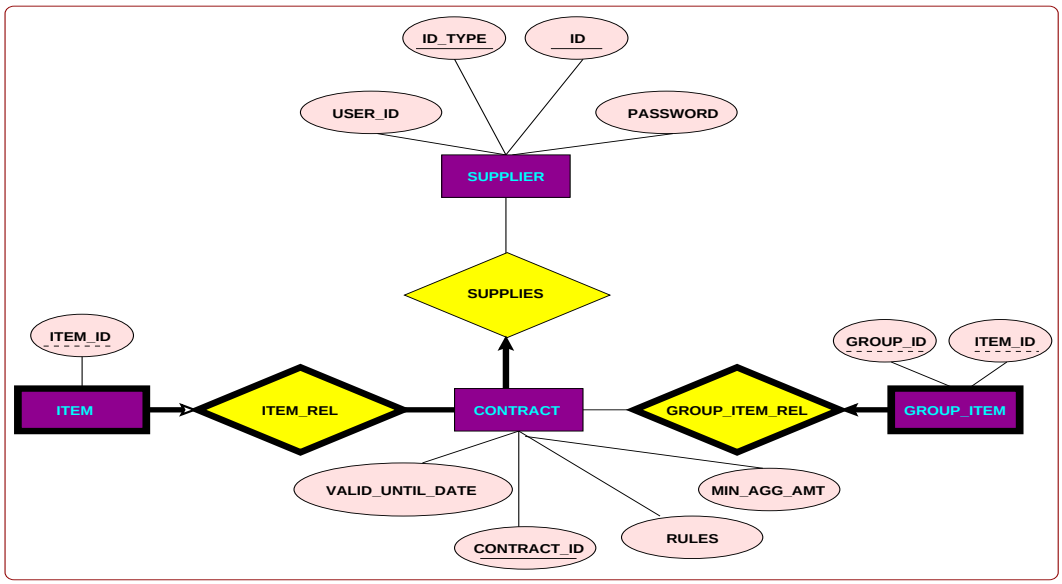

Figure 3. Entity-Relationship Diagram for CCMS

SUPPLIER
CODE_TYPE , CODE
NAME
BUSINESS_PROTOCOL
URL
USER_ID
PASSWORD

CONTRACT
CONTRACT_ID
ID_TYPE , ID
VALID_UNTIL_DATE
MIN_AGG_AMT
RULES

ITEM
ITEM_ID , CONTRACT_ID
BASE_PRICE
CONTRACT_REDUCE
MIN_QTY
MAX_QTY
DESCRIPTION

Figure 4. Relational Database Tables used by CCMS

Profile and *Current Needs* and composes a request message in accordance with the supplier server's business protocol. This message is then delivered to the *Protocol Gateway* which is responsible for communicating the message to the supplier server. After a possible series of request-reply exchanges between the supplier server and the *Contract Negotiator*, the supplier server sends out a message with the contract embedded in it. The *Contract Receiver* parses out the contract from this response message and stores the contract in the `CONTRACT` and `ITEM` database tables. The transaction submission rules of the contract are stored in the `RULES` field of the `CONTRACT` table. The *Contract Negotiator* can use the above steps repeatedly to download service contracts from multiple suppliers.

Disconnected Mode Shopping and Order Submission (Steps 11 to 16 of Fig. 5)

When the user chooses to *Browse/Submit Downloaded Contract*, the *Contract Manager* evaluates the contracts according to the conditions provided in the *User Profile* and *Current Needs*. A strategy similar to the bid evaluation model presented in [5] may be used. The user is asked to choose a particular contract for further consideration. The *Contract Manager* refers to the `RULES` field of the corresponding `CONTRACT` table entry for presenting user interfaces and guiding the user in creating a purchase order that is acceptable to the supplier. The *Purchase Order Generator* creates such a purchase order from the log of disconnected operations maintained by the *Contract Manager*. Additionally, the *Contract Manager* may maintain interaction histories of the user with its different suppliers. Such histories can be later referred to in the audit trail.

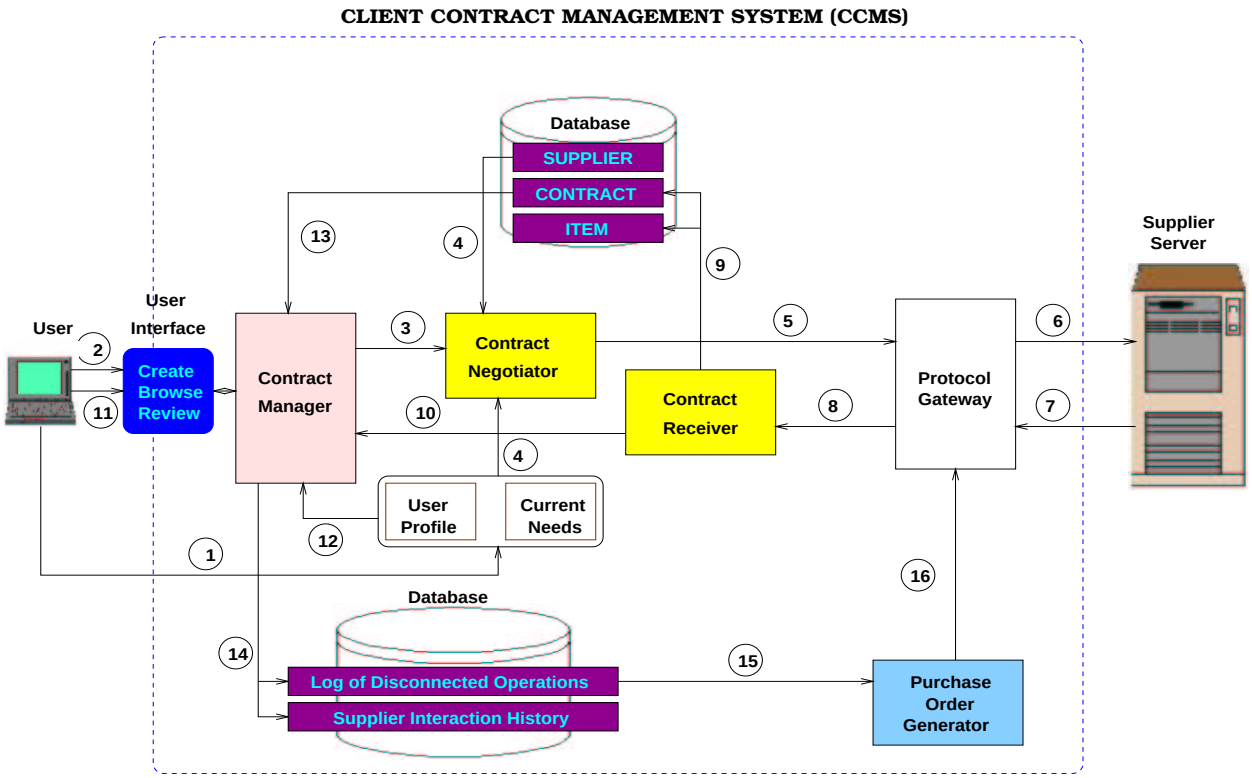

Figure 5. Flow of Execution in CCMS

5 Server-Side Contract Support

Today's supplier servers are mostly online storefronts and shopping cart systems with the necessary software to create, navigate and manage product catalogs. Such support is good enough if the supplier wants its customers to be able to browse through its online catalogs, create shopping carts and pay at the end of their shopping session. However, if the supplier wants its clients to download contracts and submit their purchase orders at a later date, it has to extend its server functionality to handle the contractual aspects of downloadable catalogs. Support for simple catalog browsing needs to be enhanced to handle contract creation and negotiation. User profiles may be maintained at the supplier site with the help of the various explicit and implicit personalization techniques proposed in [7]. Such profiles assist the supplier in deciding what are the benefits it can provide or what kind of restrictions it needs to impose on the downloadable contract. Suppliers may also need to integrate complex decision making software for dynamic pricing, that may take into account many factors including customer loyalty, and competitive pricing, etc. [3].

Information about all the pending contracts has to be kept in a manner that allows efficient queries on the states of the contracts e.g., the contracts that expire today or the contracts that include a specific item. Inventory control is also a challenging task considering the fact that simply locking the resources to guarantee the availability of an item is not feasible when the resources are at a premium. Lastly, the server should be able to verify the validity of a purchase order that gets submitted corresponding to a pending contract and take appropriate steps to notify the client about the result of her transactions. We are currently working on the development of efficient server-side support strategies for downloadable contracts.

6 Prototype Implementation

The primary goal of our system design is to extend current MRO procurement process to include contractual terms and conditions. We have implemented a prototype system in which the client procurement system uses the Ariba Commerce XML (cXML)[4] protocol to communicate with the supplier site. The supplier side has been emulated with the IBM Websphere Commerce Suite (WCS)[1], which is a business application product for manufacturers to build online stores on the Internet. A second goal is to create a

[1]Trademark or registered trademark of International Business Machines Corporation

lightweight, flexible and easily configurable system for an individual user. Furthermore, we have used a proxy-based architecture for the clients, where a proxy machine acts on behalf of multiple users to communicate with the supplier servers over the Internet.

6.1 What is cXML ?

cXML is a set of message types based on a cXML document type definition (DTD) to describe the messages exchanged between a e-Marketplace and a supplier. It is being used for exchange of catalog content and to define request/response processes for secure electronic transactions over the Internet. The processes include purchase orders, change orders, acknowledgments, status updates, ship notifications and payment transactions. It enables buyers to 'punchout' from a shopping session to a remote host to view or select catalog items without losing what is already been selected and added to their locally hosted shopping cart. The unshaded events in Fig. 6 represent a typical punchout event sequence.

The PunchOutSetupRequest message is used to communicate the identity of the originating system to the remote host. In response, the buyer receives a PunchOutSetupResponse message that directs his session to a remote HTML browsing session. In Browse Supplier Catalog phase, the buyer browses the supplier's catalog and creates a remote shopping cart. Using the PunchOutOrderMessage, the contents that are placed in the shopping cart are transferred back into the buyer's current shopping cart in preparation for check-out processing. The next step in the shopping process is the actual transmittal of the purchase order. Though cXML format is not required for transmitting purchase orders, automated order processing becomes easy if cXML's well-defined structure is used. The buyer's procurement application sends out a OrderRequest cXML message which is analogous to a purchase order. The supplier server acknowledges the receipt of the above message through the OrderResponse message.

6.2 Extensions to cXML for Downloadable Service Contracts

We have extended the cXML MRO process to handle the notion of service contracts. The shaded events shown in Fig. 6 represent our extensions to the cXML 'punchout' event sequence. First four events remain unchanged from the original event sequence. When a browser based client receives the PunchOutOrderMessage, it treats the list of items received as the "downloadable catalog". Subsequently, the server presents a predefined contract that includes the catalog returned in the earlier message. In our prototype, we allow the client to change/negotiate specific contract items (i.e., alternative quantity range hoping for a better price reduction). The client also looks for guarantee on the availability of the product. The supplier may respond with a chosen pricing scheme. Clearly, there are many aspects that can be negotiated, and other work will/have focuss(ed) on various negotiation strategies. These are complementary to our current work.

After the contract negotiation phase, the client obtains a customized contract to make off-line shopping decisions. We have also implemented a simple client prototype system for Disconnected Mode Shopping operations, during which the client selects various items based on the downloaded contract. Finally, the prototype creates a purchase order to be submitted to the supplier system. This purchase order submission is based on the cXML OrderRequest message. The supplier validates the purchase order in the Validation phase and sends a OrderResponse message back to the client. This OrderResponse message extends the cXML OrderResponse by not only acknowledging the receipt of the purchase order but also possible rejection of the order in case of contract violation.

7 Related Work

Business-to-business (B2B) integration is fundamentally about coordinating information among businesses and their information systems. Immense challenges exist due to issues of privacy, autonomy, heterogeneity in software and platforms and above all, management of complex interactions. In recent years, researchers have been trying to bridge the associated syntactic and semantic information gap between businesses with the help of contract or agreement approaches[8, 9, 2]. Milosevic & Bond proposes a business contract architectural framework for the Internet on the basis of Electronic Data Interchange (EDI) messages[8]. Dan et. al. discusses the specific functions needed in a B2B electronic contract[9]. It also presents a design for supporting the specification and enforcement of business deals among providers, based on code generation tools that generate server-side and client-side code for enforcing the terms of such electronic service contracts. They further extend their approach by proposing a specification language called tpaML in the form of an XML grammar for expressing electronic trading partner agreements (TPA)[2]. IBM Research has designed and prototyped a B2B protocol framework or BPF, a comprehensive run-time tool set which enables the automatic deployment of business protocols expressed in tpaML. The goal of CCMS (explained in Section 4) is to provide a similar kind of support for downloadable service contracts expressed using the DSCL proposed in Section 2.1. This approach signifies a way of enriching the domain of business-to-consumer electronic commerce from the techniques that have proved beneficial in the B2B do-

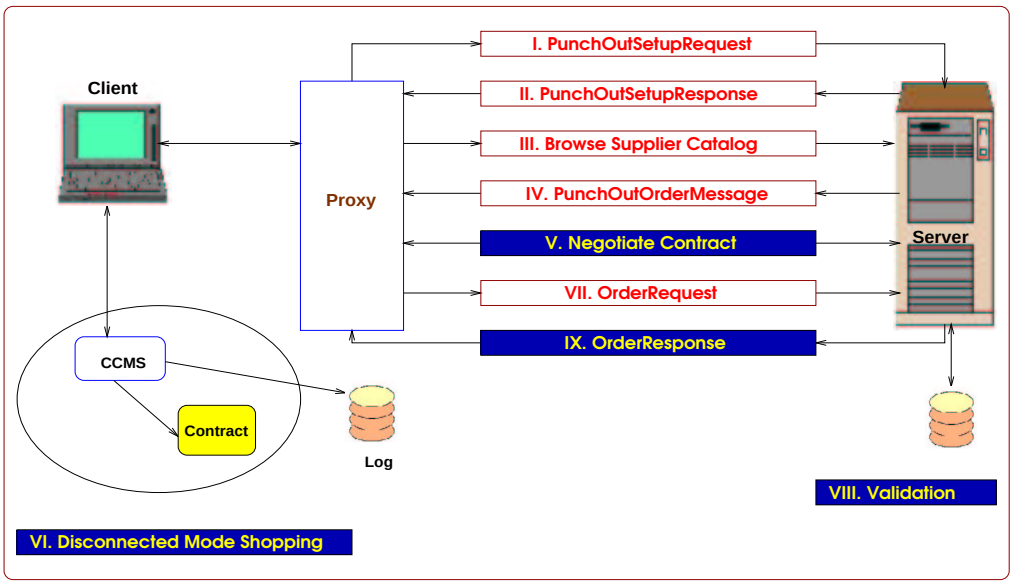

Figure 6. Extensions(shaded events) to the basic cXML messaging(unshaded events)

main.

Examples of business negotiations include auctions, competitive bids for procurement, brokerages/exchanges/cartels and two party negotiations. Kumar & Feldman models all the above types of negotiations as a generic finite state machine(FSM)[10]; such a representation might be useful in modelling the negotiation in our case as well. Mobile agent frameworks [11, 12] have been proposed in which mobile agents talk to application servers on behalf of their clients. Griffel et. al. advocates the use of mobile agents for electronic contract negotiation but assumes the existence of a shared middleware between the client and the server systems[11]. Marques et. al. presents a disconnected client's agent that has limited capabilities when it arrives at the server, and thus it is not able to capture all the requirements of the client[12]. Waern provides a agent-based service contract negotiation scheme in the domain of open service architectures like the WWW[13]. However, all the above schemes neither suggest a specific contract language nor discuss embodiment of a system based on a business contract.

Grosof et. al. presents a declarative approach to model business rules in e-commerce contracts[14]. It defines "rule" as an implication (i.e., IF-THEN) in which the antecedent (i.e., the IF part) may contain multiple conjoined (i.e., ANDed) conditions. An XML embodiment of the approach known as *Business Rules Markup Language (BRML)* has been proposed in [14]. Current policy constraints in DSCL are simple pricing schemes and/or association rules. We will explore use of BRML and other proposed rule languages to enhance policy description in DSCL.

Comparison shopping across multiple service contracts can be supported by CCMS with the help of a scheme similar to the bid evaluation system presented in [5]. It uses a hierarchical decision-criteria framework that allows the buyer to specify business rules which can then be used to find the "optimal" bid. CCMS can use such a scheme to evaluate the downloaded contracts and present the user with a contract that best matches her *User Profile* and *Current Needs*.

We also note in passing the high level similarities of proposed eCoupon system [15] where promotions and price reductions are captured as electronic coupons. The main focus of this paper is to address challenges in scalability and security to avoid duplication of electronic coupons. Contracts are personalized electronic documents that contain customized catalogs and richer pricing policies in the form of complex offers.

8 Conclusion and Future Work

Client participation in web-based services, e.g., shopping in online stores, creating service agreements as in buying a car insurance, currently demands browser based online operations with many on-the-spot decision making. Supporting disconnected operations for business-to-consumer scenario can introduce some of the same benefits, e.g, comparison shopping, satisfying client goals, at the individual consumer level as used in B2B procurements. In this paper, we proposed the concept of a downloadable service contract, that specifies unambiguous rules of interaction be-

tween an end client and a service provider. The client initially negotiates with the service provider a contract, which is subsequently used not only to guide off-line client operations, but also for validating at the service provider the log of actions, i.e., a purchase order, submitted by the client. A generic application program for browsing a downloaded contract and creating an order, may be made available by the service providers. However, a client may have its own software with many value-added services such as comparison shopping across multiple contracts, matching complex user specific goals and maintenance of interaction histories. We described a prototype system that extends the current Ariba cXML protocol for MRO procurement with additional messages.

Many challenges remain in fully realizing such a model. The online supplier systems need to be extended to negotiate customer specific deals and manage a large set of customized contracts. This however, also brings us closer to realizing agent based shopping where support for dynamic pricing and complex decision making schemes become essential. The contract language also need to be standardized in order to deploy such a model. However, the paradigm of disconnected mode shopping may become ubiquitous in facilitating many areas of personal procurement(e.g., grocery shopping).

Acknowledgements

The authors gratefully acknowledge the contributions of Dan Dias and Pradeep Janakiraman in originating some of the early ideas. The authors would also like to thank Catherine Crawford, Richard King and Hidayatullah Shaikh for their help on the server side support for cXML protocol, which is further extended for the current prototype.

References

[1] Chuck Philips and Mary Meeker, "The B2B Internet Report - Collaborative Commerce", *Morgan Stanley Dean Witter*, April 2000.

[2] A. Dan, D. M. Dias, R. Kearney, T. C. Lau, T. N. Nguyen, F. N. Parr, M. W. Sachs and H. H. Shaikh, "Business-to-business integration with tpaML and a business-to-business protocol framework", *Technology for E-Business, IBM Systems Journal, Volume 40, Number 1*, 2001 .

[3] Jeffery O. Kephart, James E. Hanson and Amy R. Greenwald, "Dynamic Pricing by Software Agents", *Computer Networks*, 2000.

[4] "Commerce XML", *http://www.cxml.org* , *http://www.oasis-open.org/cover/cxml.html*

[5] Quoc-Bao Nguyen, Mitchell A. Cohen and Jen-Yao Chung, "A Bid Evaluation System for Internet Contract Negotiation", *2nd International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS 2000)*, June 2000.

[6] Raghu Ramakrishnan and Johannes Gehrke, "Database Management Systems", *McGraw-Hill, Second Edition*, Aug. 1999.

[7] Charu C. Aggarwal and Philip S. Yu, "Data Mining Techniques for Personalization", *IEEE Data Engineering*, Vol. 23, No. 1, pp. 4–9, March 2000.

[8] Zoran Milosevic and Andy Bond, "Electronic Commerce on Internet: What Is Still Missing ?", *5th International Networking Conference (INET 1995)*, June 1995.

[9] Asit Dan, Daniel M. Dias, Thao Nguyen, Marty Sachs, Hidayatullah Shaikh, Richard King and Sastry Duri, "The Coyote Project: Framework for Multiparty E-Commerce", *2nd European Conference on Research and Advanced Technology for Digital Libraries, (ECDL 1998)*, Lecture Notes in Computer Science, Vol. 1513, p. 873, Springer-Verlag, 1998.

[10] Manoj Kumar and Stuart I. Feldman, "Business Negotiations On The Internet", *Technical Report, IBM Institute of Advanced Commerce*, March 1998.

[11] F. Griffel, M.T. Tu, M. Mnke, M.M. da Silva, M. Merz and W. Lamersdorf, "Electronic Contract Negotiation as an Application Niche for Mobile Agents", *1st International Enterprise Distributed Object Computing Conference (EDOC 1997)* , pp. 354-365, Oct. 1997.

[12] P.J. Marques, L.M. Silva and J.G. Silva, "A Flexible Mobile Agent Framework for Accessing Information Systems in Disconnected Computing Environments ," *11th International Workshop on Database and Expert Systems Applications*, pp. 173-177, Sept. 2000.

[13] A. Waern, "Service Contract Negotiation - Agent-Based Support for Open Service Environments," *Workshop on Distributed Artificial Intelligence, at the 4th Australian Conference on Artificial Intelligence*, 1998.

[14] Benjamin N. Grosof, Yannis Labrou and Hoi Y. Chan, "A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML", *1st ACM Conference on E-Commerce*, Nov. 1999.

[15] Rahul Garg, Parul Mittal, Vikas Agarwal and Natwar Modani, "An Architecture for Secure Generation and Verification of Electronic Coupons", *IBM Research Report, RI00026*, Dec. 2000.