The World of e-Business: Web-Services, Workflows, and Business Transactions

Michael P. Papazoglou

Infolab, Tilburg University, PO Box 90153, 5000 LE, Tilburg, Netherlands
{mikep}@kub.nl

Abstract. Process oriented workflow systems and e-business applications require transactional support in order to orchestrate loosely coupled services into cohesive units of work and guarantee consistent and reliable execution.
In this paper we introduce a multi-level transaction model that provides the necessary independence for the participating resource managers, e.g., local database and workflow servers of organisations, engaging in business transactions composed of interacting web-services. We also present taxonomy of e-business transactions features such as unconventional atomicity criteria, the need for support for business conversations and the need for distinguishing between three basic elements within a business transaction. In addition, we argue that an extensible framework such as the Business Transaction Protocol (BTP) proposed by OASIS is necessary for building robust and extendible e-business applications.

1 Introduction

Electronic-business applications are based on a sequence of business messages that are exchanged between enterprises, their content, precise sequence and timing, for the purpose of carrying a business activity or collaboration, e.g., securities trade settlement. Such business activities are known as business processes and could be both internal and external to organisations. We, may view an automated *business process* as a precisely choreographed sequence of activities (actions) that performs a certain business task and which operate from a start state until an end state is achieved. For example, processing a credit card number, hiring a new employee, ordering goods from a supplier; creating a marketing plan; processing and paying an insurance claim; and so on, are all examples of business processes. An activity such as a credit check, automated billing, a purchase order, stock updates and shipping on the back end systems, or such frivolous tasks as sending a document, and filling a form, constitutes part of the business process.

Business processes need to cross-organisational boundaries, i.e., they occur across organisations or between organisational subunits. Therefore, they can drive their collaboration to achieve a shared business activity by enabling highly fluid networks of collaborating web-services. The process workflow is made up of activities that follow routes, with checkpoints represented by conditions and rules. Enterprise workflow systems today support the definition, execution and monitoring of long running processes that coordinate the activities of multiple business applications. However, they do not separate internal implementation from external protocol description.

Web-services technology promises to facilitate this undertaking by replacing proprietary interfaces and data formats with a standard web-messaging infrastructure. Web messaging is sufficient for some simple application integration needs; however, it does not adequately support the complete automation of critical business processes. Process automation requires both the ubiquitously supported standards for interfaces and data that work as well across the firewall as well as within it.

With e-business applications trading partners must run their own, private business processes (workflows). The interdependent workflows among multiple trading partners need to be coordinated to ensure that the outcome of the collaborative business process is reliable. Therefore, the interdependent workflows, which may drive business transactions, must interlock at points to achieve a mutually desired outcome. This synchronization is one part of a wider business collaboration protocol that defines the public, agreed interactions between business parties. Therefore, the motivation is to create a business transaction protocol (BTP)[1] to be used in e-business applications that require transactional support beyond classical ACID and extended transactions. Classical (ACID) transactions[2] and extended transaction models based on the ACID transactions are too constraining for the applications that include activities and services that are disjoint in both time and location. Strict ACIDity and isolation is not appropriate to a loosely coupled world of autonomous trading partners, where security and inventory control issues prevent hard locking of local databases. Sometimes in a loosely coupled or long running activity it may be desirable to cancel a work unit without affecting the remainder.

The purpose of BTP is to orchestrate loosely coupled software services (e.g. web services) into a single business transaction. BTP offers transactional support in terms of coordinating distributed autonomous business functionality, in the form of services that can be orchestrated by the business application.

This paper provides an overview of current developments in the areas of business process automation and workflow systems for service-based B2B applications and outlines the requirements, essential characteristics and building blocks for BTPs based on networks of collaborating web-services. The paper introduces a set of criteria for business transaction functionality and measures standard initiatives such as the Web Service Flow Language (WSFL) and the ebXML Business Process Specification Schema (BPSS) against them.

[1] A standard BTP is currently under development by representatives of numerous software product companies, grouped in the Business Transaction Technical Committee (BTTC) of OASIS [1].

[2] ACID transactions can be still be used in the context of e-business for short duration activities.

2 Business Transactions Characteristics

A business transaction is a consistent change in the state of the business that is driven by a well-defined business function. Normally, business processes are composed of several business transactions essentially signifying interactions between businesses to accomplish some well-defined shared business process. A business transaction in its simplest form could represent an order of some goods from some company. The completion of an order results in a consistent change in the state of the affected business: the order database is updated and a document copy of the purchase order is filed. More complex business transactions may involve payment processing, shipping and tracking, coordinating and managing marketing strategies, determining new product offerings, granting/extending credit, managing market risk and so on.

Business transactions (BTs) are automated long-running propositions involving negotiations, commitments, contracts, shipping and logistics, tracking, varied payment instruments, exception handling and customer satisfaction. BTs exhibit the following characteristics:

1. They represent a function that is critical to the business, e.g., supply-chain management.
2. They can involve more than two parties (organisations) and multiple resources operated independently by each party, such as business applications, databases and ERP systems.
3. They define communications protocol bindings that target the emerging domain of web-services, while preserving the ability to carry business transaction messages also over other communication protocols. Protocol message structure and content constraints are schematised in XML, and message content is encoded in XML instances.
4. They should be based on a formal trading partner agreement, such as RosettaNet Partner Interface Processes (PIPs) or ebXML Collaboration Protocol Agreements (CPAs).

Due to their long-lived nature and multi-level collaborations, BTs require support for a variety of unconventional behavioural features that can be summarised as follows:

1. *Generic characteristics:*
   (a) who is involved in the transaction;
   (b) what is being transacted;
   (c) the destination of payment and delivery;
   (d) the transaction time frame;
   (e) permissible operations.
2. *Distinguishing characteristics:*
   (a) links to other transactions;
   (b) receipts and acknowledgments;
   (c) identification of money transferred outside national boundaries.

3. *Advanced characteristics:*
   (a) the ability to support reversible (*compensatible*) and repaired (*contingency*) transactions;
   (b) the ability to reconcile and link transactions with other transactions;
   (c) the ability to specify contractual agreements, liabilities and dispute resolution policies;
   (d) the ability to support secure transactions that guarantee integrity of information, confidentiality, privacy and non-repudiation;
   (e) the ability for transactions to be monitored, audited/logged and recovered.

Business transactions usually operate on document-based information objects such as *documents* and *forms.* A document is traditionally associated with items such as purchase orders, catalogues (documents that describe products and service content to purchasing organisations), bids and proposals. A form is traditionally associated with items such as invoices, purchase orders and travel requests. Forms- based objects are closely aligned with business transactions that have a numerical or computational/transformational nature while document-based objects are usually associated with agreements, contracts or bids. This allows business transactions to interchange everything from product information and pricing proposals to financial and legal statements.

When a business function is invoked through a web-service as part of a larger business process, the overall transactional behaviour associated with that business process depends on the transactional capabilities of the web-service. Rather than having to compose ever more complex end-to-end offerings, application developers choose those elements that are most appropriate, combining the transactional and non-transactional web- service fragments into a cohesive BT-service whole. In [18] we proposed two kinds of business transactions (on which we expand in this paper):

– *Atomic business transactions:* these are small scale interactions made up of services that all agree to enforce a common outcome (*commit* or *abort*) of the entire transaction. The atomic transaction follows the ACID properties and guarantees that all participants will see the same outcome (atomic). In case of a success all services make the results of their operation durable (commit). In case of a failure all services undo (*compensate*, *roll-back*) operations that they invoked during the transaction.
– *Cohesive business transactions* (or "cohesions[3]"): these are aggregations of several atomic transactions. Cohesions are non-atomic in the sense that they allow the *selective confirm (commit) or cancel (rollback) of participants* (even if they are capable of committing). The atomic transactions forming a particular cohesion do not necessarily have a common outcome. Under application control, some of these may be performed (confirmed), while others may fail.

To understand atomic transactions we introduce a simple example. Assume that a client application (initiator in Figure-1) decides to invoke one or more

[3] This terminology has been borrowed from the OASIS BTP specification.

operations from a particular service such as orderFulfillment. It is highly likely for the client application to expect these operations to succeed or fail as a unit. We can thus view the set of operations used by the client in each web-service as constituting an atomic unit of work (*atomic BT or service-atom*). An atomic transaction follows the traditional ACID properties and must either fully commit or fully rollback. Within an atomic transaction, the operations exposed by a single transactional web-service and the internal processes of the service, e.g., support processes, would usually make up a single atomic transaction.

Since atomic transactions use a two-phase commit protocol (with presumed abort), a coordinating process is required to manage the BTP messages sent to the participating services within a given atomic transaction. This coordinator might be implemented within the application itself, or more likely, it will be a specialised web service [2]. Once the actual work involving the consumed web services in an atomic transaction has finished, the client application can begin the two-phase commit coordination of those web- services. The client is expected to *control all aspects of the two-phase commit protocol*, i.e. prepare phase and confirm phase. to decide upon timings implicitly permits reservation-style business processes to be carried out with ease. For instance, this can apply to a hotel or aircraft reservation system, where the prepare phase of the two- phase commit protocol reserves a room or seat, and the confirm phase actually buys the reserved room or seat.

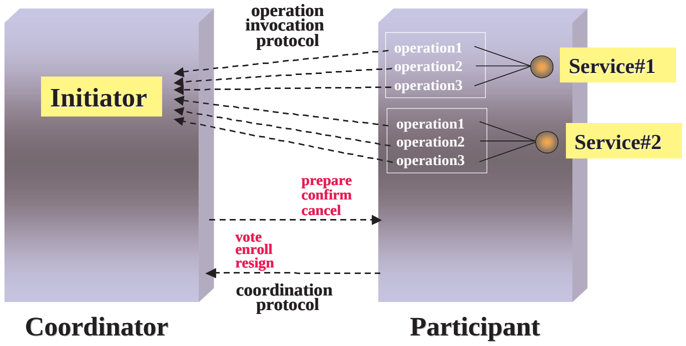

Fig. 1. BT actors and their invocations.

A cohesive transaction (or cohesion) is a set of service atoms that can be manipulated by the BT's initiator (typically a client application). With cohesion, an initiator can dictate whether a service atom within the cohesion succeeds or fails, even if the service is capable of succeeding. To exemplify this consider the following case [2]. In an e-business application one service atom arranges for the purchase of a valuable object, while a second arranges for its insurance, and a third one for its transportation. If the client application is not risk-averse, then even if the insurance atom votes to cancel, then the client might still confirm the cohesion and get the item shipped uninsured. Most likely, however, the client would simply like to re-try to obtain insurance for the item. In this case the web services infrastructure comes into play, and the client would opt for another insurer via the UDDI service infrastructure. Once the client discovers a new insurer, it can try again to complete the cohesion with all the necessary atoms voting to confirm.

One of the major drawbacks of web-services is that their underlying implementations are located remotely and hosted by third parties, thus, there is an increased risk of failure during their application. To cater for this threat a cohesion transaction may decide to selectively cancel atomic transactions and create new ones during its lifetime. Thus, the membership of cohesion is established dynamically by the action of a client application.

Using a combination of atomic transactions that deal with smaller interactions, and cohesions that deal with larger business transactions, provides a great deal of flexibility in creating an overall transaction scheme that is both robust and flexible [2].

2.1 Business Conversations

In business applications it is expected that web-services can communicate through conversation sequences. A conversation sequence is a long-running sequence of interactions, e.g., documents exchanges, between two or more interacting services [12]. For example a component manufacturer may engage in a conversation with a supplier and a conversation with a shipper to carry out the activity of purchasing parts. In many situations, the backend logic triggered as part of these conversations may be transactional. For example, it is possible to arrange for parts to be shipped and later to cancel the shipment (provided that the parts have not been actually shipped). Cancelling the shipment is an example of a compensating transaction, it compensates for the initial transaction that arranged for the shipment. Since the notion of conversation is fundamental to web-services, the exportation of transactional properties should fit within the context of conversations, giving rise to transactional conversations [12]. For example, a master purchasing agreement, which permits the placing of orders for components by known buying organisations allows a buyer and a seller to create and subsequently exchange meaningful information about the creation and processing of an order. Such agreements stem from business negotiations and are specific to a particular trading community, e.g., a vertical e-marketplace such as semiconductors, chemicals, travel industry, etc.

One important element of business conversations is to be able to demarcate the parts of a conversation that are transactional. At one extreme the whole conversation sequence may be considered transactional. However, as this is not practical, it is more likely that a conversation sequence may have multiple parts that we can view as transactional.

2.2 Atomicity Criteria

BTs are governed by unconventional types of atomicity much in the spirit of business applications that use them. We may distinguish between the following broad types of atomicity some of which were reported in [5] and [19]:

– *Service request atomicity:* a single operation on a web service occurs completely or not at all. This is a capability that the end-point publishes to users. The end-point may implement this by an internal transaction on its infrastructure, or some other mechanism. Service request atomicity simply implies that each service provider offers services and operations (and guarantees) that will complete as an atomic piece of work.
– *Payment atomicity:* payment-atomic protocols affect the transfer of funds from one party to another. Payment atomicity is the basic level of atomicity that each electronic commerce protocol should satisfy.
– *Goods atomicity:* goods atomicity protocols are payment-atomic, and also affect an exact transfer of goods for money. Goods atomicity implies that the goods will be received only if payment has been made.
– *Certified delivery atomicity:* certified delivery-atomic protocols are payment- and goods-atomic protocols that allow both transacting parties to prove exactly which goods were delivered. A goods-atomic transaction guarantees delivery to the customer, but an additional requirement is that the right goods are delivered.
– *Contract atomicity:* in addition to these basic atomicity protocols, BTs are generally governed by contracts and update accounts. These are normally based on electronic commerce protocols that include the exchange of financial information services and the exchange of bills and invoices. Thus for e-business applications payment- atomic protocols must also be contract-atomic. If a contract atomic BT succeeds it may be designated as legally binding between two (or more) partners, or otherwise govern their collaborative activity. If it fails it is null and void, and each partner must relinquish any mutual claim established by the transaction. This can be thought of as ”rolling back” the BT upon failure.
– *Non-Repudiation atomicity:* contract atomic transactions are also non-repudiation atomic. This requires the storage of all the messages and receipts for extended periods of time (months or years).
– *Conversation atomicity:* allows a pair of collaborating services to correlate sequences of requests within a logical unit of work. The pair of services uses architected conversation messages and headers to begin the conversation and end the conversation. They determine if the conversation ended successfully or if one or both participants want the conversation to rollback. The semantics of rollback is that each participant (service) can undo the operations it has performed within the conversation and expect its counterpart service to revert back to a consistent state. Furthermore, a service can rely on its counterpart service's transactional behaviour to ensure state consistency in the presence of failures, such as logical error conditions, e.g., shipping is impossible, or system-level failures, e.g., crash of a process or a network failure.

The above atomicity criteria could be seen as core programming constructs for building business applications that make use of them. I addition, they can be extended and specialised to serve the needs of a large variety of business applications.

3 Business Transaction Protocols

One of the earlier attempts to define an Internet-based transaction protocol that simplifies the distributed applications programming is the Transaction Internet Protocol (TIP) [3]. The TIP deals with the deficiencies of conventional two-phase commit (2PC) protocols. Conventional 2PC protocols employ a one-pipe model whereby the transaction protocol flows over the same conduit (pipe) as the as the application to application communications. This renders them complex and hard to implement, thereby inhibiting the development of distributed business applications. TIP is a simple two-phase commit protocol that removes the restrictions of conventional 2PC protocols by providing ubiquitous distributed transaction support in a heterogeneous and cross-domain environment. This is made possible by employing a *two-pipe* model separating the transaction protocol from the application communications protocol.

Although the TIP offers flexibility for 2PC protocol-based short-lived transactions, it falls short in the case of long-lived business transactions. Business transactions consist of a number of component transactions with largely different response times, thus blocking resources controlled by short- lived transactions for unacceptably long periods of time, rendering them unable to process new service requests. This is an undesirable option from an autonomous service provider's point of view.

3.1 Nature and Characteristics

To address the shortcomings of the TIP and at the same time provide support for business web-service enabled transactions, OASIS has introduced the concept of the Business Transaction Protocol (BTP) [1]. The BTP is designed to provide a basic framework for providing transactional coordination of participants of services offered by multiple autonomous organisations that use XML (WSDL) to exchange data.

BTP is a core-level transaction protocol that contains a set of specific messages that get exchanged between systems supporting a business application in order to achieve interoperability between web-service enabled transactions. However, as already proposed by the TIP, prior to performing a business transaction interacting systems need to exchange application-specific messages to affix tentative commitments to the terms of an interaction, e.g., price, quantity, delivery time, etc. To exemplify this, consider a manufacturing company that orders the parts and services it needs on-line. This manufacturer may have relationships with several parts suppliers and several providers of services such as shipping and insurance. Thus, within this example, the manufacturer's and a chosen supplier's applications, could exchange XML messages detailing what the goods are, their quantity, price and so on. After these terms are accepted the applications will start exchanging BTP messages implementing the business transaction. The parts of the business application (in both interacting systems) that handle these different sets of messages need to be distinguished. Therefore, for each party intending to use BTP-based messages to implement its business transactions it would be useful to distinguish between the following elements:

1. The *business application communication protocol* element (pre-transaction phase) that exchanges meaningful business terms such as order information and is the prelude to performing associated business functions. Automating the exchange of critical information prior to the actual business transaction decreases the impact of out-of- date data, reduces the potential for cancellations, and improves the odds of successfully completing transactions.
2. The *main-transaction* or *BT-based element*, which sends and receives the BTP messages, performs specific roles in the protocol (refer to the following subsection). BTP elements assist the application in executing business transactions and getting the actual work of the application done.
3. The *post-transaction* element that is responsible for observing the agreements and terms stipulated during the execution of a BT. This element corresponds, broadly speaking, to contract fulfilment phase.

In the following we examine briefly the W3C Tentative Hold Protocol (an application communication protocol) and subsequently introduce elements of the OASIS Business Transaction Protocol.

3.2 The W3C Tentative Hold Protocol

The objective of Tentative Hold Protocol (THP) is an effort to facilitate automated coordination of multi- business transactions [9]. Tentative Hold Protocol is an open, loosely coupled messaging-based protocol for the exchange of information between trading partners prior to the actual transaction itself. This framework includes a standard way for trading partners to exchange tentative commitments to the terms of an interaction, e.g., price, quantity, delivery time, and so on, and update each other when the situation changes.

THP defines an architecture that allows tentative, non- blocking holds or reservations to be requested for a business resource, e.g., the items for sale from an online retailer. In this example of online ordering, these lightweight reservations are placed prior to the sale, allowing multiple clients to place holds on the same item (thus non-blocking). Whenever one client completes the purchase of the item the other clients receive notifications that their holds are no longer valid. This provides flexibility for both ends of the transaction. The clients have the ability to request tentative reservations on the resources that they want to acquire as a single coordinated purchase, verifying price and availability before completing the transactions. If one item becomes unavailable (purchased by some other client), this client has the ability to replace it with another item without losing any holds on other items already reserved. The vendors grant non-blocking reservations on their products, retaining control of their resources, while allowing many potential customers greater flexibility in coordinating their purchases.

There is a THP coordinator on both the client and resource owner side, responsible for communicating various messages such as hold requests, cancellations, and so on. Additionally, a resource owner provides a business rules engine with which the resource side THP coordinator communicates in order to handle business rule specific actions. This gives the resource owner the possibility of providing targeted customer service with the granting of holds, specifying greater or lesser hold expirations for a given hold request, as well as the potential for notifying valued clients when some resource is being reserved by another client - allowing the preferred client the opportunity to lock in their purchase first.

Tentative Hold Protocol and Compensating Transactions. The management of cross-organisation business transactions often involves the use of compensating transactions. Under normal circumstances individual steps in a business transaction are completed immediately. However, if one of the business transaction steps fails, the other steps in the transaction are undone using a compensating transaction. A new transaction is issued to compensate for the failed transaction, e.g., an order cancellation transaction would compensate for the order placement transaction.

Not all business transactions may be cleanly and completely reversable. Consider, for example, a situation where an item was purchased across international borders. Such a purchase could conceivably involve fluctuating currency exchange rates. A compensating transaction for a cancelled sale may not result in transferring the same amount of money should the exchange rate fluctuate between the original transaction and the compensating transaction. Compensating transactions can be kept to a minimum by using the THP.

Adding a tentative hold phase for business application elements provides the following benefits [9]:

- Minimises compensations. Consider, for instance, a client application which attempts to coordinate the purchase of an item X from company A and item Y from company B. Without THP, the client application would most likely place an order for item X from company A, and then try to purchase item Y. If the application is unable to purchase item Y, then it would compensate for its earlier action in its business logic by cancelling the order with company A. However, if tentative holds were used, the client application would place tentative holds on both the items thus ensuring their availability, and thereafter complete both purchases. In this case, the application is far less likely to cancel, having determined that both X and Y are available at an acceptable price before placing any orders.
- Reduces lag between original transaction and compensation. From the previous example, it is clear that the lag time between a purchase and a compensating transaction would be reduced as the client application acquires tentative commitments from all potential business partners before it issues any actual business transactions with any one of them.

Tentative Hold Protocol and Two Phase Commit. Introducing a tentative hold phase prior to the 2PC protocol – for small scale interactions, i.e., atomic transactions, where resource locks can be tolerated – provides the following benefits [9]:

– Shortens the required 2PC lock duration by minimizing the time spent in the prepare phase since it allows applications to obtain tentative commits and make all decisions before they enter the prepare phase of a 2PC commit transaction.
– Minimises the likelihood of rollbacks since it allows the customers and vendors to exchange information on the terms they could commit to, e.g., price and quantity, and keep each other up-to-date about any change in status through proactive notifications.

3.3 The OASIS Business Transaction Protocol

The Business Transaction Protocol, BTP, is a Committee Specification of the Organisation for the Advancement of Structured Information Standards (OASIS) [1]. BTP is an interoperation protocol that defines how transactional services behave, and what messages are passed between those services during a transaction.

BTP is based on two-phase commit for small-scale (short duration) interactions known as *atoms*, which can be aggregated into larger non-ACID transactions known as cohesions [2]. BTP atoms are the atomic transactions introduced in section-2 and in that respect they posses full ACID properties and are coordinated by a standard two- phase commit protocol.

The BTP cohesion (refer to section-2 for a definition) is a transaction that is run by a *voting/enrolment process* where the client application of the transaction has the final approval or rejection vote. The client can apply business rules to its decision-making process in full light of the recommendations made by all of the atoms in the transaction. For cohesions a relaxed form of the two-phase commit protocol is used. This is illustrated in Figure 3 and will be explained later in this section.

The BTP should be implemented on the sites of all trading partners. To coordinate interaction the BTP defines the roles that trading partner applications may perform in a BTP-based interaction. It also introduces the messages, e.g., enrol, vote, prepare, etc, that pass between such actors and the obligations upon and commitments made by actors-in- roles. There are a variety of roles used in this specification:

1. A transaction is always initiated by an application of a trading partner (*initiator*). The initiator sends application messages to a web-service in order to invoke its operations.
2. A *transaction coordinator* is a software component that implements the BTP and can decide about the outcome of a single atomic transaction. It enlists and de-lists participants in a transaction and participates in the transaction execution protocol. A coordinator instructs participants to prepare, cancel, and/or confirm, see Figure 1. A transaction participant returns a successful message to a prepare instruction if it is capable of confirming or cancelling the set of operations it participates in an atomic transaction, i.e., first phase of a 2PC protocol. The cancel message is essentially the rollback operation of an atomic transaction. The confirm message instructs participants to make their current set of operations permanent, i.e., commit. The coordinator makes its

decisions based on input from BT participants and the BT initiator. As BTs are nested there might be multiple coordinators in a BT. These can play the role of a main or subordinate coordinator. There is only one main coordinator in a BT. The main coordinator drives the execution/termination protocol for that BT. The subordinate coordinators cooperate with the main coordinator for terminating a transaction with success, failure or timeout.

3. The applications of trading partners that take part in a transaction are called *participants*. These are capable of executing prepare, cancel and/or confirm operations issued by a coordinator. A participant is capable of sending vote and prepare messages to a coordinator, see Figure 1. A participant sends a vote message to a coordinator usually in response to a prepare message. A participant can vote to cancel, ready, ready with inability to cancel after timeout, or ready with cancel after timeout. An enrol message is sent from participant to a coordinator when a participant has a set of operations that it wants to participate in a service atom. A resign message is sent from a participant to a coordinator to indicate that the service should no longer be part of this atomic transaction. These signals are used by the application(s) to determine whether to confirm or cancel the results of application operations.

Since multiple application components and resources participate in a BT, it is necessary for the transaction coordinator to establish and maintain the state of the BT as it occurs. This is achieved by using a transaction context, which is a data structure propagated onto messages passed between the BT initiator and the services within a business transaction. It specifies the type of a transaction atomic transaction or cohesion and identifies it as a superior - containing both addressing information and the identification of the relevant state information. The business transaction context is unique to a transaction instance and it contains information that helps the actors to execute the BTP. The context also indicates whether this superior will behave atomically or cohesively with respect to its inferiors[4].

The propagation of the business transaction from one party to another, to establish the *superior:inferior* relationships involves the transmission of the transaction context. This is shown in Figure 2.

The Cohesion Protocol. The BTP uses a variation of the two-phase commit protocol, where participating resources are allowed to pre- commit their sub-transaction and apply a compensating action in case the main transaction terminates with a failure. First the application elements exchange messages that determine the characteristics and cause the execution of the provisional effect; then they send a separate reply message, to the BTP element, asking for confirmation or cancellation, Figure 3. This comprises the following steps [1]:

[4] The BTP distinguishes between superiors that treat their inferiors in an atomic or cohesive fashion. The former are called (atom) coordinators, while the latter are called (cohesion) composers. . we will use the term coordinator in the broad sense to encompass both types of superior behaviour.

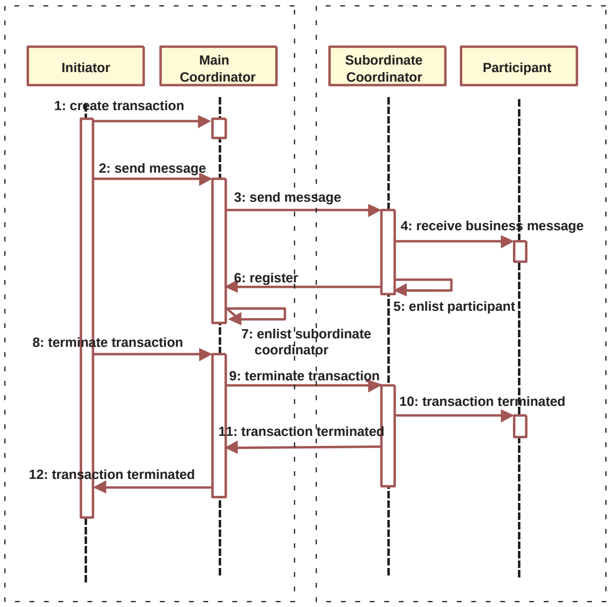

Fig. 2. Actors that can be involved in a business transaction.

– The coordinating entity decides to terminate the BTP in which case it needs to determine whether the BT participants are able either to confirm or cancel their respective operations, by sending them `prepare` messages.
– The participants report their ability to confirm-or-cancel (their preparedness) to the coordinating entity. After receiving these reports, the coordinating entity:
  1. determines which of the systems should be instructed to confirm and which should be instructed to cancel
  2. informs each system whether it should confirm or cancel by sending a message to its BTP element.
  3. The coordinating entity returns a set of results to its superior entity (the client application). The client application may decide to confirm the cohesion even in the case that some of the atoms have chosen to cancel rather than confirm.

In most standard 2PC-based systems the coordinating entity automatically commits (confirms) if all the participants vote ready. The BTP deliberately hands the decision up to the initiator (client) application. This allows the initiator to make complex decisions about the outcome of the atomic transaction (confirm, cancel). These decisions are based on business rules and other (application-related) service execution outcomes.

The two-phases of the BTP protocol ensure that either the entire attempted transaction is abandoned or a consistent set of participants is confirmed. Note

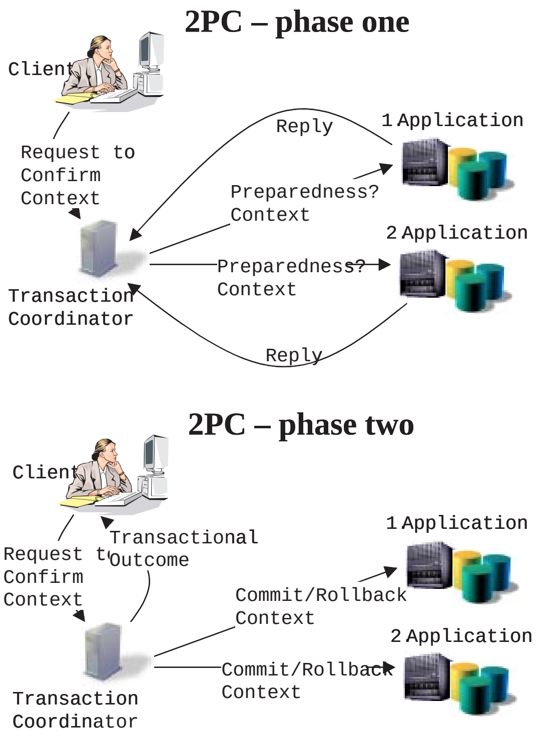

Fig. 3. The relaxed two-phase commit protocol for cohesions.

that this 2PC variant is a consensus protocol between parties and does not require two-phase locking (within the participants).

The following code fragment taken from [15] illustrates how a cohesion might look to a Java initiator. Note an API like this is not part of BTP specification, the code is used for illustrative purposes only.

```
void cohesionComposer() // an application method {
Atom orderGoods = new Atom();
Atom shippingViaGoodsSupplier = new Atom();
Atom shippingFromAnotherSource = newAtom();
// application work
Quote quoteForGoods =
    orderGoods.sendApplicationMessage ("quoteForGoods", arg, arg ...);
Quote quoteForShippingViaGoodsSupplier =
    orderGoods.sendApplicationMessage ("quoteForShipping", arg, arg ...);
Quote quoteForShippingFromAnotherSource =
    orderGoods.sendApplicationMessage ("quoteForShipping", arg, arg ...);
// ensure that the quotes are guaranteed (may be folded into app messages)
orderGoods.prepare();                // no exception, so it is ready
shippingViaGoodsSupplier.prepare();  // ditto
shippingFromAnotherSource.prepare(); // ditto
orderGoods.confirm();
QuotesOutcome quotesOutcome
  = this.decideQuotesOutcome (quoteForShippingViaGoodsSupplier,
                              quoteForShippingFromAnotherSource);
quotesOutcome.selected().confirm();
quotesOutcome.rejected().cancel();
}
```

4 Web-Services and Workflows: The Web-Services Flow Language

When looking at web-services, it is important to differentiate between the baseline specifications of SOAP, UDDI and WSDL that provide the infrastructure that supports the publish, find and bind operations in the service- oriented architecture and higher-level specifications. Higher-level specifications provide functionality that supports and leverages web-services and enables specifications for business process modelling and automation.

There are several initiatives that aim to define and manage business process activities and business interaction protocols comprising collaborating web-services. Among them is the Web Services Flow Language (WSFL), and XLANG. In the following we will give a brief overview of the capabilities of WSFL as a BTP could easily be attached to the service provider invocations (activities) within a business process defined in WSFL and transaction demarcation could be applied to the control link semantics as an extension of WSFL.

The Web Services Flow Language (WSFL) focuses on processing a core model for Workflow [10]. WSFL is an XML language for the description of web-service compositions as part of the definition of a business process. It also defines a public interface that allows business processes themselves to be defined as web-services. WSFL allows for added-value functions to be created as an aggregation of services from multiple service providers. WSFL extends the web-services architecture by providing the ability to describe compose web-services defined by WSDL into workflows by means of a *flow model.* Flow models are particularly useful when modelling business processes based on web-services. Defining the flow of control and data between web- services specifies execution orders. The power of WSFL lies in its ability to model business processes that span technology as well as business boundaries - a limitation that most workflow engines suffer from.

The unit of work in WSFL is an activity. An activity represents a business task that must be performed as a single step to completion within the context of a business process. Every activity defined in the WSFL flow model is implemented in the form of a web- service defined on the basis of WSDL. An activity has a signature that is related to the signature of the operation that is used to implement the activity. WSFL models the basic business process in the form of a directed acyclic graph that uses simple directed edges that control the flow of processing logic from one activity to the next.

A data link specifies that its source activity passes data to the flow engine as part of the context of some process instance. The data link also enables the specification of a mapping between a source and a target document, if necessary.

The WSFL *global model* provides a simple composition meta-model that describes how messages can be sent between the web-services in the flow model as the flow is executed [14]. The global model describes how the web-services involved in the business processes of the flow model are expected to interact with one another. A plug link identifies pairs of operations that communicate with each other, and describes which operation initiates this communication.

We summarise this discussion by examining how the WSFL specification addresses business process semantics, workflow and transactional properties, and collaborative agreements.

– Collaboration-based process modelling: WSFL describes processes as interactions between web-service providers, which can be abstracted using roles so collaboration-based process modelling tools could certainly be used to generate WSFL descriptions.
– Workflow support: In WSFL, the flow model defines the workflow associated with each service provider (collaboration role).
– Business transaction support: In WSFL there is no explicit mention of transactions. WSFL currently does not support transactions.
– Collaborative agreements: In a separate development IBM is currently at work expanding on XML protocols with development of a new, proposed standard – Trading Partner Agreement Markup Language (tapML) [17]. The foundation of tpaML is the Trading Partner Agreement (TPA) A TPA is an electronic contract that uses XML to stipulate the general terms and conditions, participant roles, e.g., buyers and sellers, communication and security protocols, and a business protocol (such as valid actions and sequencing rules). Although TPAs are not part of WSFL, WSFL global models give a foundation that could be used for supporting such business agreements.
– Support for BTP: BTP could be supported by the service provider invocations (activities) within a business process defined in WSFL, while transaction demarcation could be applied to the control link semantics [16].

5 Transactional Conversation Sequences

The objective of transactional conversations is to introduce a standard way of describing BT-based conversations and thereby support interactions between services. ebXML is a major initiative that ascribes to this principle by means of its Business Process Specification Schema (BPSS) [13] whose aim is to support the specification of business transactions and their choreography into business collaborations.

An *ebXML business transaction* represents business document flows between requesting and responding partners. In any BT there always is a requesting business document, and optionally, a responding business document. Each business transaction can be implemented using one of many available standard UN/CEFACT Modelling Methodology (UMM) patterns, which determine the actual exchange of business documents and business signals between trading partners to achieve the required electronic business transaction.

An ebXML *business collaboration* specifies as part of a Collaboration Protocol Agreement (CPA) an "agreement" between two or more business partners. A business collaboration is essentially the specification of business transaction activities between the two partners, their associated document flow, and the choreography of these business transaction activities. The business collaboration specifies all the business messages that are exchanged between two trading

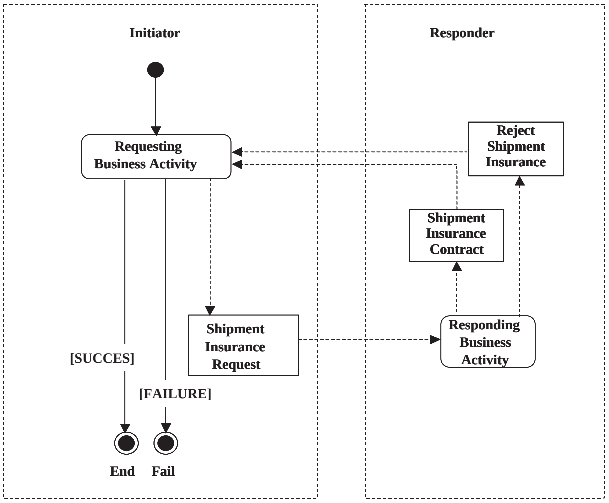
Initiator
Responder
Requesting Business Activity
Reject Shipment Insurance
Shipment Insurance Contract
Shipment Insurance Request
Responding Business Activity
[SUCCES]
[FAILURE]
End
Fail

Fig. 4. ebXML business transaction representation with a UML activity diagram.

partners, their content, and their precise sequence and timing. This part of the "agreement" provides a shared understanding of the interaction. An ebXML collaboration is conducted by two parties, each using a human or an automated business logic system that interprets the documents transmitted and decides how to (or whether to) respond. All collaborations are composed of combinations of *atomic transactions*, each between two parties. Multi- party arrangements must be decomposed into bilateral transactions. The sequencing rules contained in a collaboration definition are not between messages but between business transaction activities.

5.1 Business Collaboration Choreography

Business transaction definitions may be represented as UML activity diagrams. The UML notation represents the requesting and responding roles as swim lanes, the exchange of documents as object flows, and the start and end of the business transaction. An example of this is illustrated in Figure 4, where a party requesting to insure shipment of an object issues an insureShipment transaction, where the transaction will reach a success or a fail state based on the result of this request.

A collaboration and its sequencing rules are also represented with a UML activity diagram such the one illustrated in Figure 5. An activity within a swim lane indicates that this business transaction activity is initiated by the corresponding role.

The BPSS specification defines five states: start, success, failure, fork and join, which can be interleaved with the business transaction or collaboration activities. These states are known as pseudo-states as they have the same semantics as a state and are identical across collaboration definitions.

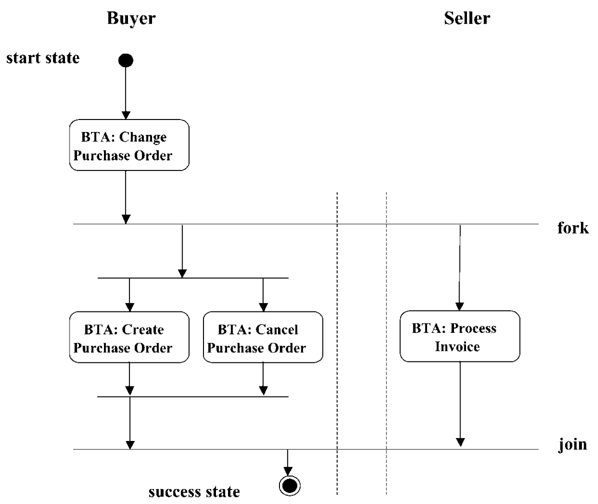
Buyer
Seller
start state
BTA: Change Purchase Order
fork
BTA: Create Purchase Order
BTA: Cancel Purchase Order
BTA: Process Invoice
join
success state

Fig. 5. ebXML collaboration activity diagram.

We summarise this discussion by examining how the BPSS addresses business process semantics, workflow and transactional properties, and collaborative agreements.

- Collaboration-based process modelling: BPSS describes public processes as collaborations between roles, with each role abstractly representing a trading partner. There are two types of collaborations: binary collaborations and multi-party collaborations. Multi-party collaborations are decomposed to binary collaborations.
- Workflow support: BPSS workflow is described by assigning a public control flow based on UML activity graph semantics to each binary collaboration. The control flow describes the sequencing of business transactions between the two roles.
- Business transaction support: BTs within the BPSS are applied to the semantic business level with a simplistic protocol defined for the interaction between two parties (requesting and responding) and determination of success or failure of the transaction. A business transaction consists of a request and optionally a response. Each request or response may require that a receipt acknowledgement be returned to the sender. For contract-forming transactions such as purchase order requests an acceptance acknowledgement may need to be returned to the requester. Time constraints can be applied to the return of responses and acknowledgements. Transactions are not nested and there is no support for specifying compensating transactions. No support for atomicity types is provided other than service request atomicity.
- Collaborative agreements: A BPSS process model can be referenced in an ebXML collaboration protocol agreement (CPA).

5.2 BPSS and the BTP

The following are a high level comparison of properties of business transactions defined in ebXML and BTP and is based on an analysis presented in [16]:

– ebXML business transactions are pre-defined re-usable interactions that include one or two business documents exchange, and one or more signals that indicate the state changes in the transaction. BTP, on the other hand, allows any BTP aware participant/service to be part of the coordinated transaction. The ebXML business transactions are atomic in nature and thus comparable to the atoms of BTP.
– ebXML business transactions currently are only between two roles - requesting and responding partners. BTP transactions have no limitation of number of participants that take part in the transactions.
– ebXML business transactions may be based on UMM [8] transaction patterns. A transaction pattern is immaterial for BTP since coordination is governed by protocol not by a pattern of message exchanges.
– The semantics of an ebXML transaction is enforced by the Business Service Interface (BSI). With the BTP a service participating in a transaction enforces the semantics of functionality, while the protocol itself supports recovery of the transaction.
– There is no support for 2PC in ebXML transactions while BTP supports an extension of the 2PC protocol.

The ”open top” coordination capabilities that BTP offers could be used to prepare multiple binary collaborations and then decide to confirm only a subset – thus allowing ebXML to support not only atomic type BTs for binary and multi-party collaborations but also cohesive transactions for multi-party collaborations.

6 Concluding Remarks

Web services and businesses-to-business collaborative systems are becoming the predominant methods of creating business applications. Process oriented workflow systems and e-business applications require transactional support in order to orchestrate loosely coupled services into cohesive units of work and guarantee consistent and reliable execution. In this paper we addressed the issues relating to the use of business transactions in web-service based applications, introduced two forms of business transactions and presented a taxonomy of e-business transaction features such as unconventional atomicity criteria, the need for support for business conversations and the need for distinguishing between three business transaction elements: the business applications communication protocol element, the main-transaction element and the post-transaction element. We have also shown that a flexible and extensible framework such as the Business Transaction Protocol (BTP) proposed by OASIS is necessary for building robust and extendible e-business applications. Finally, we introduced standard workflow web-service based initiatives, viz. the WSFL, and the Business Process Specification Schema (BPSS) of ebXML that enables the description of the public interface of e-business processes. We subsequently compared their operational characteristics against those of the proposed taxonomy.

Recently, there have been several approaches to provide support for business web-service enabled transactions in the research field. These include among other correlation mechanisms for managing multiple service conversations [4], protocols for conversational transactions [6], WSDL extensions to describe implicit transactional semantics found in business applications [8] and extensions of the two-phase commit protocol with a transactional messaging infrastructure [7]. Out of these the most notable is the work reported in [8] where the authors describe a framework that introduces transactional attitudes as extensions of WSDL to enable web-service providers to declare their individual transactional capabilities and semantics, and web-service clients to declare their transactional requirements in terms of provider supplied services.

References

1. OASIS Committee Specification "Business Transaction Protocol", version 1.0, May 2002.
2. J. Webber et. Al. "Making web services work", Application Development Advisor, Nov. Dec 2001, pp. 68-71.
3. K. Evans, J. Klein, J. Lyo, "Transaction Internet Protocol - requirements and supplemental information", 1998, http://www.landfield.com./rfcs/rfc2372.html
4. A. Sahai, J. Ouyang, V. Machiraju, K. Wurster, "End-to-End E-Service Transaction and Conversation– Management through Distributed Correlation", HP Laboratories Palo Alto, HPL-2000-145, September 2000.
5. J.D., Tygar "Atomicity in Electronic Commerce", ACM-Mixed Media, Apr. 1998.
6. J. Ouyang, A. Sahai, V. Machiraju, "An Approach to Optimistic Commit and Transparent Compensation for E-Service Transactions", HP Laboratories Palo Alto, HPL-2001-34, February 2001.
7. S. Tai, T. Mikalsen, I. Rouvellou, S. Sutton "Dependency Spheres: A Global Trnsaction Context for Distributed Objects and Messages", 5th Int'l Enterprise Distributed Object Computing Conference (EDOC), September 2001.
8. T. Mikalsen, S. Tai, I. Ravellou, "Transactional Attitudes: Reliable Composition of Autonomous Web Services", Workshop on Dependable Middleware based Systems, March 2002.
9. J. Roberts, S. Krisnamurthy, "Tentative Hold Protocol "W3C Workshop on Web Services, November 2001, http://www.w3.org/TR/tenthold-1.
10. F. Leymann, "Web Services Flow Language (WSFL 1.0), May 2001 http://www-4.ibm.com/software/solutions/webservcies/pdf/WSFL.pdf
11. S. Thatte, "XLANG - Web Services for Business Process Design", Microsoft Corporation http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm
12. S. Frolund, K. Govindarajan "Transactional Conversations", W3C web Services Workshop, http://www.w3.orgt/2001/03/WSWS-popa/, July 2001.
13. BPSS - "Business Process Specification Schema", ebXML Business Process Project Team, May 11, 2001 www.ebxml.org/specdrafts/cc_and_bp_document_overview_ver_1.01.pdf

14. J.Snell "Introducing the web services flow language", IBM developer works, June 2001, http://www-106.ibm.com/developersworks/library/
15. A. Green, P. Furniss, "Scope and Requirements, Actors and Terminology", Choreology Ltd, May 2001.
16. M. Potts, S. Temel "Business Transactions in Workflow and Business Process Management", OASIS Business Transactions Technical Committee Workflow sub- committee, Dec. 2001.
17. M. Sachs, et. al. "Executable Trading-Partner Agreements in Electronic Commerce", IBM T.J.Watson Research Center, 2000.
18. M. P. Papazoglou, A. Tsalgatidou, J Yang "The Role of eServices and Transactions for Integrated Value Chains", IDEA Publishers, 2002.
19. J. Yang and M. Papazoglou. "Interoperation Support for Electronic Business". Communications of the ACM, Vol. 43, no. 6, pp. 39-47, June 2000.