SOCA (2007) 1:51–63
DOI 10.1007/s11761-007-0002-3

ORIGINAL PAPER

Web services technology in support of business transactions

Michael P. Papazoglou · Benedikt Kratz

Received: 29 October 2006 / Revised: 9 January 2007 / Accepted: 13 January 2007 / Published online: 1 March 2007

Abstract Advanced business applications typically involve well-defined business functions such as payment processing, shipping and tracking, determining new product offerings, granting/extending credit, managing market risk and so on. These reflect commonly standard business functions that apply to a variety of application scenarios. Although such business functions drive transactional applications between trading partners they are completely external to current Web services transaction mechanisms and are only expressed as part of application logic. To remedy this situation, this paper proposes a business-aware Web services transaction model and support mechanisms, which is driven by common business functions. The model allows expressing business functions such as payment and credit conditions, delivery conditions, business agreements stipulated in SLAs, liabilities and dispute resolution policies. It allows blending these business functions with QoS criteria such as security support to guarantee integrity of information, confidentiality, and non-repudiation.

Keywords Web services transactions · Business transactions · Business functions · Business protocols · Business process orchestration · Business process integration · QoS principles

Part of this research reported is funded by the Dutch Organization for Scientific Research (NWO) under the project eXecution of Transactional Contracted Electronic Services (XTC), project No. 612.063.305.

M. P. Papazoglou · B. Kratz (✉)
INFOLAB, Tilburg University, Dept. of Information Systems & Management, P.O. Box 90153, 5000 LE Tilburg, The Netherlands
e-mail: B.Kratz@uvt.nl

M. P. Papazoglou
e-mail: mikep@uvt.nl

1 Introduction

When business processes span organization boundaries they pose a number of significant business and technology challenges. First, trading partners need to agree upon explicit and unambiguous standards that specify precisely the data and common business documents, such as purchase orders and invoices, which the disparate systems can exchange. Second, and, more importantly, they require loose coupling on the basis of precise business interaction protocols. Such protocols are by necessity message-centric: they specify the flow of messages representing business activities among trading partners (without requiring any specific implementation mechanism). Collectively, business process protocols and associated data format and message exchange standards provide the means for automated, system-to-system exchange of data and messages between trading partners. These requirements are common to all applications that involve business processes which span organizational departments and boundaries.

In environments involving business collaborations, business processes are increasingly complex and integrated both within internal corporate business functions (e.g., manufacturing, design engineering, sales and marketing, and enterprise services) and across the external supply chain. In such environments there is a clear need for advanced business applications to coordinate multiple Web services into a multi-step business transaction. This requires that several Web service operations or processes attain transactional properties reflecting business semantics, which are to be treated as a single logical (atomic) unit of work that can be performed as part of a business transaction. For example, consider, a manufacturer that develops Web service based solutions to automate the order and delivery business functions with its suppliers as part of a business transaction. The transaction between the manufacturer and its suppliers may only be considered as successful once all products are delivered to their final destination, which could be days or even weeks after the placement of the order, and payment has ensued.

In contrast to Web service transactions, which are driven by purely technical requirements such as coordination, data consistency, recovery, and so on, business transactions are driven by economic needs and their objective is accomplished only when the agreed upon conclusion among trading parties is reached, e.g., payment in exchange for goods or services.

This approach requires distilling from the structure of a business collaboration the key capabilities that must necessarily be present in a business transaction and specifying them accurately and independently of any specific implementation mechanisms. The business transaction then becomes the framework for expressing detailed operational business semantics.

Conventional approaches to business transactions, such as Open EDI, the UN/CFACT Modeling Methodology (UMM) and ebXML [1], focus only on the documents exchanged between partners, rather than coupling their application interfaces, which inevitably differ. The core idea behind this approach is to define a library of standard electronic XML business documents such as invoices, purchase orders, and ship notices—possibly described in the Universal Business Language (UBL) [2] — to provide an intuitive framework for specifying the business logic and computations that take place on each end of a document exchange. For example, if a customer sends a purchase order to manufacturer, which the manufacturer can fulfill, it will then respond with an invoice and a shipping notice. How such documents are produced and what actions result when they are consumed is strictly up to the business at each end of the document exchange.

The previous approach to business transactions focuses only on data objects, i.e., documents exchanged between partners, and ignores important constituents in a business transaction such as business operations and their behavioral semantics. A more natural approach to business transactions is to make common business operational requirements and operational level relationships between trading partners first class tenets in a business transaction. This requires providing a common core of well-understood business operational principles (or business transaction functions) such as ordering, transport, distribution and payment that can be rationalized and appropriately combined across any supply chain to create semantically enhanced business transactions. Developers can then build transactional applications by using, combining, and, possibly specializing, these constructs in a similar way that abstract data types are used in programming languages.

This paper focuses on introducing an advanced business transaction model and on providing operational business principles for specifying business transactions along with their QoS characteristics. The approach taken mimics business operational semantics and does not depend upon underlying technical protocols and implementations. The paper also presents a Business Transaction Model Language (BTML) that is used at design time to specify the elements of a business transaction. Run-time support for this environment is provided by conventional Web services standards such as BPEL, WS-Coordination and WS-Transaction and will be briefly highlighted in the context of a reference architecture. Detailed descriptions of the run-time environment as well as run-time transformations between the BTML and equivalent constructs supported by Web services transaction standards are outside the scope of this paper.

2 Business collaborations and transaction support

One key requirement in making cross-enterprise business process automation happen is the ability to describe the collaboration aspects of the business processes, such as commitments and exchange of monetary resources, in a standard form that can be used by applications and consumed by tools for business process implementation and monitoring.

Business collaborations are usually expressed with reference to business entities that are affected by a business collaboration activity, e.g., order, goods transfer, and in terms of events that trigger the state transitions of business entities and of the business collaboration, e.g., delivery of goods triggers the transition of order line status from pending to fulfilled. Business collaborations need to capture the information and message exchange requirements between trading partners, identifying amongst other things the timing, sequence and purpose of each business collaboration and information exchange. This becomes the responsibility of a business protocol that defines the ordering in which a particular partner sends messages to and expects messages from its partners based on an actual business context.

A *business protocol* captures all behavioral aspects that have cross-enterprise business significance. Each participant can then understand and plan for conformance to the business protocol without engaging in the process of human agreement that adds so much to the difficulty of establishing cross-enterprise automated business processes.

A business protocol may exhibit the flowing characteristics:

1. It invariably includes data-dependent behavior. For example, a supply-chain protocol depends on data such as the number of line items in an order, the total value of an order, or a deliver-by deadline. Defining business intent in these cases requires the use of conditional and time-out constructs.
2. It is able to specify exceptional conditions and contingency measures and their consequences including recovery sequences.
3. It is able to specify cross-partner coordination of activity outcomes (success or failure) at various levels of granularity.

A business protocol helps to choreograph the flow among business interactions. This flow depends on the states of business entities and/or the business collaboration. Business interactions are driven by significant business events and represent an atomic unit of work in a trading arrangement between two or more business partners, which is typically associated with the formation of contracts or agreements and expressed in terms of a transactional business process (or business transaction).

A business transaction is defined as an atomic business process describing a trading interaction between possibly multiple parties that strive to accomplish an explicitly shared business objective [3]. This shared business objective extends over a possibly long period of time and is terminated successfully only upon recognition of the agreed conclusions between the interacting parties. A business transaction is driven by well-defined business tasks and events that directly or indirectly contribute to generating economic value, such as processing and paying an insurance claim, and has also an associated number of quality of service (QoS) parameters that represent security and timing requirements. A business transaction always either succeeds or fails with respect to the business task (function) that initiated it and governs it throughout its execution. If a business transaction completes successfully then each participant will have made consistent state changes, which, in aggregate, reflect the desired outcome of the multi-party business interaction.

A business transaction is made up of a requesting (initiating) business activity performed by an initiating partner (party) and a responding business activity performed by the responding business partner. The initiating business activity sends a business document to a responding business activity that may return a business signal (signifying the completion of an activity) and possibly a business document as the last responding message. A transaction is associated with an SLA that describes the agreed upon QoS requirements and usually outlines what each party can do in the event the intended actions are not carried out (e.g., promised services not rendered, services rendered but payment not issued).

3 The business transaction model

An important requirement in making cross-enterprise business process automation happen is the ability to describe the collaboration aspects of the business processes, such as business commitments, mutual obligations and exchange of monetary resources, in a standard form that can be consumed by tools for business process implementation and monitoring. This gives raise to the concept of a *business transaction model* that encompasses a set of business transaction functions and several standard business primitives and conventions that can be utilized to develop complex business applications involving transactional and monetary exchanges.

Central to the business transaction model is the notion of a business transaction (see previous section for a definition). Business transactions cover many domains of activity that businesses engage in, such as request for quote, supply chain execution, purchasing, manufacturing, and so on. The purpose of a business transaction is to facilitate specifying common business procedures and practices in the form of business application scenarios that allow expressing business operational semantics and associated message exchanges as well as the rules that govern business transactions. Such rules include operational business conventions, agreements, and mutual obligations. The combination of all these factors characterizes the nature of *business relationship*s among the parties involved in a business transaction. It enforces trading parties to achieve a common semantic understanding of the business transaction and the implications of all messages exchanged.

The business transaction is initiated by a single organization and brings about a consistent change in the state of a business relationship between two or more trading parties. A business relationship is any distributed state held by the parties, which is subject to contractual

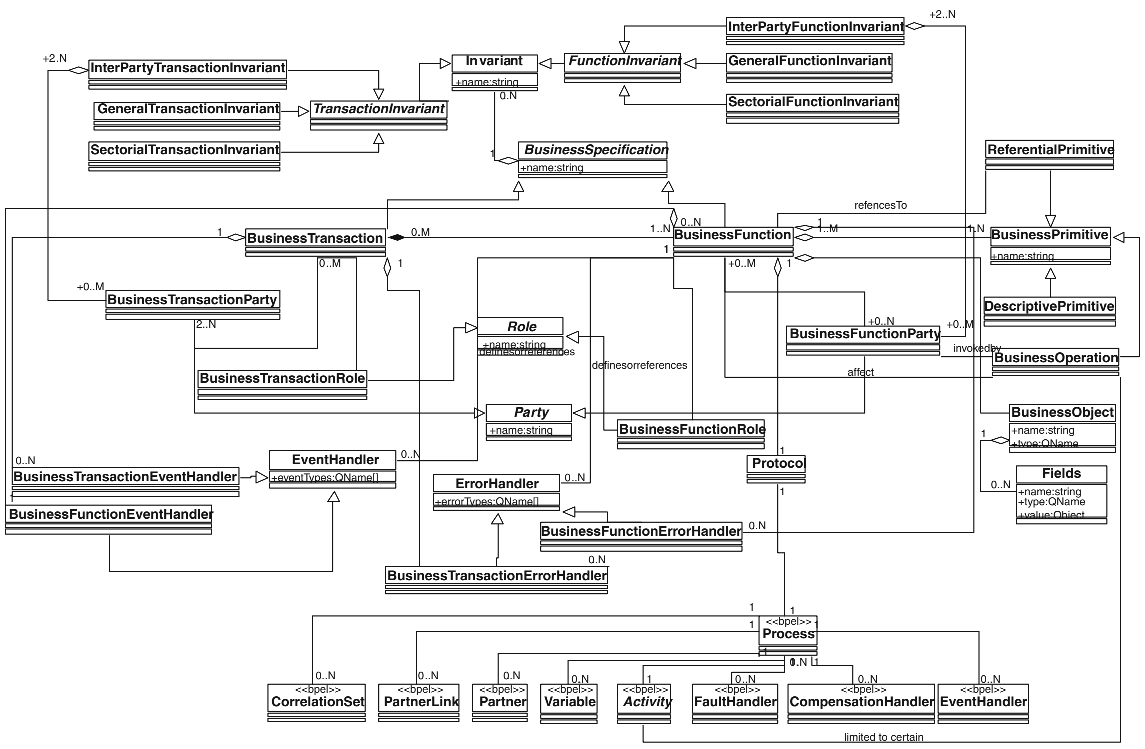


Fig. 1 Business transaction meta-model in UML

constraints agreed by those parties. A business transaction needs to express features like the parties that are involved in the transaction; the entities under transaction; the destination of payment and delivery; the transaction time frame; permissible operations; links to other transactions; receipts and acknowledgments; and finally, the identification of money transferred outside national boundaries.

The previous description of a business transaction has been derived from (classical) commerce models and serves as a common high-level, non-technical view of how business organizations interact with each other. The definition emphasizes the operational business view of a transaction. There are four key components in a business transaction model that help differentiate it from (general) message exchange that business processes involve. These are:

1. commitment exchange;
2. the party (or parties) that has the ability to make commitments;
3. business constraints and invariants that apply to the message exchanged between the interacting parties; and
4. business objects (documents) that are operated upon by business activities (transactional operations) or by processes.

These terms are introduced and explained below, while Fig. 1 represents them and their inter-relationships in UML.

A *commitment exchange* occurs between two or more interacting *parties* and concerns tasks or functions to be carried out and usually involves formal trading partner agreements that could be expressed in terms of business protocols such as RosettaNet PIPs or ebXML Business Process Specification Schema (BPSS) [4]. A commitment exchange identifies such things as the overall business process, the partner roles, the business documents used, message and document flow, legal aspects, security aspects, business level acknowledgments and status, and so on. Partners inside a transaction have distinct roles (such as *buyer* and *seller*) and the ability to make commitments, being held responsible for, having rights and obligations, in the context of the business transactions. One party can act as the initiator of the transaction while the others can act as responders.

A *business transaction constraint* is defined as an explicitly stated rule that prescribes, limits, or specifies any aspect of a business transaction that forms part of the commitment(s) mutually agreed to among the interacting parties. *Business invariants* are constraints external to constraints agreed by interacting parties in a transaction and include universal legal requirements, commercial and/or international trade and contract terms, public policy (e.g., privacy/data protection, product or service labeling, consumer protection), laws and regulations that are applicable to parts of a transaction. Invariants ensure the nature of the business transaction and/or the goods or services delivered while guaranteeing that no regulations are compromised. Business invariants are universal (or horizontal) in nature and apply regardless of the type of business or sector within which the business occurs. There are, however, constraints external to parties that are of a sectorial nature called *sectorial invariants* which can be found in sectors such as telecommunications, transportation and delivery, financial/banking, and so on. Universal and sectorial invariants can be combined with inter-party business constraints for building application use scenarios. It is important to understand that in such situations invariants take precedence over internal constraints in a business transaction.

Business transactions may be characterized by universally acceptable business operational primitives (or simply business functions), which represent functions that are critical to the conduct of business. A business function is a description of a well-defined and commonly acceptable critical business principle, e.g., payment or delivery of goods or services, that transforms business values and causes state changes to transaction participants, e.g., transforms an unpaid order to paid order. To achieve this the business function uses contextually aware polymorphic business operations, e.g., cancel an order or cancel a payment, constraints and dependencies, (see Sect. 6 for further details). Business transactions usually operate on business (document-based) objects. These are traditionally associated with items such as purchase orders, catalogues (documents that describe products and service content to purchasing organizations), inventory reports, ship notices, bids and proposals. Document objects are also associated with agreements, contracts or bids. This allows business transactions to interchange everything from product information and pricing proposals to financial and legal statements.

An important characteristic of the business transaction model is that it is extendable in nature and can be incorporate and blend together diverse business, e.g., RosettaNet PIPs, and technical protocols, e.g., security protocols.

4 Characteristics of the business transactions

Business transactions in the business transaction model are defined through two perspectives very much along the lines of the Business Operational View and the Functional Service View in OpenEDI and ebXML. These two perspectives are:

1. Business-related aspects such as critical business functions, business conventions, agreements and rules among organizations. These are limited to those aspects regarding the making of business decisions and commitments among organizations, which are needed for the description of a business transaction.
2. Systems-related aspects that are necessary for Web services transaction systems to support the execution of business transactions using Web services standards.

Important business-related aspects of a transaction include the following features:

- which parties are involved in the transaction;
- what is being transacted, e.g., what types of goods or services;
- the destination of payment and delivery;
- the transaction time frame; e.g., a timeout value that defines the maximum amount of time during which the transaction should be active;
- permissible operations;
- links to other transactions;
- receipts and acknowledgments;
- identification of money transferred outside national boundaries; and finally,
- the ability to specify contractual agreements, liabilities and dispute resolution policies.

Systems-related aspects focus on technical considerations for automating or improving the internal flow of transaction processing, spanning multiple disparate applications. An important requirement is to provide solutions for reliable, consistent, and recoverable composition of back-end services, as inconsistencies and failures cannot be tolerated. The technical infrastructure for supporting the business related aspects of a business transaction requires both short conventional ACID transactions as well as support for long-lived open nested transactions. Important systems-related aspects of a business transaction include the following features:

- the ability to support of long-running interactions (business conversations) that include multiple, often nested units of work, each with its own data requirements;
- the ability to specify exceptional conditions and their consequences, including recovery sequences;

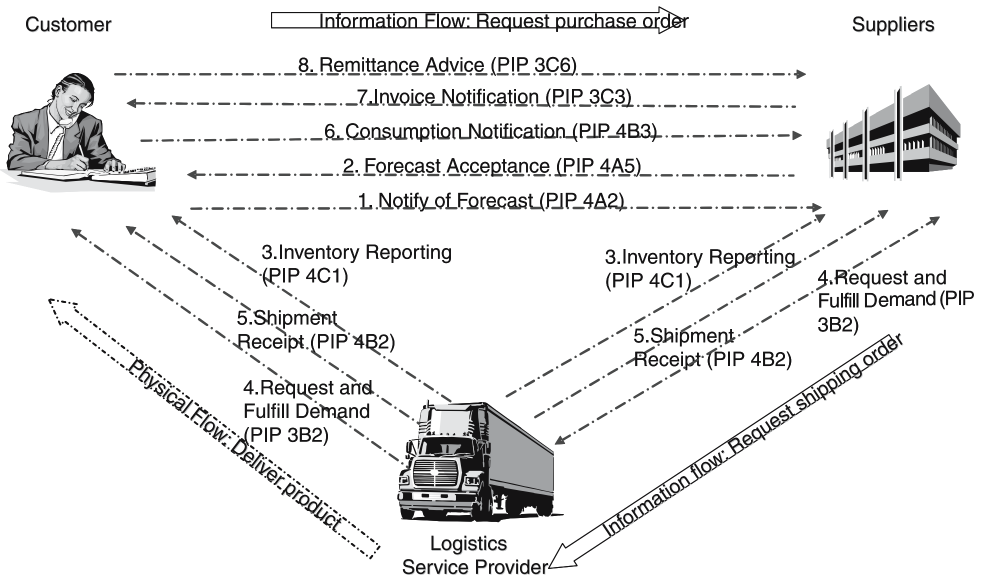

Customer
Information Flow: Request purchase order
Suppliers
8. Remittance Advice (PIP 3C6)
7.Invoice Notification (PIP 3C3)
6. Consumption Notification (PIP 4B3)
2. Forecast Acceptance (PIP 4A5)
1. Notify of Forecast (PIP 4A2)
3.Inventory Reporting (PIP 4C1)
5.Shipment Receipt (PIP 4B2)
4.Request and Fulfill Demand (PIP 3B2)
3.Inventory Reporting (PIP 4C1)
4.Request and Fulfill Demand (PIP 3B2)
5.Shipment Receipt (PIP 4B2)
Physical Flow: Deliver product
Information flow: Request shipping order
Logistics Service Provider

Fig. 2 Integrated logistics example using RosettaNet PIPs

- the ability to support reversible (compensatible) and repaired (contingency) transactions;
- use of alternate transactions that can be tried out either in sequence or in parallel;
- the ability to reconcile and link transactions with other transactions;
- the ability to support secure transactions that guarantee integrity of information, confidentiality, privacy and non-repudiation;
- the ability for transactions to be monitored, audited/logged and recovered.

In a Web services environment business transactions are used to capture and define the integration between business operational requirements and technical transactional requirements. Business transactions are found only in the application (business-logic) level and essentially trigger transactional Web service interactions between organizations at the systems-level (using Web services-based business processes and transactional standards) in order to accomplish some well-defined shared business objective. 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 back-end order database is updated and a document copy of the purchase order is filed. More complex business transactions may involve activities such as payment processing, shipping and tracking, determining new product offerings, granting/extending credit, and so on.

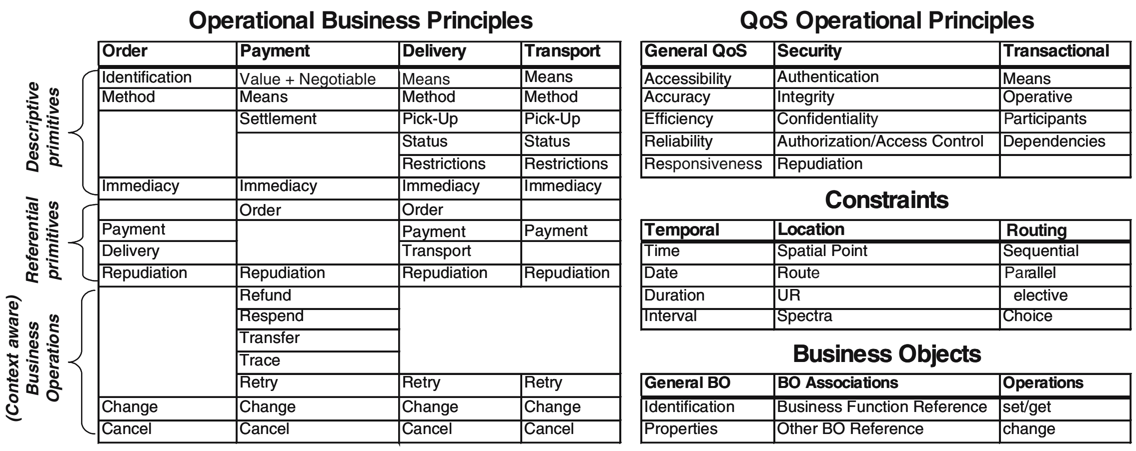
Transactional facilities such as the previous are provided by Web Services standards including the Web Services Transaction (WS-Transaction) [5,6] and the Web Services Composite Application Framework (WS-CAF) [7] standards.

5 Integrated logistics example

In this section we present a simple integrated logistics example based on standard business protocol RosettaNet PIPs [8]. We shall enhance the simple integrated logistics scenario described in the following with transactional functions and business operational semantics as well as QoS features.

RosettaNet PIPs define the specific sequence of steps required to complete a business-to-business process, such as the placement of a purchase order with a supplier, and the specific exchange of information and transactions triggered by each step in the business process. Specifically, RosettaNet PIPs create standard e-business dialogues for common business activities like order and inventory management, transportation, sales forecasting, etc. The PIPs are organized in functionally logical groupings of segments and clusters. For example, the PIP3A4 Purchase Order Request is found in the Quote and Entry Segment grouping, which belongs to the Order Management cluster [8].

Figure 2 depicts an integrated logistics scenario involving a customer, suppliers and a logistics service provider. This logistics model consists of forecast notification, forecast acceptance, inventory reporting, shipment receipt, request and fulfil demand, consumption and invoice notification processes. PIP 4A2 (*Notify of Embedded Release Forecast*) supports a process in which



Operational Business Principles

	Order	Payment	Delivery	Transport
Descriptive primitives	Identification	Value + Negotiable	Means	Means
	Method	Means	Method	Method
		Settlement	Pick-Up	Pick-Up
			Status	Status
			Restrictions	Restrictions
	Immediacy	Immediacy	Immediacy	Immediacy
Referential primitives		Order	Order	
	Payment		Payment	Payment
	Delivery		Transport	
	Repudiation	Repudiation	Repudiation	Repudiation
(Context aware) Business Operations		Refund		
		Respend		
		Transfer		
		Trace		
		Retry	Retry	Retry
	Change	Change	Change	Change
	Cancel	Cancel	Cancel	Cancel

QoS Operational Principles

General QoS	Security	Transactional
Accessibility	Authentication	Means
Accuracy	Integrity	Operative
Efficiency	Confidentiality	Participants
Reliability	Authorization/Access Control	Dependencies
Responsiveness	Repudiation	

Constraints

Temporal	Location	Routing
Time	Spatial Point	Sequential
Date	Route	Parallel
Duration	UR	elective
Interval	Spectra	Choice

Business Objects

General BO	BO Associations	Operations
Identification	Business Function Reference	set/get
Properties	Other BO Reference	change

Fig. 3 Description of common business functions, QoS principles and constraints

a forecast owner sends forecast data to a forecast recipient. The embedded release forecast contains product demand information (planning demand) and/or product release information. PIP 4A5 (*Notify of Forecast Reply*) provides visibility of available forecasted product quantity between two trading partners. PIP 4C1 (*Distribute Inventory Report*) supports a process in which an inventory information provider reports the status of the inventory to an inventory user. The inventory report can include any product, active or inactive, held in inventory. PIP 3B2 (*Notify of Advance Shipment*) allows a shipper to notify a receiver that a shipment has been assigned. This notification is often a part of the shipment process. PIP 4B2 (*Notify of Shipment Receipt*) supports a process used by a consignee to report the status of a received shipment to another interested party, such as another consignee, a transport service provider, a third-party logistics firm, or a shipper. Receipt of a shipment is reported after it has been delivered by a carrier and inspected by receiving personnel. The customer then issues an *Invoice Notification* (PIP 4B3) to communicate material consumption to the supplier, allowing the supplier to trigger invoicing for the consumed material. PIP 3C3 (*Notify of Invoice*) enables a provider to invoice another party, such as a buyer, for goods or services performed. Finally, PIP 3C6 (*Notify of Remittance Advice*) enables a payer to send remittance advice to a payee (in this case the supplier) which indicates which payables are scheduled for payment.

6 Business operational principles and QoS functions

In the previous we argued for the identification of functional capabilities necessary to support business transactions and introduced the concept of critical business functions, which divide business applications along common functional lines. Figure 3 describes the constituent elements of the business functions in the business transaction model, which is described schematically in Fig. 3. In particular, this figure illustrates that the business transaction model divides trade into four broad functional lines — ordering, paying, delivery and transportation, which are referred to as business operational principles (or business functions) in this figure. These areas represent common business functions that are generic, industry neutral and re-usable and can be used to develop business transactions in a multiplicity of business scenarios. In this way we remove excess complexity from the business transaction, allowing common business functions such as ordering, distribution and payment to be expressed in a form analogous to abstract data types and rationalizing them across an integrated supply chain. The basic structure of these business functions (which could be also appropriately extended and customized) provides a standard definition for building business solutions.

Figure 3 shows that each business function uses a number of descriptive primitives (or attributes) that describe a certain business function, e.g., the means of payment. There are also referential primitives that refer to other business functions, e.g., payment refers to an associated order. Finally, the context aware business operations introduce a set of polymorphic business operations that collectively transform business values and cause state changes to the business transaction participants, e.g., cancel and order or cancel a shipment.

Business functions not only help streamline and rationalize common business practices across an integrated supply chain, they also help enforce participant commitments. They introduce a mandatory set of four business level atomicity criteria that reflect the operational

Fig. 4 Describing the *delivery* business function attributes

```
Means <!- The means of the delivery (depends on nature of goods) -->
•setMeans
      •setChannel
            •Online <!- intangible goods --> / offline <!- tangible goods -->
      •setDeliveryMeans
            •Air / Sea / Ground / Combinations
Method <!- Method of the delivery -->
•setDeliveryMethod <!- How will goods be delivered (e.g., batch, all in once, etc) -->
      •setNumberOfDeliveries <!- How often will goods be delivered (if in batches) -->
•setDeliveryOptions
      •Express
            •Type
                  •Next day / Two day / ...
                  •Boolean Delivery_Signature_Required
•setTransportCompany <!- Which company is responsible for the transport -->
      •setTransportCompanyDetails
•setDeliveryPeriod
      •Temporal
```

semantics the four standard business functions (ordering, payment, delivery and transportation). For instance, payment atomicity affects the transfer of funds from one party to another in the transaction. This means that the transaction would fail if payment is not made within a pre-specified time period that was agreed between a supplier and a customer. Delivery atomicity, on the other hand, implies that the right goods will be delivered to a customer at the time that has been agreed.

Each atomicity criterion is treated as a single indivisible logical unit of work, which determines a set of viable outcomes for a business transaction. The outcomes of a business transaction may involve non-critical partial failures, or selection among contending service offerings, rather than the strict all-or-nothing assumption of conventional ACID transactions, and govern the duration and character of participation in a transaction. They also allow provisional results to be revealed deliberately to allow such business activities as probabilistic inventory management. Atomicity criteria can be characterized as vital or non-vital. If a business level atomicity criterion is characterized as vital and fails then the transaction aborts at the system-level. If, however, the atomicity criterion is characterized as non-vital then a contingency activity may be issued in case that a given atomicity criterion, e.g., if goods transportation fails then another shipper could be chosen and a new shipment is scheduled. The above characteristics give the ability to a business transaction to explicitly describe business operational semantics, specify the proper behavior of common business functions and their implications in case of success or failure.

The business transaction model not only expresses the purpose of each business collaboration interaction but is also capable of capturing the message and commitment exchange requirements between any trading partners, identifying the timing and sequence of message exchanges. The message exchanges in the model can be specified in such a way so as to define the ordering in which a particular partner sends messages to and expects messages from its trading partners. The model has some fixed sequencing semantics which require that *ordering* occurs first and is followed by *transport* and *delivery*. *Payment* can happen either before or after the *delivery* function. For instance, in the integrated logistics scenario described in the previous section, there might be an implicit or explicit agreement that the delivery of goods must take place before the payment and that payment always follows the confirmation of an order. This situation is depicted by the following code snippet that uses a business transaction specification language (called Business Transaction Mark Up Language or BTML) that we are currently developing for business transactions employing Web services.

```
<BTx>
 <name>LogisticsScenario</name> ...
 <sequence>
  <BF> <name>Order</name> </BF>
  <BF> <name>Delivery</name> </BF>
  <BF> <name>Payment</name> </BF> ...
 </sequence>
</BTx>
```

By using these constructs, each participant can understand and plan for conformance to the business protocol being employed.

Figure 4 shows two of the attributes of the *delivery* business function. These are *means* and *method* which describe the means and method of delivery, respectively. The *delivery* business function, as seen from Fig. 3, also uses referential primitives to refer to other functions such as *payment* and *transport*. It also uses context aware polymorphic business operations, such as *retry* to retry a failed delivery, *change* to change a delivery and *cancel* to cancel a delivery. All attributes and operations in Fig. 3 have been defined and formalized and are available on request.

Finally, an important element of the business model is the quality of service required from the functional capabilities for the business transactions. For instance, one of the referential primitives used in business functions such as ordering, payment, transport and delivery, is the issue of non-repudiation using digital signatures, see Fig. 3. Non-repudiation is a security service that supports the proof-of-origin and the proof-of-receipt services that provide proof of message origin to the recipient and proof of message receipt to the sender. In this way business transactions can also become QoS-aware and QoS principles can be blended with constraints and business requirements enforced by the business functions. Other QoS primitives that can be attached to a business transaction and govern its behavior may include general QoS primitives such as desired performance rates, mean time to respond, accessibility periods, time-to-repair a service that has failed, desirable security protocols and tokens, and so on. These are also depicted in Fig. 3.

QoS criteria can be registered in a Service Level Agreement, which specifies the agreements and commitments of trading partners involved in a business transaction and which is used to specify the conditions that initiate and drive a business transaction. More specifically, they form part of the agreed service-level objectives, which define the levels of service that both the service customers and the service providers agree on, and usually include a set of service level indicators, like availability, performance and reliability.

7 Business transaction reference architecture

Apart from providing a basis for describing partner engagements, interaction sequences and the technical services for implementing business practices in a supply chain, the business transaction model can also serve another equally important purpose. It can also serve to introduce a framework for the development of e-Business solutions needed to rationalize and simplify business transactional exchanges and applications in an integrated supply chain. The effective inter-relationship between the two perspectives of business transactions presented in Sect. 4 is a critical factor of the architectural model.

The reference architecture that supports the business transaction model is depicted in Fig. 5. Application scenarios are built using the business related aspects of the model, e.g., business principles constraints, QoS criteria, and soforth. Both the business ad technical aspects of the model must interact with each other to execute the corresponding business transaction. Each business level aspect is appropriately mapped to a corresponding infrastructure primitive that that employs Web services standards, such as BPEL, WS-Coordination [9], WS-Atomic Transaction [5,10] and WS-Business Activity [6, 10]. Currently, this trio of Web services standards is used

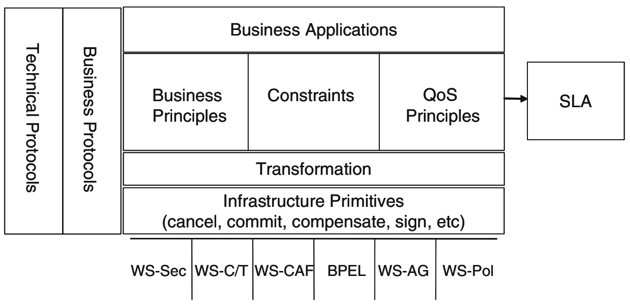


Fig. 5 Business transaction reference architecture

to implement business transactions. This infrastructure is based on open an source implementation framework provided by JBoss Transactions (http://www.jboss.org) which supports the latest Web services transactions standards, providing all of the components necessary to build interoperable, reliable, multi-party, Web services-based applications quickly and easily.

Figure 5 also shows how QoS criteria can be registered in an SLA. An SLA contains several entries that are related to a business transaction. These include the scope of the agreement (the services covered in the agreement), penalties (sanctions should apply in case the service provider under-performs and is unable to meet the objectives specified in the SLA), optional services (any services that are not normally required by the user, but might be required in case of an exception) and exclusion terms (specify what is not covered in the SLA).

QoS criteria in the context of a business transaction are expressed as assertions by an assertion sub-language of BTML. This assertion language is an extension of the WS-Policy assertion language thereby reusing existing functionality like normal form, referential and combined policies. BTML's assertion language also contains context specific assertion definitions for a business transaction. This part of the BTML can then be incorporated as guarantee terms into agreements templates specified by WS-Agreement to enable the specification, negotiation and acceptance of SLAs that are used to drive business transactions.

Finally, the reference architecture supports the use of business and technical protocols in the context of the business transaction model. Currently, the architecture supports one business protocol namely, RosettaNet, and one technical protocol, the Secure Electronic Transaction (SET) [11]. SET (already obsolete) is a technical protocol that ensures the security of financial transactions on the Internet. With SET, a user is given a digital certificate and a transaction is conducted and verified using a combination of digital certificates and digital signatures among the purchaser, a merchant, and the

Listing 1

```
<BF>
 <name>Delivery</name>
  <Means>
   <DeliveryMeans>Air</DeliveryMeans>
  </Means>
  <Method>
   <DeliveryMethod>Complete</DeliveryMethod>
   <TransportCompany>UPS</TransportCompany>...
  </Method>
  <OrderLine><!-- References the Goods  -->
  </OrderLine>
  <Change>
   <permitted>true</permitted>
   <element>location</element>
   <numberoftimes>2</numberoftimes>
   <prize monetary="$">150</prize>
   <paymentmeans>invoice</paymentmeans>
  </Change>...
</BF>
```

Listing 3

```
<activities>
 <activity>
  <name>Create&Send Invoice</name>
  <messages>
   <messageIncoming name="Invoice Message" />...
   <messageOutgoing name="Invoice Notification">
    <acknowledgeable>true</acknowledgeable>
    <tta type="maxdurationinhours">2hours</tta>...
  </messages>...
 </activity>
</activities>
```

Listing 2

```
<BF>
 <name>Payment</name>
 <BusinessProtocol>
  <participant>
   <name>SteelWorks</name>
   <role>Supplier</role>
   <activities>
    <activity>
     <name>Create&Send Invoice</name>
      <messages>
       <messageOutgoing>InvoiceMessage</messageOutgoing>...
      </messages>
     </activity>...
    </activities>
   </participant>...
   <sequence>
    <activity>Create&Send Invoice</activity>
    <activity>Receive&Check Remittance Advice</activity>
    <selective>
     <sequence>
      <activity>Accept Remittance Advice</activity>
      <activity>Process Remittance Advice</activity>
     </sequence>
     <activity>Decline Remittance Advice</activity>
    </selective>
   </sequence>...
  </BusinessProtocol>...
</BF>
```

Fig. 6 Listings of BTML snippets

purchaser's bank in a way that ensures privacy and confidentiality. SET makes use of the Secure Sockets Layer and uses several aspects of a Public Key Infrastructure.

In the following we concentrate on illustrating how to semantically enhance the RosettaNet business protocol, which lacks the notion of a business transaction as defined in Sect. 2, by injecting into it business functions, explicit sequencing of interactions, partner commitments and constraints.

8 Emulating business and technical protocols

In this section we will illustrate how we can supplant transactional primitives into the integrated logistics scenario in Fig. 2 to semantically enhance the operational characteristics of the interacting RosettaNet processes.

This procedure is performed according to the following steps. We start first by grouping the individual PIPs into related sets that realize a specific common business function. We observe that PIPs 4A2, 4A5 and 4C1 are all part of the *order* business function. PIP 3B2 is part of the *transport* function and PIPs 4B2 and 4B3 are part of the *delivery* function. The *payment* function is covered by PIPs 3C3 and 3C6.

Following this we need to capture the message and commitment exchange requirements between any trading partners, identifying the timing and sequence of message exchanges. We assume that the trading partners have agreed on a business protocol (developed on the basis of RosettaNet) which requires that *payment* follows *transport* and *delivery*. This is specified in BTML. Subsequently, we can specify the business functions using BTML. We assume that in the integrated logistics example the customer and the supplier have agreed on an all or nothing express delivery method, which specifies that if the goods are ready for transport the delivery should not take more than two days (which requires delivery by air). The specification in BTML can be found in Listing 1 of Fig 6.

The above code snippet also specifies that the delivery location is changeable at most twice at a cost of 150 $ per time and that the fees will be added to the original invoice.

Figure 7 illustrates a simple payment protocol agreed between a supplier and a customer. In this figure the payment protocol is seen from the vantage point of the supplier. The second code snippet (found in Listing 2 of Fig. 6) specifies in BTML (part of) the simple payment protocol illustrated in Fig. 7.

As a last step we may wish to add QoS properties to the elements of the business transaction. In particular, we may wish to indicate that the InvoiceNotification process (PIP 3C in Fig. 2) requires that the time to acknowledge an Invoice Notification message send from the Send Invoice activity of the Supplier to the Receive Invoice activity of the Customer is no longer then 2 h. This property can be specified using the Responsiveness primitive in the General QoS field in Fig. 3. The QoS Responsiveness primitive has an operator called setAcknowledgeable that specifies whether a particular message should or should not be acknowledgeable and also the time frame for this to happen. This can be specified in BTML as seen in Listing 3 of Fig. 6.

We may also wish to add other QoS constraints on messages or message parts. For example we may wish to specify further security primitives indicating whether or not a message or message part should be non-reputable

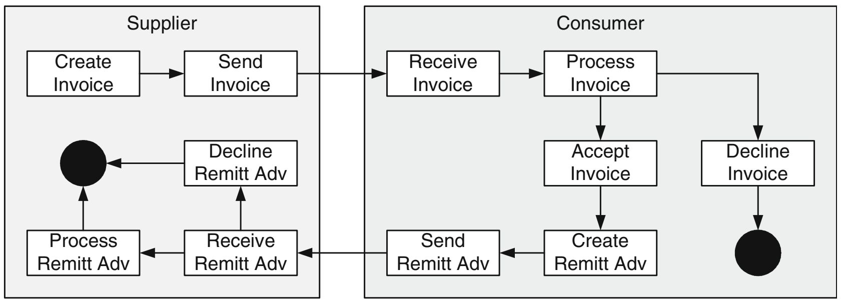


Fig. 7 Simple payment protocol agreed between a supplier and a customer

or signed with a particular hash and encryption function. This can be specified in BTML as seen in Fig. 8.

The listing expresses two alternatives in a policy specified using WS-Policy notation. This is indicated by the `ExactlyOne` policy operator, which specifies that only one of its direct elements must be applicable. Each of these alternatives is shown to be wrapped in an `All` operator. The `All` operator means that all the assertions enclosed within this operator must be applicable. This means that if the first alternative is chosen, a Kerberos token type and AES encryption algorithm must be employed. If conversely the second alternative is chosen, an X509 certificate and the 3DES encryption algorithm must be employed.

To implement the above business scenarios RosettaNet types, messages, such the ones expressed in Fig. 2, are mapped to equivalent message definitions in WSDL. The actions in a RosettaNet PIP are mapped to operations in WSDL. Partner roles are mapped to Partners in BPEL. Choreography from the RosettaNet PIPs are implemented in the choreography of the abstract and executable business process in BPEL. Finally, exception messages from RosettaNet RNIF are mapped to the exception handling mechanisms of BPEL. Transactional implementations for the above scenarios are currently being mapped to WS-Coordination and WS-Transaction constructs. Concepts that have no direct mappings either in BPEL, WS-Coordination, and WS-Transaction require specific workaround implementations.

Another important aspect of the business reference architecture is that it can blend business with technical protocols. For instance, the business application that we sketched in the previous does not have any concrete way to handle the actual payment so that it can transfer funds using a financial service provider. This situation also holds for the RosettaNet PIPs.

To remedy this situation we can extend the payment part of the business transaction described in the previous with a technical protocol that guarantees secure payments. This could be, for instance, accomplished by means of the Secure Electronic Transaction protocol. SET itself is focused on messages, their content, security aspects and authorization and it is not integrated with other business functions and business protocols and has no notion of system transactions. An implementation exists that shows how to create an executable transactional business process of the enriched SET protocol using BPEL and WS-Transaction and how to associate the payment protocol with business transaction constructs.

```
<activities>
 <activity>
  <name>Send Invoice</name>
  <Messages>
   <MessageIncoming name="Invoice Message" />
   <MessageOutgoing name="Invoice Notification">
     <wsp:ExactlyOne>
     <wsp:All>
       <wsse:SecurityToken>
        <wsse:TokenType>wsse:Kerberosv5TGT</wsse:TokenType>
       </wsse:SecurityToken>
       <wsse:Algorithm Type ="wsse:AlgSignature"
       URI="http://www.w3.org/2000/09/xmlenc#aes"/>
     </wsp:All>
     <wsp:All>
       <wsse:SecurityToken>
        <wsse:TokenType>wsse:X509v3</wsse:TokenType>
       </wsse:SecurityToken>
       <wsse:Algorithm Type ="wsse:AlgEncryption"
        URI="http://www.w3.org/2001/04/xmlenc#3des-cbc"/>
     </wsp:All>
    </wsp:ExactlyOne>...
   </MessageOutgoing>...
  </Messages>...
```

Fig. 8 Adding QoS constraints to messages

9 Related work

Automated business transactions are a new category of research, wider than historical data-centric local, distributed of federated transactions. This *third generation* of transaction management builds out from core transactional technology, particularly the concept of a open nested transactions and multi-phase distributed outcomes (*two-phase commit* in conventional database/messaging transactions).

Research in this paper was inspired by the work found in [12], which motivates the need for using transactions that mimic real business exchanges and presents an overview of several technologies and protocols that may support a business transaction framework.

Research in the business transactions area is also related to the creation of meta-models for Web service transaction models as for example reported in [13,14]. In [13] the authors propose to combine multiple transaction models as WS-C coordination types into BPEL specifications that can support transactional workflows. In [14] a meta-modeling approach to transaction management is proposed, however, this approach focuses on the modeling and representation of transaction models driven purely from database technology perspective without taking into account business and workflow requirements.

The work reported in this paper can benefit from other ongoing research in the Web services domain. Of particular interest is the work on SLAs reported in [15]. Here, the authors define a template-based approach that enables automated service provisioning. This provisioning can be guided by the WS-Agreement [16] protocol. Finally, the work reported in [17] is quite relevant as it describes many non-functional properties applicable for Web services that can also benefit business transactions.

10 Summary

In the previous we have described a business transaction model and associated reference architecture. Key characteristics of this model is that is sharply distinguishes between a business related and a systems related view of transactions. On the business-level transactions are weaved around commonly standard business functions that apply to a variety of application scenarios. Although such business functions drive transactional applications between trading partners they are completely external to current Web services transaction mechanisms and are only expressed as part of the application logic and are of course difficult to reuse and specialize. At the business level, the model can also represent business exchanges and the sequencing and timing, business agreements stipulated in SLAs, liabilities and dispute resolution policies, and blends these with QoS criteria. At the systems-level the business transaction model provides support for conventional ACID split-second duration transactions but also long-running transactions, which may endure for periods that exceed the anticipated service cycles of participant systems. Business transactions in the systems-level retain the driving ambition of consistency. Implementation of the systems-level services is currently provided by a trio of Web services standards BPEL, WS-Coordination, and WS-Transaction.

Business transactions are more flexible and sophisticated in their means reflecting real business functionality and operational semantics and also the imperfect determinism of real business processes (which are necessarily designed to cope with innumerable variations and with incremental and partial successes). This approach results in a modular, scalable, flexible integration architecture and support environment, providing a high level language to coordinate the commitment exchanges between parties and ensuing flow of messages among different business processes that require transactional support.

The potential benefits of this approach arise largely from its ability to standardize common business functions, better align business processes with business objectives and provide information to enable monitoring and troubleshooting of problems and delays. Business decisions can be made at every step of the transaction to align it with business objectives and to alleviate undesirable conditions. For example, in case of a purchase order cancellation due to a faulty part, an order transaction can automatically reserve a suitable replacement product and notify the billing and inventory processes of the changes. When all interactions between the various business processes have been completed and the new adjusted schedule is available, the purchase order Web service notifies the customer sending her an updated invoice.

Work is currently underway regarding the formalization and verification of the business transaction model.

References

1. Webber D (2000) Understanding ebxml, uddi and xml/edi http://www.touchbriefings.com/pdf/967/59.pdf
2. Meadows B, Seaburg L (2004) Universal Business Language 1.0. http://www.docs.oasis-open.org/ubl/cd-UBL-1.0/
3. Papazoglou M, Kratz B (2006) A business-aware web services transaction model. In Dan A, Lamersdorf W (eds) Proceedings of the 4th international conference on service oriented Computing (ICSOC 2006), Chicago December 4–7, 2006. Lecture Notes in Computer Science, vol 4294 Springer, Berlin, pp 352–364
4. Business Process Project Team: ebxml business process specification schema version 1.01 (2001) http://www.ebxml.org/specs/ebBPSS.pdf.
5. Cabrera LF, Copeland G, Feingold M, Freund RW, Freund T, Johnson J, Joyce S, Kaler C, Klein J, Langworthy D, Little M, Nadalin A, Newcomer E, Orchard D, Robinson I, Storey T, Thatte S (2005) Web services atomic transaction (WS-AtomicTransaction). 1.0 edn
6. Cabrera LF, Copeland G, Feingold M, Freund RW, Freund T, Joyce S, Klein J, Langworthy D, Little M, Leymann F, Newcomer E, Orchard D, Robinson I, Storey T, Thatte S (2003)

Web services Business activity framework (WS-BusinessActivity). 1.0 edn
7. Bunting D, Chapman M, Hurley O, Little M, Mischkinsky J, Newcomer E, Webber J, Swenson K (2003) Web services composite application framework (WS-CAF). 1.0 edn
8. RosettaNet: (2001) standards required to support xml-based b2b integration http://xml.coverpages.org/rosettanet-StandardsForIntegration.pdf
9. Cabrera LF, Copeland G, Feingold M, Freund RW, Freund T, Johnson J, Joyce S, Kaler C, Klein J, Langworthy D, Little M, Nadalin A, Newcomer E, Orchard D, Robinson I, Shewchuk J, Storey T (2005) Web Services Coordination (WS-Coordination). 1.0 edn
10. Cabrera LF, Copeland G, Johnson J, Langworthy D (2004) Coordinating web services activities with ws-coordination, ws-atomictransaction, and ws-businessactivity http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/wsacoord.asp
11. Merkow MS, Breithaupt J, Wheeler KL (1998) Building SET applications for secure transactions. Wiley, New York
12. Papazoglou M (2003) Web services and business transactions. World Wilde Web: Internet Web Inf Syst 6(1):49–91
13. Tai S, Mikalsen T, Wohlstadter E, Desai N, Rouvellou I (2004) Transaction policies for service-oriented computing. Data Knowl Eng 51:59–79
14. Hrastnik P, Winiwarter W (2005) Using advanced transaction meta-models for creating transaction-aware web service environments. Int J Web Inf Syst 1(2):89–99
15. Ludwig H, Gimpel H, Dan A, Kearney B (2005) Template based automated service provisioning supporting the agreement driven service life-cycle. In: Benatallah B, Casati F, Traverso P (eds) Service-oriented computing - ICSOC 2005, 3rd Proceeding of international conference, Amsterdam, December 12–15, 2005, Lecture Notes in Computer Science, vol. 3826. Springer, Berlin pp 283–295
16. Andrieux A, Czajkowski K, Dan A, Keahey K, Ludwig H, Nakata T, Pruyne J, Rofrano J, Tuecke S, Xu M (2005) Web services agreement specification (ws-agreement). Technical report, Grid Resource Allocation Agreement Protocol (GRAAP) WG
17. O'Sullivan J, Edmond D, ter Hofstede AHM (2005) Formal description of non-functional service descriptions. Technical report, Queensland University of Technology http://www.bpm.fit.qut.edu.au/about/docs/non-functional.jsp