A B2B Benchmark On Top Of UMM and TPC-App

Birgit Hofreiter, Christian Huemer, and Robert Mosser
Department of Distributed and Multimedia Systems
University of Vienna
Liebiggasse 4
1010 Vienna, Austria
{bh, ch, rom}@mminf.univie.ac.at

Abstract: The TPC-App is a B2B benchmark implementing a retail scenario. This benchmark specifies a set of web services interactions to test the performance of an application server. Today B2B systems undergo a permanent technological change. There even exist competitive technologies at the same time. From a business perspective it is important that the application systems of the business partners interoperate no matter what technology is applied. In order to measure the effects of using a combination of different technologies for the same B2B scenario, we reverse engineer the TPC-App scenario into a platform independent model following the UN/CEFACT's Modeling Methodology (UMM). The resulting UMM-App benchmark defines a conceptual B2B model that may be implemented by different IT technologies in order to evaluate feasibility and performance.

1 Motivation

Computer systems are used to solve more and more complex business problems. The more complex the problem gets the more complex it gets to compare solutions for the problem. It is hardly possible to compare different solutions by comparing their specifications. Instead a number of test and trials are performed to assess the performance of each computer system. Of course, the test must represent a typical workload for the computer systems. Since testing different computer systems in real world environments over a significant amount of time is too hard and too time-consuming, special programs have been created imposing a typical workload on the system. These synthetic programs are known as benchmarks.
It is the goal of the paper to develop a benchmark for business to business e-commerce (B2B) systems. In a B2B environment, interoperation is usually reached by the exchange of standardized business documents in an agreed choreography. Each document exchange is further characterized by reliability, authorization, non-repudiation, etc. Today, we know one B2B benchmark: TPC-App [TPC05]. It defines the workload on the application server of one business partner. This workload is defined by a set of XML documents exchanged by SOAP messages. TPC-App considers reliable and durable messaging, but not all security requirements omnipresent in inter-organizational commerce.

It follows that TPC-App defines a B2B benchmark that is limited to a certain IT technology – a certain mix of Web Services specifications. However, there exist different technologies that may be used to implement B2B systems. This technologies span from traditional EDI [HF89] over Web Services [W3C06] and ebXML [EN01], to Semantic Web technologies [BHL02] and whatever comes up in the future. Even within the family of Web Services there exist some competing specifications, e.g. BPEL [ACD03] and BPML [Ar02] to describe business processes.

From a business perspective it is not important which technology is used, as long as the business goals are reached. In order to build a test bed enabling the comparison of different B2B technologies, it is necessary to define the B2B test scenario independently of any B2B technology. UN/CEFACT's modeling methodology (UMM) [UN06a] is a well-accepted approach to model the collaboration scenario between different business partners in B2B. Thereby, UMM focuses on the business logic, commitments and constraints of a partnership independent of any platform issues.

In this paper we combine the strengths of both TPC-App and UMM in order to create a platform independent benchmark. We call this benchmark UMM-App to express its relationship to TPC-App and UMM. As a starting point we consider the B2B scenario of TPC-App which has proven to define a typical workload for B2B servers. We reverse engineer this scenario in order to create a platform independent UMM model. This model may be transformed to different other B2B technologies enabling their comparison. This approach will help the developers of B2B standards on the IT-layer to test the practicability of their specifications. It will assist B2B tool providers to decide whether to integrate a specification into their tool set or not. Last, but not least, it will aid the business partners to decide which technology to use for realizing their B2B platforms.

The remainder of this paper is structured as follows. In section 2 we give a short introduction of the TPC-App. It is followed by an overview of the most important UMM artefacts in section 3. Section 4 presents the reverse engineering of the TPC-App scenario into UMM-App. We highlight the fundamental differences between TPC-App and UMM as well as the assumptions made during the reverse engineering. Section 5 positions the role of UMM-App in developing meaningful B2B systems. The conclusion in section 6 summarizes our approach and suggests extending the UMM-App by a computational independent model to develop better socio-economic systems.

2 TPC-App

The TPC-App benchmark is designed to compare application servers executing B2B scenarios by means of web services [TPC05]. It measures the throughput rate of a commercially available application server. The benchmark not only allows the comparison of the hardware performance, but also the performance of the two supported application environments: .NET and JAVA. The throughput rate is re-

ported as Web Service Interactions per Second (SIPS). TPC-App also provides a price/performance indicator: Associated Price per SIPS ($USD/SIPS).
In order to deliver significant results, TPC-App has defined a set of basic operations that are vital and representative for B2B web service environments. The synthetic application takes place in a retail distribution environment on the Internet: A wholesaler provides retailers with web services supporting product browsing and ordering. The application server takes the role of the wholesaler. The retailers are represented by the remote business emulators (RBE) which continuously create the workload. The following self-explanatory list represents the basic operations defined by TPC-App:

- New Customer
- Change Payment Method
- Create Order
- Order Status
- Change Item (allows the retailers to change the publishing date of any given item in the wholesaler's catalog)
- New Products (returns a list of currently changed items)
- Product Detail

In addition to these services offered by the wholesaler, the TPC-App defines four emulators that are services invoked by the wholesaler. The *payment gateway emulator* (PGE) processes credit card payments. The *purchase order validation* (POV) checks incoming purchase orders against a set of business rules. The *inventory control emulator* (ICE) handles purchase orders with suppliers of the wholesaler. Finally, the *shipment notification emulator* (SNE) creates a tracking number and a shipping label for the shipment of a purchase order.
Usually, an incoming web services request from the retailer requires the wholesaler to execute one or more emulator services before returning a response to the retailer. This means that the emulators are nested within the web services provided by the wholesaler. Consequently a wholesaler's web service cannot be completed unless any nested emulator is completed. In the remainder of the paper we use the term *web services interaction* for a service offered by the wholesaler, and the term *nested interaction* for a service offered by an emulator.
The input and output of both web services interactions and nested interactions are defined as a list of parameters each complying with an XML schema data type. TPC-App utilizes SOAP to exchange messages. All communication is secured using SSL. Furthermore, the wholesaler and the emulators need to be authenticated. This is also realized by SSL. A web service interaction starts with a request message. The business response – also delivered as a SOAP message – is expected within 90 seconds. However, a web service may also lead to a control failure, i.e. no answer is given in time, a SOAP fault message is delivered or an HTTP status code in the 4xx-5xx range is indicated. In case of a control failure TPC-App allows

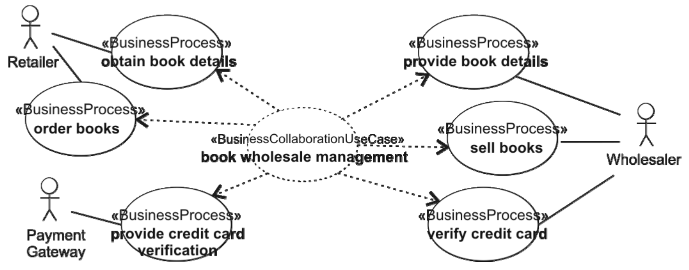
for a total of 20 resubmissions of the request. A resubmission is also necessary in case of a misshaped response document that did not fulfill the output requirements. The performance test takes place within the so-called measurement interval. The throughput rate (SIPS) is the number of successful web service interactions divided by the length of the measurement interval in seconds.
There is no choreography defined for the web service interactions. They can be requested by the retailers in any order. However, the web service interaction mix specifies how often each web service interaction must be invoked in relation to the overall number of operation calls. This means that TPC-App defines an exact percentage of operation calls for each web service interaction which must be met with a tolerance of +/- 0,5%. This regulates the mixture of web service requests in order to simulate a real world environment. In TPC-App the majority of requests generate purchase orders while only a very small portion of requests generate item catalog information.

3 UN/CEFACT Modeling Methodology

The UN/CEFACT Modeling Methodology (UMM) is a methodology for describing B2B collaborations. UMM enables capturing the business knowledge independently of the underlying implementation technology. The goal is specifying a global choreography of business document exchanges serving as an agreement between the participating business partners. UMM defines a UML profile [UN06a] i.e. a set of stereotypes, tagged values and constraints – customizing the UML meta model for the special purpose of modeling the collaborative space in B2B.
UMM consist of three views which are briefly introduced in the following subsections. Each view specifies a set of artefacts that are based on the stereotypes defined in the scope of this view. Additionally, the steps to create UMM-compliant artefacts are outlined. Later on, we demonstrate the most important steps by examples taken from our reverse engineering of the TPC-App specification. Since the reverse engineering approach results in realistic, but rather complex UMM models, we decided to simplify them due to space limitations and for educational purposes. In the simplified example the retailer is allowed to order books at a wholesaler. During the processing of the purchase order the wholesaler requests an authorization at a payment gateway.

3.1 Business Domain View (BDV)

The *business domain view* (BDV) is used to gather existing knowledge. Interviews with domain experts and stakeholder help to reveal *business processes* that require interactions with *business partners*. The *business processes* identified are recorded in a UML use-case diagram. In our example we assume that interviews with representatives of `wholesalers`, `retailers` and `payment organizations` expose the relevant *business processes* as shown in the example of figure 1.



Retailer
«BusinessProcess» obtain book details
«BusinessProcess» order books
«BusinessProcess» provide book details
«BusinessProcess» sell books
Wholesaler
«BusinessCollaborationUseCase» book wholesale management
Payment Gateway
«BusinessProcess» provide credit card verification
«BusinessProcess» verify credit card

Figure 1: Business processes of each business partner

3.2 Business Requirements View (BRV)

The *business requirements view* (BRV) builds upon the *business processes* discovered before. The challenge is finding possibilities for collaborations between the business partners. In other words, business collaborations spanning over some of the business processes are elaborated. For example, the *business processes* discovered in our example allow the creation of the *business collaboration use case* `book wholesale management` (see figure 1). Since this *business collaboration use case* must comply with all the requirements specified in the BDV's *business processes*, it is necessary to establish dependencies between them.

A *business collaboration use case* is a special kind of a *business process* where two or more *business partners* collaborate. Depending on whether two or more business partners are involved a *business collaboration use case* is either a binary or a multi-party one. The UMM artefact of the BRV is a use-case diagram depicting all relevant *business collaboration use cases*.

A *business collaboration use case* is usually a complex task that is split into subtasks. In UMM a multi-party *business collaboration use case* is always split into binary *business collaboration use cases*. This is due to fact that a UMM model serves as a kind of contract governing the data exchange and choreography commitments between business partners. Since most contracts are bilateral, UMM concentrates on binary business collaborations. Due to this fact, UMM is not able to model a nested interaction like TPC-App, since it includes the communication with a third business partner. Contacting a third partner is a decision internal to one partner and is out of scope of the collaborative space between the two original collaborating partners.

A complex *binary business collaboration use case* is also split into its subtasks. This split may be applied recursively. It ends if a split into subtasks is not possible any more. This lowest level corresponds to a so-called *business transaction use case*. It is the most basic building block to be used between two *business partners*.

A *business transaction use case* is detailed by the chorography of a *business transaction*, which is explained in more detail in subsection 3.3.

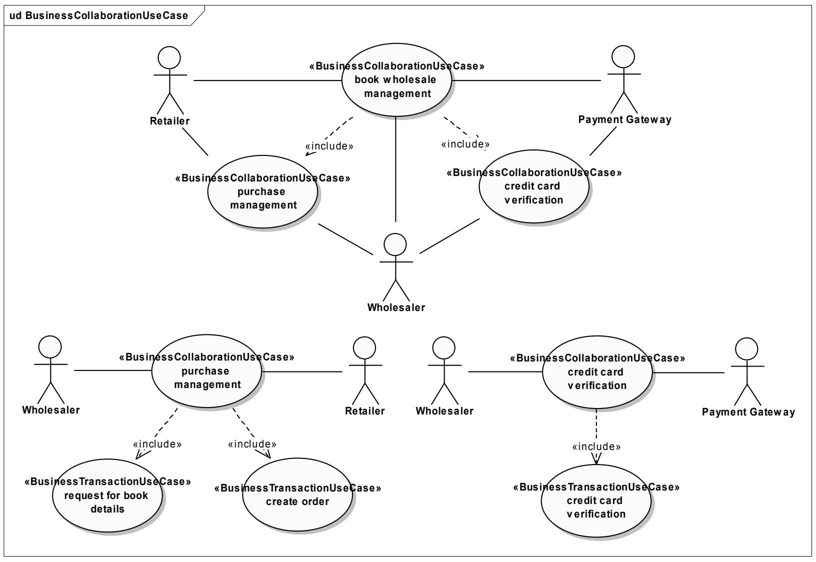

ud BusinessCollaborationUseCase
«BusinessCollaborationUseCase» book wholesale management
Retailer
Payment Gateway
«include»
«include»
«BusinessCollaborationUseCase» purchase management
«BusinessCollaborationUseCase» credit card verification
Wholesaler
Wholesaler
«BusinessCollaborationUseCase» purchase management
Retailer
Wholesaler
«BusinessCollaborationUseCase» credit card verification
Payment Gateway
«include»
«include»
«include»
«BusinessTransactionUseCase» request for book details
«BusinessTransactionUseCase» create order
«BusinessTransactionUseCase» credit card verification

Figure 2: Business collaboarion use cases

Figure 2 depicts the BRV artefacts of our simplified example: The multiparty *business collaboration use case* `book wholesale management` is decomposed into two binary *business collaboration use cases*: `purchase management` between `retailer` and `wholesaler` as well as `credit card verification` between `wholesaler` and `payment gateway`. The decomposition is denoted by a UML *<<include>>* association. The binary *business collaboration use case* `purchase management` is split into two included *business transaction use cases*: `request for book details` and `create order`. The *business collaboration use case* `credit card verification` is trivial and includes only one *business transaction use case* with exactly the same name. The fact that `credit card verification` is nested within `create order` cannot be expressed in UMM.

3.3 Business Transaction View (BTV)

The *business transaction view* (BTV) comprises amongst others three main artefacts. The first artefact is the so-called *business collaboration protocol*. It defines a choreography according to the requirements specified in the corresponding *business collaboration use case*. The resulting activity graph is built by so-called *busi-*

ness transaction activities. The transitions between these activities are guarded by the states of *business entities*.
Figure 3 shows the activity graph of the `purchase management` *business collaboration protocol*. It begins with a `request for book details`. This activity can be performed again if the result is not satisfying. It may also be the last activity with the result that no books are ordered. After a `request for book details` some books may be ordered by performing the `create order` *business transaction activity*. The *business collaboration protocol* always ends after `create order`.

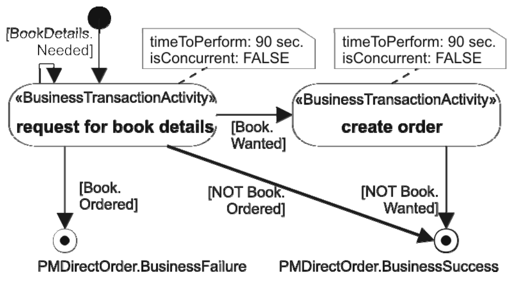


Figure 3: Business collaboration protocol

The second important artefact of the BTV is a *business transaction*. On the one hand side a *business transaction* defines a choreography according to the requirements specified in the corresponding *business transaction use case*. On the other hand side a *business transaction* presents an activity graph refining the corresponding composite *business transaction activity* of a *business collaboration protocol*.
The goal of a *business transaction* is synchronizing the *business entity states* between *business partners*. Sometimes a partner reports an irreversible state change (e.g. notification of shipment). This leads to a one-way *business transaction*. Sometimes the initiating partner sets a *business entity* into an interim state which is finally decided by the corresponding partner (e.g. create order). This leads to a two-way *business transaction*. Returning to a previous state after completion of a *business transaction* requires compensation by another *business transaction* (e.g. cancel order).
A *business transaction* always follows the same pattern. It is composed of two *partitions* – one for each participating *business partner*. The first partition includes the *requesting business activity* and the second one includes the *responding business activity*. An *object flow* from the *requesting business activity* to the *responding business activity* denotes the *request document*. An *object flow* in the reverse direction is optional and represents the *response document* in two-way transactions. The activity graph does not show the flow of *business signals* like *acknowledgments of receipt* and *acknowledgments of processing*. These are specified as *tagged values* of the *requesting/responding business activities*. Additional *tagged values* are *time to perform*, *authorization required*, *non-repudiation of origin and*

content, *non-repudiation of receipt*, and a *retry* counter. The document exchanges carry *tagged values* to signal *confidentiality*, *authentication* and *tamper proofness*. Figure 4 depicts the activity of the `create order` *business transaction*. The `buyer` performs the initiating role and starts a `place order` activity. This activity outputs the `order request envelope` which triggers the `process order` activity performed by the `seller`. The `place order` activity of the `buyer` does not end after sending the envelope, it receives the `order results envelope` from the `process order` activity of the `seller`.

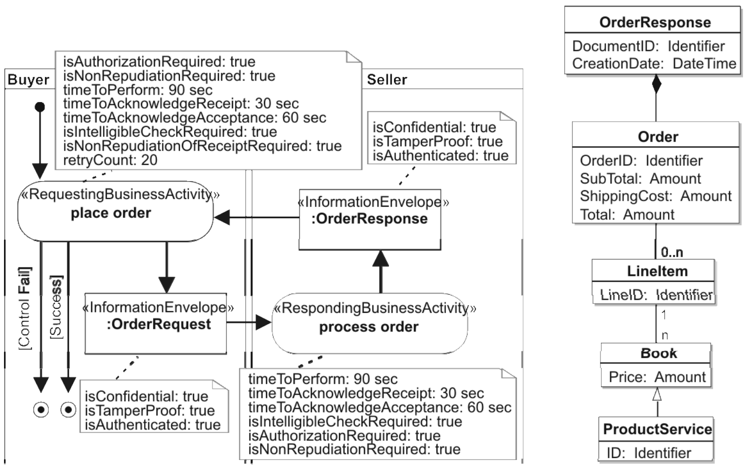


Figure 4: Business transaction and business document

The third artefact of the BTV is a *class diagram* describing the *business documents* exchanged. It is a structured representation of the *information envelopes* exchanged during a *business transaction*. The relevant information pieces are identified and structured. The final *class diagram* must be built from reusable building blocks guaranteeing reusability. The UN/CEFACT *core components* [UN03a] represent a library of such reusable building blocks. The class diagram of the `order response envelope` depicted on the right hand side of figure 4 uses *core components*.

4 UMM-App: Reverse Engineering TPC-App into UMM

4.1 Differences

As outlined in section 1, our goal is delivering a framework for B2B scenarios in order to measure the effects of using different technologies for the same business case. For this purpose we reverse engineering the TPC-App scenario into the technology independent UMM notation. Since our benchmark is based on both UMM and TPC-App we call it UMM-App. The resulting UMM-App scenario may be supported by another mix of technologies than the ones in TPC-App.

Reverse engineering would be trivial, if TPC-App and UMM were using exactly the same semantic concepts. However, there exist substantial differences between TPC-App and UMM. These differences are a result of the different goals of a benchmark and a modeling methodology. TPC-App was created to test an application server. Its only goal is to create a reasonable workload. Thus, it is designed to make the scenario most easy by providing the simplest implementation for the performance test while delivering reasonable results. A proprietary notation is sufficient, and reuse is no issue. UMM was created to unambiguously define B2B collaborations. It is suited to describe complex, real world B2B scenarios. Parts of UMM models may be used in other UMM models. Thus, reuse is a key issue.
TPC-App defines its B2B scenario from the local perspective of the wholesaler. This means that the application server to be tested must implement the role of the wholesaler. The benchmark program produces the workload created by the retailer. Furthermore, the application server must call emulators to create responses to the retailer. In contrary, UMM defines a collaboration from a global and neutral perspective. This means each business partner is treated equally. Thus, an implementation may not only test the application server of the wholesaler, but also of any other business partner involved.
As mentioned earlier, a retailer's request triggers nested operations along the supply chain (performed by emulators) before the wholesaler returns a response. UMM considers binary collaborations only. It is impossible to specify an interaction sequence that spans over more than two business partners. Thus, an additional UML activity diagram (which is not in the pure UMM scope) is needed to model the triggered operation calls.
TPC-App's web service interactions are not choreographed. TPC-App does not specify interdependencies between the different web services operations as they exist in the real world – e.g. a purchase order follows a previous request for quote. Although this negates B2B requirements, it helps to easily balance the number of calls of each web services interaction as defined in the web service interaction mix. Since UMM models real world scenarios, the definition of a choreography amongst the business transactions is of great importance.
UMM and TPC-App also differ in exception, retry, and failure handling. TPC-App specifies the acknowledgments on the network layer using a given set of technologies, like HTTP status messages and SSL. UMM defines acknowledgments on the business layer. An acknowledgment of receipt signals that the exchanged document is valid and kept the sequence defined in the choreography. An acknowledgment of processing indicates that the document has passed additional business rules

and is imported into the business application. There do not exist any of these business acknowledgments in TPC-App.
Finally TPC-App states the exchanged data as sequential parameter lists, whereas UMM assembles core components to a business document defined in class diagram.

4.2 A different level of complexity for different kinds of business transactions

TPC-App concentrates on the amount of the workload on the application server only. All the messages used as input and output to the web services must be syntactically valid. However, it is not evaluated whether their business content makes sense or not. For example, the payment gateway emulator always returns the same authorization code. Of course, this would be nonsense in a real business environment, but it is sufficient for pure workload tests.
In real world environments there exist different kinds of business transactions, each of a different level of complexity and, consequently requiring different level of technical support. A business transaction may lead to a new legally binding contract or not. For example, a purchase order usually leads to a new contract, whereas a request for book details does not. Commercial business transactions leading to a new contract have the highest security requirements, like non-repudiation, authentication, etc.
Business transactions not leading to a new contract may differ significantly according to their requirements. For example, a one-way transaction may be non-reputable due to a previously negotiated contract or not. For example, a shipment notification must be sent indicating that canceling the products is not possible anymore. Furthermore, different transactions require a different kind of processing by the responding partner. A business transaction may result in a simple database query where no input processing is required. Other ones lead to database transactions requiring input processing before the information can be processed. In some cases even pre-context validation is required. More complex business transactions may require modifications of the stored data. Updating the data in a database requires more complex operations than read-only database queries. UMM knows six different types of business transactions covering all the different situations described above. These six types of business transactions were also used in RosettaNet [Ro02] and are able to handle all interaction types as defined in the ISO standard of the Open-edi reference model [ISO97]. A B2B benchmark that considers the business level must provide a representative mix of different types of business transactions.
Furthermore, there does not exist a unique way of managing the sales/procurement process. Different buying scenarios can be identified. The search catalog scenario describes the order process via a catalog search. The buyer searches a catalog. She might order products of this catalog or she might request more information on a product prior to ordering. In addition to an order from catalog scenario, a so-called

direct order may be supported. In this scenario the buyer is aware of the catalog and has already decided what products to order. Consequently, there is no need to search the catalog or to obtain detailed information on the products. The buyer can order the products directly. Of course other scenarios to manage procurement/sales are feasible. A good B2B benchmark must consider a representative mix of different purchasing scenarios.

4.3 Adjustments made when moving from TPC-App to UMM-App

The differences between TPC-App and UMM described in section 4.2 and the complexity of B2B collaborations as described in 4.3 require some assumptions and adjustments when moving from TPC-App to UMM-App.
The most important adoption is made by adding a choreography to UMM-App. Since TPC-App is missing a choreography, the UMM-App choreography is defined according to best business practice. UMM-App does not mandate a single business collaboration protocol for the purchase management. Instead it specifies different alternative business collaboration protocols for the same purpose. Even within the same business collaboration protocol there exist different paths to follow. UMM-App defines how often (in percentage) each of the paths is executed. Since different business collaboration protocols partially use the same business transactions the re-use within a UMM model is demonstrated.
TPC-App defines the scenario from the local perspective of the wholesaler. We assume that the local perspectives of the other business partners are complementary to this one. In this case the global perspective mandated by UMM can simply be reverse engineered.
TPC-App does not distinguish different kinds of business transactions. By reverse engineering the TPC-App into UMM business transaction, we took care that each business transaction type is represented at least once in the UMM-App. Furthermore, the TPC-App does not specify any acknowledgements on the business level. Since these acknowledgments are vital for different kinds of business transactions, they have been added to UMM-App.
A UMM transaction identifies the document types to be exchanged. In order to define the documents we had to manually reverse engineer the TPC-App parameter lists. For each parameter we looked for a semantically matching core component in the library. Having identified all core components we assembled them to an appropriate class diagram.
UMM defines tagged values for different security requirements, e.g. non-repudiation, authentication, authorization, tamper-proofness, confidentiality. The protocols used in TPC-App give a hint on the security requirements involved. However, they are the same for any interaction in TPC-App. Thus, we declined to base the reverse engineering on these protocols. Instead we defined the security requirements according to the sensibility of the business transaction. Furthermore, UMM-Apps defines a percentage for failing to comply with these security re-

quirements for each business transaction. This mandates the application server to cope with business fault handling scenarios.
It is not possible to keep the exact TPC-App web service interaction mix in the UMM-App. However, UMM-defines how often each business collaboration protocol is executed. Since the flow of transactions as part of the business collaboration protocol is well defined and the percentages of executing alternative paths are specified, the UMM-App defines also a representative mix, which comes close to the one of TPC-App.
In order to keep the information about the nested interactions of TPC-App, we added an activity graph to the UMM-App that is not UMM-compliant, but specifies a flow of activities executed by a single business partner due to an incoming call in order to produce the response.

5 Positioning UMM-App

Our UMM-App is a technology independent B2B benchmark. Best to our knowledge, no other similar approach is described in the literature. Accordingly, a comparison with similar work is impossible. Instead, we want to position the role of UMM-App in delivering meaningful B2B systems.
The model driven architecture (MDA) approach of the OMG distinguishes three different views of a system that result in different kind of models [MM03]: (1) The *computation independent viewpoint* focuses on the environment of the system and its requirements. The details of the structure and processing of the IT system are unspecified. It leads to a *computation independent model (CIM)* that is familiar to the practitioners of the domain under consideration who do not need to care how to realize the functionality of an IT system. (2) The *platform independent viewpoint* focuses on the operation of an IT system while hiding the details necessary for a particular platform. It leads to a *platform independent model (PIM)* that exhibits a certain degree of platform independence in order to be suitable for a number of different platforms of similar type. (3) The platform specific viewpoint extends the *platform independent viewpoint* with an additional focus on the detail of the use of a specific platform by an IT system. It leads to *a platform specific model (PSM)* that is limited to a particular platform.
Crucial questions of the Pragmatic Web are first how to model and analyze collaboration, context, organizational commitments, and meaning negotiation; and second how to use these conceptual models in the design and implementation of real-world tools and applications [Sch06]. According to the Pragmatic Web Manifesto [SMD06] insights from the language action perspective (LAP) may help to answer the first set of question and may serve as a theoretical foundation for communication modeling and system design. An example of an LAP-based methodology is DEMO revealing the essential structure of business processes [Di06]. DEMO concentrates on a socio-economic system in which human beings in their role of social individuals involve in commitments and are bringing about the goods

or services that are delivered to the environment. It follows that a DEMO model must be considered as a kind of computation independent model.
The second set of questions mentioned above is already directed towards *platform specific models* and their implementations. A UMM model provides the intermediate layer – the platform independent model. It does not mandate a specific platform – like traditional EDI, ebXML or Web Services. However, it assumes that possible implementation technologies are of a similar type, i.e. some kind of message-oriented middleware connecting autonomous applications of business partners. It follows that some concepts that are kept abstract in the platform independent UMM models must be transformed when moving towards implementation to test a certain mix of technologies.
UMM describes business documents as an assembly of transfer-syntax independent core components [UN03a]. The document assembly may be mapped to traditional EDI standards like UN/EDIFACT [UN06b], to XML business document standards such as UBL [OA04a] and those mentioned in a survey by Li [Li00], or to RDF(S) and OWL base approaches as described in [HHW02,Om01]. The security requirements captured in UMM models must be reflected in the data exchanges. Some security requirements may be captured by the protocols used on the network layer, (e.g. HTTP, SMTP or Websphere MQ) or extensions to them such as the Secure Socket Layer (SSL). Other B2B requirements like business level acknowledgements may be solved on a "higher" network level. SOAP and its extensions WS-Reliable Messaging [OA04b], WS-Security [OA05], WS-Addressing [W3C04] are candidates for implementation as well as the ebXML messaging specification [OA02]. Also the chorography defined in UMM models may expressed in different machine-readable languages which allow the application server to dynamically load the choreography and to track and to validate corresponding business process instances. Candidate languages are BPEL [ACD03], BPML [Ar02], WS-CDL [Ka04] and ebXML BPSS [UN03b].

6 Conclusion

In this paper we have presented the UMM-App benchmark which is a platform-independent benchmark. It defines a representative B2B scenario. This scenario is based on the scenario used in the TPC-App benchmark which is a platform dependent benchmark based on a particular mix of Web Services specifications. We reverse engineered the TPC-App scenario to a platform independent UMM model. It is envisioned that the resulting UMM-App scenario is tested on different B2B platforms in order to prove the feasibility of each technological mix and its performance.
The current UMM-App defines a platform independent model that assumes to fulfill the requirements of a representative socio-economic system. However, the requirements of the underlying socio-economic system are not explicitly captured. A possible approach to describe this socio-economic system may use the language

action perspective such as provided by the DEMO methodology. This would require a transformation of the computation independent DEMO model to a platform independent UMM model. Although this transformation needs further investigations, it seems to be possible by looking at the main concepts of the two methodologies. The UMM model will make use of those coordination acts of a DEMO model in which a business application will support the individual in performing its act. The concept of DEMO transactions may be represented by UMM business transactions where the requesting business activity in UMM covers the request and accept acts of DEMO and, correspondingly, the responding business activity in UMM includes the promise and state acts of DEMO. The dependencies between acts of different transactions in DEMO (e.g. waiting conditions) must be reflected in the choreography of business transaction activities within the UMM business collaboration protocol. By extending the UMM-App with a computational independent DEMO model in the future, we hope to help operationalizing LAP-models into concrete benchmarks creating better socio-technical systems.

References

[ACD03] T. Andrews, F. Curbera, H. Dholakia, et al. *Business Process Execution Language for Web Services, Version 1.1.*, 2003, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbizspec/html/bpel1-1.asp

[Ar02] A. Arkin. *Business Process Modeling Language (Version 1.0), 2002,* http://xml.coverpages.org/BPML-2002.pdf

[BHL02] T. Berners-Lee, J. Hendler, O. Lassila. *The Semantic Web*. Scientific American, 2001

[Di06] J.L.G. Dietz. *The Deep Structure of Business Processes*, Communications of the ACM, Vol. 49, No. 5, 2006

[EN01] B. Eisenberg, D. Nickull D. *ebXML Technical Architecture Specification v1.0.4.*, 2001, ebXML Specifications, http://www.ebxml.org/specs/ebTA.pdf

[HF89] N.C. Hill, D.M. Ferguson. *Electronic data interchange: A definition and perspective*. EDI Forum: The Journal of Electronic Data Interchange Vol. 1, No. 1, 1989

[HHW02] B. Hofreiter, C. Huemer, W. Winiwarter. *Towards Syntax-Independent B2B*. ERCIM News, No. 51, October 2002

[ISO97] ISO. *Open-edi Reference Model*, ISO/IEC JTC 1/SC30 ISO Standard 14662, 1997, http://www.disa.org/international/is14662.pdf

[Ka04] N. Kavantzas et al. *Web Services Choreography Description Language, Version 1.0.*, W3C, 2004, http://www.w3.org/TR/ws-cdl-10

[Li00] H. Li. *XML and Industrial Standards for Electronic Commerce*. Knowledge Information Systems, Vol. 2, No. 4, 2000

[MM03] J. Miller, J. Mukerji, J.; *MDA Guide Version 1.0.1*; OMG omg/2003-06-01, 2003, http://www.omg.org/docs/omg/03-06-01.pdf

[OA02] OASIS. *ebXML Message Service Specification, Version 2.0*, 2002, http://www.oasis-open.org/committees/download.php/272/ebMS_v2_0.pdf

[OA04a] OASIS. *Universal Business Language 1.0*, 2004, http://docs.oasis-open.org/ubl/cd-UBL-1.0/

[OA04b] OASIS. *Web Services Reliable Messaging TC, WS-Reliability 1.1*, 2004, http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf

[OA05] OASIS. *Web Services Security: SOAP Message Security 1.1*, 2005, http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-pr-SOAPMessageSecurity-01.pdf

[Om01] B. Omelayenko. *Ontology Integration Tasks in Business-to-Business E-Commerce*, Proc. of 14th Int. Conf. on Industrial and engineering applications of artificial intelligence and expert systems: engineering of intelligent systems (IEA/AIE '01), Springer LNCS, June 2001

[Ro02] RosettaNet. *RosettaNet Implementation Framework: Core Specification, V02.00.01*, 2002, http://www.rosettanet.org/rnif

[Sch06] M. Schoop. *PragWeb 2006 – Call for Papers*, 2006, http://www.pragmaticweb.info/

[SMD06] M. Schoop, A. de Moor, J.L.G. Dietz. *The Pragmatic Web: a Manifesto*, Communications of the ACM, Vol. 49, No. 5, 2006

[TPC05] TPC. *TPC Benchmark App (Application Server) Specification, Version 1.1.1*, 2005, http://www.tpc.org/tpc_app/spec/TPC-App_V1.1.1.pdf

[UN03a] UN/CEFACT TMG. *Core components technical specification, Version 2.01*, 2003, http://www.untmg.org/dmdocuments/CCTS_v201_2003_11_15.pdf

[UN03b] UN/CEFACT TMG. *ebXML business process specification, Version 1.10*, 2003, http://www.untmg.org/dmdocuments/BPSS_v110_2003_10_18.pdf

[UN06a] UN/CEFACT TMG. *UMM Foundation Module, Version 1.0,* 2006, http://www.untmg.org/index.php?option=com_docman&task=docclick&Itemid=137&bid=15, 2

[UN06b] UN/CEFACT ICG. *UN/EDIFACT Standard Directories*, 2006, http://www.unece.org/trade/untdid/welcome.htm

[W3C04] World Wide Web Consortium. *Web Services Addressing*. W3C Member Submission, 2004, http://www.w3.org/Submission/ws-addressing/

[W3C06] World Wide Web Consortium. *Homepage of the Web Services Activity*, 2006, http://www.w3.org/2002/ws/