ISSN: 2167-1508
v7 n3318

Review of Web Service Specifications for Long-running Conversations

Chirag N. Rana
chirag.for.portal@gmail.com
Florida Blue
Jacksonville, FL - 32256

Karthikeyan Umapathy
k.umapathy@unf.edu
School of Computing
University of North Florida
Jacksonville, FL - 32224

Abstract

Despite the growing number of standards and interest in web services, support for implementation of long-running conversations is inadequate. Most real world business transactions typically consists of series of business activities. Such transactions originate from different sources which have multiple web services running to achieve a specific result. In this paper, we provide an overview of long-running conversation properties, and a review of relevant web service specifications. Our analysis indicates that WS-Coordination and WS-BusinessActivity specifications are the best option for implementing long-running conversations using web services.

Keywords: Web Service, Long-running Conversations, Transactions, Standards, and Specification Analysis.

1. INTRODUCTION

Business conversations are sequences of message exchanges among software components within a distributed system (Papazoglou, 2003). Conversations are of two types: short-lived and long-running. Short-lived conversations contain atomic transaction with a single unit of task or activity (Little, Maron, & Pavlik, 2004, p. 32). Long-running conversations can be a series of smaller transactions or activities and can take minutes, hours or even days to complete the transactions (Bowles & Moschoyiannis, 2008). Conversations are integral to system integration as they provide a means to model and implement the interactions between components to achieve interoperability. Web service is an ideal technology for systems integration including implementation of business conversations.

Web service is primarily a technology to integrate software systems and support machine-to-machine interactions over a network (Booth et al., 2004). It has emerged as a leading technology for systems integration as it is device-, language-, operating system-, and platform-neutral. Web service builds upon Service-oriented Architecture (SOA) to allow applications developed by different providers to be composed and coordinated in a loosely-coupled fashion. Web service provides its capability through a collection of standards. Web service technology provides standards such as WSDL (Web Services Description Language) for describing how to interface with a service, SOAP (Simple Object Access Protocol) for exchanging messages between services, and WS-BPEL (Web Services Business Process Execution Language) for composing multiple individual services into a single composite service.

Web services aide systems integration by supporting conversations between software components within and cross networks. Short-lived conversations are supported through WS-Atomic Transactions (WS-AT) standard (Newcomer, Robinson, Little, & Wilkinson, 2009). However, long-running conversations are supported by several competing standards such as Web Service Choreography Description Language (WS-CDL), Web Services Business Activity (WS-BA), Business Transaction Protocol (BTP), and Web Service Composite Application Framework (WS-CAF). The existences of competing standards cause difficulty for developers with selecting appropriate specification for supporting long-running conversations. In this paper, we provide a review and a comparison of support provided by these standards for implementing long-running conversations.

2. LONG-RUNNING CONVERSATIONS

Business conversation (a.k.a., business transactions) involves ordered sequence of interactions among multiple partners. Business conversations comprise of multiple sub-transactions or activities (Bowles & Moschoyiannis, 2008). Each activity involves execution of the underlying services and/or software components. These activities would need to share results with other services participating in the transaction. Dependencies between activities and corresponding services need to be coordinated in order to successfully complete a transaction. Thus, conversations can take minutes, hours, or even days to complete. Hence, they are known as long-running conversations (Bowles & Moschoyiannis, 2008). For example, Amazon's product authentication process could take a few minutes to days, depending upon the response received from the customer (Amazon-DevPay, 2014).

It is possible that sub-transactions within a long-running conversation consists of several sub activities that may be interdependent on each other (Bowles & Moschoyiannis, 2008). It means output of one sub activity may be the input of another sub activity. If one or more of the sub activities do not provide output to an activity whose execution is dependent on previous activity's results, then the subsequent activity may not execute and may result in failure of an entire transaction. For a long-running conversation to be considered as completely successful it should result in an acknowledgement or return code from the sub-transactions from which it expects the output.

Let us consider a travel booking scenario as an example of long-running conversations. In this scenario, a customer wants to book a flight, a hotel room, and a rental car. This scenario consists of multiple sub-transactions, namely, flight booking, hotel booking, rental car booking, and payment processing. The travel booking scenario involves following participants: customer, travel agent service, flight service, hotel service, rental car service, and payment service. Interactions between these participants must be orchestrated for successfully completing the scenario and achieving the business objective. It should be noted that travel agent service can interact with multiple flight, hotel, and rental car services. Figure 1 shows the sequence of interactions that occur in a successful completion of a travel booking scenario.

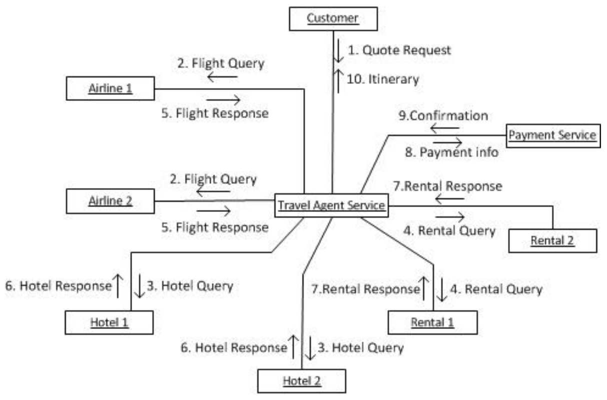

Figure 1. Collaboration Diagram for Travel Booking Scenario

3. LONG-RUNNING CONVERSATION PROPERTIES

The Service-oriented Architecture (SOA) provides an infrastructure to organize different applications involved in a long-running conversation and can be used, composed, and coordinated in the form of loosely coupled services. Each business transaction involves interaction between multiple service providers which needs to be structured and arranged in a definite sequence. These services require appropriate web service specifications that support long-running conversations.

In order to review the ability of web service standards to support long-running conversations, we need to be aware of the properties of long-running conversations. Below, we have identified key properties of long-running conversations in the context of web services.

ACID Properties

Ensuring consistency of information shared amongst services despite of concurrent accesses and system failure is very important for long-running conversations. Transaction systems achieve this through well-known ACID properties of Atomicity, Consistency, Isolation and Durability (Little, 2003). Atomicity property states that either all activities in a transaction are completed successfully, or none succeeds. Consistency property states that a transaction works on a consistent set of data and leaves a consistent state after the transaction is completed. Isolation property states that transactions should have concurrent access to the same data while maintaining integrity and correctness of data. Isolation also requires keeping a transaction hidden from other concurrently running transactions. Durability property states that when a transaction completes all changes made becomes persistent even if the system subsequently fails.

Relaxed ACID

It should be noted that not all long-running conversations need to maintain rigid ACID properties. Most long-running conversations may not be able to possess full atomicity and isolation properties (Dalal, Temel, Little, Potts, & Webber, 2003). Long-running conversations require an extended period of time to complete all sub-transactions. A participant might be inactive for an extended time period as often waiting to receive messages from other participants. Data used within conversations might also be shared among participants, thus, it cannot be locked by a participant (Razavi, Moschoyiannis, & Krause, 2007). For those scenarios, isolation of data is not required. Participants might be required to share partial results before sub-transactions being committed, thus, not all transactions will be atomic transactions (Little, 2003). Therefore, long-running specifications should provide support for relaxed ACID properties.

Fault Handling

Standard specification should provide ability to handle faults that arise from execution of business conversations. A business transaction needs to handle exceptions or errors that are thrown at any stage during the execution. A long-running conversation should successfully complete (commit) each of the sub-transactions or none at all (Bowles & Moschoyiannis, 2008). A long-running transactions may be atomic i.e. either complete successfully (commit) or not take place at all (roll back) or may not be atomic. But in either case, if there is a point of failure, there should be some mechanism which can revert back all the previous sub-transactions which are successful till the point of failure.

Therefore, in case of failure within a transaction, there must be an ability to undo portions of transactions that may have been completed so far. The Long-running transaction is an aggregation of sub-transactions, thus, there is likelihood that at some point a sub-transaction might fail. Failure can occur within a transaction for several reasons such as service unavailability, network disconnection, and application errors. Thus, support for compensating failure is a necessity.

Coordination Support

Long-running conversations, typically, consists of a series of sub-transactions. Executions of sub-transactions need to be coordinated to ensure interactions among participants follow expected logical sequence (Razavi, et al., 2007). Standards for long-running conversations should provide a centralized controller to manage distributed transactions in a loosely coupled manner. Specifications should also provide relevant protocols to ensure that the controller and the participating services are working together to complete transactions or cancel when it fails.

2PC Protocol

Two-phase commit protocol is a necessity for complex transactions with ACID like behaviors. The first phase is called the prepare phase which involves participants preparing for the transaction (declaring dependencies, setting up the relationships, and indicating scope and side effects of updates) (Razavi, et al., 2007). The second phase is called the commit phase, in which participants finalizes or aborts the transaction. Long-running specifications should provide support for two-phase commit protocol to ensure a group of distributed transactions succeed or fail as if they were a single transaction.

Arbitrary Transaction Models and Semantics

Transaction models and semantics supported within the specification should not be catered to a specific set of business domains. Transaction processing policies vary from one business domain to another (Hrastnik & Winiwarter, 2004). If a specification focuses on few domains, it cannot be used by other domains. Thus, specifications should support arbitrary transaction models and semantics.

Series of Message Exchanges

A business conversation is a series of message exchanges that occurs between participating entities involved in a business process (Papazoglou, 2003). Participating entities share information with each other through message exchanges. Transactional operations are also performed via message exchanges. Thus, support for series of message exchanges is a critical requirement for long-running specifications.

Multi-Participant Support

A typical long-running conversation not only includes a series of message exchanges but also two or more participants. Thus, the long-running specification supporting multiple participants is equally important.

State Information

Maintaining the current state of interactions is essential for monitoring progress of the transactions. State information of transactions, typically include transactional operation status such as pre-prepare, preparing, committing, aborting, and done. State information can also contain information about the interaction and message exchange between participants. Without state information, transaction systems could not track and report on the progress made within long-running conversations. Thus, long-running specifications should provide the ability to maintain state information.

Complements WS-BPEL specification

WS-BPEL specification provides ability to create a composite service from a set of individual services (Jordan & Evdemon, 2007). WS-BPEL specification can be used to define which service accomplishes a specific task within a business process. Identified services can be assembled in the order specified in the business process to function as a single composite service. WS-BPEL specifies and manages order of invocations of participating services, but it does not manage business conversations and transactional operations between participating services. For some simple business processes, WS-BPEL might be sufficient to achieve the business objectives. However, for scenarios involving long-running conversations, there is a need of complementing specifications. Any long-running specification should complement WS-BPEL without which its purposes might be meaningless for users.

Tool Support

Availability of tool support is critical for adoption of any standard specifications, without which a specification is just an abstract entity. Thus, availability of tool support is important for adoption of long-running conversation specifications as well. Tool support is required for modeling (to design) conversations and mechanisms that act as transactional systems to execute conversations.

4. SPECIFICATIONS THAT SUPPORT LONG RUNNING CONVERSATIONS

In this section, we provide an overview of web service specifications designed to support long-running conversations. We also provide a review of support provided to long-running conversation properties identified in the previous section.

Web Services Choreography Description Language (WS-CDL)

The WS-CDL specification is a set of rules and regulations defined at a global level which consists of common ordering conditions and restrictions for the participants exchanging messages (collaborate) with each other (Kavantzas et al., 2005). The primary goal of the WS-CDL specification is to specify a declarative, XML based language that defines from a global viewpoint the common and complementary observable behavior specifically, the information exchanges that occur and the jointly agreed ordering rules that need to be satisfied. Some of the goals of this specification are re-usability, cooperation, multi-party collaboration, semantics, composability, modularity, information driven collaboration, information alignment, exception handling, and transactionality (Kavantzas, et al., 2005). The participants develop their solutions which comply with these global definitions of rules in order to achieve successful messaging within or across their domains.

WS-CDL is not an "executable business process description language" or an implementation language (Fredlund, 2006). The implementation is handled by the participants who are involved in the conversation. WS-CDL specification is developed by W3C. The current version is 1.0 with candidate recommendation status.

Below, we provide a list of the main elements in the WS-CDL specification (Kavantzas, et al., 2005):

Choreography: *Choreography* is the root element of the specification. It defines rules regulating the ordering of message exchanges and pattern of collaborative behaviors agreed upon between interacting participants.

Collaborating participant: This element defines how a participant is capable of engaging in collaborations with different participants. *RoleType*, *RelationshipType*, *ParticipantType*, and *ChannelType* are the elements that define collaborating participants and their coupling. All interactions specified occur between *roleTypes* being exhibited by *participantTypes* and constrained by *relationshipTypes*.

Information driven collaborations: Observable collaborating behaviors of participants can be defined within a choreography element via *variable*, *token*, and *informationType elements*. *Variable* captures information regarding message exchanges, state changes of a *roleType*, *channel* information, and exceptions. *Tokens* can be used to refer a part of a variable. *InformationType* defines the type of information contained within a *variable* or referenced by a *token*.

Activities: Activities describe actions performed within the choreography via *basic activity*, *ordering structure*, and *workunit* elements. A *basic activity* is the lowest level of actions performed within *choreography*. The *Ordering structure* combines *basic activities* in a nested way to describe ordering rules within *choreography*. *Workunit* allows defining conditions and repetition of a group of activities.

Table 1 provides an overview of long-running conversation properties supported by WS-CDL specification. From the table, it can be noted that WS-CDL provides full support for 6 out of 11 identified properties and provides partial support for 3 other properties.

Web Service Coordination (WS-Coordination) and Web Service Business Activity (WS-BusinessActivity)

WS-Coordination and WS-BusinessActivity specifications collectively are designed to support long-running conversations. WS-Coordination is a building block that is used in conjunction with other specifications and application-specific protocols to accommodate a wide variety of protocols related to the operation of distributed web services. Web services can be used to bind together a large number of participants to form a large distributed computational unit. WS-Coordination defines a coordinator and a set of coordination protocols for coordinating activities performed by participating web services (Newcomer, Robinson, Feingold, & Jeyaraman, 2009). WS-Coordination should be used when interoperability is needed across vendor implementations and trusted domains. Thus, the protocols defined in this specification can be combined with proprietary protocols within the same application (Newcomer, Robinson, Feingold, et al., 2009).

WS-Coordination specification became an industry standard in 2009 and the current version is 1.2. WS-Coordination is developed by and recommended by OASIS. Below, we provide a list of the main elements in the WS-Coordination specification (Newcomer, Robinson, Feingold, et al., 2009):

CoordinationContext: The *CoordinationContext* element is used by applications to pass coordination information such as *registration service* and *coordination type* to participants involved in an activity.

CoordinationService: CoordinationService represents coordinator, which is essentially an aggregation of *activation service*, *registration service*, and *coordination type*. *Activation service* defines a CreateCoordinationContext operation which creates a new activity and returns a *CoordinationContext*. Registration service defines a Register operation which allows a web service to register for an activity using CoordinationContext. *Coordination type* defines coordination behaviors such as accepted service contexts, protocol registrations, and protocols associated within an activity.

Fault: The coordination *fault* allows defining endpoints that can be used when preset fault conditions are met.

Security Model: WS-Coordination specification works in conjunction with WS-Security and WS-Trust specifications to secure message exchanges and other communications between participants.

WS-Business Activity defines protocols that enable existing business process and workflow systems to wrap their proprietary mechanisms and interoperate across trusted boundaries and different vendor implementations (Newcomer, Robinson, Freund, & Little, 2007). WS-BusinessActivity builds its protocols based on the extensible coordination framework from WS-Coordination specification. Business activities can be partitioned into hierarchical nested scopes. Results of completed tasks (e.g., transactions) within business activities are visible prior to the completion of the business activity, thereby relaxing isolation. These tasks are in fact tentative, thus, the business logic for compensation relies on the outcome (Newcomer, Robinson, et al., 2007).

To undo completed child tasks, compensating actions are registered with the parent activity. Exceptions are handled by exception handlers (which may be compensating actions) using application logic in such a way that the overall business activity can continue (i.e., forward recovery) (Newcomer, Robinson, et al., 2007). Business activities are allowed to query multiple participants (i.e., child tasks) in order to finally select the most appropriate one and cancel others. The participants of a business activity also could be coordinated in an all or nothing fashion by the coordinator. The coordinator in a business activity is not as restricted as the coordinator in an atomic transaction. The behavior of the coordinator will be determined by the application. Participants are allowed to exit activities autonomously thereby, depending on the position of the protocol, delegating processing to other scopes or exiting without knowing the outcome of the protocol.

WS-BuinessActivity specification became an industry standard on 2007 and the current version is 1.1. WS-BusinessActivity is developed by and recommended by OASIS. Below, we provide a list of the main elements in the WS-BusinessActivity specification (Newcomer, Robinson, et al., 2007):

Coordination Types: WS-BusinessActivity supports two coordination types, which are AtomicOutcome and MixedOutcome. For AtomicOutcome, the coordinator must direct all participants to either close or compensate. For MixedOutcome, the coordinator can direct individual participants to either close or compensate.

Coordination Protocols: WS-BusinessActivity supports two coordination protocols, which are BusinessAgreementWithParticipantCompletion (A participant knows when it has completed a business activity) and BusinessAgreementWithCoordinatorCompletion (A participant relies on coordinator to know when it has completed a business activity).

Table 2 provides an overview of long-running conversation properties supported by WS-Coordination and WS-BusinessActivity specifications. From the table, it can be noted that WS-Coordination and WS-BusinessActivity specifications provide full support for 8 out of 11 identified properties and provides partial support for one other property.

Business Transaction Protocol (BTP)

BTP specification defines possible roles within a transaction, message exchanges between roles, meanings of messages and their permitted ordering sequences. Its purpose is to provide the interactions (or signaling) required to coordinate the effects of application protocols to achieve a business transaction (Ceponkus et al., 2002).

In a real world business-to-business (B2B) paradigm, maintaining the ACID property in a loosely coupled, distributed web services are practically not possible. We cannot satisfy the ACID property 100% because in case of complex failures, it requires to use compensating transactions. Typical locking techniques introduce problems in long running transactions, so there is a need to design complex lock management algorithms or new interaction techniques (Dalal, et al., 2003).

BTP is designed to allow coordination of web services using a two-phase coordination protocol to ensure that consistent results are achieved. BTP also provides the participants' ability to record before or after images of a transaction operation. It also provides flexibility for participants to compensate with rollfoward-rollbackward capability.

BTP specification is developed by OASIS and it is currently a committee specification as of June 2002. Below, we provide a list of main concepts in the BTP specification (Ceponkus, et al., 2002):

Elements: Each participant in BTP has two elements, application element and BTP element. Application element holds the order of message exchange information and associated business functions to be performed. BTP elements assist application to get work done by sending and receiving messages.

Actors and Roles: BTP establishes bilateral relationships between two applications using actors and role concepts. The role concept refers to the role played by a participating application within a transaction. Participating application is called as an actor. An actor can perform different roles within or in different transactions. There are two key roles in BTP - superiors (nodes that coordinate a transaction) and inferiors (nodes that participate in a transaction coordinated by another node) (Dalal, et al., 2003). There is also a possibility that the roles can be interchanged during a transaction. A BTP element can also implement both roles, which allows the creation of tree structures.

Atoms and Cohesions: There are two kinds of transaction behaviors supported by BTP: atoms and cohesive. In transactions with atomic behavior (also called atoms) all elements contributing to a transaction must eventually reach the same conclusion about a transaction (confirm or cancel) (Schmit & Dustdar, 2005). Atoms behavior supports all or nothing transactions, i.e., either the transaction is 100% successful or it does not take place at all. Isolation is relaxed in atom transactions to support long-lived transactions. Transaction with cohesive behavior (also called Cohesions) allows some sub elements to cancel while others confirm, which is useful in the case of different providers offering similar services (Schmit & Dustdar, 2005). Cohesions relax the atomicity of the transactions. The behavior can be different for different nodes within a BTP transaction tree, which allows for the construction of complex transaction patterns. The set of participants with "confirm" actions are the ones who successfully complete the business transactions and are known as confirm-set. Complex long running business transactions are modeled using cohesions.

Table 3 provides an overview of long-running conversation properties supported by BTP specification. From the table, it can be noted that BTP provides full support for 10 out of 11 identified properties and provides partial support for one other property.

Web Service Composite Application Framework (WS-CAF)

WS-CAF defines a framework for composite services that needs to specify boundaries for activities, manage context information, and inform participants of changes to an activity (WS-CAF, 2005). The purpose of WS-CAF is to enable development of composite services encompassing range of transaction models, coordination of activities, and recoverable long-running activities.

WS-CAF consists of three specifications (WSCAF-XML, 2003): Web Service Context (WS-Context) - a framework for managing contextual information such as IDs, tokens, channels, and address (Newcomer et al., 2007); Web Service Coordination Framework (WS-CF) - a sharable mechanism to manage context augmentation and lifecycle, and guarantee message delivery (WS-CF, 2004); and Web Service Transaction Management (WS-TXM) - comprising protocols for interoperability across multiple transaction managers and supporting two phase commit, long running actions, and business process flows transaction models (Bunting et al., 2003b).

An implementation of WS-CAF can start with a simple context management using WS-Context. Subsequently, additional context management features and message delivery guarantees can be added using WS-CF, and finally, transactional recovery mechanisms can added using WS-TXM (WSCAF-XML, 2003). Under WS-CAF umbrella, multiple web service specifications can be combined in various ways to achieve a common goal. Hence, the minimum requirement is to share a common context, and the maximum requirement is to coordinate results in a potentially long-running unit of work with predictable results including failure conditions (WSCAF-XML, 2003).

WS-CAF is developed by OASIS. WS-CF and WS-TXM were proposed but were not developed into standard specifications. Only, WS-Context was recommended as an OASIS standard. Below, we provide a list of main concepts in the WS-Context specification (Newcomer, Chapman, et al., 2007):

Context structure: Context structure defines nested structured models for organizing context information.

Context service: Context service defines scope of an activity and how context information can be referenced and broadcasted.

ISSN: 2167-1508
v7 n3318

Context manager: Context manager defines how applications can retrieve and set associated data with a context.

We have also provided an overview of WS-TXM and WS-CF specifications even though they were proposed but were never recommended as standards. WS-TXM supports three kinds of transactional models (Bunting, et al., 2003b):

ACID transactions: ACID transaction model supports short-running transactions that require ACID properties. It supports two-phase protocols and fault handling for recovering from exceptions.

Long-running action transactions: Long-running action (LRA) transactions are designed for transactions with long duration. It does not support ACID transactions but supports all or nothing properties. It provides compensation support for faults, but does not support two-phase protocol.

Business process transactions: Business process transaction model allows an activity or a group of activities that is responsible for performing tasks pertaining towards a specific business domain. A business process transaction can be structured as a collection of ACID or LRA transactions depending upon the business domains and process requirements.

WS-CF specification consists of four main components (WS-CF, 2004): registration service (that allows a participant to register with a specific protocol), participant service (allows defining operations performed by a participant as a part of the protocol), registration context (allows a participant to join an activity group), and recovery service (provides capability for transaction systems recover from failures).

Table 4 provides an overview of long-running conversation properties supported by WS-CAF specification. From the table, it can be noted that WS-CAF provides full support for 6 out of 11 identified properties and provides partial support for 3 other properties.

5. SPECIFICATION ANALYSIS DISCUSSION

The analysis presented in tables 1 to 4 indicates that none of web service long-running conversation specifications provide full support for all 11 properties identified in section 3.

In regards to WS-CDL, service interactions are represented in a peer-to-peer structure and not at the individual participant level. WS-CDL specification does not include a coordination mechanism to coordinate interactions between services. The purpose of WS-CDL specification is to provide the ability to specify message exchange sequences but not as an actual implementations of long-running conversations. The actual implementation is handled by the participants involved in the conversation. WS-CDL does not provide support for ACID and 2PC protocol transactions. Thus, industry adopters did not find WS-CDL specification to be useful. WS-CDL specification stagnated at the candidate recommendation stage and never progressed to become a full W3C recommended standard. Development of tools implementing WS-CDL was a necessity. Due to lack of industry adoption and tool development, WS-CDL standardization process stagnated (Umapathy, Purao, & Bagby, 2012).

Pi4SOA is an eclipse based tool which provides a graphical editor to write choreographies and generate BPEL from those choreographies. The tool can validate whether a developed specification document conforms to WS-CDL rules. However, it doesn’t identify any design errors. Pi4SOA does not provide any tools to verify whether the document is fault free (Caliz, Umapathy, Sánchez-Ruíz, & Elfayoumy, 2011). Given the lack of adequate tool support, coordination mechanism, and transaction protocol support, we can conclude that WS-CDL specification is not a good fit for web service long-running conversations.

WS-Coordination and WS-BusinessActivity specifications are designed for handling business transactions. Thus, they support many of the identified properties. They do not provide full support for ACID transactions and 2PC protocol, as they have developed WS-AtomicTransaction specification for that purpose. Thus, one can use WS-Coordination with WS-AtomicTransaction for those contexts. Depending upon the requirements, multiple short-lived transactions might have to be created to satisfy the long-running conversation requirements.

WS-Coordination and WS-BusinessActivity does not provide any support for managing and monitoring peer-to-peer service interactions. For long-running conversations in the context of integrating disparate systems, having an ability to model and monitor peer-to-peer service interactions are important (Umapathy, 2009). WS-CDL specification is designed for specifying peer-to-peer service interactions, so far, we

ISSN: 2167-1508
v7 n3318

have not identified any work that brings together WS-CDL, WS-BusinessActivity, and WS-Coordination. Despite lack of peer-to-peer message exchange support, WS-Coordination and WS-BusinessActivity is the best option for developers as these specifications are OASIS standards and some tool supports are available for technical implementation.

BTP specifications provide more coverage than other three specifications considered. Similar to WS-BusinessActivity, BTP does not provide support for managing and monitoring peer-to-peer service interactions. However, the major problem with BTP specification is that it never became an OASIS standard. It remains as a committee draft and the business transaction committee which was responsible for developing BTP was closed in 2006 due to the lack of progress and inactivity with specification development (BTPMailArchive, 2006). BTP does not have enough activities in tools development arena either. Java Open Transaction Manager (JOTM)-BTP was last updated on 2004. Given that BTP is not an OASIS standard and lack of updated tool support, we can conclude that BTP is not a viable option. It should be noted that the combination of WS-Coordination, WS-AtomicTransaction, and WS-BusinessActivity provides coverage similar to BTP.

WS-CAF provides a layered implementation that supports long running conversations and is compatible with any transaction protocol (Bunting et al., 2003a). The benefit of layered implementation is that development can be started with basic and move towards complex implementation based on the requirements. This framework breaks down the problem into layers and hence the implementation is modularized into three specifications. WS-Context provides the basic context sharing mechanism and defines the context as a web service. WS-TXM defines semantics of each business transaction which does not try to define everything globally but makes model for each problem domain. WS-CF defines coordinator for transaction models. WS-CAF is designed to support both short-lived and long-running conversations. However, many of the properties made available for short-lived transactions are not made available for long-running conversations.

Major problem with WS-CAF is lack of progress with the standardization process. Only WS-Context was declared as an OASIS standard. Both WS-TXM and WS-CF was not declared as standards. Similarly, the lack of standardization led to a lack of industry adoption and tool development. Thus, there is no tool support for WS-CAF, which makes it the least viable specification for long-running conversations.

6. CONCLUSION

Long-running conversations are required in cases where transactions are complex and consist of a series of sub-transactions. Most of the established companies rely upon long-running conversations to provide their services. A typical real world business process involves dozens of activities with multiple participants and several interactions between them. Standards developed specifically for long-running conversations should provide infrastructure for interconnecting services and orchestrating dependencies and interactions between services.

A comprehensive review of web service standards that support long-running conversations is possible only when properties of long-running conversations are known. We have identified 11 properties related to long-running conversations. We reviewed four relevant web service specifications that support long-running conversations. Our analysis reveals that two specifications – WS-Coordination & WS-BusinessActivity and BTP – have more coverage for long-running conversations in comparison to other specifications.

WS-Coordination & WS-BusinessActivity is a better option than BTP as it is an OASIS standard, and has adequate and up-to-date tool support. Another limitation of BTP in comparison to WS-Coordination & WS-BusinessActivity is that BTP does not have an ability to provide partial results. It is the responsibility of business process designers to implement such mechanism. This results in introducing new transactions to get around the problem in case of failure conditions. However, WS-BusinessActivity provides the ability to see results of completed tasks within business transactions prior to the completion of the transaction. In long-running conversations, a transaction can span a long period of time involving multiple participants and rely on the outcomes of the task that is to be compensated. Thus, partial results of tasks within a business transaction prior to completion can be helpful in determining appropriate business logic to compensate and recover from a failure.

We conclude with a recommendation to use WS-Coordination and WS-BusinessActivity for implementing long-running conversations in the context of web service based solutions. WS-Coordination and WS-BusinessActivity does have potential for further improvement. It does not provide support for managing and monitoring peer-to-peer interactions. Additional research work is necessary for addressing this requirement.

7. REFERENCES

Amazon-DevPay. (2014). Amazon DevPay Desktop Product Authentication Process Retrieved September 3, 2014, from http://docs.aws.amazon.com/AmazonDevPay/latest/DevPayDeveloperGuide/DesktopOverallProcess.html

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., & Orchard, D. (2004, 11 February 2004). Web Services Architecture - W3C Working Group Note Retrieved March 25, 2014, from http://www.w3.org/TR/ws-arch/

Bowles, J., & Moschoyiannis, S. (2008). When Things Go Wrong: Interrupting Conversations. In J. L. Fiadeiro & P. Inverardi (Eds.), *Fundamental Approaches to Software Engineering* (Vol. 4961, pp. 131-145). Berlin Heidelberg: Springer

BTPMailArchive. (2006, February 01). Plan to close OASIS Business Transaction committee Retrieved July 12, 2014, from https://lists.oasis-open.org/archives/business-transaction/200602/msg00000.html

Bunting, D., Chapman, M., Hurley, O., Little, M., Mischkinsky, J., Newcomer, E., . . . Swenson, K. (2003a, July 28). Web Services Composite Application Framework (WS-CAF) Primer Retrieved July 12, 2014, from https://www.oasis-open.org/committees/download.php/4343/WS%EE%93%93CAF%20Primer.pdf

Bunting, D., Chapman, M., Hurley, O., Little, M., Mischkinsky, J., Newcomer, E., . . . Swenson, K. (2003b, July 28). Web Services Transaction Management (WS-TXM) Retrieved July 11, 2014, from http://xml.coverpages.org/WS-TXM-200310.pdf

Caliz, E., Umapathy, K., Sánchez-Ruíz, A. J., & Elfayoumy, S. A. (2011). Analyzing Web Service Choreography Specifications Using Colored Petri Nets. In H. Jain, A. Sinha & P. Vitharana (Eds.), *Service-Oriented Perspectives in Design Science Research* (Vol. 6629, pp. 412-426). Berlin Heidelberg: Springer

Ceponkus, A., Dalal, S., Fletcher, T., Furniss, P., Green, A., & Pope, B. (2002, 3 June). Business Transaction Protocol Retrieved July 9, 2014, from https://www.oasis-open.org/committees/download.php/1184/2002-06-03.BTP_cttee_spec_1.0.pdf

Dalal, S., Temel, S., Little, M., Potts, M., & Webber, J. (2003). Coordinating business transactions on the Web. *IEEE Internet Computing, 7*(1), 30-39. doi: 10.1109/mic.2003.1167337

Fredlund, L.-Å. (2006). *Implementing WS-CDL.* Paper presented at the Spanish workshop on Web Technologies, University of Santiago de Compostela.

Hrastnik, P., & Winiwarter, W. (2004). *An Advanced Transaction Meta-Model for Web Services Environments.* Paper presented at the Information Integration and Web-based Applications & Services, Jakarta, Indonesia.

Jordan, D., & Evdemon, J. (2007, April 11). Web Services Business Process Execution Language (WS-BPEL) Version 2. Retrieved July 5, 2014, from http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., & Barreto, C. (2005, 9 November). Web Services Choreography Description Language (WS-CDL) Version 1.0 Candidate Recommendation. Retrieved July 5, 2014, from http://www.w3.org/TR/ws-cdl-10/

Little, M. (2003). Transactions and Web services. *Communications of the ACM, 46*(10), 49-54. doi: 10.1145/944217.944237

Little, M., Maron, J., & Pavlik, G. (2004). *Java Transaction Processing: Design and Implementation* (1st ed.). Upper Saddle River, NJ, USA: Prentice Hall.

Newcomer, E., Chapman, M., Little, M., Little, M., Newcomer, E., & Pavlik, G. (2007, 2 April). Web Services Context Specification (WS-Context) Retrieved July 11, 2014, from http://docs.oasis-open.org/ws-caf/ws-context/v1.0/OS/wsctx.html

Newcomer, E., Robinson, I., Feingold, M., & Jeyaraman, R. (2009, 2 February 2009). Web Services Coordination (WS-Coordination) Version 1.2 Retrieved July 8, 2014, from http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec-os/wstx-wscoor-1.2-spec-os.html

Newcomer, E., Robinson, I., Freund, T., & Little, M. (2007, 12 July). Web Services Business Activity (WS-BusinessActivity) Version 1.1 Retrieved July 8, 2014, from http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec/wstx-wsba-1.1-spec.html

Newcomer, E., Robinson, I., Little, M., & Wilkinson, A. (2009, 2 February). Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.2 Retrieved March 25, 2014, from http://docs.oasis-open.org/ws-tx/wstx-wsat-1.2-spec-os/wstx-wsat-1.2-spec-os.html

Papazoglou, M. P. (2003). Web Services and Business Transactions. *World Wide Web, 6*(1), 49-91.

Razavi, A. R., Moschoyiannis, S. K., & Krause, P. J. (2007, 21-23 Feb. 2007). *A Coordination Model for Distributed Transactions in Digital Business EcoSystems.* Paper presented at the IEEE-IES Digital EcoSystems and Technologies Conference, Cairns, Australia.

Schmit, B. A., & Dustdar, S. (2005, 19 July 2005). *Towards transactional Web services.* Paper presented at the IEEE International Conference on E-Commerce Technology Workshops, Munich, Germany.

Umapathy, K. (2009). *From Service Conversation Models to WS-CDL.* Paper presented at the Americas Conference on Information Systems (AMCIS), San Francisco, California, USA.

Umapathy, K., Purao, S., & Bagby, J. (2012). Empirical analysis of anticipatory standardization processes: a case study. *Information Systems and E-Business Management, 10*(3), 325-350. doi: 10.1007/s10257-011-0169-1

WS-CAF. (2005). Web Services Composite Application Framework (WS-CAF) Retrieved March 25, 2014, from http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-caf

WS-CF. (2004, December 22). Web Services Coordination Framework Specification (WS-CF) Retrieved July 11, 2014, from https://www.oasis-open.org/committees/download.php/10889/WSCF-Working-12-22.pdf

WSCAF-XML. (2003). Web Services Composite Applications Framework (WS-CAF) Retrieved July 11, 2014, from http://xml.coverpages.org/WSCAF-Announce.html

Appendices

Table 1. WS-CDL support for long-running conversations

Properties	Support	WS-CDL Elements	Remarks
ACID Properties	Partial support	Consistency supported by workunit element and isolation supported by choreography element, but no support for atomicity and durability	It should be noted that WSCDL is not designed for supporting transactional systems. Support for isolation and consistency from a message exchange perspective not from the transactional data perspective.
Relaxed ACID	No support	None	
Fault Handling	Full support	Workunit and Choreography	Faults can be handled by creating exception workunit within the exceptionblock of a choreography.
Coordination Support	Partial support	Choreography and coordination protocol	WS-CDL does define a coordination protocol. Participating applications would have to come up agreed upon coordination protocol.
2PC Protocol	No support	None	
Arbitrary Transaction Models and Semantics	Full support	Not applicable	WS-CDL is domain and transactional model independent.
Series of Message Exchanges	Full support	Activities	Ordering structure within activities allows arranging different basic activities in a nested structure to achieve an objective.
Multi-Participant Support	Full support	Roletype, relationshipType, and participantType.	Specified elements can be collectively used to define engagement among multiple participants.
State Information	Full support	Variables	Variables can be used to capture the state information such as observable changes of a roleType.
Complements WS-BPEL specification	Full support	Not applicable	WS-CDL functions on a layer top of WS-BPEL.
Tool Support	Partial support	Pi4SOA (http://sourceforge.net/projects/pi4soa/), last updated on April 25, 2013.	Supports only development of WS-CDL specification but not executing it.

Table 2. WS-Coordination and WS-BusinessActivity support for long-running conversations

Long-Running Properties	Support	WS-Coordination & WS-BusinessActivity Concept/Elements	Remarks
ACID Properties	Partial support	Coordination types and protocol from WS-BusinessActivity	Provides weaker support for isolation and durability. It is possible to use a combination of shorter WS-AtomicTransactions to achieve full ACID properties.
Relaxed ACID	Full support	Coordination types and protocol from WS-BusinessActivity	Offers flexibility for atomicity, isolation, and durability.
Fault Handling	Full support	Faults from WS-Coordination	Provides five different preset fault message
Coordination Support	Full support	Coordination service from WS-Coordination	Supports coordination between participants via activation, registration, and coordination protocol services.
2PC Protocol	No support	Coordination service from WS-Coordination	Participants can commit immediately after an activity
Arbitrary Transaction Models and Semantics	Full support	Not applicable	WS-BusinessActivity is domain and transactional model independent.
Series of Message Exchanges	No support	None	Only support pre-defined messages for coordination service purposes, but does not provide support for application specific message exchanges.
Multi-Participant Support	Full support	Coordination service from WS-Coordination	Allows multiple participants to register for a coordinationcontext.
State Information	Full support	Coordination protocol from WS-BusinessActivity	Provides accepted states for coordinator and participant.
Complements WS-BPEL specification	Full support	Not applicable	Both specifications can function independently and along with WS-BPEL
Tool Support	Full support	JBoss Transactions – Narayana (http://narayana.jboss.org/), last updated on June 2, 2014.	Provides full implementation of WS-Coordination, WS-BusinessActivity, and WS-AtomicTransaction

Table 3. BTP support for long-running conversations

Long-Running Properties	Support	BTP Concept/Elements	Remarks
ACID Properties	Full support	Atoms	Supported via business transactions consisting of superior and inferior nodes, with a superior being atom coordinator.
Relaxed ACID	Full support	Cohesions	Supported via business transactions consisting of superior and inferior nodes, with superior being cohesion composer.
Fault Handling	Full support	Recovery and failure handling	Handles communication, network, and system failures.
Coordination Support	Full support	Business transaction (superior)	Superior nodes in business transaction trees act as a coordinator.
2PC Protocol	Full support	Business transaction	Two-phase outcome is managed by transitioning events between superior and inferior nodes.
Arbitrary Transaction Models and Semantics	Full support	Not applicable	BTP is domain and transactional model independent.
Series of Message Exchanges	Partial support	Message sequence	Provides guidance on message exchange sequence for transaction coordination but not for coordinating conversations to achieve an objective.
Multi-Participant Support	Full support	Business transaction trees	Superior and inferior relationships allows multiple participants to take part in a transaction.
State Information	Full support	State tables	Provides state tables for superior and inferior roles along transitions between states.
Complements WS-BPEL specification	Full support	Not applicable	BTP can function independently and along with WS-BPEL
Tool Support	Full support	JOTM-BTP (http://jotm.objectweb.org/jotm-btp.html) Last updated: July 1, 2004	BTP extension implemented on top of Java Open Transactions Manager.

Table 4. WS-CAF support for long-running conversations

Long-Running Properties	Support	WS-CAF Concept/Elements	Remarks
ACID Properties	Full support	ACID transaction model from WS-TXM	Provides full functionality of ACID transactions.
Relaxed ACID	Full support	Long running action (LRA) from WS-TXM	Considers each activity to be a LRA. Multiple LRA's are nested to create a long-running conversations.
Fault Handling	Full support	Recovery and compensator from WS-TXM	Recovery is used for ACID transactions and compensator is used for LRA.
Coordination Support	Full support	Coordinator from WS-TXM	Provides separate coordinator for ACID and LRA transactions.
2PC Protocol	Partial support	ACID transaction model from WS-TXM	Supports 2PC protocol only for ACID transactions but not for LRA.
Arbitrary Transaction Models and Semantics	Partial support	Business process transactions from WS-TXM	While ACID and LRA are domain independent, business process transactions allows creation of transaction model for a domain.
Series of Message Exchanges	No support	None	Provides accepted coordinator exchanges for ACID transactions, but none for application oriented coordinated message exchanges.
Multi-Participant Support	Full support	Not applicable	All three transaction model supports multiple participants.
State Information	Partial support	LRA WS-TXM	Provides expected state transitions but does not provide ability to manage state information.
Complements WS-BPEL specification	Full support	Not applicable	WS-CAF can function independently and along with WS-BPEL
Tool Support	No support	None	As key specifications such as WS-TXM and WS-CF were only proposal that did not become standards, there was no tool support for WS-CAF.