A Causal Tracking Monitoring Approach for Business Transaction Management

Yehia Taher[1,2], Willem-Jan van den Heuvel[1], Salima Benbernou[3] and Rafiqul Haque[4]
[1]*European Research Institute in Service Science, Tilburg University, Tilburg, The Netherlands*
[2]*Laboratoire PRiSM, Université Versailles ST Quentin-En-Yvelines, Versailles, France*
[3]*Laboratoire LIPADE, Université Paris Descartes, Paris, France*
[4]*LERO, University of Limerick, Limerick, Ireland*

Keywords: Business Transaction, SLA, Monitoring, CEP, Adaptation.

Abstract: Business transaction are automated, agreed-upon, long running propositions between multiple trading partners, that are specified over Service Level Agreement as constraints that need to be compliant. However, these constraints could be violated at any time due to various unexpected events. In this paper we propose a Business Transaction Management (BTM) system that amalgamates existing service composition and monitoring techniques ($e.g., BPEL$) with complex event processing (CEP) technology. The proposed system is a first important step toward effective support of end-to-end processes that can monitor their own performance by sensing and interacting with the physical word, and repair, upgrade, or replace themselves proactively.

1 INTRODUCTION

Business transactions can be loosely defined as trading interactions involving multiple parties, spanning many organizations, that strive to accomplish an explicitly shared business objective that extends over a possibly long duration and which is terminated successfully only upon recognition of the agreed conclusions between the interacting parties (Papazoglou and Kratz, 2007).

To successfully implement business transactions, there is a critical need of concepts, mechanisms and technologies that go far beyond their static analysis, design and implementation (e.g., in terms of functional and SLA-related terms), supporting also runtime execution, monitoring and analysis to dynamically enforce them to remain compliant to the end-to-end SLAs, pointing towards comprehensive business transaction management (BTM).

As illustrated in Figure 1, an integrated business transaction management system basically requires support for three major lifecycle phases. *1). Negotiation and Commitment*, which deals with mutual commitments that parties are accepting to integrate and govern their shared business processes. Particularly, business commitments mandate the shared goals and policies of business collaborations and glue shared goals and policies (e.g., business constraints and KPIs) with process definitions that collectively implement a business transaction (Weigand and van den Heuvel, 2002). *2). Design, Implementation, and Deployment*, which concerns with designing, implementing, testing and deploying the partners processes including their functional and QoS features that are rendered in their *local SLA*, in such a way that it meets the *global SLA* that is associated to the business transaction. *3). Execution, monitoring, and adaptation*, nce the business process - realized during the previous phase - becomes operational, its execution needs to be managed and monitored to gain a clear insight in how the services that run within the context of a business transaction perform against the QoS policies. The management of QoS policies in a

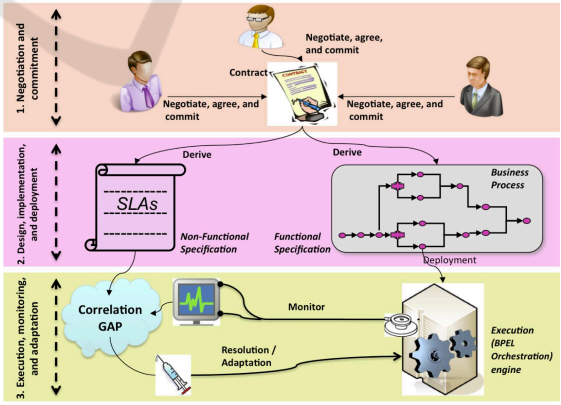

Figure 1: Layered structure of Business Transactions.

140 Taher Y., van den Heuvel W., Benbernou S. and Haque R..
A Causal Tracking Monitoring Approach for Business Transaction Management.
DOI: 10.5220/0004380601400144
In *Proceedings of the 3rd International Conference on Cloud Computing and Services Science* (CLOSER-2013), pages 140-144
ISBN: 978-989-8565-52-5

business transaction is of paramount importance due to their cross-organizational nature, ensuring that the delivery of services by providers and the consumption of services by consumers are performed within the threshold of the service level objectives stipulated in the local SLAs. Important questions have to address during this phase, notably: How can a business transaction remedy a potential future violation of the local SLA, such as the unavailability or failure of a service, local SLA violation, etc.? Is there any means to enforce the transaction to comply with its global SLAs even after one of the local SLA has been violated? Is there a need to make a correlation between local and global SLA?

In paper, we firstly introduce a BTM model that allows correlating critical business activities and events, QoS requirements, and application significant business data in an end-to-end process at runtime. Secondly, we propose a causal-tracking and monitoring approach to track the progress of business transactions against multi-party business agreements (e.g., a Service Level Agreement). Lastly, we introduces a preventive adaptation framework that preempts end-to-end process SLA violations, and automatically adapt the transaction execution in a way that meets the agreed end-to-end SLAs.

2 MOTIVATING EXAMPLE

Let's consider the following running example: an e-commerce trading system supports requisition of car parts between a car manufacturer and part automotive supplier through a third tier, a shipper, as depicted in Figure 2. We assume that the part supplier produces highly specialized car parts for a car manufacturer, and arranges the transportation. The car manufacturer pays a fixed rate for both the part and the transportation services. The part order business transaction will involve a sequence of messages, flowing between the car manufacturer, part supplier, and shipper. Figure 2 graphically depicts the message exchanges that conform to the contract between the car manufacturer, part supplier and shipper. It is basically structured as follows:

1. Initiate an Order: A message of type MT502 may be sent from the car manufacturer to the supplier to initiate a new order or to cancel a previous order, which is identified by the values of its data fields.
2. Confirm Order: A message of type MT513 must be issued from the supplier to the customer in response to an MT502 new order, confirming the execution of the order.

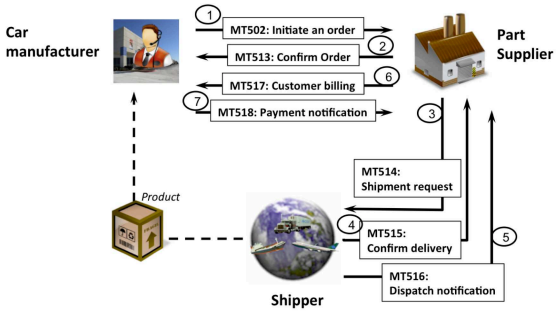

Figure 2: Steps capturing the collaboration protocol between the parties

3. Shipment Request: A message of type MT514 must be sent by the supplier to the shipper to determine delivery dates, shipment, loading, timing, and routing data-possibly offering multiple shipping options.
4. Confirm Delivery: A message of type MT515 must be sent by the shipper to the supplier in response to an MT514 shipment request, confirming the product delivery, including the shipment schedule and charges.
5. Product Dispatch Notification: A message of type MT516 should be sent by the shipper to the supplier as soon as the product has been dispatched.
6. Customer Billing: A message of Type MT517 should be sent upon receiving an MT517 message by the supplier to the customer, billing the latter with the final price of the order including the shipment charges.
7. Payment Notification: Finally, a message of type MT518 should be submitted by the customer to the supplier notifying the payment achievement.

Details about the data content of the messages would be spelled out as part of the contract. The contract might also contain local and global constraints of the business transaction. Constraints play important role in crafting agreements. Here are some possible examples of constraints:

Global Timing Constraint:

- C1. The transaction should be completed within five working days.

Local Timing Constraints:

- C2. The supplier should issue an MT513 message within 1 day upon receiving an MT502 message.
- The shipper should issue an MT515 message within 2 days upon receiving an MT514 message.
- C4. The shipper should issue an MT516 message within one day after issuing and MT517 message.

- C5. The customer should issue an MT518 message within one day upon receiving an MT517 message.

In order to ensure correct and consistent execution of business transactions, coordination protocols between the business transaction collaborating processes - derived on the basis the contract discussed above - have been incorporated in Web service composition and transaction languages (Papazoglou and Kratz, 2007). The car manufacturer plays the role of the transaction central coordinator to ensure atomicity over the set of the three collaborating processes.

3 BUSINESS TRANSACTION MANAGEMENT

The steering philosophy of our BTM framework is distilled from the old nursery rhyme, for a want or a nail[1]. The rhyme is about the transitivity of causality between events in a battlefield system. It teaches us, at a young age, to pay attention to details because a small event can lead to successive larger event and finally grow into a catastrophic event in a transitive manner.

3.1 BTM Approach

Our BTM framework is intended to not only dynamically check whether contractual stipulations are met, but also, provide means for preventing failure occurrences at an early stage. The BTM framework is grounded on a fine-grained, coherent, and consistent correlation/mapping between the events generated from the process execution and the process SLAs. The system comprises five steps:

1. Identify the events of interest. Each of the SLA clauses are analysed in order to identify which are the events of interest from the events generated by the process execution. The mappings between SLA and events are monitored. For instance:

   C2: The supplier should issue an MT513: Order Confirmation message within 1 day upon receiving an MT502: Initiate Order message.

   The analysis of this SLA clause will easily pinpoint the two following events of interest, *MT513: Order Confirmation* and *MT502: Initiate Order*.

2. Construct the causal model (Poset, see section 3.2) of the transaction. The causal model of a transaction is the set of event pattern rules that enable us to pinpoint which of the events - we are observing from the transaction - are causally related. The causal model will help to analyse and reconstruct the chain of events that led to the execution trail logged by the transaction monitor.

3. Generate the rules that specify violation situations. On the basis of the causal model constructed at the previous step, we generate the (ESL) rules that capture the SLA violations that may lead to the disruptions of the end-to-end process SLAs. For instance, the local SLA clause C2 is violated if the supplier hasnt issued an MT513 message within 1 day upon receiving an MT502 message. This violation is expressed by the following ESL expression:

   ```
   E1: (Sequence (MT502:Initiate
   Order, Negation (MT513: Order
   Confirmation))) Within (24h))
   ```

4. Generate the executable code to run into a CEP engine. The last step concerns with automatically generating the executable CEP code to deploy on the CEP engine. We particularly use the Continuous Computation Language (CCL) (http://www.sybase.com/products/financialservicessolutions/complex-event processing,) to develop executable code in terms of Coral8 CCL continuous queries. The snippet below presents an excerpt of a CEP rule for monitoring the violation of the constraint C2, i.e., the occurrence of the event E1.

   INSERT INTO OUTPUTSTREAM ALERTVIOLATION
   SELECT LOCAL VIOLATION OF THE LOCAL SLA CLAUSE C2
   FROM INPUTSTREAM ESB.MT502 INITIATEORDER, ESB.MT513 ORDERCONFRIMATION
   MATCHING [24 HOURS:INITIATEORDER,!ORDERCONFRIMATION]
   ON INITIATEORDER.ID = ORDERCONFRIMATION.ID

CEP allows for constant, real time accumulation of events to evaluate the event patterns specified within the rules. The rule pattern subscribes to the event input streams MT502 and MT513 from the ESB, and publishes to the event output streams AlertViolation. The rule is activated upon occurrence of the MT502, and initializes a 24h counter check. If the time-frame 24h get expired before the occurrence of the MT513 event, the rule pattern will be satisfied and a violation alert will be generated to trigger a process adaptation action.

[1] http://en.wikipedia.org/wiki/For_Want_of_a_Nail_(proverb)

3.2 Event Causal Tracking Monitoring

To achieve pro-active transaction management, we have made use of the causality notion introduced by Luckham (Luckham, 2001).

Causality Approach. Causality denotes the relationship between events that is defined in a transitive and asymmetric manner. As such, causality establishes a partial ordering of events. A set of events together with their casual relationship is called a Transaction Poset (Partially Ordered Set of events) or map. As illustrated in Figure 3, Posets are rendered as directed acyclic graphs (DAGs). A node in a DAG represents an event (and its data) and a directed arc represents a dependency relationship. It allows analysing the casual history of events leading up to the event under scrutiny.

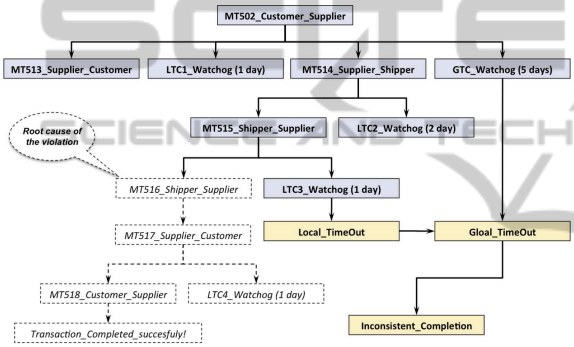

Figure 3: Casual history of successful transaction.

Let's illustrate the approach towards the example of Figure 3. An order event from the manufacturer to the supplier (signifying an MT502 message) causes an MT513 event from the supplier to the manufacturer, an MT514 event from the supplier to the shipper, and two watchdog events. The first is initializing a check for the global timing constraint (5 days) while the second is initializing a check for the local timing constraint related to the events MT502 and MT13 (1 day). In turn, an MT14 event causes an MT515 event from the shipper to the supplier, and a watchdog event initializing a check for the local timing constraint related to the events MT514 and MT515 (2 days). Subsequently, an MT515 event triggers an MT516 event from the shipper to the supplier, and a watchdog event initializing a check for the local timing constraint (1 day). In this example we assume that the time frame initialised by LTC3_Watchdog got expired while the MT516 event haven't yet occurred. At this stage, LTC3_Watchdog will take the hand over the transaction and trigger a Local_TimeOut event, which in turn will cause a Global_TimeOut event, and subsequently the transaction will result in an inconsistent completion.

Causally relating event in a transaction helps to analyse the root cause of the Global-TimeOut event. In particular, the global timing constraint was violated because MT502 event did not cause a consistent completion of the transaction within 5 days. And this is because the local constraint C3 was violated. the root cause of this violation is the non-occurrence of the event MT516 in time.

Obviously, it is important to get deep understanding of the root cause of any transaction failure. A typical solution to deal with such situations consists of improving the transaction diagnostics by checking the event executions online for constraint violations. Reducing constraint violations starts from having the flexibility to change the process. Essentially, processes need to be changed to conform to constraints and to handle exceptions.

Complex Event Specification Operators to Causally Relate Events. Event specification deals with primitive and complex events. Primitive events are basic, yet meaningful system-level changes of an application or business transactions, for instance, Initiate order, Confirm order, Shipment request, confirm delivery, Customer billing, Payment notification, etc. A rule associated to the detection of primitive event is fired as soon as the primitive event happens. But in real world applications, a complex sequence of events may need to be detected for a rule firing. A complex event logically chains together primitive- or other complex events occurring at time points other than the time specified composite event happens. For instance the complex event:
E1: The non-occurrence of an event confirming the execution of an order within one day upon receiving the initiate order.

In order to get to grips with event complexity, researchers have initially focused on the definition of event algebra composed of a set of event operators. Complex events are specified as event expressions that make use of events operators to relate events including primitive and complex ones - occurring at different time points.

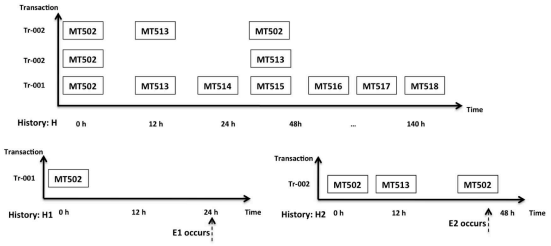

Figure 4: Event history mapping.

As illustrated in Figure 4, an event expression constitutes the mapping from one event history to another.

The result of applying the expression $E1$ to the history H is the history $H1$ that contains the event occurrences of H at which the events specified by $E1$ take place. Particularly, history $H1$ shows that the complex event specified by $E1$ takes place at $t = 24(h)$ because the event $MT513$ did not happen within $24h$ upon receiving the event $MT502$. Similarly, the result of applying the expression $E2$ to the history H is the history $H2$ that contains the event occurrences of H at which the events specified by $E2$ take place. Particularly, History $H2$ shows that the complex event specified by $E2$ takes place at $t = 36(h)$ where the event $MT502$ happened after an event $MT513$.

For a space limitation reason, we only describe the following operators. **Sequence:** the event type E characterizing the sequence of two events e1 and e2 is of the form *E= Sequence(e1, e2)*. An instance of E occurs iff e1 occurs before e2. **Negation:** the event type E characterizing the negation of an event e is of the form *E= Negation (e)*. An instance of E occurs iff e doesnt occur in a validity interval that depends on a context of detection. **Within:** the event type E defining a complex event P that occurs within a time period T is of the form *E= P Within (T)*. An instance of E occurs iff the complex event P occurs within T time period.

3.3 Adaptation Process

During the monitoring, in case of unfulfilment of duties, we introduce a proactive adaptation framework that pre-empts end-to-end process SLA violations. In this scenario, the soft constraints are used to relax and decide how to rebuild a composition involved in a transaction. Let 's recall in a nutshell the Soft constraint Solving Problem (SCSP, (Bistarelli et al., 2002)). Traditional CSPs is an assignment of a value to *every* variable, it can either be fully solved (when all requirements are satisfied) or not solved at all (some requirements cannot be satisfied). Solving techniques for *soft* CSPs can generate solutions for overconstrained problems by allowing some constraints to remain unsatisfied. Soft constraints generalize classical CSPs by adding a preference level to every tuple in the domain of the constraint variables. This level can be used to obtain a suitable solution which may not fulfill all constraints, which optimizes some metrics, and which in our case will be naturally applied to the requirements of the business transactions. The basic operations on soft constraints (building a constraint conjunctions and projecting on variables) need to handle preferences in a homogeneous way. It is based on mathematical structure of semiring algebra, enriched with additional properties and termed a *C-semiring*. For instance, let's consider that the service execution related to the constraint $C2$ got 1.5 day instead of 1 day. That means that the execution time of services related to the remaining constraints is 3.5 day instead of 4 days. Thus, we relax the individual constraints in a way that their aggregate execution time satisfy the new global constraint (3.5). In case we have four services to be executed, we may have alternative selections by applying the SCSP techniques. One could be 0.5, 1, 1, and 1 days respectively for services $S2$, $S3$, $S4$, and $S5$, second could be 1.5, 0.5, 0.5, and 1 days respectively for services $S'2$, $S'3$, $S'4$, and $S'5$.

4 CONCLUSIONS

The approach presented in this paper offers a structure for managing and controlling the QoS of a business transaction at run-time. The proposed framework monitors business transactions, computes potential risks, and performs proactive adaptation actions in order to prevent the possible risks of violating global SLA. We identify the actual possible cases of SLA violation during run-time and provide an approach for mitigating them by substituting services that could have failed or triggering changes of the composite services in terms of its compounding components by relaxing the constraints. The limitation of the framework is that it in its current version cannot resist the violation of local SLAs. Extending the functionality of the framework in terms of local SLA violation prevention is our future work.

REFERENCES

Bistarelli, S., Codognet, P., and Rossi, F. (2002). Abstracting Soft Constraints: Framework, Properties, Examples. *Artificial Intelligence*, 139(2):175–211.

http://www.sybase.com/products/financialservicessolutions/complex-event processing, S. C. E. P.

Luckham, D. (2001). *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman.

Papazoglou, M. P. and Kratz, B. (2007). Web services technology in support of business transactions. *Service Oriented Computing and Applications*, 1(1):51–63.

Weigand, H. and van den Heuvel, W.-J. (2002). Cross-organizational workflow integration using contracts. *Decision Support Systems*, 33(3):247–265.