

Formal Model for Business-aware Transaction Management *

Francois Hantry

Mohand-Said Hacid

Université Claude Bernard Lyon 1, LIRIS CNRS UMR 5205, France

francois.hantry@liris.cnrs.fr

mohand-said.hacid.hantry@liris.cnrs.fr

Mike Papazoglou

European Research Institute in Service Science, University of Tilburg, PO Box 90153, Tilburg 5000 LE, The Netherlands

mikep@uvt.nl

A few service-based languages explicitly define business artifacts including critical business activities, business events, business data, QoS requirements and SLAs in a choreographic style. Here, we propose formal ingredients for a business aware transaction framework for end-to-end business processes.

1 Introduction: The Need for Business-Aware Transactions (BAT)

Currently, Service Based Applications (SBAs) concentrate on composing software services into business processes. However, a few languages explicitly define business artifacts [6] including critical business activities, business events, business data, QoS requirements and SLAs. This is surprising since end-to-end processes typically involve well defined standard processes or segments (eg. payment processing, shipping and tracking....). Additionally, SLA is a key concept in end-to-end processes. Furthermore, no well defined language enables an holistic (choreographic) description of these necessarily related granular business tenets in an end-to-end process constellation [19]. However this is critical for obtaining a reliable message level protocol, tackling monitoring issues, and enabling reuse. Last, although declarative languages exist for service composition [17] to tackle flexibility issue and reuse, they are not business aware and do not emphasize on the key concept of transaction and SLAs [19] for SBA. However Business Transaction and SLAs concepts often need to be declarative [20], [5], [19] and seems convenient for reuse. To illustrate some of the above remarks, we consider the standard BPEL [18] and a current technique for monitoring and recovery. In BPEL, focus is on messages and control flow rather than on active business object [6]. But business analyst may wish focusing only on the effect of business transaction such as "Shipping" on corresponding real object such as "good" (standed by a business object), rather than dealing systematically with the manipulation of numerous Business document (or attributes) aliases while sending messages. Enforcing message dependencies should be presented as an enactment of a stronger declarative business goal. Moreover, different encoding may satisfy business requirement. For instance, it is not possible to specify in a direct way in BPEL the business compliance pattern of "prerequisite" wich means that, the absence of one activity implies that another is also absent. Last, The "mixing" of SAGA style [9] with fault-handler mechanism coming from programing community (C++, Java) [18], is an ad-hoc approach [12]. Furthermore, BPEL does not tackle decentralized distributed transactional coordination mechanism. In [3], monitoring techniques enable separation between monitoring data from analysis. However, the business artifact dependencies maps, are not considered as a

*The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube)

critical issue. Those are critical at design-time [6], [2], runtime [15] and monitoring [4] in a end-to-end choreographic style. For instance, a possible decoupling of payment information in payment and invoicing business processes from sales order and delivery information of goods in order management and shipment business processes, increases risks, could violate the integrity and accuracy of business objects and compliance with contractual agreements.

Here, we formalize the complete business aware transaction framework [19] for end-to-end business processes. Considering the above remarks we claim that our scheme is declarative, human understandable, convenient for choreography style, monitoring issue, flexibility, transactional theories and reuse. Additionnally, the use of formal model would contribute to ease many technical issues from planning [13], to verification as model checking [1] or theorem proving [21] and to runtime monitoring [14].

2 Formal specification for BAT : an overview

2.1 Transactional and Duration patterns

BAT specification is strongly built on the MITL formalism [1]. Precisly a BAT specification is based on temporal property patterns convenient for business analysts [7]. We have enhanced temporal property pattern [7] with new transactional pattern, and new duration pattern. The key ideas are to emphasize transactional statement, introduce fundamental transactional operator of "vitality" and extend sequencing pattern to "duration" sequencing pattern. Vitality is a convenient pattern block to construct many of the advanced business transaction concepts. Following the style it is declared weak or strong. For instance, let us assume T and T' are two sub-transactions. T' is weakly vital for T if and only if

- at any time, if T is weakly/fully waiting , then T' must also be fully waiting
 $style(t, strict) \Rightarrow G(f - waiting - T \Rightarrow f - waiting - T') \wedge style(t, flexible) \Rightarrow G(w - waiting - T \Rightarrow f - waiting - T')$
- at any time, if T' has been aborted or compensated then T is aborted or compensated in a near future
 $G[(aborted - T' \vee Compensated - T') \Rightarrow F((aborted - T) \vee (Compensated - T))]$

Furthermore, sequencing from [7] are enhanced as for instance $after(response_{[i,j]}(st, st'), st'')$ wich means after st'' , if st occurs then st' must occur at delay between i and j. It is translated into $MITL_{0\infty}$ as $G(st'' \Rightarrow G(st \Rightarrow (F_{>i}st' \wedge F_{<j}st')))$. Duration is one critical performance for end-to-end business processes. Our new formal intermediate language supports the following BAT model.

2.2 BAT model and specification

several styles for hierachy of transactions The emphasis is on nested transaction that may be categorized by ACID [11] transaction, closed nested [8], Long Running flexible business transaction [9], unconventional and application level atomicity [19] and advanced business transactional pattern [12]. We provide temporal pattern for those hierarchies.

Transaction and Active business object Any subtransaction enacts on Business artifact [6]. Business artifact range from business object (eg. Account, good), to coordinate artifact (eg. invoice), to business document or message(eg. acknowledge invoice). We enable to declare a kind of functional dependencies such as in distributed database [16], [10] but here dependencies are dynamic. We enable to elicitate input-output dependencies as for instance $G(w - commit - shipment \Rightarrow Billoflading.namecust :=$

sales.namecust) i.e while shipment commits then the input attribute namecust from the object Sales must be the same as billoflading. Those declarative constraints should be enforced by reliable message protocol and/or aliasing at a lower level. We also propose event dependencies [2] between low and high level object such as $G(w - commit - message - *price \Rightarrow invoice.price := message.changeprice.price)$ linking invoice object to a change-price message. These dependencies are critical for enabling data integrity, compensative phase while using SAGA style, enabling reliable monitoring and reuse.

SLAs Declarative business constraints are Master end-to-end SLAs and Local SLAs. SLAs include temporal performance, rules as sequencing, occurrence and compliance property, right and responsibilities of roles. Any property must hold on a definite time frame. A Master SLA is dependent of some local SLAs. These dependencies should be deduced while checking [1] or proving [21] the consistency of the specification. The consistent dependencies are also critical to provide reliable monitoring [14]. For instance, the below master SLA is dependent of the local SLA.

MASTER.SLA : $Process\ cycle - time < 7days(Payment\ after\ order)$
LOC.SLA : $process\ cycle - time < 24hrs(Order\ to\ Acknowledge)$

Business Aware Transaction and vital transaction To make a link between SLA and transaction, each SLA may be declared vital for some subtransaction. In other words, a subtransaction may be aware of this SLA. Furthermore, the vitality pattern also allow to customize inter-transaction dependencies.

vital-for(Integrated-Logistics[payment, shipment, order]) vitalfor(sequence((payment after shipment) before (delivery)), Integrated logistic) vitalfor(Good atomic, Integrated logistic) vital-for(Global(boundedexistence(retry(Inventory), <,2)), Integrated-Logistics)

Contingence and declarative fault handler Although the vitality pattern enables a strict "undo", many SLAs specify a fault handling as repairing, retrying or reaching second order goal [12]. We then isolate particular SLA patterns using failure transaction event (eg. abort-shipment) and often of the form of a boolean implication. This declarative temporal extension of fault handler in BPEL is more convenient for reuse. For instance contingent pattern are retry, alternative and compensation. Retry(t) is the sequence pattern: Response(abort-t ,started-alias(t)) where alias(t) means that t gets the same functional name and the same specification over business objects and constraints ; alternative(t) is the Pattern Response(abort-t,started-alternative(t)), compensation(t) is a Pattern Response(compensated-t,started-compensation(t)). Alternative and compensation must also be specified according to their effect on business objects, and transactions. Finally, a declarative fault handling wich may specify the contrary of a "SAGA style" compensation [12] is in BAT language :

Global (existence(shipment-aborted) IMPLY Global (precedence(compensate(shipment),compensate(payment))))

3 conclusion and perspective

We have provided formal ingredient for BAT language for SBA. We have taken care to the feasibility of our language using formal model and complexity result. Our ongoing works are now to check the validity of our scheme in implementing the following points. We have been investigating querying language to emphasis the dependencies of *MITL* formulas using model checking or theorem proving technics. We have been also taking care of the extra complexity of collecting data while monitoring. Finally we believe that enforcing declarative active business object dependencies in choreography style is a challenging research theme and further investigations are needed.

References

- [1] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.
- [2] Youcef Baghdadi. A business model for deploying web services: A data-centric approach based on factual dependencies. *Inf. Syst. E-Business Management*, 3(2):151–173, 2005.
- [3] Luciano Baresi, Sam Guinea, and Liliana Pasquale. Towards a unified framework for the monitoring and recovery of bpel processes. In *TAV-WEB '08: Proceedings of the 2008 workshop on Testing, analysis, and verification of web services and applications*, pages 15–19, New York, NY, USA, 2008. ACM.
- [4] Catriel Beeri, Anat Eyal, Tova Milo, and Alon Pilberg. Bp-mon: query-based monitoring of bpel business processes. *SIGMOD Rec.*, 37(1):21–24, 2008.
- [5] Panos K. Chrysanthis and Krithi Ramamritham. Synthesis of extended transaction models using acta. *ACM Trans. Database Syst.*, 19(3):450–491, 1994.
- [6] David Cohn and Richard Hull. Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Data Eng. Bull.*, 32(3):3–9, 2009.
- [7] Amal Elgammal, Oktay Turetken, Willem-Jan van den Heuvel, and Mike Papazoglou. On the formal specification of business contracts and regulatory compliance. In *FLACOS 2010 Fourth Workshop on Formal Languages and Analysis of Contract-Oriented Software*.
- [8] J. Eliot, J. Eliot, B. Moss, John Eliot, and Blakeslee Moss. Nested transactions: An approach to reliable distributed computing, 1981.
- [9] Hector Garcia-Molina and Kenneth Salem. Sagas. In *SIGMOD Conference*, pages 249–259, 1987.
- [10] Dimitrios Georgakopoulos, George Karabatis, and Sridhar Gantimahapatruni. Specification and management of interdependent data in operational systems and data warehouses. *Distrib. Parallel Databases*, 5(2):121–166, 1997.
- [11] Jim Gray. The transaction concept: Virtues and limitations (invited paper). In *VLDB*, pages 144–154, 1981.
- [12] Paul Greenfield, Alan Fekete, Julian Jang, and Dean Kuo. Compensation is not enough. In *EDOC*, pages 232–239, 2003.
- [13] Jörg Hoffmann, Piergiorgio Bertoli, and Marco Pistore. Web service composition as planning, revisited: In between background theories and initial state uncertainty. In *AAAI*, pages 1013–1018, 2007.
- [14] Oded Maler, Dejan Nickovic, and Amir Pnueli. Real time temporal logic: Past, present, future. In *FORMATS*, pages 2–16, 2005.
- [15] Thomas W. Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26:87–119, 1994.
- [16] Enric Mayol and Ernest Teniente. A survey of current methods for integrity constraint maintenance and view updating. In *In ER Workshops*, pages 62–73. Springer, 1999.
- [17] Marco Montali, Maja Pesic, Wil M. P. van der Aalst, Federico Chesani, Paola Mello, and Sergio Storari. Declarative specification and verification of service choreographies. *TWEB*, 4(1), 2010.
- [18] OASIS. Ws-bpel 2.0. <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>, 2006.
- [19] Mike P. Papazoglou and Benedikt Kratz. A business-aware web services transaction model. In *ICSOC*, pages 352–364, 2006.
- [20] Indrakshi Ray and Tai Xin. Analysis of dependencies in advanced transaction models. *Distributed and Parallel Databases*, 20(1):5–27, 2006.
- [21] P.-Y. Schobbens, J.-F. Raskin, T. A. Henzinger, and L. Ferrier. Axioms for real-time logics, 1999.