Model Checking Inconsistency Recovery Costs

Gang Zhou
Supervised by: Heinz Schmidt, Ian Peake, and Lawrence Cavedon

School of CSIT, RMIT University, Melbourne, Australia
g.zhou@student.rmit.edu.au

Abstract. Typically, when services become inconsistent from a business viewpoint, it is expected that compensation be used to recover from the inconsistency, by undoing the executed operations. In reality, compensation may incur additional costs, however existing approaches to recovery do not take such costs fully into account. We identify some major underlying gaps in SOC (Service Oriented Computing) related to compensation modelling, inconsistency identification, recovery and cost calculation. To make services more reasonable and predictable in dealing with inconsistencies from a cost perspective. We propose a cost-aware compensation framework modelling service compensations and compositions by a Petri Net-based model, reasoning about inconsistency recovery behaviours by model checking the *LTL* properties corresponding to business rules, and computing costs of recovery behaviours by parameterised cost calculation.

1 Introduction

Services are distributed self-describing open units that can be recursively composed for meeting business goals[25]. During transactions, services may become inconsistent with respect to business rules due to exceptions such as service failure, network error and human behaviours. Database transaction management[18] for resolving data integrity inconsistency is not suitable in SOC as the 2PC (2 Phase Commit) cannot be realised[19,15]. Handling compensation is firstly studied in the context of *sagas*[17], in which a compensable transaction consists of a sequence of smaller ACID transactions[17,19]. Nowadays, compensation is extensively applied in SOC to maintain consistencies by undoing the executed operations in case of transaction failures [26,8].

In reality, however, compensation may incur costs such as time, efforts and money etc. It is challenging for a service provider to determine such costs, as exceptions at any time may lead to different service inconsistencies that are the service's states and behaviours where business rules are violated. Recovery from different inconsistencies may incur different costs. For example, a customer may cancel an order from an online shop after the goods has been packed, where the inconsistency is identified as $goods\_packed \wedge order\_canceled$. To recover this inconsistency, the action *unpack_goods* is invoked to undo the effects of the *pack_goods* and costs y man-hours. Furthermore, the costs vary when the cancelation is raised at different time due to different compensation behaviours involved

G. Pallis et al. (Eds.): ICSOC 2011, LNCS 7221, pp. 195–200, 2012.

e.g. the order is canceled after the goods have been shipped, then the compensation may include *reshipping* and *unloading* which may cost more resources.

In concurrency, the cost of compensation-based recoveries depends on not only the compensation behaviours, but also the types of resource being consumed. For instance, the time costs of two concurrent actions may be computed by *max*, while the financial costs are summed; however for *choice* actions, the overall time and money costs are both computed by *max*. Hence cost calculation must be parameterised with respect to different kinds of composition and resources.

Moreover, cost-effective recoveries should be automated. Thus there is a clear need for a formal architecture framework to reason about the costs of compensation-based service inconsistency recovery. In such a framework, services inconsistencies would be handled more reasonably and predictably, e.g. by determining the inconsistency recovery behaviours and the minimal cost of such behaviours. At present, no such framework exists, due to some unsolved fundamental gaps such as the lack of a formal model of service composition bundled with compensation with supporting inconsistency identification and recovery, and the lack of a method for cost calculation parameterised with respect to different kinds of compositions and resources.

In this research, we will propose a formal framework to model and reason about costs of compensation-based service inconsistency recoveries. Motivated by the gaps above, we propose a research program as follows. Firstly, we will formalise an extended Petri net model modelling service composition, compensation and costs. Secondly, we will describe business rules in terms of *LTL* [29] properties and use model checking to identify inconsistency recoveries. Finally, we will reflect these recoveries into the Petri net model to calculate costs.

The remainder of this paper is organised as follows. Current issues are identified by literature review in section 2. The overview of the approach is sketched in section 3. Research plan and ongoing work are briefed in section 4 presents. Finally we conclude with a discussion in section 5.

2 Related Work and Research Gaps

Current models and methods cannot fully handle costs in compensation-based service inconsistency recoveries due to three major fundamental gaps. Firstly, ***compensation in SOC is not formally modelled***. Compensations in SOC are not formalised and automated in existing web service frameworks [5,13,10,27] in which developers need to program the *compensation handlers* to deal with ad hoc inconsistencies. Existing formal transaction models such as *sagas* [17], *StAC* [11], *t-calculus* [22], *cCSP* [12] and *cWFN* (Compensational Workflow Net) [2] that formally describe compensations in LRTs (Long Running Transactions) are not applicable directly in SOC, because services' interfaces and message-exchange based compositions are not considered. On the other hand, some models formalise services but do not take compensations into account, such as *oWFN* (open Workflow Net) [28] suggested by Reisig and service automata [24]. Secondly ***there is no support for reasoning about inconsistency recovery***

behaviours. Existing frameworks for ensuring service states based consistencies such as *WTDP* [14], *set consistency* [16] and *webservice interfaces*[6] are concerned only with the consistencies of end states. However, it is overlooked that the inconsistencies may be tolerated and recovered eventually before a service ends. Finally, ***parameterised cost calculations are not studied***. Costs in compensations discussed by Biswas [8] and Li [23] are not formalised. Timed automata[3] and weighted timed automata [4] formalise the linear costs based on state transition models in which actions are interleaved and studied in sequences. Consequently, *truly concurrency* and resource types introduced above are not included. Some timed Petri net based models[20,1] reason about linear time costs, however, they are based on reachability graphs, that are converted interleaved models where information about concurrency has been lost.

3 Overview of the Proposed Approach

We propose a novel approach to tackle the overall gaps. For modelling compensation in SOC environments, a Petri Net based model is proposed due to its algebraic and graphic capabilities, and the elegance of describing *true concurrency*. We propose the CSN (Compensable Service Net) by combining *cWFN* and *oWFN*. The former models normal tasks, failure tasks and compensation tasks; the latter models services interfaces and compositions. Thus we combine their strengths including cost modelling. For model checking inconsistency, a business rule denoted by φ is defined by two *LTL* properties separately: the *safety* property $G\varphi$ to ensure φ is satisfied in all states (markings) and the end-state property $F\varphi$ to ensure φ is met in the end. We define an inconsistency of a service as a service behaviour that violates the *safety* property of a business rule. By model checking these properties on every execution trace, inconsistency recovery paths are reasoned from counterexamples if any. For cost calculation, first inconsistent states and final consistent states obtained from the recovery paths are reflected to CSN and costs are calculated in the net.

Service Compensation Modelling. In CSN, all actions including communication, normal, failure or compensation actions are represented by transitions while the preconditions (effects) of actions are represented by places. Four special places such as *receiveIn*, *successOut*, *receiveCancel* and *sendAbort*, called *interfaces places* are distinguished for modelling communications with environments including e.g. other CSNs. A number of algebraic operators defined in [22], e.g. *sequence* and *parallel* etc. are translated to the service composition operators for CSNs' compositions by gluing all shared *interface places*, therefore, both normal and the compensation processes are constructed.

Model Checking Inconsistency. We propose a model checking procedure which has the inputs of a finite set of execution paths extracted from the reachability graph of CSN and the two *LTL* properties: *safety* and *end-state*, from business rules; its output is a set of recovery paths in each of which a first inconsistent state and a final consistent state will be reflected back to the CSN

for costs calculations. This procedure is briefed as follows. For each execution trace, model checking both *LTL* properties, if the end state consistency is satisfied and the *safety* consistency is violated, then we locate the last element in the counterexample, which is the first action breaking the *invariance*, the consistency specification. Therefore, the recovery path in this execution is identified from the located action to the final action. This procedure is recursively applied on the sub CSNs to find out all sub paths that record all inconsistencies' tolerances. For the sub CSN model checking, the properties are refined in terms of the sub CSN's observable actions, thus the specification decomposition must be considered.

Parameterised Cost Calculation. Resources and their quantities are represented by a set of special places called *resource places* and the tokens within the places respectively. A cost function of a transition is derived from its firing rule by projecting only the arcs connecting with *resource places*. Therefore, given a parameterised cost specification that represents the cost calculation rule for a resource type, for instance, time cost is computed by *max* for parallel transitions, and by + for sequence transitions, the costs from one state to another in a CSN system can be decidable with some restrictions and the algorithm is currently under development.

4 The Research Plan and Current Work

This ongoing research is planned in three stages according to the proposed approach and correspondingly, the framework consists of three major modules: Service Net Compositor (SNC), Inconsistency Recovery Model Checker (IRMC) and Parameterised Cost Calculator (PCC) (see Fig. 1). In the first stage, we define and implement the CSN with a number of composition operators. Hence, the formal model of service composition bundled with compensation is defined and implemented in SNC. In the second stage, the IRMC module is realised in which all compensation-based recovery paths from inconsistencies are identified by model checking the *LTL* inconsistency properties on a reachability structure that is derived from a CSN. In the last stage, parameterised costs are calculated in the PCC.

The plan is carried out incrementally and iteratively and we have implemented the first two stages in a prototype applying several case studies. The SNC is built on *PIPE* [9]. The sub CSNs are input in the forms of *PNML* [7] files and their operators are input as the algebra operators. These operators i.e. *parallel*, *sequence* and *choice* have been translated and implemented to the net composition operators. More operators will be considered when necessary e.g. *alternative forwarding* and *iteration*. The *LTL* properties are generated according to business rules input by users and checked in the IRMC module which has been built based on *LTSA* [21]. All inconsistency recovery behaviours are output by IRMC and will be provided to PCC for computing costs.

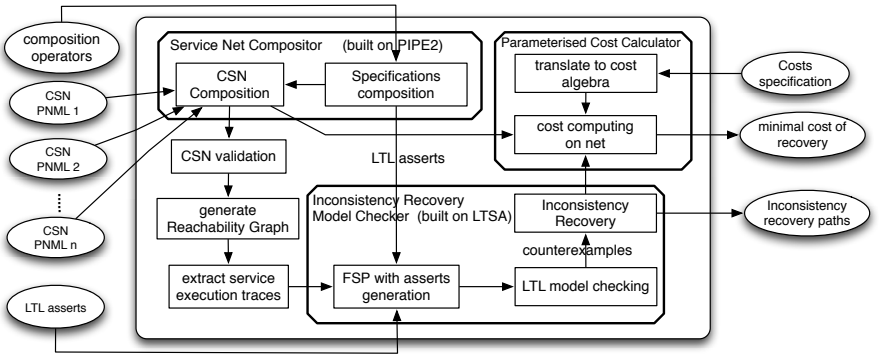

Fig. 1. Compensation-based inconsistency recovery and costs reasoning framework

5 Discussion

In SOC, compensation with costs to recover from inconsistencies is not formally studied. We identify three fundamental gaps related to compensation modelling, inconsistency recovery and parameterised cost calculation. The contribution of this paper is to propose a formal framework tackling all these gaps, reasoning about costs of compensation-based solutions. The inconsistencies captured by this research are the services' states and behaviours that violate the *safety* property of business rules. In some cases, not all inconsistencies can be recovered even when they are detected and these situations should be diagnosed. We are scoping the inconsistencies that can be recovered. Complex case studies suggest we must face and handle the well-known state space explosion problem. The efficiency of model checking algorithms can be improved by reducing the redundant computations on identifying the same inconsistencies. The cost model needs to be enhanced to support resource sharing and racing problems in the future.

References

1. Abdulla, P.A., Mayr, R.: Minimal Cost Reachability/Coverability in Priced Timed Petri Nets. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 348–363. Springer, Heidelberg (2009)
2. Acu, B., Reisig, W.: Compensation in Workflow Nets. In: Donatelli, S., Thiagarajan, P.S. (eds.) ICATPN 2006. LNCS, vol. 4024, pp. 65–83. Springer, Heidelberg (2006)
3. Alur, R., et al.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
4. Alur, R., La Torre, S., Pappas, G.J.: Optimal Paths in Weighted Timed Automata. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 49–62. Springer, Heidelberg (2001)
5. Andrews, T., et al.: Business process execution language for web services, version 1.1. Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation (2003)
6. Beyer, D., Chakrabarti, A., Henzinger, T.A.: Web service interfaces, pp. 148–159 (2005)

7. Billington, J., Christensen, S., van Hee, K.M., Kindler, E., Kummer, O., Petrucci, L., Post, R., Stehno, C., Weber, M.: The Petri Net Markup Language: Concepts, Technology, and Tools. In: van der Aalst, W.M.P., Best, E. (eds.) ICATPN 2003. LNCS, vol. 2679, pp. 483–505. Springer, Heidelberg (2003)
8. Biswas, D.: Compensation in the World of Web Services Composition. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 69–80. Springer, Heidelberg (2005)
9. Bonet, P., et al.: PIPE v2. 5: A Petri net tool for performance modelling. In: Proc. 23rd Latin American Conference on Informatics (CLEI 2007). Citeseer (2007)
10. Bunting, D., et al.: Web Services Composite Application Framework (WS-CAF). Ver1. 0 (2003)
11. Butler, M., Ferreira, C.: A process compensation language, pp. 61–76 (2000)
12. Butler, M., Hoare, T., Ferreira, C.: A Trace Semantics for Long-Running Transactions. In: Abdallah, A.E., Jones, C.B., Sanders, J.W. (eds.) CSP25. LNCS, vol. 3525, pp. 133–150. Springer, Heidelberg (2005)
13. Cabrera, F., Copeland, G., Cox, B., Freund, T., Klein, J., Storey, T., Thatte, S.: Web services transaction (WS-transaction). In: Microsoft, IBM etc (2002)
14. Choi, S., et al.: A framework for ensuring consistency of Web Services Transactions. Information and Software Technology 50(7-8), 684–696 (2008)
15. Coleman, J.: Examining BPEL's compensation construct. In: Rigorous Engineering of Fault-Tolerant Systems (REFT 2005), p. 122 (2005)
16. Fischer, J., Majumdar, R.: Ensuring consistency in long running transactions. In: Proceedings of the Twenty-Second IEEE/ACM International Conference on Automated Software Engineering, pp. 54–63. ACM (2007)
17. Garcia-Molina, H., Salem, K.: Sagas. ACM SIGMOD Record 16(3), 249–259 (1987)
18. Gray, J.: The transaction concept: Virtues and limitations. In: Proceedings of the Very Large Database Conference, pp. 144–154. Citeseer (1981)
19. Greenfield, P., et al.: Compensation is not enough. In: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing, p. 232. IEEE Computer Society, Washington, DC (2003)
20. Holliday, M.A., et al.: A generalized timed Petri net model for performance analysis. IEEE Transactions on Software Engineering (12), 1297–1310 (1987)
21. Kramer, J., McGee, J.: Concurrency: State Models and Java Programs (1999)
22. Li, J., Zhu, H., Pu, G., He, J.: Looking into compensable transactions. In: 31st IEEE Software Engineering Workshop, SEW 2007, pp. 154–166. IEEE (2007)
23. Lin, L., Liu, F.: Compensation with dependency in web services compositions. In: International Conference on Next Generation Web Services Practices, NWeSP 2005, pp. 183–188. IEEE (2006)
24. Massuthe, P., Schmidt, K.: Operating guidelines-an automata-theoretic foundation for the service-oriented architecture. In: Fifth International Conference on Quality Software (QSIC 2005), pp. 452–457. IEEE (2005)
25. Papazoglou, M.P., Georgakopoulos, D.: Service-oriented computing. Communications of the ACM 46(10), 25–28 (2003)
26. Pires, P.F., et al.: Webtransact: A framework for specifying and coordinating reliable web services compositions (2002)
27. Potts, M., et al.: Business Transaction Protocol Primer. OASIS Committee Supporting Document (2002)
28. Reisig, W.: Simple Composition of Nets. In: Franceschinis, G., Wolf, K. (eds.) PETRI NETS 2009. LNCS, vol. 5606, pp. 23–42. Springer, Heidelberg (2009)
29. Vardi, M.: An Automata-Theoretic Approach to Linear Temporal Logic. In: Moller, F., Birtwistle, G. (eds.) Logics for Concurrency. LNCS, vol. 1043, pp. 238–266. Springer, Heidelberg (1996)