Business Modeling via Commitments

Pankaj R. Telang and Munindar P. Singh

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
prtelang@ncsu.edu, singh@ncsu.edu

Abstract. Existing computer science approaches to business modeling offer low-level abstractions such as data and control flows, which fail to capture the business intent underlying the interactions that are central to real-life business models. In contrast, existing management science approaches are high-level but not only are these semiformal, they are also focused exclusively on managerial concerns such as valuations and profitability.
This paper proposes a novel business metamodel based on commitments that considers additional agent-oriented concepts, specifically, goals and tasks. It proposes a set of business patterns and algorithms for checking model completeness and verification of agent interactions. Unlike traditional models, our approach marries rigor and flexibility, providing a crisp notion of correctness and compliance independent of specific executions.

1 Introduction

Real-life service engagements generally involve long-lived, complex interactions among two or more autonomous business partners. We define a *business model* as a specification of a way in which a service engagement is carried out. We address the problem of creating, enacting, and verifying business models from a high-level, yet rigorous standpoint.

Service organizations form complex business relationships with other organizations to exchange value. Competition continually forces organizations to improve their operations. Such improvements include outsourcing or insourcing business tasks based on appropriate strategic considerations. Mergers, acquisitions, and alliances change the partners of a value network. The business processes needed to support such dynamic interactions tend to be complex.

Existing techniques for modeling, operationalizing, and evolving such processes are inadequate, because they are based on low-level abstractions at the level of data and control flows, expressed in orchestrations or choreographies. These specifications do not capture the business intent of the interactions. They tend to over-constrain business behavior by mandating the exchange of a predetermined set of messages usually in an unnecessarily restrictive temporal order.

This paper proposes a commitment-based business metamodel, which captures value exchanges among business partners in terms of their commitments. Further, this paper defines patterns based on the above metamodel as well as algorithms to verify the correctness of service engagements with respect to their designs. The main benefits of this approach are to meld rigor and flexibility thereby improving the quality of service engagement specifications and their instantiations.

Contributions. The main contributions of this paper are (1) a commitment-based metamodel that describes value exchanges among business partners, (2) a set of business modeling patterns, and (3) algorithms for verifying (a) implemented agent interactions with respect to a business model and (b) the completeness of a business model.

Organization. Section 2 presents the business metamodel and a set of business patterns. Section 3 applies the patterns to create a model for an insurance claim processing scenario. Sections 4 and 5 introduce notions of compliance and completeness respectively, and provide algorithms for checking them. Section 6 compares our approach with related work.

2 Metamodel and Patterns

A business model seeks to capture value exchanges and the evolution of commitments among business partners. Figure 1 illustrates our metamodel.

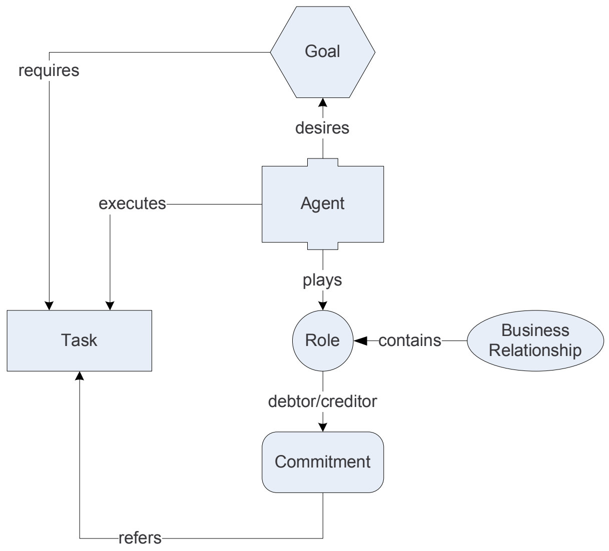

Fig. 1. Metamodel for commitment-based business models

We characterize a business model via a set of *business relationships*, the participants of which we term *(business) partners*. The partners execute tasks for each other that enable achieving their respective *goals*. Importantly, our approach defines relationships in terms of the creation and manipulation of *commitments* among the partners. To enter into a business relationship, each partner takes on the commitments that the relationship specifies. The partner presumably possesses the *capabilities* that the relationship requires—these are presumably required to perform the *tasks* that would discharge the specified commitments.

We associate interaction protocols with business relationships in two main ways. Interaction protocols are crucial both to (1) creating or modifying a business relationship, such as via negotiation and (2) to enacting a business relationship. The following paragraphs describe the key concepts of this metamodel.

Agent: a computational representation of a business partner. An agent captures the autonomy and heterogeneity of a real-world business. An agent has goals and possesses a set of capabilities that enable it to execute business tasks. For each business relationship in which an agent participates, it enacts one or more roles in that relationship.

Role: an abstraction over agents that helps specify a business relationship. Each role specifies the commitments expected of the agents who play that role along with the capabilities they must possess to function in that role.

Goal: a state of the world that an agent desires to be brought about [3]. In simple terms, an agent's goals are its ends. An agent achieves a goal by executing appropriate tasks.

Task: a business activity viewed from the perspective of an agent. Value transfers between the agents when they execute tasks for one another.

Capability: an abstraction of the tasks that an agent can perform.

Commitment: A commitment C(DEBTOR, CREDITOR, antecedent, consequent) denotes that the DEBTOR is obliged to the CREDITOR for bringing about consequent if antecedent holds [8]. A commitment C(BUYER, SELLER, goods, pay) means that buyer commits to paying the seller if goods are delivered. When the seller delivers goods, the buyer becomes unconditionally committed to paying. In the event that the buyer makes the specified payment, this commitment is discharged.

Business relationship: a set of interrelated commitments among two or more roles that describe the value to be exchanged among the roles. In simple terms, each agent's main motivation behind forming a business relationship is to access the capabilities of others.

At run-time, commitments arise between agents, but at design-time we specify them between roles. Being able to manipulate commitments yields the flexibility needed in open interactions. A commitment may be *created*. When its consequent is brought about, regardless of whether antecedent holds or not, it is *discharged*, i.e., satisfied. If its antecedent is brought about then it is *detached*. The creditor may *assign* a commitment to another agent. Conversely, a debtor may *delegate* a commitment to another agent. A debtor may also *cancel* a commitment and a creditor may *release* the debtor from the commitment. Further, a commitment moves among four main states: *active* (when it is created and (presumably) being worked upon), *pending* (when it has been delegated and is not being worked upon), *satisfied*, and *violated*.

2.1 Running Example

We evaluate the proposed metamodel and patterns via a real-world insurance claim processing use case involving AGFIL, an insurance company in Ireland [4]. AGFIL underwrites automobile insurance policies. Fig. 2 shows the parties and processes involved in the business service of (emergency) claim processing that AGFIL provides.

To provide this service, AGFIL must provide claim reception and vehicle repair to the policy holders. Additionally, it needs to assess claims to protect against fraud. AGFIL depends on its partners, Europ Assist (EA), Lee Consulting Services (CS), and repairers, for executing these tasks. EA provides a 24-hour helpline for customers to

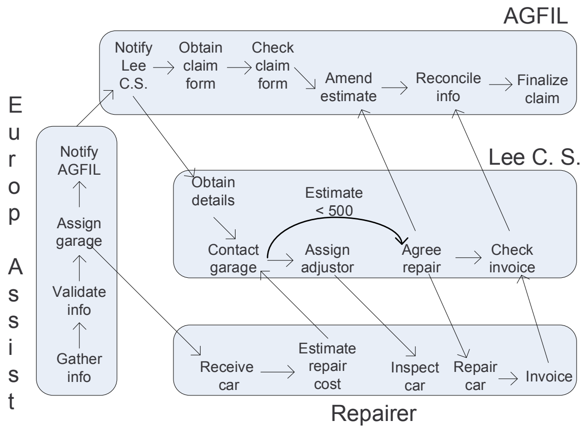

Fig. 2. Insurance claim processing [4]

report a claim and provides an approved repairer garage. CS assesses and presents invoices to AGFIL on behalf of the repairers. A network of approved repairers provide repair services. AGFIL retains the authority for issuing final claim approvals.

2.2 Patterns

A pattern, in the present setting, is a recipe for modeling recurring business scenarios. This section describes a key set of such patterns, which could seed a potential business model pattern library. Section 3 demonstrates the effectiveness of this simple set of patterns on an existing use-case based on a real-life scenario.

Of the 13 attributes in the classical template for object-oriented design patterns [6], we use *name*, *intent*, *motivation*, *implementation*, and *consequences* to describe our patterns. Here the *consequences* of a pattern allude to the practical consequences of applying the pattern, i.e., the assumptions underlying the model. The pattern figures use the notation of Fig. 1, and additionally show two directed edges for each commitment: from the debtor to the commitment and from the commitment to the creditor. The subscript on a commitment indicates its state: **A** for active, **D** for detached, **S** for satisfied, and **P** for pending. The patterns are expressed in terms of roles and would be instantiated by the agents who adopt the specified roles. Each role of a pattern must be adopted by some agent in order for the resulting business relationship to be executed.

2.3 Unilateral Commitment

Intent: A performer commits to a beneficiary for value transfer. There is no “converse” commitment from the beneficiary.

Motivation: For example, a conference committee member commits to a program chair to review a paper that the program chair asks the member to review. The chair makes no converse commitment.

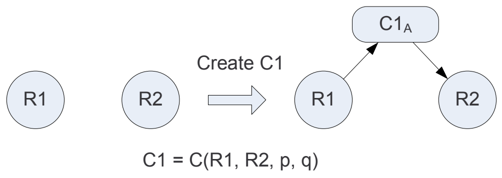

Fig. 3. Unilateral commitment

Implementation: A commitment is created from the performer (R_1) to the beneficiary (R_2) for a value transfer. Figure 3 shows this pattern.

Consequences: This presumes a side benefit to the performer (debtor) from the antecedent of the commitment.

2.4 Commercial Transaction

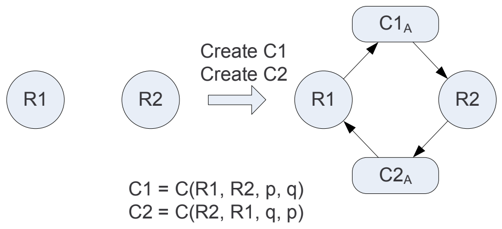

Fig. 4. Commercial transaction

Intent: This pattern expresses a value exchange between two trading partners. The trading partners negotiate and, upon agreeing, commit to each other for the specified value transfers.

Motivation: A typical barter motivates this pattern. For example, a seller and a buyer agree to exchange goods for payment. A more conventional barter would be when the parties exchange goods and services rather than money for goods or services.

Implementation: A pair of reciprocal commitments between the trading partners (R_1 and R_2, treated symmetrically) specify the pattern. Figure 4 shows this pattern.

Consequences: In general, the antecedents and consequents of the commitments are both composite expressions. Importantly, we need a mechanism to ensure progress by in essence breaking the symmetry, e.g., via a form of concession [10].

2.5 Outsourcing

Intent: An outsourcer delegates a task to a subcontractor, typically because the outsourcer lacks the necessary capabilities or expects some other benefit such as a more efficient solution or a lower risk of failure.

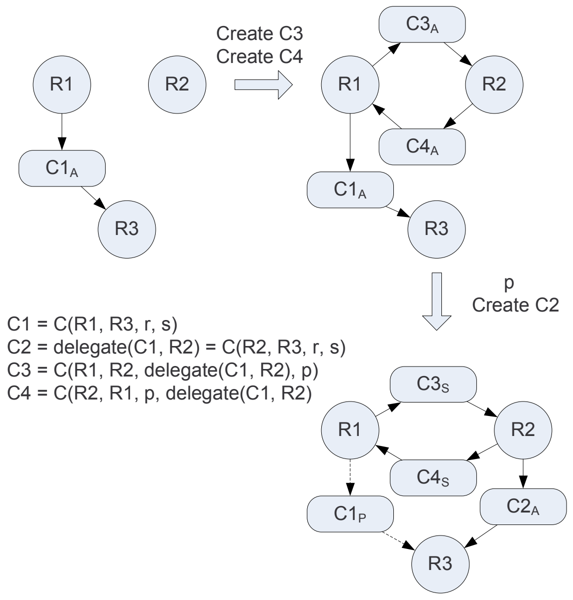
Create C3
Create C4
R1
R2
C1A
R3
C3A
C4A
p
Create C2
C1 = C(R1, R3, r, s)
C2 = delegate(C1, R2) = C(R2, R3, r, s)
C3 = C(R1, R2, delegate(C1, R2), p)
C4 = C(R2, R1, p, delegate(C1, R2)
C3S
C4S
C1P
C2A

Fig. 5. Outsourcing

Motivation: Many business organizations outsource noncore activities. As an example, consider a customer who signs up for cable television service. The cable operator commits to the customer for installation. Instead of staffing its entire service area directly, the cable operator outsources the installation task in several regions to its local partners in those regions.

Implementation: The outsourcer is the current debtor (R_1). The current debtor and the new debtor (R_2) create a relationship, following which the current debtor delegates the commitment to the new debtor. The existing commitment becomes pending; the new commitment becomes active. The creditor is unchanged. Figure 5 shows this pattern.

Consequences: The business relationship between the new and the previous debtors would be a standing arrangement, which must have a scope and lifetime no smaller than that of the delegated commitment. The commitment from the previous debtor is pending and must either be considered discharged or reactivated depending on how the new debtor performs.

2.6 Standing Service Contract

Intent: A service provider negotiates with a consumer for providing service over a specified duration, and creates a pair of commitments. The consumer's request for a service instance detaches the standing commitment. The provider then creates one or more commitments for providing the service instance.

Motivation: A business service such as plumbing maintenance or a line of credit from a bank refers to (potentially) numerous service instances. Whenever the faucet leaks

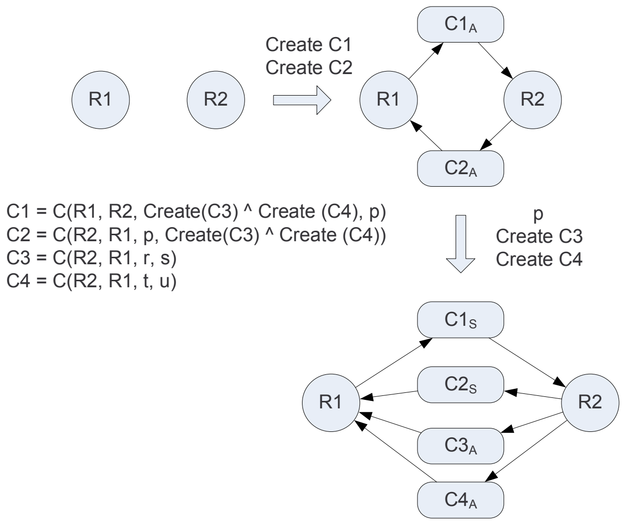

Fig. 6. Standing service contract

(within specified limitations), the plumber will fix it. Whenever the customer submits a check for an amount up to the specified credit limit, the bank will disburse funds.

Implementation: The service provider (R_1) and consumer (R_2) enter into the following commitments. Here, C_1 and C_2 are reciprocal commitments (as in the commercial transactions pattern) that describe the standing service contract. C_3 and C_4 arise from the consumer exercising the service contract. Figure 6 shows this pattern.

Consequences: The standing contract must be of sufficiently large scope to cover the cases of interest but should generally be bounded in the effort it requires. This pattern can be applied multiple times as when a consumer pays a subscription every month to obtain a continuing plumbing warranty.

3 AGFIL Business Model

This section applies the patterns to the AGFIL scenario and describes the resulting business model. AGFIL, an insurer (I), has the goal of providing emergency service, which requires the capabilities for claim reception, claim assessment, claim finalization, and vehicle repair. Except claim finalization, which it possesses locally, AGFIL acquires the remaining capabilities from its business partners.

The insurer delegates to the call center its claim reception commitment to the policy holder. Although the commitment from the insurer to the policy holder for claim reception is not created yet, the insurer chooses to set up the delegation earlier. The *outsourcing pattern* models this scenario. The insurer selects EA as a call center provider (C). The selection process is out of our present scope. Figure 7 shows how the outsourcing pattern applies.

C1. C(C, I, payCallcenter, create(C3))
C2. C(I, C, create(C3), payCallcenter)
C3. C(C, P, reportAccident, receiveClaim)

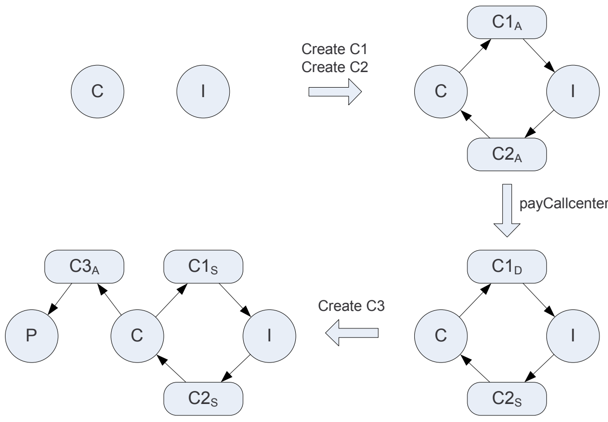

Fig. 7. Claim reception: Outsourcing

The insurer and the call center create commitments C1 and C2 when they agree upon the payment that the insurer makes to the call center, for providing claim reception to the policy holder. The commitment C1 means the call center commits to the insurer for creating commitment C3, which is to receive claims from the policy holder, provided the insurer pays the call center. The commitment C2 means the insurer commits to the call center for payment if the call center creates C3. The insurer pays the call center, and therefore discharges C2 and detaches C1. Later, the call center creates C3 and discharges C1.

The insurer outsources the claim assessment capability to Lee CS, an assessor. In this case, the *outsourcing pattern* does not apply since the insurer is not delegating a commitment. That is, the insurer requires claim assessment for itself. Instead, the *commercial transaction pattern* models this scenario.

C4. C(A, I, payAssessor $\wedge$ reqAssessment, agreeToRepair)
C5. C(I, A, agreeToRepair, payAssessor)

The commitment C4 means the assessor commits to the insurer, for negotiating repair cost and to bring about the agreement to repair with the repairer, provided the insurer pays the assessor and makes a request for assessment. The commitment C5 means the insurer commits to the assessor for the payment provided the assessor brings about agreement to repair.

The assessor outsources the vehicle inspection to an adjuster (D). The *commercial transaction pattern* models this scenario. Since this scenario is similar to the claim assessment scenario, to save space, we do not describe it in detail.

A policy holder (P) desires to get insurance. Through a directory service, the policy holder locates AGFIL, the insurer. The policy holder and the insurer interact to setup the insurance service contract. The *service contract pattern* models this scenario. Figure 8 shows how the service contract pattern applies.

C8. C(P, I, insurance, payInsurer)
C9. C(I, P, payInsurer, insurance)
C10. C(I, P, reportAccident, receiveClaim)
C11. C(I, P, requestService, repairVehicle)

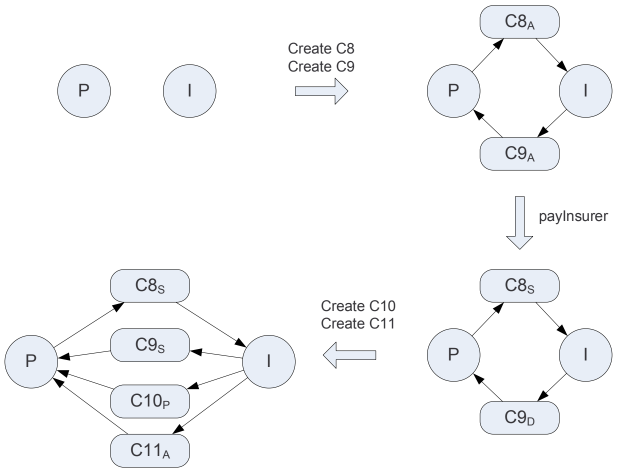

Fig. 8. Insurance purchase: Service contract

Commitment C8 means the policy holder commits to the insurer for payment if insurance is provided, and commitment C9 means the insurer commits to the policy holder for insurance if the policy holder pays the insurer. To provide insurance, the insurer creates the commitments C10 and C11, that is, insurance = create(C10) $\wedge$ create(C11). Commitment C10 means the insurer commits to receiving claim if the policy holder reports an accident, and in commitment C11, the insurer commits to repairing the (insured) vehicle if the policy holder requests repair service for it. The insurer changes the status of commitment C10 to pending, since it has delegated that commitment to the call center. Recall that C3 results from the delegation of C10. That is, C3 = delegate(C10, C).

To assess a claim, the assessor has the adjuster inspect the vehicle. The assessor negotiates with the repairer. By bringing about an agreement to repair, the assessor satisfies its commitment to the insurer C4. Figure 9 shows how the *outsourcing pattern* now applies between the insurer, the repairer, and the policy holder.

C12. C(I, R, delegate(C11, R) $\wedge$ agreeToRepair, payRepairer)
C13. C(R, I, payRepairer, delegate(C11, R))

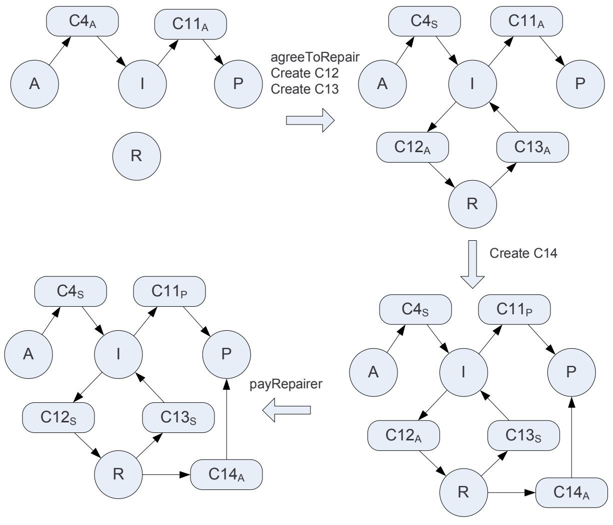

Fig. 9. Vehicle repair: Outsourcing

C14. delegate(C11, R) = C(R, P, requestService, repairVehicle)

Commitment C12 means the insurer commits to the repairer for paying the repair charges, if the repairer accepts the delegation of C11 and creates C14. Commitment C13 means the repairer commits to accepting the delegation of commitment C11 if the insurer pays. In the delegated commitment C14, the repairer commits to the policy holder for vehicle repair when the policy holder requests for repair. The repairer satisfies the commitment C13 by creating C14, and detaches C12. Later the insurer discharges C12 by paying the repairer. Note that it is not necessary for the insurer to pay the repairer at this time, and other evolutions are possible. For example, the repairer may repair the vehicle, that is, satisfy the commitment C14, before the insurer pays. We describe one possible model evolution above.

4 Verifying Agent Interactions

This section presents an algorithm for verifying if each partner complies with a business model. An agent complies with a business model if it discharges each detached commitment of which it is the debtor. We consider a UML sequence diagram as a low-level model for agent interactions. The agents may exchange multiple messages for executing one task. For example, the policy holder may report an accident by sending a message to the insurer; the insurer may request additional information, leading to further messages. In the interaction model (based on a sequence diagram), we assume that upon completing a task, the executor of the task sends a message asserting its completion.

Given a business model and an interaction model, Algorithm 1 returns a set of violated commitments. We assume that the interaction model captures all agent interactions. The algorithm iterates over the commitments from the business model and eval-

Algorithm 1: verifyInteractions(m, i): Verify agent interaction model i with respect to business model m

```
1  $C = m.C$; // Model Commitments
2  $CS = ()$; // Satisfied commitments
3  $CV = ()$; // Violated commitments
4  $T = i.T$; // Tasks completed in the interaction model
5  foreach $c \in C$ do
6      if (eval(c.consequent, T) = true) then
7          $CS.add(c)$;
8  foreach $((c \in C) \wedge (c \notin CS))$ do
9      if (eval(c.antecedent, T) = true) then
10         $CV.add(c)$
11 return $CV$;
```

uates the antecedent and consequent of each using the tasks asserted in the interaction model. The antecedent and consequent of a commitment are formulae, each containing a disjunction of tasks. The $eval$ procedure evaluates these based on the tasks asserted in the interaction model. The commitments whose consequent evaluates to true are satisfied, whereas the commitments whose antecedent evaluates to true, but whose consequent evaluates to false, are detached commitments that are violated. The debtors of the violated commitments are the agents that do not comply with the given business model (within the scope of the given interaction model).

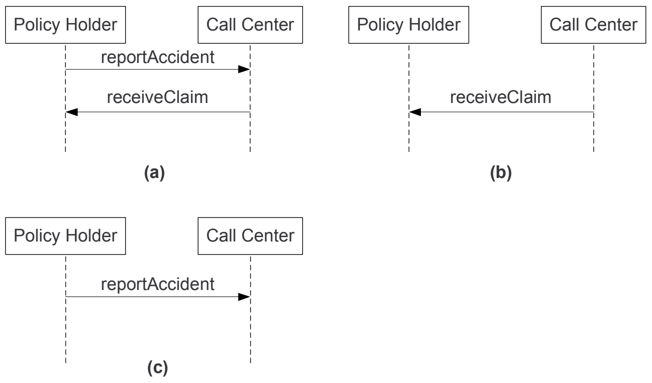

Fig. 10. Verifying agent interactions

For example, in the AGFIL business model, consider the commitment C10 = C(C, P, reportAccident, receiveClaim). An interaction model in which neither of the tasks, reportAccident and receiveClaim, are asserted, is a trivial case where both agents, the policy holder (P) and the call center (C), comply with the business model. In Fig. 10(a), the policy holder reports an accident, and detaches the commitment C10. The call center receives the claim and, therefore, satisfies the detached commitment C10. In this case, both the agents comply with the business model. In Fig. 10(b), the call center receives the claim and satisfies the commitment C10. This is another case where both the agents comply with the business model. In Fig. 10(c), the policy holder reports an accident, but the call center does not receive the claim. The call center violates the detached commitment C10, and lacks compliance with the business model.

5 Completeness

Agents enter into a business relationship for achieving their respective goals. A business model in which all agents achieve their goals is complete. It is important to check for model completeness, since in its absence, some agents will not achieve their goals and therefore desire to leave the relationship. That is, the business model will not be stable.

Algorithm 2: verifyCompleteness(m): Verify completeness of business model m

```
1  $C = m.C$; // Model Commitments
2  $A = m.A$; // Agents
3  foreach $(a \in A)$ do
4      $G = a.G$; // Agent goals
5      foreach $(g \in G)$ do
6          $GT = g.T$; // Tasks for goal
7          $AT = a.T$; // Agent tasks
8          $task$: foreach $((t \in GT) \wedge (t \notin AT))$ do
9              foreach $(c \in C)$ do
10                 if $((c.creditor = a) \wedge$
                   $(t \in tasks(c.consequent)) \wedge (tasks(c.antecedent) \subset AT))$ then
11                     next $task$;
12             return false;
13 return true;
```

The Algorithm 2 checks a business model for completeness. For each agent, the algorithm checks if the agent can achieve all of its goals. An agent a can achieve a goal g, if it can execute all the tasks required for that goal. In case where the agent cannot execute all the tasks required for its goal, the model must contain commitments from other agents to execute the remaining tasks. Additionally, the agent a should be able to execute the tasks specified in the antecedents of those commitments. In the model, if there is an agent who cannot achieve a goal, then the algorithm returns false indicating that the model lacks completeness. Otherwise, the algorithm returns true.

For example, consider the AGFIL business model. The assessor has the goal of claim assessment. To assess a claim, the assessor needs to inspectVehicle and agree-ToRepair. The assessor has the capability of bringing about agreeToRepair, but it lacks the capability to inspectVehicle. In this case, for completeness, the model must contain commitment from some other agent to inspectVehicle. Additionally, the assessor should be able to bring about the antecedent of that commitment. For example, C(D, A, payAdjuster, inspectVehicle) is a commitment required for model completeness, assuming the assessor can perform payAdjuster.

6 Discussion

This section compares our approach with some existing approaches. Existing high-level approaches capture business organizations and value exchanges among them [1]. Many of these approaches are informal or semiformal and are developed for valuation and profitability analysis. They lack a rigorous treatment of business relationships (as via commitments) and lack a corresponding business-level notion of compliance.

Gordijn and Wieringa [7] propose the e^3-value approach, which captures a business organization as an actor. This is similar to the notion of an agent from our model. Actors execute value activities similar to the tasks in our model. In e^3, a value interface aggregates related in and out value ports of an actor to represent economic reciprocity. This concept is close to our concept of commitment, but it lacks formal semantics and doesn't yield equivalent flexibility. For example, unlike value interfaces, commitments can be delegated. Due to this, an e^3 model may capture value exchange among two actors, but during execution, the exchange and interaction may take place between two different actors.

Tropos [2] is an agent-oriented software methodology based on concepts of actor, goal, plan, and actor dependencies. The concepts of role, goal, and task from our model are similar to the Tropos concepts of actor, goal, and plan, respectively. A key difference between our model and Tropos is the concept of commitment. In Tropos, a dependency means that a depender actor depends on a dependee actor, for executing a plan or achieving a goal. This concept of dependency does not model what is required of the depender, and the dependee unconditionally adopts the dependency. Our debtor, creditor, and consequent are similar to the Tropos dependee, depender, and dependum, respectively. Unlike a dependency, a commitment includes an antecedent that brings it into full force. This allows modeling of reciprocal relationships between economic entities, which is lacking in the concept of dependency.

Opera is a framework for modeling multiagent societies [9], though from the perspective of a single designer or economic entity. In contrast, we model interactions among multiple entities. Opera's concepts of landmark, scene, and contract are close to our concepts of task, protocol, and commitment, respectively. However, Opera uses traditional obligations, which lack the flexibility of commitments.

Amoeba [5] is a process modeling methodology based on commitment protocols. This methodology creates model in terms of fine-grained messages and commitments. In contrast, our approach lies at a higher level of abstraction containing business goals, tasks, and commitments.

Conclusions. The main contributions of this paper are a business metamodel, a set of modeling patterns, and algorithms for verifying compliance and completeness of service engagements to business models. Our set of business model patterns is clearly not exhaustive; nor do we expect any set of patterns to be exhaustive—hundreds of patterns exist for programming and for software architecture, and the domain of business models is at least as complex as those. However, our core set of patterns shows how we may construct additional patterns. Future work includes development of a methodology for business modeling, model formalization and complexity analysis, and graphical tools for creating business models.

References

1. Birger Andersson, Maria Bergholtz, Ananda Edirisuriya, Tharaka Ilayperuma, Paul Johannesson, Jaap Gordijn, Bertrand Grégoire, Michael Schmitt, Eric Dubois, Sven Abels, Axel Hahn, Benkt Wangler, and Hans Weigand. Towards a reference ontology for business models. In David W. Embley, Antoni Olivé, and Sudha Ram, editors, *Conceptual Modeling - ER 2006, 25th International Conference on Conceptual Modeling, Tucson, AZ, USA, November 6-9, 2006, Proceedings*, volume 4215 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2006.
2. Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
3. BRG. The business motivation model, 2007.
4. Sinead Browne and Michael Kellett. Insurance (motor damage claims) scenario. Document Identifier D1.a, CrossFlow Consortium, 1999.
5. Nirmit Desai, Amit K. Chopra, and Munindar P. Singh. Amoeba: A methodology for modeling and evolution of cross-organizational business processes. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2009. In press.
6. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Professional Computing Series. Addison-Wesley, Reading, MA, 1995.
7. Jaap Gordijn and Roel Wieringa. A value-oriented approach to E-business process design. In Johann Eder and Michele Missikoff, editors, *Advanced Information Systems Engineering, 15th International Conference, CAiSE 2003, Klagenfurt, Austria, June 16-18, 2003, Proceedings*, volume 2681 of *Lecture Notes in Computer Science*, pages 390–403. Springer, 2003.
8. Munindar P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7:97–113, 1999.
9. Hans Weigand, Virginia Dignum, John-Jules Ch. Meyer, and Frank Dignum. Specification by refinement and agreement: Designing agent interaction using landmarks and contracts. In Paolo Petta, Robert Tolksdorf, and Franco Zambonelli, editors, *ESAW*, volume 2577 of *Lecture Notes in Computer Science*, pages 257–269. Springer, 2002.
10. Pınar Yolum and Munindar P. Singh. Enacting protocols by commitment concession. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 116–123, May 2007.