

# Enhancing Tropos with Commitments

## A Business Metamodel and Methodology

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.** This paper motivates a novel metamodel and methodology for specifying cross-organizational business interactions that is based on Tropos. Current approaches for business modeling are either high-level and semiformal or formal but low-level. Thus they fail to support flexible but rigorous modeling and enactment of business processes. This paper begins from the well-known Tropos approach and enhances it with commitments. It proposes a natural metamodel based on commitments and a methodology for specifying a business model. This paper includes an insurance industry case study that several researchers have previously used.

## 1 Introduction

Modern organizations form complex business relationships with their partners to create and provide value to their customers. Due to competitive pressures, modern organizations are continually forced to improve their operations. Such improvements increasingly involve outsourcing noncore business tasks, and other redistributions and realignments of business functions. A business model serves as a starting point for realizing the IT systems required to support the operations of these organizations.

Current approaches are inadequate for capturing business models in a manner that is both flexible and formal. On the one hand, management scientists have proposed a number of high-level business metamodels. However, these models are semiformal, and are useful primarily for valuation and profitability analysis. On the other hand, computer scientists have proposed low-level business metamodels, which consider abstractions centered on data and control flow. These approaches fail to capture the business meaning of the interactions.

This paper motivates a novel business metamodel and methodology based on Tropos [2], an established agent-oriented engineering methodology, as enhanced with commitments. Tropos provides a well-defined requirements engineering approach for modeling agents, and their mutual dependencies. However, it lacks appropriate treatment of agent commitments. Commitments [6] are a well-studied concept for modeling business interactions. Commitments help capture the business meaning of interactions in a manner that supports judgments of compliance

while enabling flexible enactment. Our proposed metamodel benefits from the goal, task, and dependency modeling offered by Tropos, and the semantics and flexibility offered by commitments. We exercise our approach on an insurance industry scenario, a well-known case study from the literature.

*Contributions.* The main contribution of this paper is a novel agent-oriented business metamodel and methodology. A real-life insurance claim processing scenario validates our proposal.

*Organization.* Section 2 presents our business metamodel and methodology. Section 3 applies the methodology to create a business model for an insurance claim processing scenario. Section 4 introduces the notion of agent conformance and presents an approach for verifying it. Section 5 compares our approach with related work.

## 2 Metamodel and Methodology

In our treatment, a business model captures the business organizations involved in conducting a specified element of business and the commitments between them. An organization executes business tasks either to achieve its own goals, or to satisfy its commitments toward another organization. We now define the concepts used by our metamodel using which such models may be expressed. With the exception of commitment, these concepts are based on Tropos.

**Agent:** A computational representation of a business organization. An agent abstraction captures the autonomy and heterogeneity of a real-world business. An agent has goals and capabilities to execute tasks. It enters into business relationships with other agents to acquire capabilities, that is, to get other agents to execute tasks on its behalf.

**Role:** An abstraction over agents via their related sets of commitments and tasks. An agent can adopt one or more roles in a business model.

**Goal:** A state or condition of the world that an agent would like to bring about or have brought about. In simple terms, a goal is an end. AND-OR decomposition helps decompose a root goal into subgoals.

**Task:** An abstract way of executing a business activity, that is, the means by which a goal can be achieved. Similar to a goal, a root task can be decomposed into subtasks.

**Dependency:** A relationship between two roles where one role depends upon the other for achieving a goal or executing a task. The former role is called the *dependor*, the latter is called the *dependee*, and the object of the dependency is called the *dependum*.

**Commitment:** A commitment  $C(R1, R2, p, q)$  denotes that the role  $R1$  is responsible to the role  $R2$  for bringing about the condition  $q$  if  $p$  holds. In this commitment,  $R1$  is the *debtor*,  $R2$  is the *creditor*,  $p$  is the *antecedent*, and  $q$  is the *consequent*. A commitment may be *created*. When its condition is brought about, it is *discharged*. 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. The above operations describe how a commitment can be manipulated.

A traditional obligation specifies a condition that an agent ought to bring about. Unlike commitments, an obligation cannot be assigned, delegated, canceled, or released.

Table 1 outlines the steps in our proposed methodology. The subsections below describe each step in detail.

**Table 1.** Methodology steps

Step	Description	Input	Output
S1	Identify agents and roles	Scenario description, process flows, domain knowledge	Agents and roles
S2	Determine goals and goal dependencies	Roles, scenario description, process flows, domain knowledge	Goals and goal dependencies
S3	Identify tasks and task dependencies	Roles, goal dependencies, scenario description, process flows, domain knowledge	Tasks and task dependencies
S4	Identify commitments	Task dependencies, scenario description, process flows, domain knowledge	Commitments

### 2.1 Step S1: Agent and Role Identification

Agents represent the business organizations participating in the scenario of interest. A scenario description typically specifies the agents using terms like company, partner, and organization. If there is a single agent of its kind, then the scenario description usually specifies a unique name for it. In case we have multiple agents of the same kind, a scenario description may specify a role name. For each uniquely named agent, the business functions it provides yields the associated role.

A traditional process flow, if available, can help us identify the roles. For example, a *role* in a choreography corresponds to a role in our business model. Similarly, *partner link* in an orchestration, corresponds to a role in our model.

### 2.2 Step S2: Determine Goals and Dependencies

This step iteratively determines the goal dependencies between the roles. First, it identifies the main roles and their high-level goal dependencies. Second, using means-end and AND-OR decomposition analysis, it refines the high-level root

goals into subgoals. Third, this step introduces roles that adopt these subgoals. It then iteratively refines the subgoals until no new dependencies arise.

The scenario description may explicitly or implicitly specify business dependencies between participating organizations. These dependencies yield the goal dependencies between the roles. In cases where the dependencies are not clear from the scenario, additional business insight may be needed.

Thus, Step S2 identifies goals adopted by different roles, that is, the goals of each dependee role.

### **2.3 Step S3: Determine Tasks and Dependencies**

For each role, this step refines the goal dependencies from Step S2 into task dependencies. A set of tasks achieves a goal. The means-end analysis identifies the set of tasks required for achieving a goal. A task is refined into subtasks using AND-OR decomposition. The refined tasks identify dependencies that are not evident at the higher level task. The decomposition iterates until no new task dependencies emerge.

As we analyze additional roles, we may discover new task dependencies, requiring the addition of any missing tasks and goals to the (potentially already analyzed) roles.

### **2.4 Step S4: Identify Commitments**

This step identifies commitments between roles in terms of tasks. It analyzes each task dependency from Step S3 to identify if a commitment exists corresponding to that dependency. For a task dependency, if the dependee is obliged to the depender for executing the dependum task, then a commitment exists, where the depender is the creditor, the dependee is the debtor, and the dependum task is the consequent. The antecedent of the commitment is determined by identifying the tasks that the debtor requires as prerequisites for executing the consequent task. If the dependee is not obliged to the depender for executing a task, then no commitment exists. This implies that the dependee executes the task to achieve its internal goal.

Although the scenario description and process flow may contain information that yields commitments, additional human insight is typically required to correctly identify the commitments.

## **3 Methodology Applied to a Real-World Case**

This section describes a real-world insurance claim processing use case. It further describes an application of our methodology and the resulting model for that use case. Figure 1 clarifies the notation used in the figures that follow.

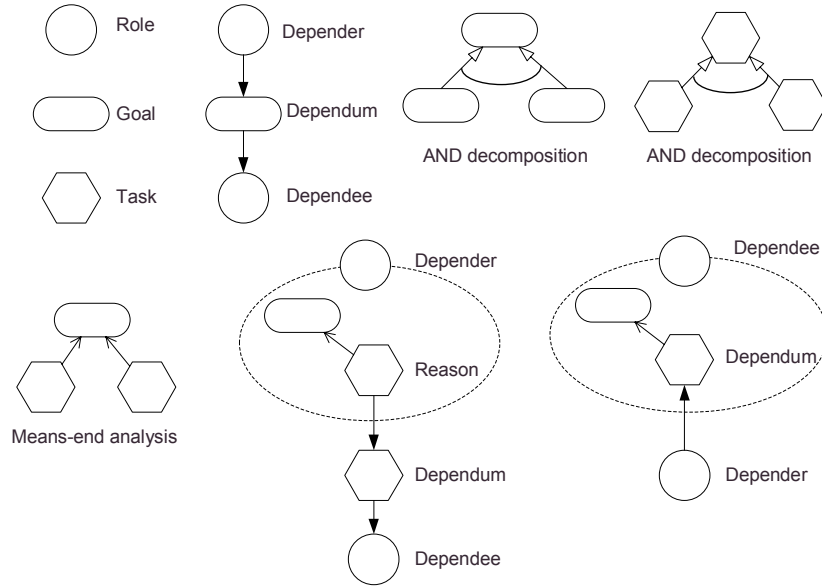


Fig. 1. Notations

### 3.1 Insurance Claim Processing Scenario

AGFIL [3] is an insurance company in Ireland. It underwrites insurance policies and covers losses incurred by policy holders. AGFIL provides an emergency service to its policy holders. Figure 2 shows the parties involved in providing emergency service, their individual processes, and the control and data flows among these processes.

To provide emergency service, AGFIL must provide claim reception and vehicle repair to the policy holders. Additionally, it needs to assess claims to protect itself against fraud. AGFIL uses its partners, Europ Assist, Lee Consulting Services (CS), and various repairers, for executing these tasks. Europ Assist provides a 24-hour help-line to customers for reporting a claim, and provides them the name of an approved repairer facility. Lee CS performs the necessary assessment and additionally presents invoices to AGFIL on behalf of the repairers. Several approved repairers provide repair services. AGFIL remains in charge of making the final decision on claim approvals, and making the payment.

### 3.2 Step S1

The insurance claim processing scenario from Section 1 specifies AGFIL, EA, and Lee CS as uniquely named agents. The process flow also shows these agents. These agents serve the business functions of insurer, call center, and assessor, respectively. Therefore, we can induce the corresponding roles from them. The description additionally specifies the repairer, claim adjustor, and policy holder roles that are enacted by multiple agents.

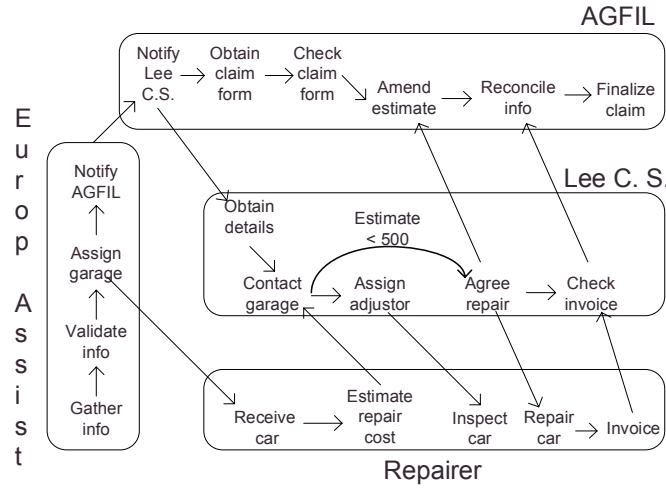


Fig. 2. Insurance claim processing [3]

### 3.3 Step S2

In the AGFIL scenario, the main roles are insurer and policy holder. A policy holder depends upon the insurer for receiving emergency service and, in exchange, the insurer depends upon the policy holder for the insurance premium payment. Fig. 3 shows these dependencies using the Tropos notation.

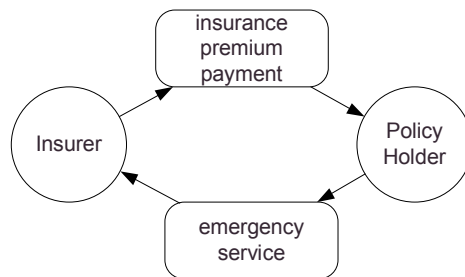


Fig. 3. Insurer and policy holder goal dependencies

Using AND decomposition, in Fig. 4, the insurer’s goal of emergency service yields subgoals of claim reception, claim assessment, vehicle repair, and claim finalization. Among these, the policy holder depends on the insurer for vehicle repair and claim reception. The insurer requires additional subgoals to provide emergency service but these additional goals do not involve a dependency from the policy holder. As the goal structure is refined and later, in Step S3, when the task structure is identified, a dependency between one of these subgoals and the policy holder may be discovered.

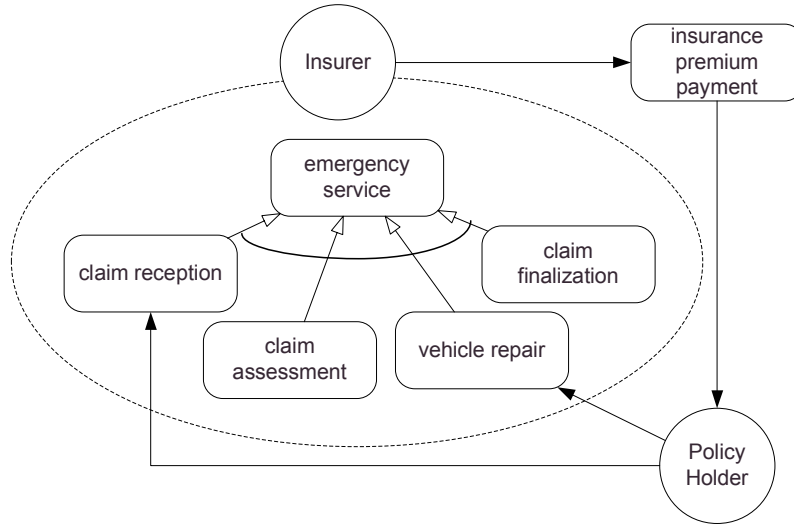


Fig. 4. Emergency service goal decomposition

In Fig. 5, the insurer delegates claim reception to the call center. The policy holder now depends upon the call center for claim reception. In exchange for claim reception, the insurer pays service charges to the call center. Fig. 5 omits the claim assessment and finalization subgoals as they are not dependent upon other roles.

In Fig. 6, the insurer delegates claim assessment to the assessor. The assessor role and a dependency from the insurer to the assessor is added to the model. In exchange, the assessor depends upon the insurer for the payment of assessment fees.

Fig. 7 shows all roles and goal dependencies. The assessor delegates vehicle inspection, which is a subgoal of claim assessment, to an adjustor. The adjustor,

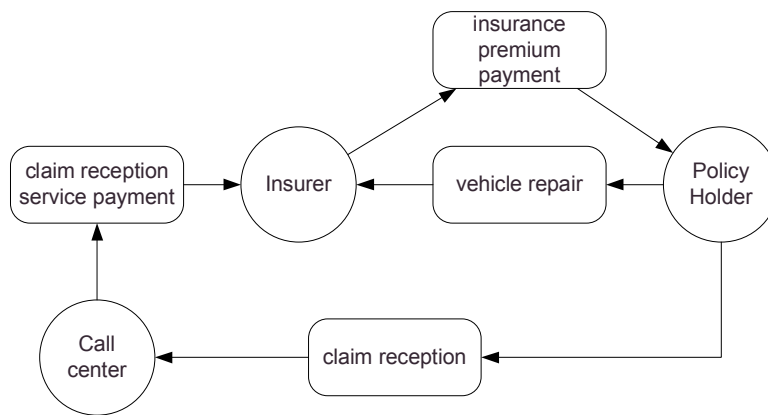


Fig. 5. Insurer delegates claim reception

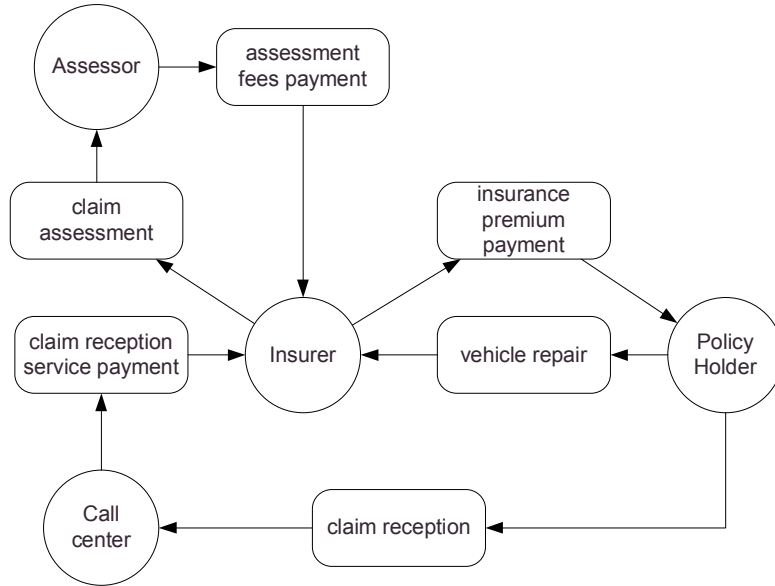


Fig. 6. Insurer delegates claim assessment

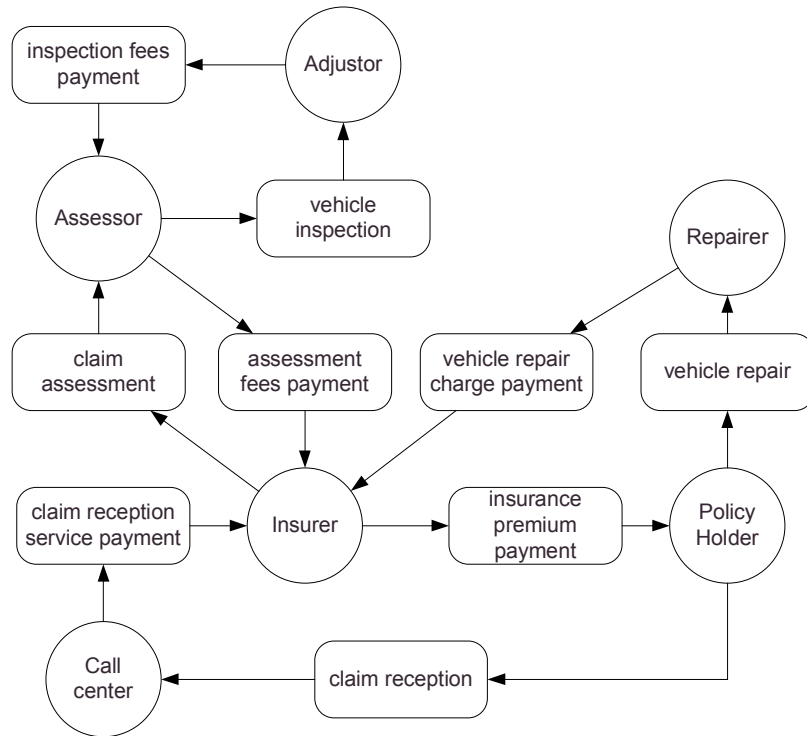


Fig. 7. AGFIL goal dependency model

in exchange, depends upon the assessor for inspection fees payment. The insurer delegates vehicle repair to a repairer. In exchange, the repairer depends upon the insurer for vehicle repair payment.

### 3.4 Step S3

Fig. 8 shows the tasks and the task dependencies we identify by analyzing the call center role. From S2, the call center has the goal of claim reception. By performing means-end analysis on this goal, we obtain tasks of gathering claim information, assigning garage, sending claim, and validating claim.

When the policy holder reports an accident, the call center gathers claim information and assigns a garage. This means that the policy holder depends upon the call center for gathering claim information and assigning a garage. Additionally, the repairer depends on the call center to assign a garage.

Using AND-OR decomposition of the validate claim information task, we obtain two subtasks: request policy information and validate. The call center depends upon the insurer for providing policy information, and it performs validation without any dependency.

From Step S2, the call center depends on the insurer for payment of the claim reception charge. This yields a dependency from the call center to the insurer for the task of paying claim reception charge. The call center executes a task of receiving this payment and it derives a new call center goal of service charge collection.

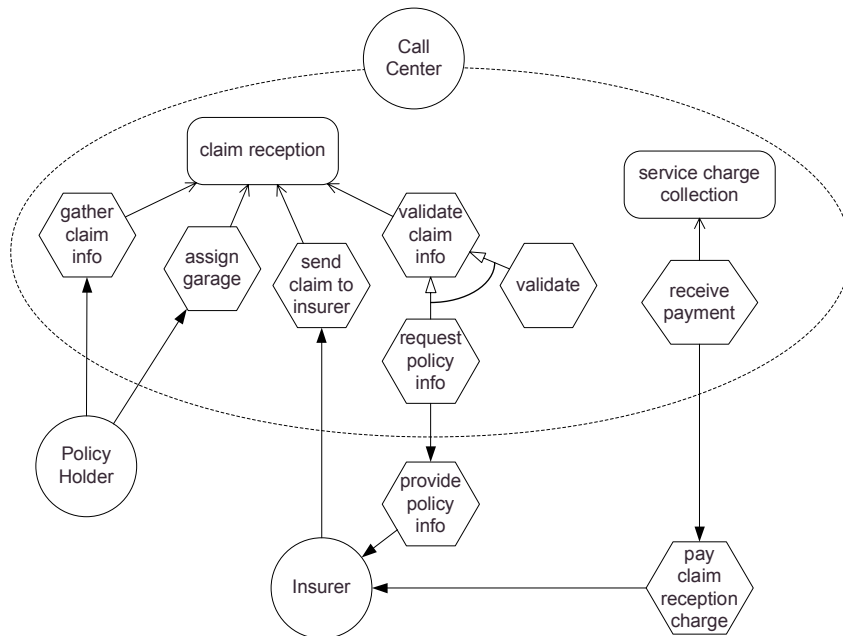
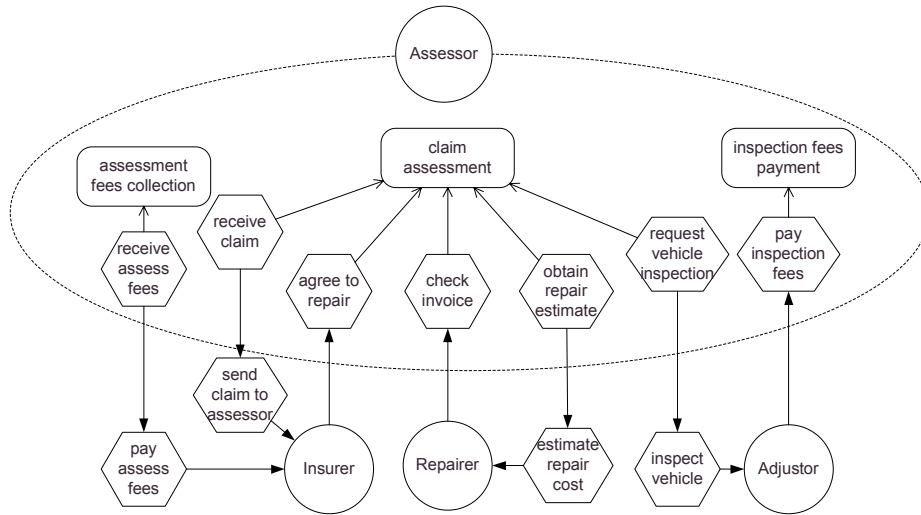


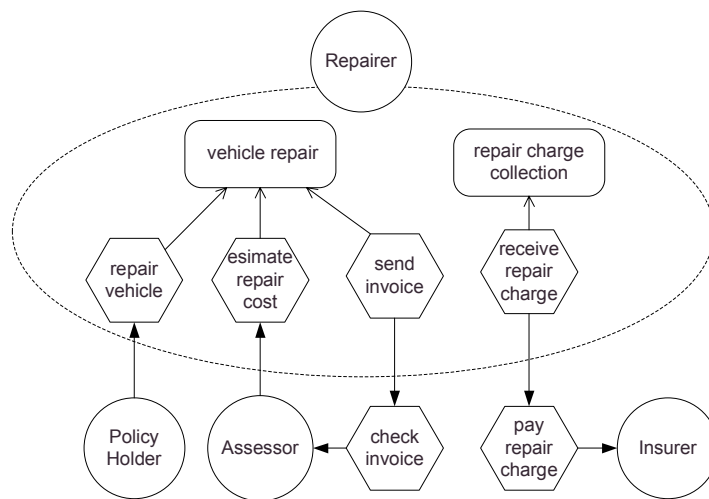
Fig. 8. Call center task dependencies



**Fig. 9.** Assessor task dependencies

The call center sends a validated claim to the insurer for further processing. This yields a dependency from the insurer to the call center for sending the claim.

Fig. 9 shows the tasks and the task dependencies derived for the assessor role. The assessor has goals of claim assessment and inspection fees payment. A new goal of assessment fees collection is derived similarly to the goal of receiving service payment of the call center.



**Fig. 10.** Repairer task dependencies

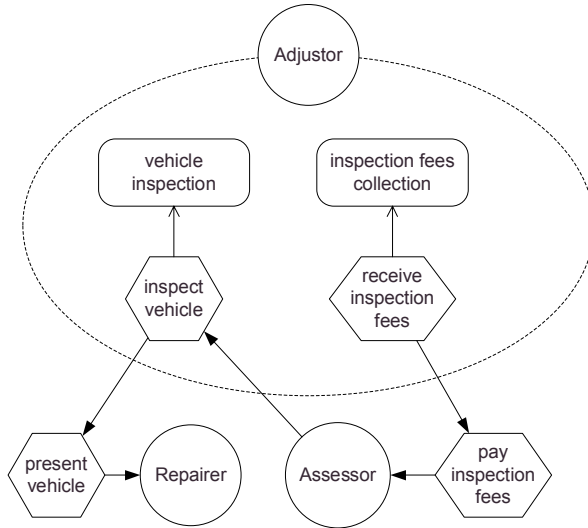


Fig. 11. Adjustor task dependencies

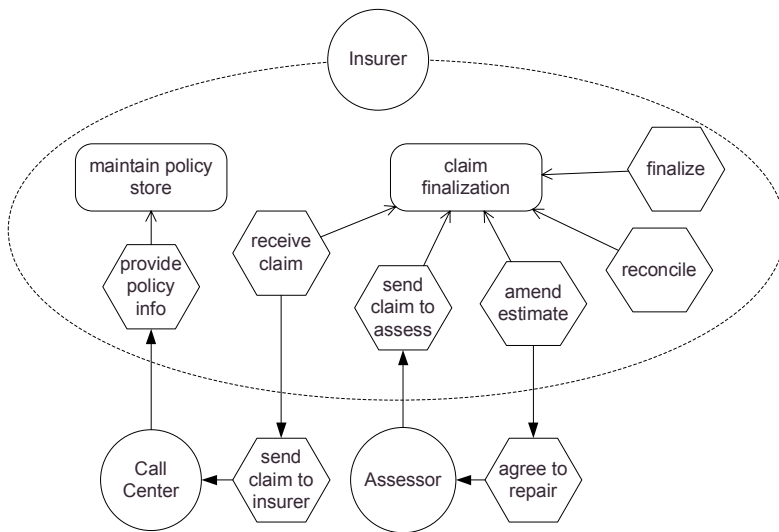
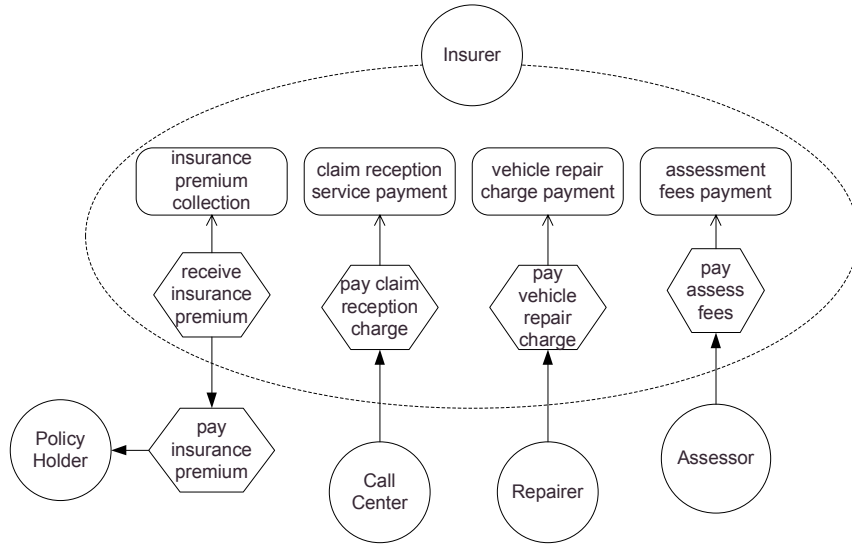


Fig. 12. Insurer task dependencies: Business function goals

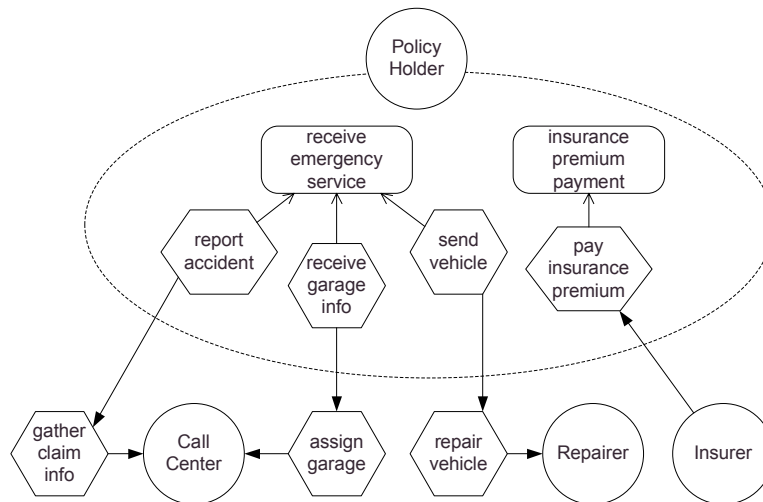
The goal of assessment fees collection requires a receive assessment fees task, which depends upon the insurer’s pay assessment fees task. The tasks needed to cover claim assessment goal are receive claim, check invoice, agree to repair, obtain repair estimate, and inspect vehicle. The receive claim task depends upon the insurer’s task of sending claim. The insurer depends upon the check invoice and agree to repair tasks performed by the assessor. The repairer depends upon



**Fig. 13.** Insurer task dependencies: Payment related goals

the assessor for checking the invoice and agreeing to repair. For obtaining the repair estimate, the assessor depends upon the repairer to estimate the repair cost. The assessor depends upon the adjustor to inspect a vehicle. The goal of inspection fees payment requires a task of paying inspection fees to the adjustor.

Similarly, we analyze repairer, adjustor, insurer, and policy holder roles to obtain the task dependencies shown in Figs. 10, 11, 12, 13, and 14 respectively.



**Fig. 14.** Policy holder task dependencies

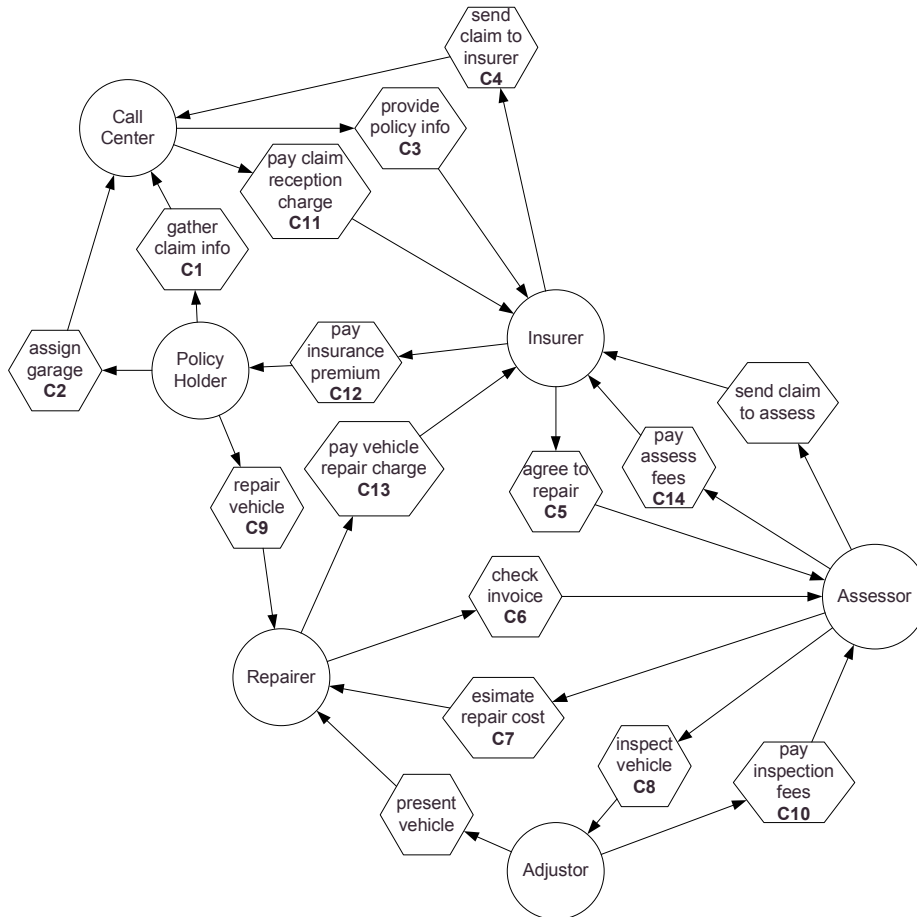


Fig. 15. All task dependencies

Fig. 15 consolidates the task dependencies among all roles. This figure shows only the dependum tasks and hides the tasks that are reasons for the individual dependencies. For example, Fig. 15 shows the pay repair charge task but does not show the receive repair charge task.

### 3.5 Step S4

This step identifies the commitments for each task dependency from Step S3. Fig. 15 annotates each task dependency with the corresponding commitment. Table 2 summarizes these commitments.

The commitment C1 means that the call center commits to the policy holder for gathering claim information if the policy holder reports an accident. In C2, the call center commits to assigning a garage if the policy holder reports an accident and if the claim request is valid. In C3, the insurer commits to providing

**Table 2.** Commitments for AGFIL scenario

<b>Id</b>	<b>Task</b>	<b>Commitment</b>
1	gather claim info	$C(\text{CALL CENTER, POLICY HOLDER, reportAccident, gatherInfo})$
2	assign garage	$C(\text{CALL CENTER, POLICY HOLDER, reportAccident} \wedge \text{validClaim, assignGarage})$
3	provide policy info	$C(\text{INSURER, CALL CENTER, reqPolicyInfo, providePolicyInfo})$
4	send claim to insurer	$C(\text{CALL CENTER, INSURER, reportAccident} \wedge \text{validClaim} \wedge \text{pay-ClaimRecCharge, sendClaimToInsurer})$
5	agree to repair	$C(\text{ASSESSOR, INSURER, sendClaimToAssess} \wedge \text{payAssessFees, agreeToRepair})$
6	check invoice	$C(\text{ASSESSOR, REPAIRER, sendInvoice, checkInvoice})$
7	estimate repair cost	$C(\text{REPAIRER, ASSESSOR, reqEstimate, estimateRepairCost})$
8	inspect vehicle	$C(\text{ADJUSTOR, ASSESSOR, reqInspection} \wedge \text{payInsFees, inspectVehicle})$
9	repair vehicle	$C(\text{REPAIRER, POLICY HOLDER, validClaim, repairVehicle})$
10	pay inspection fees	$C(\text{ASSESSOR, ADJUSTOR, inspectVehicle, payInsFees})$
11	pay claim reception charge	$C(\text{INSURER, CALL CENTER, C1} \wedge \text{C2} \wedge \text{C4, payClaimRecCharge})$
12	pay insurance premium	$C(\text{POLICY HOLDER, INSURER, C9, payInsurancePremium})$
13	pay vehicle repair charge	$C(\text{INSURER, REPAIRER, repairVehicle, payRepairCharge})$
14	pay assessment fees	$C(\text{INSURER, ASSESSOR, agreeToRepair, payAssessFees})$

policy information if the call center requests it. The commitment C4 means that the call center commits to sending a claim to the insurer if it receives a valid claim and the insurer pays claim reception charge. In C5, the assessor commits to the insurer to negotiate and bring about the agreement to repair provided the insurer requests claim assessment and pays the assessment fees. Commitment C6 means that the assessor commits to the repairer for checking the invoice and for forwarding it to the insurer, provided the repairer sends the invoice. In C7, the repairer commits to the assessor for estimating the repair cost if requested. In C8, the adjustor commits to the assessor for inspecting the vehicle if the assessor requests inspection, and pays for it. The commitment C9 means that the repairer commits to the policy holder for repairing the vehicle provided the claim is valid. In C10, the assessor commits to paying the adjustor for inspection if the vehicle is inspected. The commitment C11 means that the insurer commits to the call center for paying if the call center creates commitments C1, C2, and

C4. That is, if the call center commits to gathering claim information, assigning a garage, and sending the claim to insurer. In C12, the policy holder commits to paying the insurance premium to the insurer if commitment C9 is created for repairing the vehicle. In C13, the insurer commits to the repairer for payment if the vehicle is repaired. The commitment C14 means that the insurer commits to the assessor for payment if the assessor brings about the agreement to repair.

There is no commitment associated with some of the task dependencies. For example, the assessor depends upon the insurer for sending a claim for assessment. In this case, the insurer does not commit to sending the claim to the assessor.

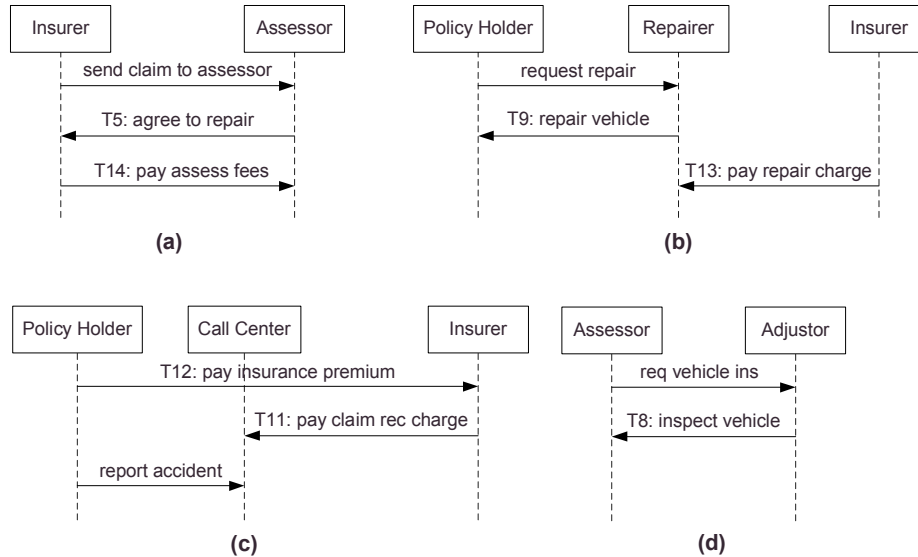
## 4 Verifying Agent Interactions

Since agents are autonomous (based on the fact that they represent autonomous business organizations), they may not conform to a given model. So it is important to verify agent interactions with respect to a model. A business model captures commitments between agents. The commitments provide a basis for verifying agent interactions for conformance with the specified business model.

We consider a UML sequence diagram as a low-level model for agent interactions as they are realized. The roles appear as objects in this diagram and they exchange messages. Roles may exchange multiple messages for executing one task. For example, consider the task of reporting an accident. The policy holder sends a message to the insurer for this task. If the information in that message is incomplete, the insurer may send a message to the policy holder requesting additional information. This would repeat until the insurer receives all information, at which point the task of reporting accident would complete.

An agent conforms to a business model if it satisfies each commitment of which it is the debtor and whose antecedent holds. To verify conformance, we iterate over active commitments from the business model. For each commitment, we evaluate its antecedent and consequent using the tasks and the domain facts asserted in the interaction model. Each commitment whose consequent evaluates to true is satisfied. Each commitment whose antecedent evaluates to true, but consequent to false, is a detached commitment that is violated. Hence, the debtor of a violated commitment is the agent that fails conformance with respect to the business model.

The agent interactions can be verified either at design time or at run time. At design time, a low-level interaction model design can be verified against a business model. There can be many interaction models that satisfy a given business model. At run time, the emergent agent behavior, captured in the form of a low-level interaction model, can be verified against a business model. When such verification is performed, some interactions may still be pending, and therefore, some commitments may be eventually satisfied. Therefore, to detect violations we must model the various tasks as being time bounded, that is, as including timeouts. A commitment whose antecedent evaluates to true but consequent to false (taking timeouts into consideration) is violated. Figure 16 shows a series of conforming and nonconforming interactions for AGFIL scenario. A message labeled  $T_i$  corresponds to the task  $i$  in Table 2.



**Fig. 16.** Conforming and nonconforming agent interactions

- Fig. 16(a) shows an interaction between the insurer and the assessor. The insurer sends a claim for assessment to the assessor. The assessor negotiates the repair charge with the repairer, and brings about an agreement with the repairer for vehicle repair. This satisfies commitment C5, and detaches commitment C14. The insurer pays the assessment fees to the assessor, and satisfies C14. Since, the insurer and the assessor satisfy their commitments, they conform to the business model.
- Fig. 16(b) shows another example of conforming interaction between a policy holder, a repairer, and the insurer. The policy holder provides claim information, and requests the repairer for vehicle repair. The repairer finds the claim to be valid, and repairs the vehicle. The repairer satisfies commitment C9 and detaches commitment C13. By paying the repair charge, the insurer satisfies commitment C13.
- Fig. 16(c) shows an example of nonconforming interaction between a policy holder, the call center, and the insurer. The policy holder obtains insurance by paying the requisite premium to the insurer. The insurer pays the call center for providing claim reception service. Then the policy holder reports an accident and detaches commitments C1, C2, and C4, assuming the claim to be valid. But the call center does not gather the claim information, assign a garage, or send the claim to the insurer. Therefore, the call center violates commitments C1, C2, and C4, and does not conform to the model.
- Fig. 16(d) shows another example of a nonconforming interaction. The assessor requests a vehicle inspection from the adjustor. The adjustor inspects the vehicle, and detaches commitment C10. However, the assessor violates commitment C10 by not paying the inspection fees to the adjustor, and therefore fails to conform with the model.

## 5 Discussion

This paper proposes an agent-oriented business metamodel based on Tropos. The model uses the mental and social concepts of Tropos. It also uses the goal, plan, and dependency modeling techniques from Tropos. The goal and plan modeling are based on the means-end and AND-OR analyses.

Our approach offers two major benefits. One, the high-level metamodel captures the business relationships directly. Thus it shows how the business model may be modified in the face of changing business needs. For example, the insurer may decide to outsource claim handling (as in the above case) or may decide to insource it. A business analyst can readily determine if the resulting business model is sound. Likewise, each participant can evaluate a potential change to the business model in terms of whether it would affect the commitments of which it is the debtor or the creditor. Two, the high-level metamodel yields a natural basis for reasoning about correctness. We can use the business relationships as a basis for determining whether a particular enactment is conforming and whether a particular way to generate an enactment is sound.

Tropos is a general purpose agent-oriented software engineering methodology. It can be applied to a wide range of software applications, and it covers all phases of software development. In contrast, our proposed methodology is tailored specifically for business modeling.

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. The concept of dependency does not model what is required of the depender, and the dependee unconditionally adopts the dependency. The debtor, creditor, and consequent of a commitment are similar to the Tropos dependee, depender, and dependum, respectively. However, unlike a dependency, a commitment includes an antecedent that brings it into full force. This enables modeling reciprocal relationships between economic entities, which is lacking in the concept of dependency.

Andersson *et al.* [1] present what they call a “reference” ontology for business models based on concepts from three approaches, namely, REA, BMO, and e<sup>3</sup>-value. Table 3 compares their ontology concepts to the concepts from our model. Their concepts of actor and actor type are similar to our agent and role, respectively. A domain ontology captures resource, resource type, feature, and right in our approach. Our task abstraction is similar to the concept of event without additional classification into types. Andersson *et al.*’s notion of commitment is close to our concept of commitment. Their concepts of exchange, transaction, contract, agreement, and reciprocity are reflected as two or more commitments in our approach. Unlike our metamodel, Andersson *et al.* do not model actor goals.

Gordijn and Wieringa [5] propose e<sup>3</sup>-value business model. Unlike our metamodel, e<sup>3</sup>-value is a semiformal model based on economic concepts, and is mainly intended for profitability analysis. A value interface in e<sup>3</sup>-value 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 similar semantics and flexibility. For example, unlike a value interface, a commitment can be delegated.

**Table 3.** Proposed business metamodel related to Andersson *et al.*'s ontology

Reference ontology concept	Business metamodel concept
Actor	Agent
Actor type	Role
A pair of transfers	A pair of commitments
Resource, resource type, feature, right	Defined in domain ontology
Event	Task
Commitment	Commitment
Exchange	A pair of commitments
Transaction	Set of commitments
Contract	Set of commitments
Agreement	Commitment
Reciprocity	A pair of commitments
Claim	Detached commitment

Due to this, an e<sup>3</sup>-value model may capture value exchange among two actors, but during execution, the exchange and interaction may take place between two different actors, and without a clear notion of delegation, it is not obvious how the latter is selected.

Opera is a framework for modeling multi-agent societies [7], though from the perspective of a single designer or economic entity. In contrast, we model interactions among multiple entities. Opera's concepts of landmark and contract are close to our concepts of task and commitment, respectively. However, Opera uses traditional obligations, which lack the flexibility of commitments. Unlike obligations, a commitment can be manipulated as Sec. 2 describes.

Amoeba [4] is a process modeling methodology based on commitment protocols. This methodology creates a process model in terms of fine-grained messages and commitments. In contrast, our model is at a higher level of abstraction and includes business goals and tasks in addition to commitments.

## References

1. Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T., Johannesson, P., Gordijn, J., Grégoire, B., Schmitt, M., Dubois, E., Abels, S., Hahn, A., Wangler, B., Weigand, H.: Towards a reference ontology for business models. In: Embley, D.W., Olivé, A., Ram, S. (eds.) ER 2006. LNCS, vol. 4215, pp. 482–496. Springer, Heidelberg (2006)
2. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems* 8(3), 203–236 (2004)
3. Browne, S., Kellett, M.: Insurance (motor damage claims) scenario. Document Identifier D1.a, CrossFlow Consortium (1999)

4. Desai, N., Chopra, A.K., Singh, M.P.: Amoeba: A methodology for modeling and evolution of cross-organizational business processes. *ACM Transactions on Software Engineering and Methodology, TOSEM* (to appear, 2009)
5. Gordijn, J., Wieringa, R.: A value-oriented approach to E-business process design. In: Eder, J., Missikoff, M. (eds.) *CAiSE 2003. LNCS*, vol. 2681, pp. 390–403. Springer, Heidelberg (2003)
6. Singh, M.P.: An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law* 7, 97–113 (1999)
7. Weigand, H., Dignum, V., Meyer, J.-J.C., Dignum, F.: Specification by refinement and agreement: Designing agent interaction using landmarks and contracts. In: Petta, P., Tolksdorf, R., Zambonelli, F. (eds.) *ESAW 2002. LNCS*, vol. 2577, pp. 257–269. Springer, Heidelberg (2003)