ONLINE APPENDIX

MAPPING SEQUENCE DIAGRAMS TO NUSMV MODULES

The UML 2.0 specification [8] defines the *sequence diagram* to model the behavior of a system. A sequence diagram represents each participant as a vertical line. A directed horizontal line between any two participants represents a message. The ordering of the horizontal lines gives the temporal order of the messages. To modularize large and complex interactions, the specification further defines operators *alt*, *opt*, and *loop*.

We now describe how we build a NuSMV FSM from a sequence diagram. A sequence diagram yields a finite state machine with Boolean state variables, one per message. A state variable is false until the participants exchange the corresponding message, at which time it becomes true. We assume that the messages do not cross, that is, messages arrive at a recipient in the same order in which they are sent by a sender. In the finite state machine, each message exchange causes a state transition. We consider four fundamental patterns that compose a sequence diagram: a pattern containing n message exchanges, a pattern containing *alt* (alternate), a pattern containing *opt* (optional), and a pattern containing *loop*. An algorithm maps each of these patterns to the NuSMV language, and composes them to obtain the mapping for a sequence diagram. Due to lack of space, we cannot present the details of the mapping.

Listing 2. FSM model (in NuSMV) for Figure 16(c).

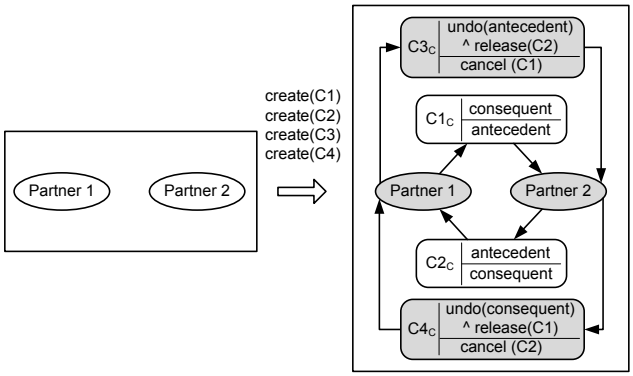
```
1  VAR
2  reqQuoteDS: boolean;
3  quoteSD: boolean;
4  ...
5
6  ASSIGN
7  init(reqQuoteDS) := 0;
8  init(quoteSD) := 0;
9  ...
10
11 TRANS
12 confirmOrderDC & !reqQuoteDS
13 & next(reqQuoteDS) = 1
14 & next(quoteSD) = quoteSD
15 & next(acceptDS) = acceptDS
16 & next(confirmShipReqSC) =
       confirmShipReqSC |
17
18 reqQuoteDS & !quoteSD & next(quoteSD) = 1
19 & next(reqQuoteDS) = reqQuoteDS
20 & next(acceptDS) = acceptDS
21 & next(confirmShipReqSC) =
       confirmShipReqSC |
22
23 ...
```

As an example, Listing 2 shows the mapping of sequence diagram from Figure 16(c) to NuSMV. The VAR and the ASSIGN sections of this listing declare and initialize the FSM state variables respectively. Lines 11–15 declare a transition from a state in which the guard confirmOrderDC holds and reqQuoteDS does not hold, to a state in which both of them hold. In this transition, the rest of the variables remain unchanged between the two states.

CANCEL ORDER SCENARIO

Section 3 models the scenario in which a customer places an order. At a later time, the customer may decide to cancel the order. This section presents revert commercial transaction pattern, which enables modeling of scenarios such as order cancellation.

Revert Commercial Transaction



C_1 C(PARTNER 1, PARTNER 2, antecedent, consequent)
C_2 C(PARTNER 2, PARTNER 1, consequent, antecedent)
C_3 C(PARTNER 1, PARTNER 2, cancel(C_1), undo(antecedent) $\wedge$ release(C_2))
C_4 C(PARTNER 2, PARTNER 1, cancel(C_2), undo(consequent) $\wedge$ release(C_1))

Fig. 19. Revert Commercial transaction.

Intent: In the commercial transaction pattern, trading partners commit to executing certain tasks for each other. This pattern captures the scenario in which the partners desire to cancel the commercial transaction.

Motivation: In a commercial transaction, a buyer and a seller agree to exchange goods for payment. Later the buyer or the seller may desire to cancel the exchange.

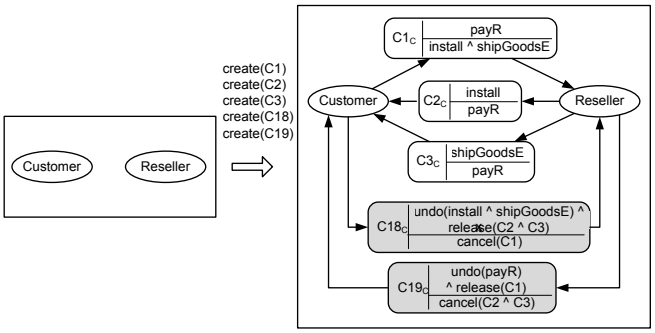
Implementation: In case of cancellation, every debtor commits to undoing the tasks that the creditor executed, that is the antecedent, and releases the creditor from the reciprocal commitment. Figure 19 shows this pattern. The meaning of undo depends upon the task. For example, undo(goods) means returning the goods, and undo(pay) means refunding the payment.

Consequences: This pattern applies in conjunction with the commercial transaction pattern.

Applying revert commercial transaction pattern

To each of the commercial transactions in the QTC model from Section 3, we apply the revert commercial transaction pattern. Next, we describe a few applications of this pattern.

- Commercial transaction between the customer and the reseller: Figure 20 shows this pattern.

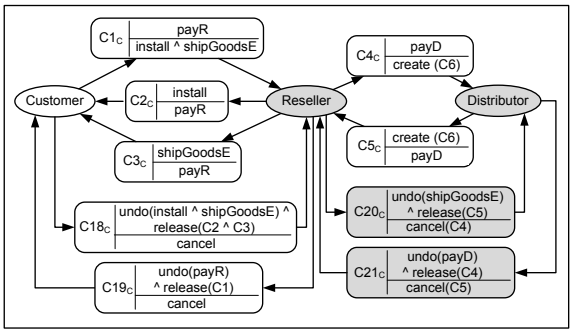


C_{18} C(CUSTOMER, RESELLER, cancel(C_1), undo(install $\wedge$ shipGoodsE) $\wedge$ release($C_2 \wedge C_3$))
C_{19} C(RESELLER, CUSTOMER, cancel($C_2 \wedge C_3$), undo(payR) $\wedge$ release(C_1))

Fig. 20. Revert Commercial transaction: Customer purchases the goods and the installation service from the reseller.

If the customer or the reseller request cancellation (cancel), the customer commits (C_{18}) to returning the goods if the goods are shipped, paying for the installation if the installation is done (undo(install $\wedge$ shipGoodsE)), and releasing the reseller from C_2 and C_3. Conversely, the reseller commits (C_{19}) to the customer to refunding the payment (undo(payR))if the customer paid, and releasing the customer from C_1.

- Commercial transaction between the reseller and the distributor: Figure 21 shows this pattern.

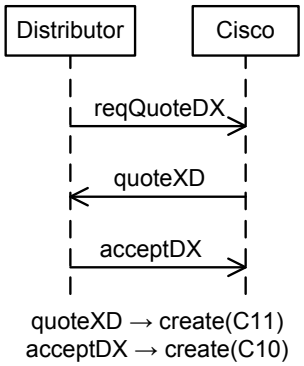


C_{20} C(RESELLER, DISTRIBUTOR, cancel(C_4), undo(shipGoodsE) $\wedge$ release(C_5))
C_{21} C(DISTRIBUTOR, RESELLER, cancel(C_5), undo(payD) $\wedge$ release(C_4))

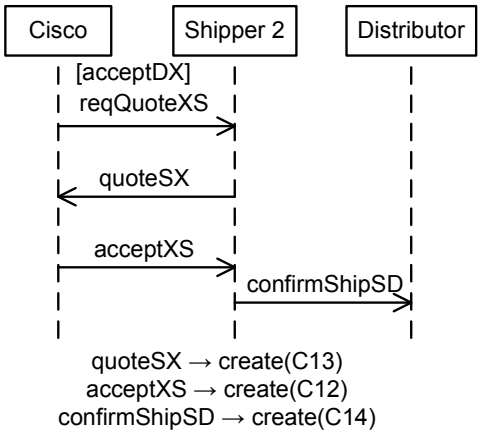
Fig. 21. Revert Commercial transaction: Reseller outsources the shipping of the goods to the distributor.

As Section 3 describes, the reseller outsources the shipping of goods to the distributor, and the outsourcing pattern (C_4, C_5, and C_6) captures it. The outsourcing pattern includes a commercial transaction pattern (C_4 and C_5) to which we apply the revert pattern. The reseller commits (C_{20}) to the distributor to returning the goods if the distributor shipped the goods (undo(shipGoodsE)), and to releasing the reseller from C_5. Note that the reseller commits to undoing shipGoodsE which is the consequent of the commitment C_6, instead of undoing create of C_6. The distributor commits to the reseller to refunding the payment if the reseller paid (undo(payD)), and to releasing the reseller from C_4.

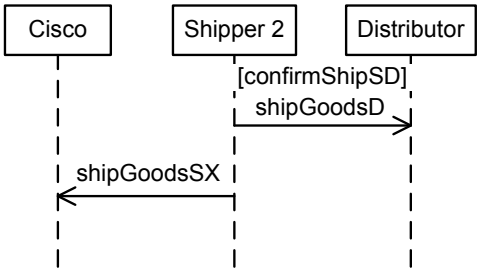
Similarly, the revert commercial transaction pattern applies to the remaining commercial transactions in the QTC model.



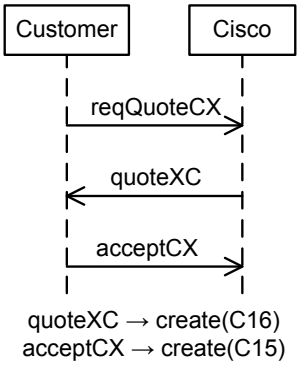
(e) A distributor requests a quote from Cisco.



(f) Cisco requests a quote from a shipper.

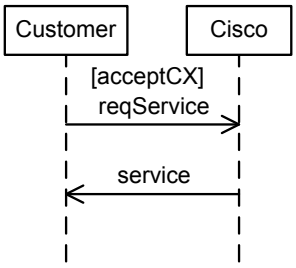


(g) The shipper ships goods to the distributor.

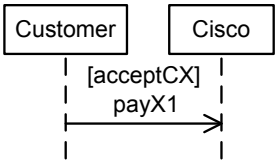


(h) A customer requests a quote from Cisco to purchase product support service.

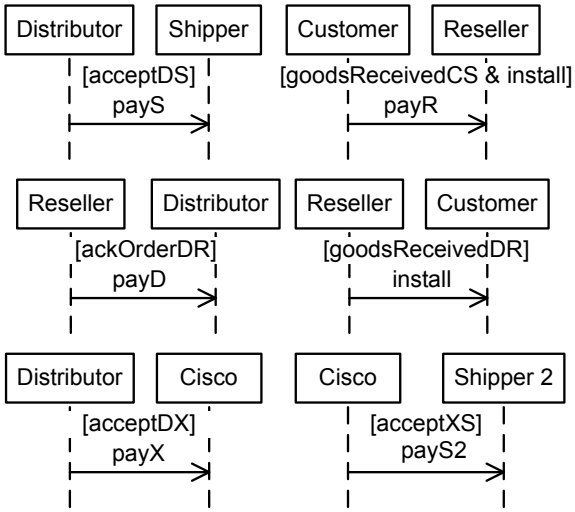
Fig. 22. Operational interactions in a QTC implementation.



(i) A customer requests service from Cisco.



(j) Customer pays Cisco



(k) Various payments, and install

Fig. 23. Operational interactions in a QTC implementation.

Business Model for Insurance Scenario

To substantiate the generality of the patterns from Section 2.1 we evaluate them on a second real-world scenario involving AGFIL, an insurance company in Ireland [30]. AGFIL provides vehicle insurance service to the policy holders. As part of the vehicle insurance service, AGFIL provides claim reception and vehicle repair to the policy holders. AGFIL outsources the claim reception to Europ Assist, a call center provider, and vehicle repair to various repairers in its partner network. To guard against fraud, AGFIL needs to assess claims, which it outsources to Lee Consulting Services. Lee Consulting Services employs inspectors to inspect the vehicle damage. It negotiates and approves the repair charge as reported by the repairers. Further, it presents invoice of repair charge to AGFIL. Figure 24 shows the AGFIL cross-organizational scenario expressed as a business model. The figure describes the commitments that model the scenario.

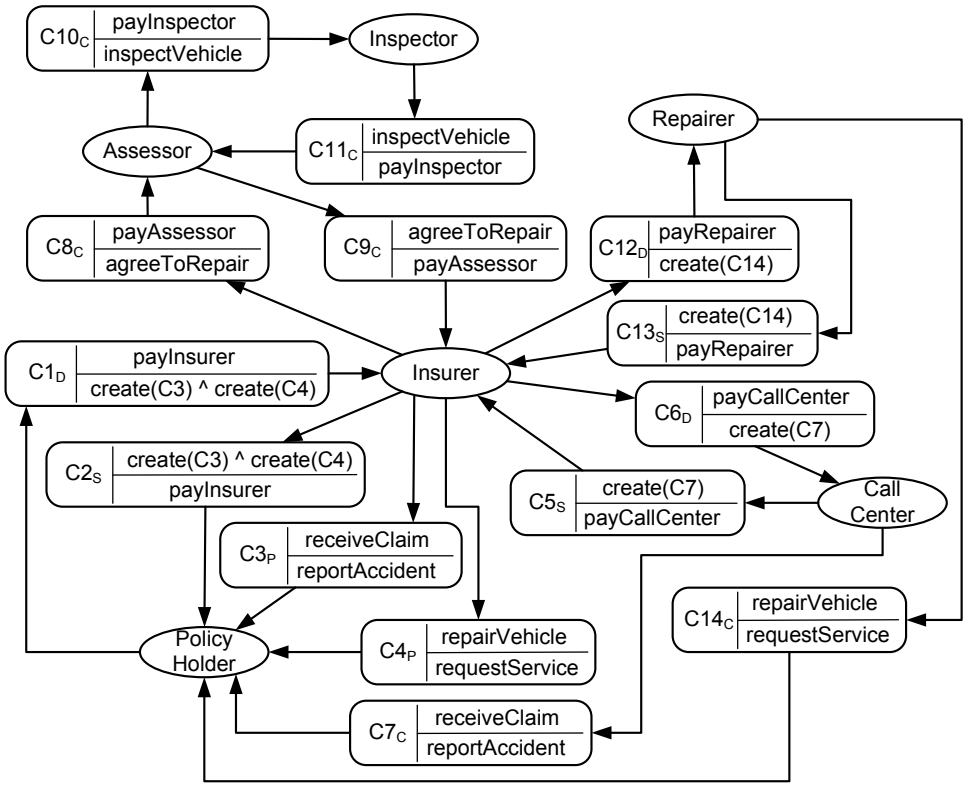
A policy holder and AGFIL (insurer) negotiate and agree upon the insurance payment. The *commercial transaction pattern* models this interaction. The policy holder commits (C_1) to the insurer to paying if the insurer creates commitments C_3 and C_4. The insurer commits (C_2) to create C_3 and C_4 if the policy holder pays the insurer. As part of the insurance service, the insurer commits (C_3) to receiving a claim if the policy holder reports an accident, and commits (C_4) to repairing the vehicle if the policy holder requests repair service.

AGFIL (insurer) outsources the claim reception to Europ Assist (call center). The *outsourcing pattern* models this interaction. The call center and the insurer agree upon the payment for the claim reception service, and create the reciprocal commitments C_5 and C_6. The insurer commits (C_5) to the call center to paying if the call center creates commitment C_7, and the call center commits (C_6) to the insurer to creating C_7 if the insurer pays the call center. In this case, C_7 is the outsourced commitment. The call center commits (C_7) to the policy holder to receiving the claim if the policy holder reports an accident.

AGFIL (insurer) and Lee Consulting Services (assessor) negotiate the assessment fees. The *commercial transaction pattern* models this interaction. The insurer and the assessor create commitments C_8 and C_9. The insurer commits (C_8) to the assessor to paying the assessment fees if the assessor assesses the claim and brings about agreement to repair. Conversely, the assessor commits (C_9) to the insurer to assess the claim if the insurer pays the assessor.

The assessor and the inspector negotiate the vehicle inspection fees. The *commercial transaction pattern* models this interaction. The assessor and the inspector create reciprocal commitments C_{10}, and C_{11}.

The insurer outsources the vehicle repair to a repairer. The *outsourcing pattern* models this interaction. The insurer, and the repairer create the commitments C_{12}, C_{13}, and C_{14}. The insurer commits (C_{12}) commits to the repairer to paying if the repairer creates the commitment C_{14}. Conversely, the repairer commits (C_{13}) to the insurer to create commitment C_{14} if the insurer pays the repairer. Here, C_{14} is the outsourced commitment. The repairer commits (C_{14}) to the policy holder to repairing the vehicle if the policy holder requests for the repair service.



C_1 C(POLICY HOLDER, INSURER, create(C_3) $\wedge$ create(C_4), payInsurer)
C_2 C(INSURER, POLICY HOLDER, payInsurer, create(C_3) $\wedge$ create(C_4))
C_3 C(INSURER, POLICY HOLDER, reportAccident, receiveClaim)
C_4 C(INSURER, POLICY HOLDER, requestService, repairVehicle)
C_5 C(CALL CENTER, INSURER, payCallCenter, create(C_7))
C_6 C(CALL CENTER, INSURER, create(C_7), payCallCenter)
C_7 C(CALL CENTER, POLICY HOLDER, reportAccident, receiveClaim)
C_8 C(INSURER, ASSESSOR, agreeToRepair, payAssessor)
C_9 C(ASSESSOR, INSURER, payAssessor, agreeToRepair)
C_{10} C(ASSESSOR, INSPECTOR, inspectVehicle, payInspector)
C_{11} C(INSPECTOR, ASSESSOR, payInspector, inspectVehicle)
C_{12} C(INSURER, REPAIRER, create(C_{14}), payRepairer)
C_{13} C(REPAIRER, INSURER, payRepairer, create(C_{14})
C_{14} C(REPAIRER, POLICY HOLDER, requestService, repairVehicle)

Fig. 24. AGFIL expressed as a business model.