Noname manuscript No.
(will be inserted by the editor)

Muon: Designing Multiagent Communication Protocols from Interaction Scenarios

Anup K. Kalia · Munindar P. Singh

Received: date / Accepted: date

Abstract Designing a suitable communication protocol is a key challenge in engineering a multiagent system. This paper proposes Muon, an approach that begins from representative samples of interactions or *scenarios*. Muon identifies key semantic structures and patterns based on (social) commitments to formally analyze the scenarios and offers a methodology for designing protocols that would meet stakeholder needs. Interestingly, Muon applies its formal representations to suggest ways to identify additional scenarios needed to address exceptions arising in the interactions.

This paper contributes (1) a conceptual model of message types and causal relationships among them as a foundation for developing commitment-based communication protocols; (2) a robust, reusable characterization of semantic structures reflecting the above model; (3) a mapping from an annotated scenario to causally related interactions; and (4) a methodology to synthesize specifications of communication protocols. This paper reports on an empirical evaluation involving developers creating protocols from two real-life cases.

Keywords commitments · protocols · interactions · Dooley graphs

1 Introduction

A time-honored approach to building a multiagent system is, first, to develop a communication protocol that identifies one or more roles and how they may exchange messages and, second, to develop agents who would play one or more roles in one or more protocols.

Currently dominant software engineering approaches emphasize procedural representations. Specifically, they focus on message occurrence and ordering, but

Anup K. Kalia
North Carolina State University, Raleigh, NC 27695-8206, USA
E-mail: akkalia@ncsu.edu

Munindar P. Singh
North Carolina State University, Raleigh, NC 27695-8206, USA
E-mail: singh@ncsu.edu

neglect the meanings of the messages. The patterns they identify are likewise formulated at a low level, e.g., a request followed by a response. Notable existing approaches are the following. RosettaNet (2009) originated in the 1990s and is a widely deployed standard that describes interactions for realizing supply chains. RosettaNet underlies well-known business process types such as *Quote to Cash* (Oracle, 2009), which is extensively used in practice. TWIST (2008) is an industry group formalizing foreign exchange processes, which are also naturally viewed as protocols. Health Level Seven (HL7) (ISO/HL7, 2009) is the leading health care information technology standard, parts of which deal with messages, such as for ordering laboratory tests for a patient. Cross-organizational settings, where the above approaches apply, are architecturally well-suited to protocols. These approaches represent protocols semiformally in sequence diagrams, standardized as UML 2.0 Sequence Diagrams Object Management Group (2004).

However, traditional approaches fail in one major respect: they support only low-level correctness criteria. As a result, they do not readily accommodate flexible enactment, and thereby interfere with participants' autonomy. In contrast, meaning-based approaches for protocols promote flexibility through a high-level correctness criterion: a deviation to an enactment is correct if it preserves meaning. For example, a customer may wish to pay and a merchant may wish to ship goods earlier than absolutely necessary to take advantage of tax laws regarding business expenses. A traditional approach would not allow such deviation unless it was already encoded as a possible flow.

The multiagent systems community has developed a family of approaches under the rubric of *commitment protocols* (Yolum and Singh, 2002a), which tackle the challenge of specifying communication protocols purely or largely in terms of the meanings of the messages. Commitments provide a declarative basis for judging correctness, i.e., verifying compliance, thereby enabling participants to interact flexibly based on their internal policies (Singh, 1999; Fornara and Colombetti, 2003; Singh, 2008). In addition, meaning provides a systematic basis for expanding a protocol. For example, a simple purchase protocol may be expanded to accommodate delivery of goods and payments via third parties, a shipper and a bank, respectively. Practitioners engaged in advanced development in industry recognize the potential value of meaning-based approaches, e.g., (Desai et al., 2007a), but require clearer methodologies to adopt such approaches at a larger scale.

Despite the benefits of commitment protocols, we must overcome two challenges to bring the benefits of commitment protocols to practical settings:

- How can we design protocols with some assurance of addressing stakeholder needs?
- How can we map a commitment protocol to an operational representation that could be absorbed into traditional software engineering processes?

1.1 Contributions

Our proposed approach, Muon, addresses these challenges by incorporating (1) a commitment-based *model* for protocols that captures the meanings of messages along with any causal relationships among them, (2) semantic *structures* and pragmatic patterns based on commitments, and (3) a *methodology* for specifying protocols that address stakeholder needs. For clarity, we adopt the UML 2.0 sequence diagram notation as the representation for protocols, but Muon would equally produce any other operational notation.

1.2 Running Example

We adopt a well-known car insurance claim handling case study (Browne and Kellett, 1999). Here, an insurance company (AGFIL itself, as the insurer) provides automobile insurance to an insured party, which means that the insurer will pay for any damage to the automobile if the insured files a claim, the damage does not total the automobile, and the repair is assessed as being acceptable by the insurer or its appropriate business partners. Typically, the insurer contracts out key functionalities to independent agencies. For example, it would outsource the handling of claims filed by the insured party and the verification of whether the reported damage is covered and whether the resulting repair costs are legitimate.

The following case fleshes out the elements of the case we consider here. AGFIL commits John Doe (JD) to provide insurance coverage for his car provided JD pays the premium. Similarly, JD commits to paying the premium provided AGFIL insures his car. When JD's car is damaged in an accident, JD files an insurance claim with the call center, Europ Assist (EA), which helps JD in filing his claim. EA helps identify a mechanic M; asks JD to take the car to M; and informs AGFIL of the claim being filed and the mechanic being assigned. AGFIL turns over the claim handling to Lee Consulting Services (LCS). M estimates his charges for repairing JD's car and informs LCS. If LCS accepts M's estimates, it authorizes M to repair the car. In the meanwhile, as a way to reduce fraud, LCS may inspect the car at M's premises. Once M repairs the car, he submits an invoice to LCS, which forwards it to AGFIL. If AGFIL agrees, it pays M for the car repair. M asks JD to come retrieve his car.

1.3 Organization

Section 2 presents our model and semantic structures based on commitments. Section 3 presents important pragmatic patterns built on our communication model. Section 4 describes our methodology for inducing protocols from scenarios. Section 5 describes an empirical evaluation of our approach using two cases adapted from real-life studies: one on automobile insurance and one on breast cancer diagnosis. Section 6 concludes with a discussion of our main claims, the relevant literature, and limitations and future directions.

2 Model and Semantic Structures

Muon's model of protocols begins from the idea that each message in a protocol can be understood in terms of the high-level or social actions it brings about. An important feature of Muon is that it assigns a declarative meaning to the messages in terms of their effects on the social state of an interaction, especially as the state is expressed via the commitments among the stakeholders. Section 2.1 provides the necessary background; the rest of this paper is new.

2.1 Background: Commitments and Operations on Them

An *agent* is an active computational entity that reflects the autonomy of the party it represents. A protocol involves one or more *roles*, which an agent would adopt to participate in an enactment of a protocol. Agents (and roles) capture the interacting parties enter into and manipulate publicly observable *commitments* to one another (Singh, 1999). We understand a commitment as a conditional relationship directed from a debtor to a creditor (agents or roles, at run time and design time, respectively). A commitment $c = \mathsf{C}(\mathit{debtor}, \mathit{creditor}, \mathit{antecedent}, \mathit{consequent})$ means that the debtor commits to the creditor to bring about the consequent provided the antecedent holds. In effect, if the commitment is *detached* (i.e., its antecedent holds), the debtor becomes unconditionally committed to ensuring the consequent. For example, a *quote* message from a mechanic to a customer creates a commitment from the mechanic (debtor) to the customer (creditor) that if the customer agrees to pay the quoted amount (antecedent), the mechanic will make the specified repairs (consequent). Our approach is minimalistic in that we allow important patterns of applying commitments, including reciprocity, but do not make them a mandatory part of the semantics. Singh (2012) provides additional conceptual motivation for commitments.

Typically, but not necessarily, the creditor would bring about the antecedent and the debtor would bring about the consequent. Quite often, the antecedent would be an environmental property describing when the given commitment would be detached. The following operations on commitments are from the literature (Singh, 1999). Muon introduces partial detach and discharge.

1. *Create*(c) creates the commitment c. An example is the *quote* message described above. The create operation can be performed only by c's debtor, which is essential to preserving the debtor's autonomy. If one party were to make another party committed, it would infringe upon the latter's autonomy.
2. *Discharge*(c) c's debtor fulfills the commitment by bringing about the *consequent*. This operation is trivially performed when the consequent holds through environmental events or the actions of others.
3. *Cancel*(c) cancels the commitment c. This operation can only be performed by c's debtor. For convenience, we treat any violation of a commitment as a cancellation. For example, if a commitment from a merchant to a customer is detached and yet the merchant fails to deliver the goods on time, the timeout as counts as the merchant canceling the commitment. As we explain in connection with pragmatic patterns below, the cancellation of a commitment would often, but not always, be compensated for in business settings.
4. *Release*(c) releases c's debtor from commitment c. This operation can only be performed by the creditor. The mechanic, for instance, can release the customer of its commitment to pay. A more typical case concerns a commitment that makes an offer. When the creditor rejects the offer, that corresponds to a release of the concomitant commitment.
5. *Assign*(c, z) replaces z as c's creditor. This operation can only be performed by the creditor. For instance, the mechanic can assign the customer's commitment to another party if the mechanic does not wish to carry out the repairs.

6. *Delegate*(c, z) replaces z as c's debtor. The insurance company can delegate its commitment to pay the mechanic to a bank. Such a delegation can be done before or after the mechanic performs the ordered repairs.

We do not contribute a formal semantics for commitments. We adopt the semantics proposed by Singh (2008). This semantics justifies our notions of (partial or full) detach and (partial or full) discharge, and supports partial assignment and delegation. Specifically, a commitment whose consequent is $p \wedge q$ corresponds to two commitments of the same debtor and creditor, one for p and one for q. The debtor may delegate each of these commitments separately and the creditor may assign them separately. When p and q both come to hold, these two commitments as well as the original $p \wedge q$ commitment are discharged.

2.2 Message Meanings and Relationships

We understand each message as having an explicit meaning. In particular, messages in domains of interest, such as business, often carry meanings that correspond to the above commitment operations. Within the scope of a protocol, the meanings of the messages are fixed. Thus, a mechanic who sends a *quote* message to a customer creates the associated commitment (if that is how *quote* is defined). In other words, the agent's autonomy extends its decisions about sending the messages, not to the meanings of the messages that it sends. Fixing such meanings across agents is essential for collaboration, and one of the key motivations behind specifying a protocol in the first place. Below $m(s, r)$ is a message from s to r; for brevity, we omit the sender and receiver where they are understood. Here, x, y, z, and a are the participants in a protocol, and $c = \mathsf{C}(x, y, p, q)$ is a commitment.

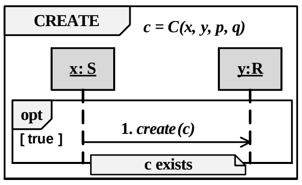
We identify the following semantic relationships among messages. These relationships describe the fundamental *semantic structures* since the messages are interpreted as operations on commitments and the relationships are based on the semantics of commitments. The semantic structures capture the elements of a protocol that reflect the essential logical content of a commitment. For example, let $c = \mathsf{C}(x, y, p, q)$. Now if q occurs, c is discharged: there is no choice about the discharge given our commitment semantics. A protocol specifies the meanings of its messages; an agent exercises its discretion to decide which messages to send.

Figure 1a shows the elementary semantic structure of a commitment being created by an explicit *create*$(\cdot)$ message from its debtor. (We show the postcondition of each structure at the bottom of its diagram.) A commitment may be created by the delegation or assignment of another commitment—we handle these cases separately since they presuppose an already existing commitment.

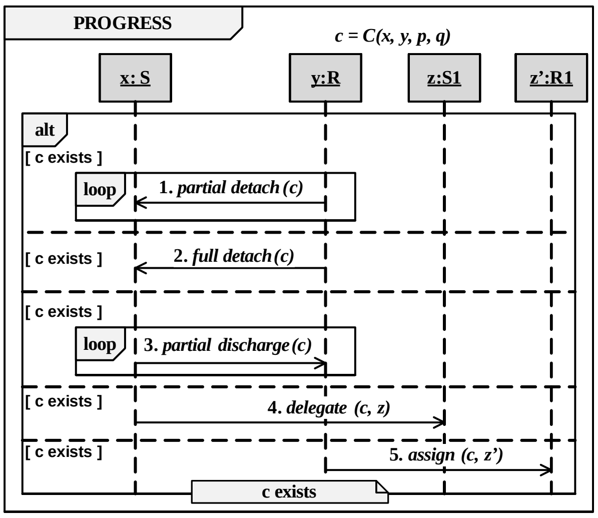
2.2.1 Progress a Commitment

Message m_j *progresses* Message m_i if and only if m_j logically affects any commitments referenced by m_i but does not eliminate such commitments. Figure 1b illustrates this structure.

A commitment is (fully) *detached* when its antecedent becomes true. To accommodate practical cases, we introduce a *partial detach* as when the antecedent of a commitment is weakened but does not become true. Likewise, a *partial discharge*



(a) Create.



(b) Progress.

Fig. 1: Semantic structures: create and progress. (In all figures in this paper, S, S', and S1 are senders, and R, R', and R1 are receivers.)

weakens the consequent of a commitment but does not make it true. Partial detach and discharge are respectively ways to bring about the antecedent and consequent in a "piecemeal" manner. We can think of a series of messages, each accomplishing a fraction of the total work, such as a shipment being completed in parts. In essence, each such message modifies a commitment based on suitable logical operations (Singh, 2008), and is related to the initial creation via the Progress relationship.

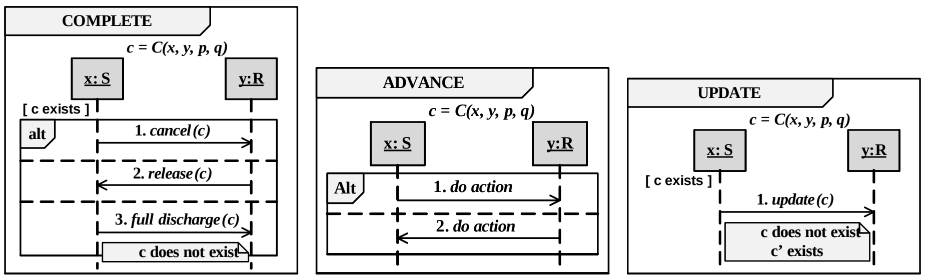
If Message m_j partially discharges or (partially or fully) detaches a commitment created by m_i, then m_j progresses m_i. Likewise, m_j progresses m_i if m_j delegates or assigns a commitment created by m_i.

2.2.2 Complete a Commitment

Message m_j *completes* Message m_i if and only if m_j eliminates a commitment created or progressed by m_i, as shown in Figure 2a.

A commitment can simply be discharged by its debtor by performing the consequent. One can think of discharge as a message that finishes the job. We make the simplifying assumption that even when the commitment's consequent becomes true due to an event in the environment, the debtor sends a message claiming "credit" for it, which makes for safe protocols wherein the debtor observes the relevant event and informs the creditor.

Typically, but not necessarily, the debtor would discharge a commitment after the creditor brings about the antecedent. Note that although a full or partial detach and a partial discharge merely progress a commitment, a full discharge completes it.



(a) Complete. (b) Advance. (c) Update.

Fig. 2: Semantic structures: complete, advance, and update.

Additionally, a commitment may be completed by its debtor canceling it or its creditor releasing it. Thus if m_j (fully) discharges, cancels, or releases a commitment created or progressed by m_i, then m_j completes m_i. Any commitment created or progressed by m_i is no longer active after m_j occurs.

2.2.3 Advance an Interaction

Message m_j *advances* Message m_i if and only if m_i is causally prior to m_j but m_j does not affect any commitment created by m_i. That is, m_j does not rely upon the commitment semantics. For example, a price quote sent in response to a request for quotes (RFQ) advances the latter. In particular, we can understand the price quote as creating a commitment, which advances the prior RFQ. In general, if a commitment creation relates to a prior message, it can only advance it. Notice that the advancing message, m_j, could be sent by the sender or receiver of the advanced message, m_i. Figure 2b illustrates this structure.

2.2.4 Update: Complete plus Advance

Message m_j *updates* Message m_i if and only if m_i is causally prior to m_j and m_j affects a commitment created by m_i *extralogically*, that is, an update is not based on the formal semantics of commitments. Figure 2c illustrates this structure. The debtor of a commitment can update its antecedent or consequent. Because a commitment results from an update, for reasons of autonomy, only the debtor can perform an update. In essence, an update is equivalent to canceling the existing commitment and creating a new one. In terms of Figure 3, an update would affect two commitments, one being completed and one being created. A creditor may request an update, e.g., during negotiation. Such an update is equivalent to releasing the existing commitment and requesting a new one from the debtor. For instance, a mechanic may update its estimate and an insurance company may change its premium quote. Updating a commitment is distinct from causing progress of that commitment.

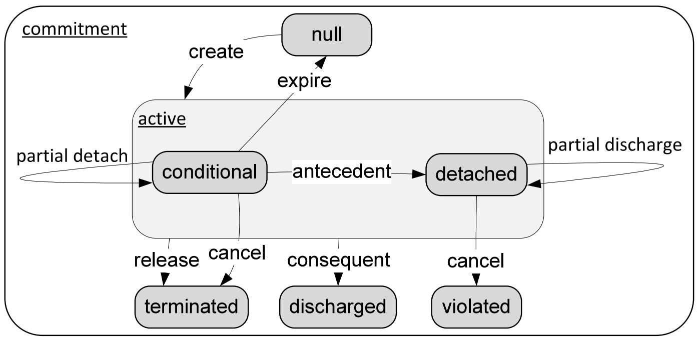


Fig. 3: The life cycle of a commitment, enhanced from Singh et al. (2009). Section 2.2.1 explains partial detach and partial discharge.

2.3 Grouping and Adequacy of the Semantic Structures

Figure 3 shows the life cycle of a commitment as a state diagram, enhanced as needed for Muon. The debtor brings a commitment from the *null* state to the *conditional* state by creating it. Once the antecedent comes to hold, the commitment transitions to the *detached* state; otherwise, if the antecedent times out, the commitment transitions to the *expired* state. From the *active* state (a superstate of conditional and detached), if the consequent comes to hold, the commitment is *discharged*. From the detached state, if the consequent fails to become true, the commitment is *violated*. The commitment transitions to the *terminated* state when it is either canceled by the debtor or released by the creditor.

Our semantic relationships help group messages. A message that progresses another must (partially or fully) detach, partially discharge, delegate, or assign it. A message that completes another must fully discharge, cancel, release, or update it. A message that advances another could have any content.

The semantic structures capture, based on logic alone, how a commitment goes from being created to terminated. Because the semantic structures respect the logical properties of commitments, we can see that they are adequate in the following sense. The semantic structures act upon the two logical components of a commitment—its antecedent and consequent—in all possible ways, namely, via the partial and complete discharge and detach operations. Doing so covers all paths in the commitment life cycle that involve a modification of the antecedent and consequent. The release and cancel operations provide additional transitions. Thus every possible path is covered. In addition, the update structure cancels a commitment and creates another, and the advance structure has no logical effect on a commitment. Update and advance do not affect the above adequacy.

2.4 Commitment-Causal Graphs

We propose *commitment-causal* or *CC* graphs to capture the essential semantic and causal properties of an interaction scenario. CC graphs build on the notion of a *Dooley graph* (Parunak, 1996). Parunak's approach, although a crucial precursor to Muon, suffers from significant limitations for our present purposes. Specifically, it considers a fixed set of message types and does not accommodate their meanings. Moreover, it organizes interactions in terms of fragments that involve exactly two parties. We exploit Singh's (2000) modular approach for constructing Dooley graphs but we further introduce meanings (expressed via commitments) and relationships based on meanings that are missing from Singh's approach.

The following terminology is important for understanding Muon. A *(interaction) scenario* is a sequence of messages corresponding to a specific instance or enactment of a protocol. Parunak (1996) uses the term "discourse" for these scenarios, but we adopt the terminology from requirements engineering (Filippidou, 1998) to simplify the connection with software engineering and to avoid confusion with natural language discourse. Scenarios in Muon pertain only to communications among autonomous roles. A *role* is a participant in a scenario. A *conversation* is a fragment of a scenario that reflects a causally self-contained logical unit. A *character* is a part played by one role in a conversation.

3 Pragmatic Patterns

The pragmatic patterns represent heuristic ways in which designers in an application domain may construct suitable protocols. For example, the designers may decide that the cancellation (1) of a typical commitment is addressed by the creditor *escalating* a complaint to a specified authority, (2) of a delegated commitment by *protesting* to the original debtor, and (3) of an assigned commitment by *informing* the original creditor. Here we identify three protocol patterns that the designers could apply. In general, there is a huge variety of such patterns and nowhere should one assume that a particular pattern must be applied. Such a variety is reminiscent of situational method engineering (SME) (Chopra and Singh, 2011; Qumer and Henderson-Sellers, 2008), wherein a software design method comprises reusable fragments selected based on application and organizational requirements. In the present setting, a selected set of pragmatic patterns can be thought of as a situation-specific set of methods for designing a protocol. For example, we may (or may not) capture that a creditor would escalate a canceled commitment.

Previous research has examined patterns for business processes. Van der Aalst et al. (2003) consider the operational structure of workflows, e.g., that two subworkflows may run in parallel or that one may invoke the other, which apply close to the implementation. In contrast, Muon supports patterns expressed via commitments: not which workflow invokes another but which business partner delegates a commitment to another or escalates a commitment violation to another.

Singh et al. (2009) propose patterns that highlight business relationships among the participants. Their patterns apply from the standpoint of maintaining the social state of two or more business partners. Specifically, they state nested commitments expressions and how the state of interrelated commitments progresses, e.g., in delegation, but not how the interaction may be operationalized via messages. In contrast, Muon patterns describe how communications among the participants accomplish the corresponding business-level interaction.

We now introduce some important pragmatic patterns. For each pattern, we provide six key attributes, loosely based on Gamma et al. (1995). *Intent* describes the purpose for adopting the pattern; *motivation* a potential use case; *applicability* the situations where the pattern is beneficial; *consequences* the implications of and assumptions necessary as backdrops for the pattern; *implementation* how the pattern contributes to the solution being generated; and *known uses* some real-life cases demonstrating this pattern.

3.1 Accommodating Cancellation via Escalation

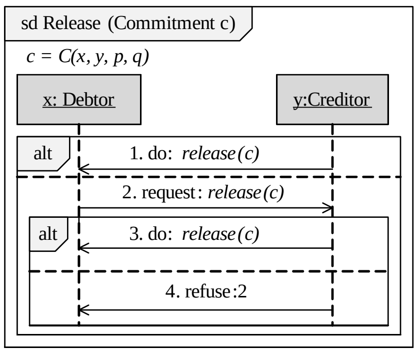
As an autonomous party, a debtor may cancel a commitment. (Recall that a timeout of bringing about the consequent for a detached commitment counts as a cancel.) However, since such an exit is usually not desirable, a protocol may include support for a jilted creditor to send an *escalation* message to an appropriate authority, which might (or might not) proceed to sanction the errant debtor.

– *Intent*: The protocol should handle a debtor dropping a commitment.
– *Motivation*: An insurance company does not pay an insured car owner for repairs to the insured car.
– *Applicability*: Supports dealing with undesirable conduct by a debtor.
– *Consequences*: Presumes a higher authority to whom the creditor can complain about the cancellation.
– *Implementation*: Creditor sends a complaint to a higher authority.
– *Known uses*: Informally, a car owner may complain to the State Insurance Regulatory Commission.

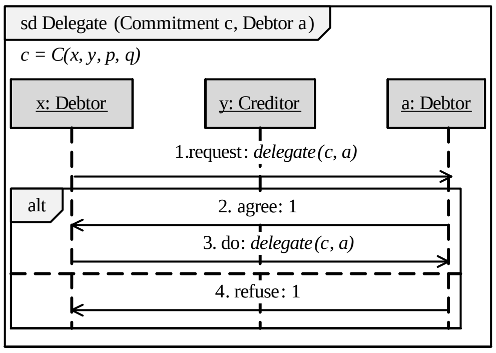
3.2 Accommodating Release

As in Figure 4a, the creditor may unilaterally release a commitment. Alternatively, the debtor can request a release from the creditor, in which case the creditor may release the debtor, or refuse the debtor's request.

– *Intent*: The protocol should allow a creditor to relieve a debtor from a commitment.
– *Motivation*: A car owner rejects an offer from a mechanic for repairs for a specified price: the mechanic need no longer honor that offer. Or, the mechanic cannot repair the car on time and requests an extension.
– *Applicability*: Supports incorporating negotiation (where parties may reject each other's offers) and enacting parties who seek a graceful exit when one of them cannot (conveniently) discharge a commitment.
– *Consequences*: The parties involved are cooperative: they explicitly reject an offer or consider a request for release.
– *Implementation*: The creditor notifies the debtor with or without the debtor's prior request.
– *Known uses*: RosettaNet's (2009) Request Shipping Order Cancellation (PIP 3B14), Request Purchase Order Cancellation (PIP 3A23).



(a) Release.



(b) Delegate.

Fig. 4: Pragmatic patterns: release and delegate.

3.3 Accommodating Delegation

In principle, the debtor of a commitment, the *delegator*, can delegate the commitment to another party, the *delegatee*. Since such a unilateral delegation may fail, it is helpful to consider the pattern shown in Figure 4b. Here, the delegator requests the delegatee and only upon its agreement performs the delegation.

- *Intent*: The protocol should enable flexible allocation of tasks while leaving the debtor responsible.
- *Motivation*: AGFIL delegates its commitment to repair a damaged car to a mechanic. If the mechanic repairs it, AGFIL's commitment is discharged; if the mechanic fails to repair it, AGFIL is still on the hook for the repairs.
- *Applicability*: Supports a debtor getting some work done even when it is unable or unwilling to perform its committed task.
- *Consequences*: Suitable social relationships are in place between the delegator and the delegatee so that the request is a reasonable one. However, if the delegator possesses the power to force the delegatee, the pattern can be simplified by directly performing the delegation.
- *Implementation*: The delegator sends a request message to the delegatee. If the delegatee agrees, the original commitment becomes pending but is not discarded. If the delegatee fails to discharge the commitment, the original commitment is revived.
- *Known uses*: The MIT Process Handbook (MITPH) (Malone et al., 2003) includes business processes that demonstrate the delegate pattern, e.g., Deliver Stock Product (SCOR D1), where a seller outsources its shipping duty to a shipper or a carrier. Similarly, (2009) Request Shipping Order (PIP 3B12) and Notify of Shipping Order Confirmation (PIP 3B13) reflect the delegate pattern.

A useful variation of this pattern is to have the delegator inform the creditor: doing so ensures the creditor knows who its current debtor is, which can facilitate compliance checking by the creditor.

This pattern handles the case where the delegator retains responsibility for the satisfaction of the commitment. An important, and simpler, alternative is where the delegator does not retain responsibility. Both variants are discussed in Singh et al. (2009).

Yet another pattern would be one where the delegator requests permission from the creditor before proceeding with the delegation. Specifically, if the delegator is additionally committed to the creditor to not delegate the main commitment, the delegator may request the creditor to release it from the additional commitment. We can accommodate the effect of the permission pattern via a composition of the release and delegation patterns.

3.4 Accommodating Assignment

An assignment is analogous to a delegation. It involves an *assigner* and an *assignee* in addition to the debtor. The main difference from a delegation is that the creditor initiates it by sending a message to the debtor. The creditor can perform an assignment unilaterally unless some other commitment (whose debtor is the assigner and whose creditor is the current debtor) prevents it, in which case the assigner can request a release from it. As above, the assigner can additionally inform the assignee so the assignee is aware that it is the creditor of some commitment, which can help it verify the debtor's compliance.

4 The Muon Methodology

Muon greatly expands on the well-known Class, Responsibility, and Collaboration (CRC) card methodology for object-oriented analysis (Beck and Cunningham, 1989), especially as introduced to multiagent systems by Parunak (1996). Like the CRC methodology, Muon relies upon stakeholders playing roles to extract the requirements for a technical solution.

The Muon methodology guides the designers of a protocol. Muon presupposes that experts answer the questions *Who* the stakeholders are (Step M1); *What* messages they exchange (Steps M2 and M3); and *Why* they exchange such messages, indicating the effects of those messages on the social state (Step M4). Muon provides guidance in validating the scenarios to handle deviant enactments (Step M5). Muon includes an automated method to generate a modularized protocol based on the model and semantic structures introduced above.

4.1 M1: Identify Stakeholders as Roles

The first step in our methodology is the identification of all the independent stakeholders involved in the interaction.

- *Input:* A business problem and domain expertise.
- *Output:* A set of autonomous entities whose interaction is being modeled.
- *Description:* Domain experts identify the independent stakeholders participating in the interaction. The key to identifying stakeholders is that they are autonomous and thus can enter into and modify social relationships with others. Specifically, they can create and otherwise manipulate their commitments.

The stakeholders are generalized into the *roles* they will play in the final protocol. The roles are groupings of stakeholders who engage in similar interactions. For example, if two stakeholders both handle insurance claims, we might combine them into the single role of call center.

We identify the following roles in the insurance case.

- Insurer. Insurance company or provider.
- Insured. Automobile owner covered through an insurance policy.
- Call Center. Customer-facing party that handles insurance claim requests.
- Mechanic. Estimator of repair costs and performer of any repairs.
- Claims Handler. Independent estimator of claimed damages to vehicle and payer of authorized repair charges to mechanics.

4.2 M2: Record One or More Typical Happy Path Scenarios

Once the roles are identified, the protocol designers specify a *typical* successful or *happy path* scenario as an instance of the interaction enacted by domain experts—in the spirit of role playing, as in the CRC methodology (Beck and Cunningham, 1989). Muon is neutral as to whether the experts belong to one organization or many, but it presumes that they have good knowledge of the stakeholders so that they can create realistic scenarios.

- *Input:* The output of Step M1, the business problem, and domain expertise.
- *Output:* Illustrative scenarios of the interaction being modeled in terms of messages: one primary and zero or more secondary.
- *Description:* The experts enact the roles identified in these scenarios.

The following are some guidelines for selecting scenarios.

- *Represent the core positive outcomes.* A scenario wherein John Doe's car is repaired and all payments are made captures desirable enactments and provides a basis for deviations. A trivial scenario, e.g., in which the mechanic refuses to repair the car, is not as useful.
- *Reflect social or organizational relationships.* These relationships specify the commitments among the roles. For example, it helps to capture that Europ Assist and Lee Consulting Services work for AGFIL.
- *Omit irrelevant messages.* In particular, omit messages pertaining to transactions outside of the current scenario, e.g., customers unrelated to AGFIL.
- *Omit irrelevant roles and role instances.* For example, omit a bank role since payments are not part of the scenario. Restrict instances of a role to those that interact in distinct ways. For example, we consider two mechanics only because they interact differently with others in the scenario.

Table 1 shows a typical scenario, as might be recorded from an enactment session (Parunak, 1996).

4.3 M3: Record One or More Scenarios Demonstrating Exceptions

The domain experts identify one or more scenarios corresponding to exceptions.

Table 1: A successful scenario for the AGFIL example. ID, S, and R are respectively the sequence number, sender, and receiver of a message.

ID	**S**	**R**	**Content**
1	AG	JD	If you pay premium p, I will insure your car
2	JD	AG	If you insure my car, I will pay premium p
3	AG	JD	I will respond to and resolve claims
4	AG	JD	EA will handle claims
5	AG	EA	If you will handle claims, I will pay
6	EA	AG	OK, I will handle claims
7	JD	AG	OK, I accept EA as claims handle
8	AG	EA	JD has agreed
9	AG	JD	EA has agreed
10	JD	EA	Insurance claim
11	EA	JD	Take the car to M
12	EA	AG	JD's car is with M
13	AG	LCS	If you resolve the claim (verify estimate and handle car repair), I will pay
14	LCS	M	If you provide the estimate and I accept it, I will pay
15	M	LCS	Here is the estimate for the repairs
16	LCS	AG	I request you to pay M2
17	AG	LCS	I agree to pay
18	LCS	AG	OK, I have verified the estimate
19	LCS	M2	If you repair the car, I will pay
20	M2	LCS	OK
21	M2	LCS	Car is repaired
22	LCS	AG	Car is repaired
23	AG	JD	Your premium has increased to p'
24	JD	AG	Here is (the updated premium) p'
25	AG	JD	Take back the car from M2
26	AG	EA	Here is payment for handling claims
27	AG	LCS	Here is payment for resolving the claims

– *Input:* The output of Step M2, business problem, and domain expertise.
– *Output:* Scenarios exhibiting exceptions in the interaction being modeled.
– *Description:* The experts identify possible exceptions that can occur in a previously determined typical scenario.

Table 2: A scenario demonstrating an exception in the AGFIL example.

ID	**S**	**R**	**Content**
15a	LCS	M	Reject the estimate from M
15b	LCS	AG	Suggest another mechanic
15c	AG	EA	Suggest another mechanic to LCS
15d	EA	LCS	contact M2
15e	EA	JD	Take the car to M2
15f	LCS	M2	If you provide the estimate and I accept it, I will pay
15g	M2	LCS	Here is the estimate for the repairs
15h	LCS	M2	I accept it

Table 2's columns show an exceptional scenario. When M provides an estimate for repairing JD's car to LCS, LCS may reject the estimate. LCS may find another mechanic, M2. The rest of the scenario would proceed just as when LCS succeeds with M. Messages 15a to 15h represent a scenario to be merged with the scenario provided in Table 1.

4.4 M4: Annotate Messages in Scenarios

– *Input:* The outputs of Steps M2 and M3.
– *Output:* Annotations for each of the messages produced by Steps M2 and M3.
– *Description:* Annotate each message in the scenario with the commitment operations it involves, if any, and its relationship with other messages.

For example, Message 1 is an offer from AG to JD to pay a premium. Message 15 progresses Message 14 by detaching the commitment. Message 3 advances Message 1 where AG commits to handling and resolving claims for JD. Message 25 completes Messages 24 and 3, where AG provides the repaired car when JD claims it.

4.5 M5: Verify Robustness of Scenarios

– *Input:* The annotations of Step M4 and the scenarios of Steps M2 and M3.
– *Output:* Annotations on commitments that fail robustness checks.
– *Description:* Check if the scenarios produced in Steps M2 and M3 are complete given the annotations of Step M4. If not, iterate Steps M2, M3, and M4.

Muon lends itself to determining and addressing possible exceptions and opportunities based on the commitments of the parties involved. Specifically, for any scenario that involves the creation of a commitment we can ask what would happen if (1) the debtor canceled or updated it; (2) the debtor attempted to delegate to another party; or (3) the creditor attempted to release it or assign it to another party. The above questions serve as sanity checks, and arise naturally from the commitment life cycle. Notice they do not include advance messages because those are not constrained by the model.

Prompted by these questions, designers would iterate over Steps M2 and M3 by recording multiple exceptional scenarios. In doing so, the designers would demonstrate how to handle exceptions and our approach would incorporate such exception handling into the resulting protocol. Note that Muon does not demand whether and how designers address exceptions: such decisions presupposing application-specific knowledge and some imagination.

In Table 3, JD never pays the insurance premium, LCS never pays M for the estimates and repairs, and AG never pays for the estimate it obtains from M2. This prompts the designers to introduce Message 2a, Message 22a, and Message 18a to complete the respective commitments.

In this step, we identify the incompleteness and iterate Steps M2 and M3. As Table 4 shows, Step M2 now yields Message 2a, which completes the commitment to pay the premium by discharging it, and Message 22a, which completes the

Table 3: An initial scenario for the AGFIL case that combines the happy path and exception scenarios. The annotations relate pairs of messages. (Here, A: advance; P: progress; U: update; C: complete; crt: create; det: detach; dis: discharge; cnl: cancel; rel: release; del: delegate; pdet: partial detach.)

ID	**S**	**R**	**Content**	**A**	**P**	**U**	**C**	**Operation**
1	AG	JD	If you pay premium p, I will insure your car					crt(C_1)
2	JD	AG	If you insure my car, I will pay premium p					crt(C_2)
3	AG	JD	I will handle and resolve claims	1				crt(C_3)
4	AG	JD	EA will handle claims	3				
5	AG	EA	If you will handle claims, I will pay	4	3			crt(C_4)
6	EA	AG	OK, I accept to handle claims	5				
7	JD	AG	OK, I accept EA as claims handler	4				
8	AG	EA	JD has agreed	5, 7				
9	AG	JD	EA has agreed	4, 7				
10	JD	EA	Insurance claim	4				
11	EA	JD	Take the car to M	10				
12	EA	AG	JD's car is with M		5			det(C_4)
13	AG	LCS	If you resolve the claim (verify estimate and handle car repair), I will pay		3			crt(C_5)
14	LCS	M	If you provide the estimate and I accept it, I will pay	13				crt(C_6)
15	M	LCS	Here is the estimate for the repairs		14			det(C_6)
15a	LCS	M	Reject the estimate from M	5			15	cnl(C_6)
15b	LCS	AG	Suggest another mechanic	14				
15c	AG	EA	Suggest another mechanic to LCS	15b		12		
15d	EA	LCS	contact M2	15c				
15e	EA	JD	Take the car to M2	15d				
15f	LCS	M2	If you provide the estimate and I accept it, I will pay	13				crt(C_7)
15g	M2	LCS	Here is the estimate for the repairs		15f			pdet(C_7)
15h	LCS	M2	I accept the estimate		15g			det(C_7)
16	LCS	AG	I request you to pay M2		15f			crt(C_8), det(C_8)
17	AG	LCS	I agree to pay	16				
18	LCS	AG	OK, I have verified the estimate		13			pdet(C_5)
19	LCS	M2	If you repair the car, I will pay	3				crt(C_9)
20	M2	LCS	OK	19				
21	M2	LCS	Car is repaired				19	det(C_9)
22	LCS	AG	Car is repaired	21	18			det(C_5)
23	AG	JD	Your premium has increased to p'			1		
24	JD	AG	Here is (the updated premium) p'			23		det(C_1)
25	AG	JD	Take back the car from M2				24, 3	dis(C_1), dis(C_3)
26	AG	EA	Here is the payment for handling the claims				12	dis(C_4)
27	AG	LCS	Here is the payment for resolving the claims				6	dis(C_5)

Table 4: Completeness check for the happy path and exceptions in Table 3.

ID	S	R	Content	A	P	U	C	Operation
2a	JD	AG	Pay premium p		1		2	det(C_1), dis(C_2)
22a	LCS	M2	Here is the payment for performing the repairs				21	dis(C_9)
18a	AG	M2	Here is the payment for the estimate				16, 15h	dis(C_8), dis(C_7)

detached commitment from AG to M. And, as Table 4 shows, Step M3 yields Message 18a, which completes the commitment to pay for the estimate by discharging it. Table 6 (in the appendix) provides the resulting scenario.

4.6 M6: Map the Scenarios to a Commitment-Causal Graph (Automatic)

- *Input:* The output of Step M4, once verified in Step M5.
- *Output:* A commitment-causal (CC) graph of the scenario, identifying semantic structures in the interaction.
- *Description:* Generate a CC graph via a conversation graph.

An annotated scenario maps to a directed graph whose vertices are messages and whose edges are relationships between messages such as P (progress), C (complete), A (advance), or U (update). For example, if m_j progresses m_i, there is an edge labeled P from m_i to m_j. Figure 5 shows the directed graph generated from the annotated scenario in Table 6.

A *conversation graph* G is a minimal graph component that contains at least one create message and contains all the messages that relate via progress or completion (hence also update, but not advance) to any message that occurs in G. That is, a conversation graph is a logically coherent scenario fragment beginning from a commitment creation. For example, Message 1 forms a conversation graph with Messages 2a, 23, 24, and 25.

Figure 6 shows a library of all of Muon's conversation templates. These templates are based on the semantic structures and pragmatic patterns discussed in Sections 2.2 and 3, respectively. Each specific template is quite simple and self-explanatory. Together, the templates capture the possible ways in which a commitment may progress, complete, or be updated. One can verify by inspection that these templates cover the life cycle of a commitment discussed in Section 2.3 and the pragmatic patterns introduced in Section 3. Notice that in our approach, delegate and assign are included in progress and function as implicit creates. Thus, a delegate or assign message can itself be progressed through a detach or partial discharge, and completed through a discharge.

An annotated scenario maps via these templates to multiple conversation graphs. Notice that the same create message would generally feature in more than one conversation graph, e.g., one for detach and one for discharge.

The vertices of each CC graph are characters. A role in a protocol corresponds to one or more characters, reflecting its distinct facets. We generate the CC graph

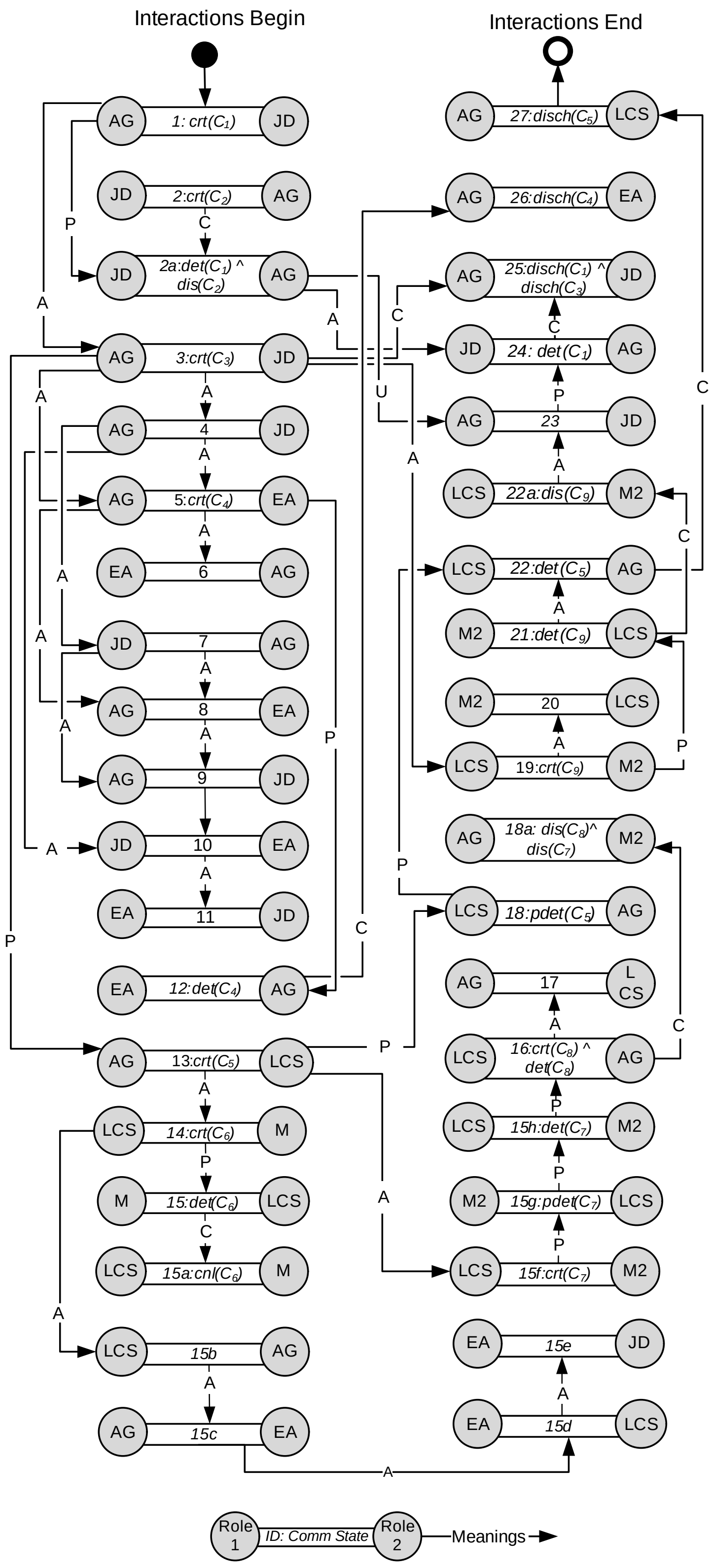


Fig. 5: A directed graph showing the semantic relationships in the AGFIL case.

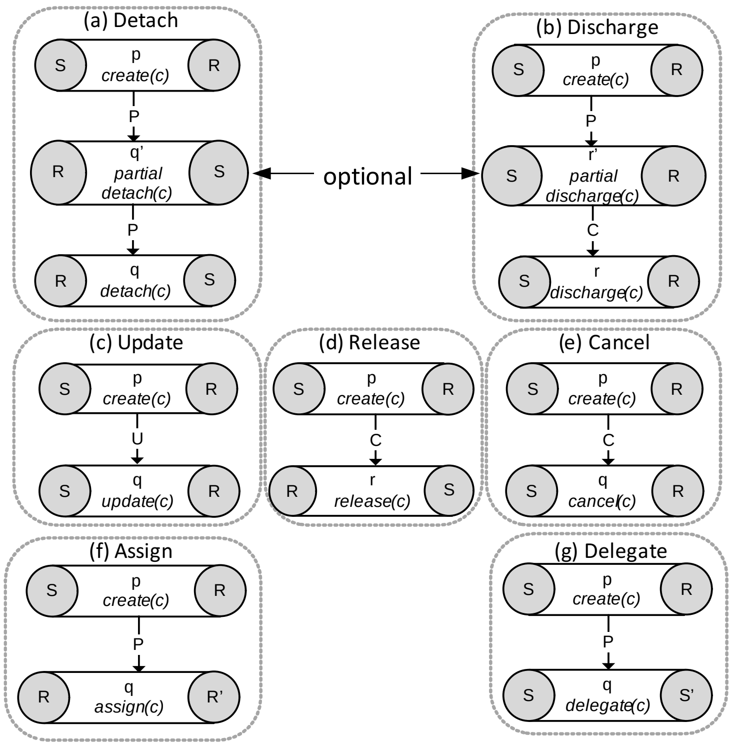


Fig. 6: Template library: each box shows a conversation fragment pattern based on the semantic structures and pragmatic patterns. (Partial detach and partial discharge are each optional in their respective templates.) (S-sender, R-receiver, S'-new sender, R'-new receiver)

of the scenario from its conversation graphs. Each conversation graph maps to a connected component in a CC graph. The senders and receivers of messages become the vertices in the CC graph and the messages become the edges. Additionally, the edges (messages) within each component are causally ordered. Figure 7 illustrates how we can create a CC graph by mapping a conversation graph to its corresponding CC component. Each CC component aggregates characters and messages. Here, S and R are the characters and p, q, and r are the messages exchanged between the characters. For example, discharging a commitment after it is delegated is a variant of Figure 7(a) wherein the commitment is discharged after being created and detached. Figure 8 shows the CC graph for the AGFIL case.

Mutually independent or nonoverlapping scenarios map to distinct conversation graphs and hence to separate CC components. An interaction scenario I_1 that is causally affected by a message in another scenario I_2 (the advance relationship) would yield a protocol that hangs off the protocol for I_2. Such causally connected protocols can spawn off new commitments or perform other actions. Each such commitment would be treated in its own logical right. For example, the cancellation of a commitment might lead to the creditor sending an escalation message,

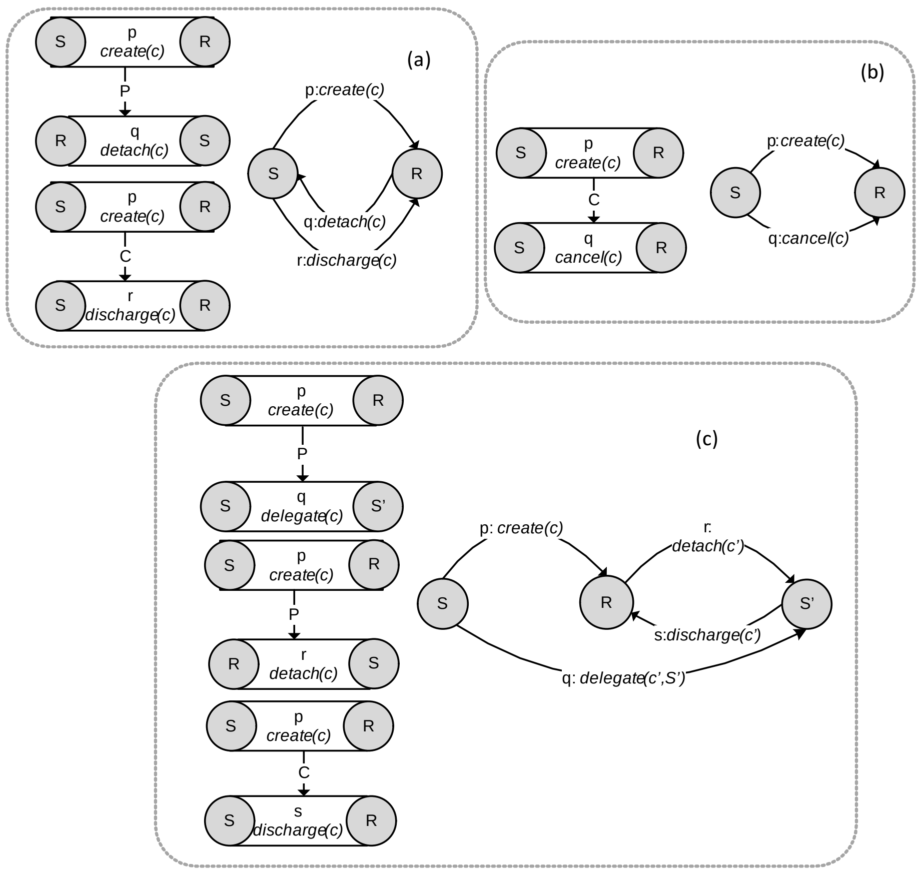


Fig. 7: Mapping possible conversation fragments to corresponding CC graph components. S' is a delegatee and R' is an assignee.

which might lead the recipient of the message to create another commitment to compensate the creditor.

This method is robust in cases of a single message affecting multiple commitments. For example, consider two commitments created by separate messages. If one message fully discharges both of them, their respective conversation graphs would be merged and would demonstrate a tree-like structure. Our mapping would apply to each branch of that tree and the resulting CC component would show all their messages with the discharge message ordered after each of the create messages.

4.7 M7: Generate a Protocol (Automatic)

Identify semantic structures and pragmatic patterns that match components of the CC graph. For instance, the graph in Figure 8 consisting of Messages 1, 2a, 23, 24, and 25 corresponds to the conversation initiated by Message 1. It matches a

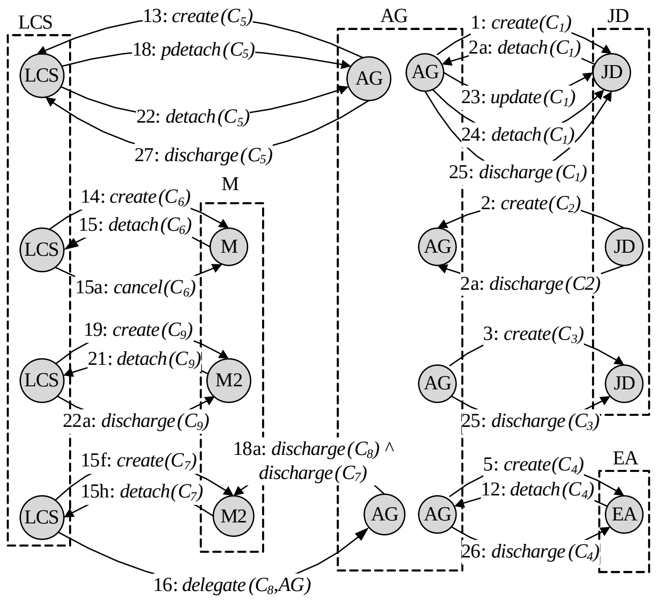


Fig. 8: The CC graph for the example of Table 6. The connected components correspond to conversations and indicate the modularity of the protocol to be created. The circles are the characters; all characters with the same label are aggregated into one role. Messages within each component are partially ordered, here indicated by their numbers. Advance relationships link components but are omitted to reduce clutter.

composition of the Create, Progress, Update, and Complete structures introduced in Section 2.2. Figure 9 shows the protocols generated from the CC graphs of Figure 8.

– *Input:* The output of Step M6.
– *Output:* A protocol based on the input CC graph.
– *Description:* Create a protocol from the generated CC graph by mapping the graphs to our structures.

Define the protocol in terms of commitments among the independent entities and the restrictions on the operations that can be performed on those commitments. In the AGFIL example, we obtain:

– Create a commitment from the insurer AG to the insured JD: C(AG, JD, pay premium p, provide insurance).
– Create commitments from AG, to the claim handler EA and the estimate verifier LCS, respectively: C(AG, EA, handle claims, pay) and C(AG, LCS, resolve claims, pay).
– Create a commitment from LCS to the mechanic M: C(LCS, M, provide estimate, pay).
– Create a commitment from AG to M: C(AG, M, repair, pay).
– Any commitment may be delegated, assigned, canceled, released, or updated.

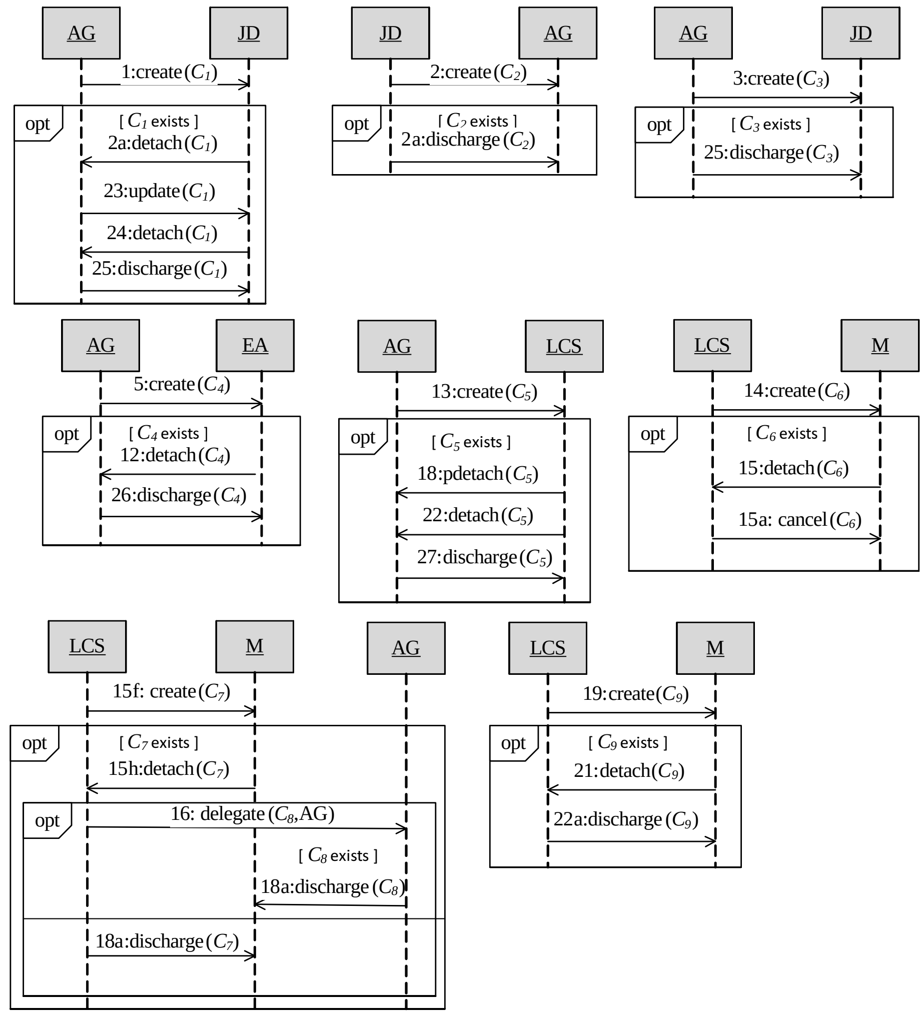


Fig. 9: The generated protocol for the AGFIL case.

The above steps incorporate a design decision to first create an intermediate CC graph representation and then map it to a protocol. The main benefit of doing so is to have a description that is independent of the final syntax. Potentially, depending on the software engineering processes adopted in a particular setting, a CC graph could be mapped not only to SDs but to some other operational representation.

Muon can deal with settings where an ordering among messages is important. Such ordering requirements on protocols would first appear in the scenarios. Some such requirements relate to the advance relationship, which captures any causal connection, e.g., a request must precede a response. Such requirements eventually show up in the final protocol as guards on some of the SDs that refer to conditions brought about through messages in other SDs. Other ordering requirements would apply within the antecedents and consequents of commitments, e.g., that the insurance company will verify the car owner's driving record and then his credit history before offering an insurance policy (thus the car owner's credit history is not reviewed if his driving record is not acceptable). A designer may model such cases in one of two ways: (1) via the advance relationship (trigger the credit check request after receiving the driving record) or (2) embedding the temporal condition within the antecedent or consequent. In the latter case, the detach or discharge messages would embed the requirement, and would be sent as appropriate.

5 Empirical Evaluation and Results

We now describe a partial empirical evaluation to compare Muon and a traditional approach (dubbed *Trad* below) for creating a protocol from a textual description along with happy path and exception scenarios (given in the appendix). Trad is based on UML 2.0 Sequence Diagrams (SDs) Object Management Group (2004) and their precursor, AUML (Odell et al., 2001). The study sought to compare the quality of the protocols produced and expressed as SDs.

- **Trad:** Subjects create a combined scenario; check its completeness with respect to the provided description; and create a protocol.
- **Muon:** Subjects create a protocol using Muon, described as http://research.csc.ncsu.edu/mas/code/chor/Methodology.pdf.

We designed our study in a way as to eliminate the following threats.

- **Expertise differences** among the subjects may affect the study results. To eliminate this threat, we divided the 58 subjects into two sets (Group A and Group B) with equal mean industry experience, as reported by subjects.
- **Learning effect**, wherein subjects have unequal opportunities to learn about a domain. Our cases were the AGFIL case and a breast cancer diagnosis case (ASPE, 2010). First, Group A designed a protocol for AGFIL and Group B for ASPE using the traditional method. Next, the groups applied Muon though with cases switched (Group A on ASPE and Group B on AGFIL).
- **Instrumentation** refers to the effect of tooling on performance. All subjects used IBM Rational Software Architect 8.0.2 for developing their solutions.

5.1 Descriptive Results

The subjects were 58 computer science graduate students, 10 of whom were already familiar with commitments. The labels Trad-expert and Muon-expert represent the results from these 10 subjects whereas Trad-novice and Muon-novice represent the results from the remaining 48 subjects. We measured the following dependent variables for each group and each approach.

- **Flexibility** measures the number of possible executions.
  - *Sequence Diagram* (SD) count indicates modularity (higher is better).
  - *Count of* ALTERNATIVE (ALT)*,* OPTIONAL (OPT)*, and* PARALLEL (PAR) fragments indicating the number of possible executions (higher is better).

- **Objective quality** measures model quality as assessed in objective terms.
  - *Number of missing guards*, indicating spurious enactments (lower is better).
  - *Errors in SDs*, indicating failure to tackle complexity (lower is better).
- **Subjective quality** measures model quality as assessed subjectively.
  - *Scenario coverage* is the distribution of subjects with respect to the coverage of the scenario ranging over high (entire scenario), medium, and low.
  - *Comprehensibility* measures the rater's ability to understand a protocol and ranges over high (easy to comprehend), medium, and low.

Figure 10 shows boxplots of flexibility based on objective criteria. Muon yields higher medians of ALT, OPT, PAR counts (11) than Trad (6), indicating greater flexibility. The median number of SDs for Trad (6) is slightly higher than Muon (5) though the third quartile value for Muon (12.45) is higher than Trad (10.22). A possible reason is that we specifically directed Trad subjects to produce modular diagrams. Notice that Muon promotes modularity without special attention by the designer. Trad-novice yields a higher median (6) than Muon-novice, whereas the medians for Trad-expert and Muon-expert are the same (4).

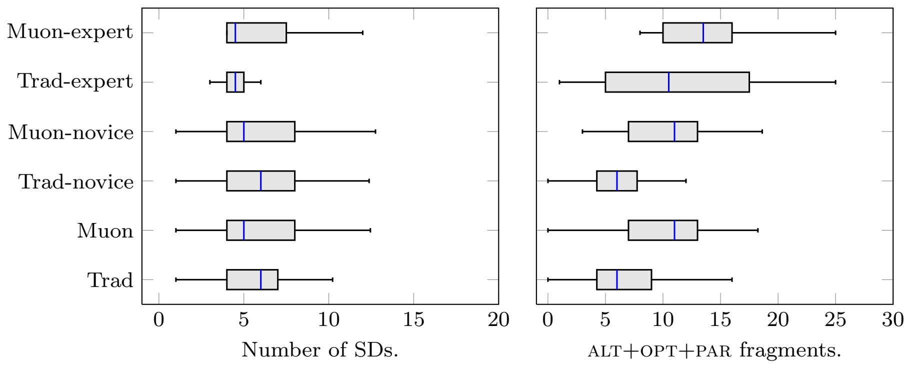


Fig. 10: Flexibility, as judged based on objective criteria.

Figure 11 (left) shows the boxplots for the number of missing guards for Trad and Muon. The results show that the number of subjects with missing guards is lower (1) in case of Muon than Trad (3). We obtain similar results for Trad-novice, Muon-novice, Trad-expert, and Muon-expert. Figure 11 (right) shows the boxplots representing the numbers of errors in SD created by the subjects. The median counts for these errors are the same in Trad, Muon, Trad-novice, and Muon-novice whereas the median count for Muon-expert is slightly lower than Trad-expert.

Figure 12 shows the results for scenario coverage and comprehensibility, respectively. Scenario coverage is higher (57%) for Muon than for Trad (37%) and likewise for Muon-novice and Muon-expert over Trad-expert and Trad-novice, respectively. Comprehensibility is higher in Muon (44%) than Trad (31%) and likewise for Muon-novice and Muon-expert over Trad-expert and Trad-novice, respectively. Muon yields higher scenario coverage and comprehensibility.

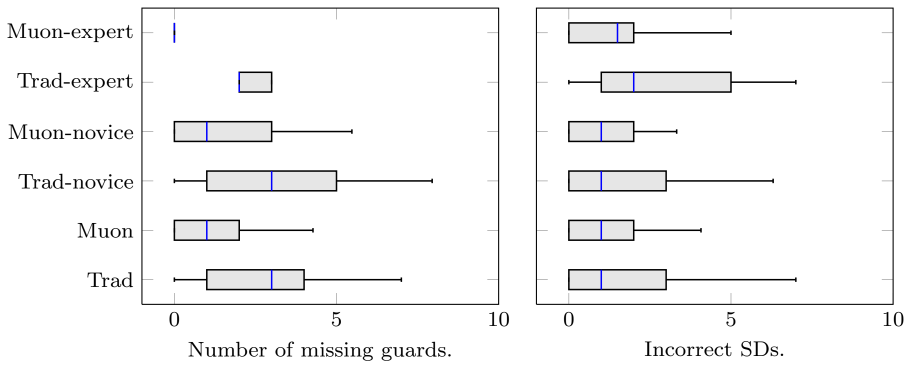


Fig. 11: Objective measures of quality.

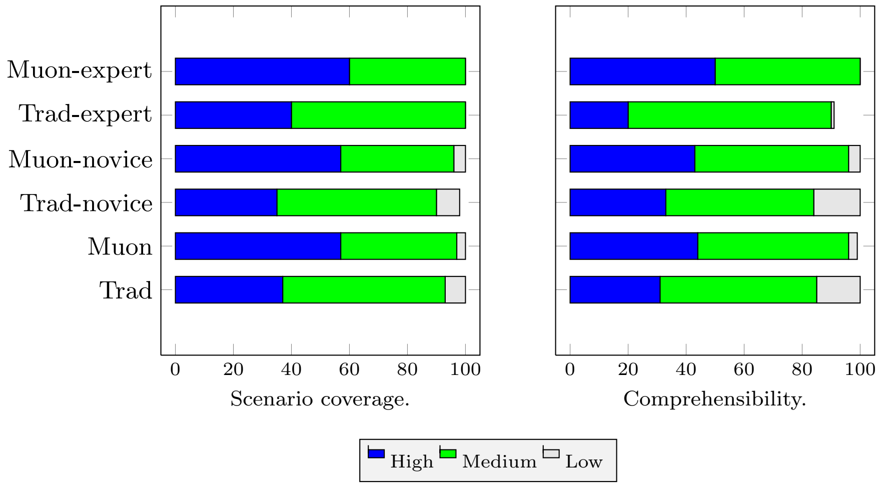


Fig. 12: Distributions of quality, as judged subjectively.

5.2 Hypotheses and Statistical Testing

Table 5 lists four alternative hypotheses, which claim that the means in the Muon and Trad distributions of the referenced variables favor Muon. Following standard practice, for each alternative hypothesis, we formulate a null hypothesis that there is no significant difference between the two means and apply Student's t-test to determine whether each null hypothesis can be rejected at the 5% confidence interval.

Table 5 shows the p-values for the above hypotheses. It shows we reject all null hypotheses except the one for SD count at 5% confidence. That is, we find that subjects employing Muon produce more flexible SDs, with fewer missing guards, and fewer incorrect SDs than subjects employing Trad. However, the number of SDs is not significantly higher among the Muon subjects.

Table 5: Hypothesis testing.

Hypothesis	**Muon Mean** (μ_m)	**Trad Mean** (μ_t)	**Null Hypothesis** [$\mu_m = \mu_t$] **p-value**	**Accepted at p-value 5%?**
Muon yields higher mean count of ALT, OPT, PAR fragments than Trad	10.21	6.93	0.001	×
Muon yields higher mean SD count than Trad	5.94	5.35	0.142	✓
Muon yields lower mean count of missing guards than Trad	1.26	2.89	0.001	×
Muon yields lower mean count of incorrect SDs than Trad	1.18	1.90	0.014	×

5.3 Threats to Validity

We described above how our study design mitigated the threats of expertise differences, learning effect, and instrumentation. An external threat is that our subjects are differently qualified and motivated than actual industry designers: most have less work experience and were required to learn Muon. To control for the subjects' lack of expertise in the chosen domains, we provided multiple scenarios and textual descriptions. In essence, Steps M1, M2, and M3 were performed for the subjects. However, they had to apply the descriptions to annotate the scenarios, determine which exceptions were relevant, carry out a sanity check on the commitments, and expand the scenarios on their own.

6 Conclusions and Discussion

Muon addresses the problem of specifying protocols that not only capture stakeholder needs but also reflect a commitment-based understanding of interactions among autonomous parties. Capturing scenarios is a natural way to elicit stakeholder requirements and to provide a design rationale for the protocol. Work on scenarios (Filippidou, 1998) shares the intuition that the concreteness of scenarios helps in coming up with models. Notice that the primary work done by the designers is in coming up with the scenarios; understanding and semantically annotating them; and deciding how to expand scenarios. Muon introduces semantic structures and patterns for commitments and helps designers by offering sanity checks based on commitments and converting annotated scenarios into a protocol. It provides a natural approach to determine a modular protocol from a set of scenarios. Our empirical findings suggest that Muon produces gains in flexibility and quality, thereby showing the promise of introducing multiagent concepts into software engineering practice.

A common challenge to all protocol approaches is one of perspective. That is, what should happen if the parties involved do not agree as to the facts? There is no fundamental solution to this challenge except to introduce an authoritative party or an agreed upon infrastructure that captures the facts definitively for the interaction. This challenge, however, is orthogonal to our approach since we are concerned with specifying a protocol, and an appropriate protocol should reflect such considerations in its messages. Our causal approach to patterns points to a natural way to create protocols that avoid problems such as race conditions.

6.1 Related Work

Muon extends Parunak's (1996) work on deriving Dooley graphs from a scenario. Wan and Singh (2003) relate communications to Commitment Causality Diagrams (CCDs), which depict the causal relationships between operations on the commitments in a scenario. Their treatment, however, creates processes for specific scenarios, not reusable protocols that can be recombined in novel ways. Further, Wan and Singh assume that operations on commitments are preceded by explicit proposals, which restricts agent autonomy. Unlike in the CRC (Beck and Cunningham, 1989) and scenario-based methodologies (Filippidou, 1998), in Muon, the experts play roles corresponding to autonomous entities, e.g., real-life business partners, not to programming objects. Not only are contents of the interactions higher level, they can be initiated by a stakeholder and can advance a prior communication. Additionally, Muon uses the scenarios to create meaning-based representations, which earlier approaches ignore.

Commitments have previously been applied in understanding communications, e.g., (Flores et al., 2007). Colombetti and colleagues (Verdicchio and Colombetti, 2003; Fornara and Colombetti, 2003) share our intuitions, though their life cycle involves a precommit state, which we avoid by making commitments conditional and potentially unilaterally created by their debtors. Previous works have realized commitments in a variety of logic-based languages, such as event calculus (Chesani et al., 2009; Yolum and Singh, 2002b), rules (Desai et al., 2005), UML's object constraint language (OCL) (Robinson and Purao, 2009), and nonmonotonic causal logic (Desai et al., 2007b).

Winikoff's (2006) approach for designing agent interactions begins from a sequence of steps, each performed by a role accomplishing a goal or action. He inserts conditions on each step and specifies commitments to fill in any gap between required and available conditions, and generalizing the interaction by incorporating some variations. In contrast, Muon begins from scenarios consisting of messages, identifies their causal relationships and annotates them with relevant propositions and commitment operations. It uses commitments as a basis for considering possible exceptions and accommodating the exceptions the stakeholders consider worthwhile.

Günay et al. (Günay et al., 2012) describe how to generate a protocol from an analysis of the beliefs and goals of the participants. Carabelea and Boissier (2006) show how to design a model that represents the expected behavior of an agent using commitments to use as a basis for rewarding or punishing an agent depending upon the state of a commitment. These approaches go top-down from models of rational agents to determine how they should interact whereas Muon works bottom-up from annotated scenarios and build protocols independent of the participants' internal models.

Flores and Kremer (2004) apply the *protocol for proposals* to propose every commitment operation, thus reducing the autonomy of the participants. They use obligations to denote which agents' turn it is to perform some action in response to another action. Such obligations, however, are inimical to autonomy. Flores and Kremer address runtime state maintenance, not the creation of a specification.

Our solution compares well to Desai et al.'s (2009) solution, who describe a hand-crafted protocol for the AGFIL case. Their solution is modular but does not highlight the causal and meaning-based relationships we emphasize. Desai et al. do not capture the design rationale of their protocol and must rely on recreating the necessary domain knowledge to make any modifications to their solution. However, their emphasis is requirements evolution, which we disregard here.

Chopra and Singh (2011) identify commitment-based business patterns and antipatterns corresponding respectively to appropriate or inappropriate uses of commitments. They do not provide a methodology for eliciting protocol requirements as Muon does.

Commitment protocols are either operationalized directly via a logic-based (e.g., event calculus (Chesani et al., 2009; Yolum and Singh, 2004) and causal logic (Chopra and Singh, 2004)) or a rule-based (e.g., Jess (Desai et al., 2005)) approach. In essence, these approaches build a logic-based representation of commitments and implement the commitment life cycle, describing how that representation progresses based on operations on commitments.

A traditional use of rules is to embed the semantics in the rules alone (Grosof and Poon, 2003) and to enable intelligent decision making. But unless the semantics is part of the protocol, the partners cannot use it to support their collaboration. And, revealing internal details—even if those details are expressed in high-level terms as rules—means the partners' implementations are more tightly coupled than they should be. However, protocols can be naturally applied in combination with rules, e.g., as in (Desai et al., 2005).

BPMN (OMG, 2010) is a standard notation that shows the internal flows of the participants and how they interact with one another. BPMN does not cleanly separate the internal implementation from the interactions, and omits the meanings of the interactions. As a result, it yields models that cannot easily handle exceptions (e.g., cancellations) or opportunities (e.g., delegations). More generally, any operational representation (e.g., tasks and messages) does not natively represent social relationships among partners. A designer can deal with exceptions and opportunities in BPMN, but only by reducing them to operational details. The difficulty in producing and maintaining such reductions tends to lead designers to become over-cautious and capture nearly serial execution paths. Thus, such representations suffer from limiting autonomy and flexibility of the participants while lacking a clear basis for verifiability.

6.2 Limitations and Directions

First, Muon relies upon designers coming up with natural scenarios. Although this can be a benefit if it helps designers imagine relevant interactions, it can prove to be a significant limitation if the designers fail to come up with consider normal interactions or omit any important exceptions. Either of those can result in an imperfect final protocol. Although the designers ought to possess the requisite knowledge, they can be guided through methods such as the *critical incident technique* from psychology, which is applied in some information systems problems (Serenko and Turel, 2010), albeit not for interactions as required for protocols.

Second, a limitation is the manual effort required, which we hope to address through improved tooling. One, to come up with the correct annotations is nontrivial—and erroneous annotations would yield a flawed protocol. However, if the effort expended in annotation forces clarity among the designers, that can be beneficial in the long run. Additionally, natural language processing can facilitate this task. Kalia et al. show how to automatically annotate email and chat communications exchanged in technical support with operations on commitments (Kalia et al., 2013). It would be interesting to enhance such techniques to automatically infer causal relationships between communications as a basis for seeding the set of scenarios considered. Two, it would help to enhance our commitment verification tool (Telang and Singh, 2012), possibly by leveraging newer approaches for commitment verification, e.g., (El Menshawy et al., 2013; Baldoni et al., 2014), to capture scenarios and flag possible exceptions for a designer. The designer would review such flagged scenarios and, if the exceptions make sense in the domain, potentially generate additional scenarios to handle them. Once we improve our tools, it would be worth repeating an evaluation with expanded hypotheses, incorporating not only flexibility, quality, and comprehensibility, but also efficiency.

Third, Muon considers only commitments. Other approaches (Álvarez-Napagao et al., 2009; Artikis et al., 2009; Dignum et al., 2005; Fornara and Colombetti, 2009) consider organizations and norms, which are relevant for understanding communications. These approaches do not provide specific guidelines for capturing stakeholder requirements based on explicit scenarios as we have. To enhance Muon to accommodate an additional construct, we need to specify (1) its life cycle and semantics to characterize progress, completion, and advance and (2) sensible pragmatic patterns.

Acknowledgments

Special thanks to Ashok Mallya for early discussions that led to this work. Thanks to Amit Chopra, Nirmit Desai, Jon Doyle, Scott Gerard, Emerson Murphy-Hill, Pankaj Telang, and the anonymous reviewers for helpful comments. This work was partially supported by the NSF under grant 0910868 and by the U.S. Army Research Office (ARO) under grant W911NF-08-1-0105 managed by the NCSU Secure Open Systems Initiative.

References

Álvarez-Napagao, S., Cliffe, O., Padget, J. A., Vázquez-Salceda, J., 2009. Norms, organisations and semantic web services: The ALIVE approach. In: Proceedings of the 2nd Multi-Agent Logics, Languages, and Organisations (MALLOW). Vol. 494 of CEUR. CEUR-WS.org, Torino.

Artikis, A., Sergot, M. J., Pitt, J. V., Jan. 2009. Specifying norm-governed computational societies. ACM Transactions on Computational Logic 10 (1), 1:1–1:42.

ASPE, Nov. 2010. The importance of radiology and pathology communication in the diagnosis and staging of cancer: Mammography as a case study. Office of the Assistant Secretary for Planning and Evaluation, U.S. Department of Health and Human Services; available at http://aspe.hhs.gov/sp/reports/2010/PathRad/index.shtml.

Baldoni, M., Baroglio, C., Marengo, E., Patti, V., Capuzzimati, F., 2014. Engineering commitment-based business protocols with 2CL methodology. Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS), 1–37In press.

Beck, K., Cunningham, W., 1989. A laboratory for teaching object-oriented thinking. In: Proceedings of the 4th Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA). ACM, New Orleans, pp. 1–6.

Browne, S., Kellett, M., 1999. Insurance (motor damage claims) scenario. Document D1.a, CrossFlow Consortium.

Carabelea, C., Boissier, O., May 2006. Coordinating agents in organizations using social commitments. Electronic Notes in Theoretical Computer Science 150 (3), 73–91.

Chesani, F., Mello, P., Montali, M., Torroni, P., 2009. Commitment tracking via the reactive event calculus. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI). IJCAI, Pasadena, California, pp. 91–96.

Chopra, A., Singh, M. P., 2004. Nonmonotonic commitment machines. In: Dignum, F. (Ed.), Advances in Agent Communication: Proceedings of the 2003 AAMAS Workshop on Agent Communication Languages. Vol. 2922 of LNAI. Springer, pp. 183–200.

Chopra, A. K., Singh, M. P., May 2011. Specifying and applying commitment-based business patterns. In: Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS). IFAAMAS, Taipei, pp. 475–482.

Desai, N., Chopra, A. K., Arrott, M., Specht, B., Singh, M. P., 2007a. Engineering foreign exchange processes via commitment protocols. In: Proceedings of the 4th IEEE International Conference on Services Computing (SCC), Application and Industry Track. IEEE Computer Society, Salt Lake City, pp. 514–521.

Desai, N., Chopra, A. K., Singh, M. P., Jul. 2007b. Representing and reasoning about commitments in business processes. In: Proceedings of the 22nd Conference on Artificial Intelligence (AAAI). AAAI Press, Vancouver, pp. 1328–1333.

Desai, N., Chopra, A. K., Singh, M. P., Oct. 2009. Amoeba: A methodology for modeling and evolving cross-organizational business processes. ACM Transactions on Software Engineering and Methodology (TOSEM) 19 (2), 6:1–6:45.

Desai, N., Mallya, A. U., Chopra, A. K., Singh, M. P., Dec. 2005. Interaction protocols as design abstractions for business processes. IEEE Transactions on Software Engineering 31 (12), 1015–1027.

Dignum, V., Vázquez-Salceda, J., Dignum, F., 2005. OMNI: Introducing social structure, norms and ontologies into agent organizations. In: Programming Multi-Agent Systems. Vol. 3346 of Lecture Notes in Computer Science. pp. 181–198.

El Menshawy, M., Bentahar, J., Kholy, W. E., Dssouli, R., Nov. 2013. Reducing model checking commitments for agent communication to model checking ARCTL and GCTL*. Journal of Autonomous Agents and Multi-Agent Systems

(JAAMAS) 27 (3), 375–418.

Filippidou, D., Mar. 1998. Designing with scenarios: A critical review of current research and practice. Requirements Engineering 2 (1), 1–22.

Flores, R. A., Kremer, R. C., 2004. A principled modular approach to construct flexible conversation protocols. In: Proceedings of the 17th Canadian Conference on Artificial Intelligence. Vol. 3060 of LNCS. Springer, London, Ontario, pp. 1–15.

Flores, R. A., Pasquier, P., Chaib-draa, B., Apr. 2007. Conversational semantics sustained by commitments. Autonomous Agents and Multi-Agent Systems 14 (2), 165–186.

Fornara, N., Colombetti, M., Jul. 2003. Defining interaction protocols using a commitment-based agent communication language. In: Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). ACM Press, Melbourne, pp. 520–527.

Fornara, N., Colombetti, M., 2009. Specifying and enforcing norms in artificial institutions. In: Declarative Agent Languages and Technologies VI, Revised Selected and Invited Papers. Vol. 5397 of LNCS. Springer, Berlin, pp. 1–17.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Professional Computing Series. Addison-Wesley, Reading, MA.

Grosof, B. N., Poon, T. C., 2003. SweetDeal: Representing agent contracts with exceptions using XML rules, ontologies, and process descriptions. In: Proceedings of the 12th International Conference on the World Wide Web. pp. 340–349.

Günay, A., Winikoff, M., Yolum, P., 2012. Commitment protocol generation. In: Proceedings of the 10th AAMAS Workshop on Declarative Agent Languages and Technologies (DALT). pp. 51–66.

ISO/HL7, Jun. 2009. Data exchange standards – Health Level Seven version 2.5 – an application protocol for electronic data exchange in healthcare environments. TC 215's ISO/HL7 27931 http://www.iso.org/iso/catalogue_detail.htm?csnumber=44428.

Kalia, A. K., Nezhad, H. R. M., Bartolini, C., Singh, M. P., 2013. Monitoring commitments in people-driven service engagements. In: Proceedings of the 10th IEEE International Conference on Services Computing (SCC). IEEE Computer Society, Santa Clara, California, pp. 160–167.

Malone, T. W., Crowston, K., Herman, G. A. (Eds.), 2003. Organizing Business Knowledge: The MIT Process Handbook. MIT Press, Cambridge, MA.

Object Management Group, Jul. 2004. UML 2.0 Superstructure Specification. Framingham, Massachusetts, http://www.omg.org/spec/UML/2.0/Superstructure/PDF/.

Odell, J., Parunak, H. V. D., Bauer, B., 2001. Representing agent interaction protocols in UML. In: Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE 2000). Vol. 1957 of LNCS. Springer, Toronto, pp. 121–140.

OMG, Jun. 2010. Business process model and notation (BPMN), version 2.0 beta. Object Management Group. http://bpmn.org/.

Oracle, Jun. 2009. Automating the Quote-to-Cash process. http://www.oracle.com/us/industries/045546.pdf.

Parunak, H. V. D., 1996. Visualizing agent conversations: Using enhanced Dooley graphs for agent design and analysis. In: Proceedings of the 2nd International

Conference on Multiagent Systems. AAAI Press, Kyoto, pp. 275–282.

Qumer, A., Henderson-Sellers, B., Mar. 2008. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. Information and Software Technology 50 (4), 280–295.

Robinson, W. N., Purao, S., Mar. 2009. Specifying and monitoring interactions and commitments in open business processes. IEEE Software 26 (2), 72–79.

RosettaNet, 2009. Home page. http://www.rosettanet.org.

Serenko, A., Turel, O., Apr. 2010. Rigor and relevance: The application of the critical incident technique to investigate email usage. Journal of Organizational Computing and Electronic Commerce 20 (2), 182–207.

Singh, M. P., Mar. 1999. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. Artificial Intelligence and Law 7 (1), 97–113.

Singh, M. P., Jun. 2000. Synthesizing coordination requirements for heterogeneous autonomous agents. Autonomous Agents and Multi-Agent Systems 3 (2), 107–132.

Singh, M. P., Jul. 2008. Semantical considerations on dialectical and practical commitments. In: Proceedings of the 23rd Conference on Artificial Intelligence (AAAI). AAAI Press, Chicago, pp. 176–181.

Singh, M. P., 2012. Commitments in multiagent systems: Some history, some confusions, some controversies, some prospects. In: Paglieri, F., Tummolini, L., Falcone, R., Miceli, M. (Eds.), The Goals of Cognition: Essays in Honor of Cristiano Castelfranchi. College Publications, London, Ch. 32, pp. 613–638, available at http://www.csc.ncsu.edu/faculty/mpsingh/papers.

Singh, M. P., Chopra, A. K., Desai, N., Nov. 2009. Commitment-based service-oriented architecture. IEEE Computer 42 (11), 72–79.

Telang, P. R., Singh, M. P., Jul. 2012. Specifying and verifying cross-organizational business models: An agent-oriented approach. IEEE Transactions on Services Computing 5 (3), 305–318, appendix pages 1–5.

TWIST, Nov. 2008. Transaction workflow innovation standards team. http://www.twiststandards.org.

van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., Barros, A. P., Jul. 2003. Workflow patterns. Distributed and Parallel Databases 14 (1), 5–51.

Verdicchio, M., Colombetti, M., Jul. 2003. A logical model of social commitment for agent communication. In: Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS). ACM Press, Melbourne, pp. 528–535.

Wan, F., Singh, M. P., Jul. 2003. Commitments and causality for multiagent design. In: Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS). ACM Press, Melbourne, pp. 749–756.

Winikoff, M., 2006. Designing commitment-based agent interactions. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology. Hong Kong, pp. 363–370.

Yolum, P., Singh, M. P., 2002a. Commitment machines. In: Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages (ATAL 2001). Vol. 2333 of LNAI. Springer, Seattle, pp. 235–247.

Yolum, P., Singh, M. P., Jul. 2002b. Flexible protocol specification and execution: Applying event calculus planning using commitments. In: Proceedings of the 1st

International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS). ACM Press, Bologna, pp. 527–534.

Yolum, P., Singh, M. P., Sep. 2004. Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. Annals of Mathematics and Artificial Intelligence 42 (1–3), 227–253.

A Appendix: A Complete Scenario

Step M4 yields a complete scenario containing the happy path and exception scenarios. Table 6 shows this complete scenario.

B Appendix: Empirical Study Details

B.1 The ASPE Breast Cancer Diagnosis Case

We provided the following passage to study subjects.

A patient (P) finds symptoms of breast cancer and reports a primary care physician (PCP). If the patient is new, PCP immediately starts examining the patient or else he collects the history of the patient before the examination. PCP thoroughly examine the breasts for lumps or suspicious areas. He then sends the patient to a radiologist (R) for a mammography or imaging. R performs a diagnostic imaging and reports the results to PCP. PCP reviews the results. If PCP finds P's condition not worrisome, then he asks P to visit just for an yearly checkup. If PCP finds the P's tumor benign, then he asks her to come back after four to six months. If PCP finds the tumor suspicious then he orders R for a biopsy. R performs a biopsy and forwards the tissue specimens to a pathologist (PT). PT accesses the specimen and conducts several laboratory examinations to determine the nature of the cancer and comes up with a report. PT then holds a conference with R and ensures their results are concordant. R then forwards the integrated reports produced by him and P to PCP. The registrar (RG) registers the patient under a breast cancer registry. PCP checks the integrated report. If the tissue sample is benign, the tumor is removed by a surgeon (S). If the sample is malignant, PCP discusses the treatment steps with P. P pays PCP for the checkup, imaging, and biopsy. PCP pays R for imaging and biopsy. R pays PT for preparing the biopsy report.

B.2 AGFIL Automobile Insurance Case

We provided the following passage to study subjects.

AGF Irish Life Holdings (AG) is an insurer and covers the losses incurred by policy holders. John Doe (JD) is a policy holder. AG creates a policy with JD such that if JD pays the premium, AG will insure his car. JD pays the premium and gets his car insured. AG requests Europ Assist (EA) to receive claims from policy holders to which EA agrees. AG pays EA for receiving the claims. When JD meets with a car accident, he requests for a claim to EA. EA asks JD to take his car to a mechanic (M) and reports AG about the claim made by JD. AG hires Lee CS (LCS) for handling the claims made by JD. LCS offers M to pay if M estimates the repair costs for the JD's car. When M provides the estimates, LCS verifies it. If the estimates are reasonable, he offers M to repair the car. When M repairs the car, LCS delegates the payment to AG. AG pays M for the repair and informs JD. JD gets his repaired car from M. AG pays EA for receiving the claims.

B.3 Scenarios for the ASPE Case

We provided the following happy path and exception scenarios to study subjects.

B.4 Scenarios for the AGFIL Case

We provided the following happy path and exception scenarios to study subjects.

Table 6: A complete scenario for the AGFIL case. (A: advance; P: progress; U: update; C: complete; crt: create; det: detach; dis: discharge; cnl: cancel; rel: release; del: delegate; pdet: partial detach.)

ID	**S**	**R**	**Content**	**A**	**P**	**U**	**C**	**Operation**
1	AG	JD	If you pay premium p, I will insure your car					crt(C_1)
2	JD	AG	If you insure my car, I will pay premium p					crt(C_2)
2a	JD	AG	Pay premium p		1		2	det(C_1), dis(C_2)
3	AG	JD	I will handle and resolve claims	1				crt(C_3)
4	AG	JD	EA will handle claims	3				
5	AG	EA	If you will handle claims, I will pay	4	3			crt(C_4)
6	EA	AG	OK, I accept to handle claims	5				
7	JD	AG	OK, I accept EA as claims handle	4				
8	AG	EA	JD has agreed	5, 7				
9	AG	JD	EA has agreed	4, 7				
10	JD	EA	Insurance claim	4				
11	EA	JD	Take the car to M	10				
12	EA	AG	JD's car is with M		5			det(C_4)
13	AG	LCS	If you resolve the claim (verify estimate and handle car repair), I will pay		3			crt(C_5)
14	LCS	M	If you provide the estimate and I accept it, I will pay	13				crt(C_6)
15	M	LCS	Here is the estimate for the repairs		14			det(C_6)
15a	LCS	M	Reject the estimate from M	5			15	cnl(C_6)
15b	LCS	AG	Suggest another mechanic	14				
15c	AG	EA	Suggest another mechanic to LCS	15b		12		
15d	EA	LCS	contact M2	15c				
15e	EA	JD	Take the car to M2	15d				
15f	LCS	M2	If you provide the estimate and I accept it,I will pay	13				crt(C_7)
15g	M2	LCS	Here is the estimate for the repairs		15f			pdet(C_7)
15h	LCS	M2	I accept the estimate		15g			det(C_7)
16	LCS	AG	I request you to pay M2		15f			del(C_8, AG), crt(C_8), det(C_8)
17	AG	LCS	I agree to pay	16				
18	LCS	AG	OK, I have verified the estimate		13			pdet(C_5)
18a	AG	M2	Here is the payment for the estimate				16, 15h	dis(C_8), dis(C_7)
19	LCS	M2	If you repair the car, I will pay	3				crt(C_9)
20	M2	LCS	OK	19				
21	M2	LCS	Car is repaired				19	det(C_9)
22	LCS	AG	Car is repaired	21	18			det(C_5)
22a	LCS	M2	Here is the payment for repairs				21	dis(C_9)
23	AG	JD	Your premium has increased to p'			2a		
24	JD	AG	Here is (the updated premium) p'			23	det(C_1)	
25	AG	JD	Take back the car from M2				24, 3	dis(C_1), dis(C_3)
26	AG	EA	Here is payment for handling claims				12	dis(C_4)
27	AG	LCS	Here is payment for resolving the claims				6	dis(C_5)

Table 7: The breast cancer case (happy path I).

ID	**S**	**R**	**Content**
1	P	PCP	I'll pay a, if you do a checkup
2	P	PCP	Here is the payment a for the checkup
3	PCP	P	Performs the checkup
4	PCP	P	I did not find the breast lumps suspicious, you can go home

Table 8: The breast cancer case (exceptions I).

ID	**S**	**R**	**Content**
1	PCP	P	I will not perform the checkup and will assign you another physician
2	PCP	PCP_1	Will you checkup the patient P?
3	PCP_1	PCP	yes I will
4	PCP	P	you can visit PCP_1

Table 9: The breast cancer case (happy path II).

ID	**S**	**R**	**Content**
1	P	PCP	Requests a checkup
2	PCP	P	Performs the checkup
3	PCP	P	If I find the lumps suspicious, I will order for an imaging
4	P	PCP	I agree for the imaging
5	PCP	P	The lumps are suspicious and I am ordering the imaging
6	R	PCP	If you order an imaging, I will do the imaging and provide you a report
7	PCP	R	Here is the order for a imaging with P's information
8	R	PCP	Here is the imaging report
9	PCP	P	The report suggests the tumor is benign, you can come after 4-6 months

Table 10: The breast cancer case (exceptions II).

ID	**S**	**R**	**Content**
1	R	PCP	I am on leave, I will ask R_1 to do the imaging
2	R	R_1	Will you do the imaging
3	R_1	R	Yes I will
4	R	PCP	R_1 will do the imaging

Table 11: The breast cancer case (happy path III).

ID	S	R	Content
1	P	PCP	Requests Checkup
2	PCP	P	Performs the checkup
3	PCP	P	I will order for an imaging if I find the lumps suspicious
4	P	PCP	I agree
5	PCP	P	The lumps are suspicious and hence ordering for the imaging
6	R	PCP	I will do the imaging and provide you a report
7	R	PCP	Here is the imaging report
8	PCP	P	I will order for a biopsy
9	P	PCP	I agree for a biopsy
10	PCP	R	Here is the order for a biopsy with P's information
11	R	PT	Here are the P's tissue specimens
12	PT	R	Here is the biopsy report
13	R	PCP	Here is the integrated report
14	PCP	P	Your results are positive and here are steps for the treatment
15	P	PCP	Agree for the treatment
16	PT	RG	Add P's information under cancer registry
17	RG	PT	Information added

Table 12: The breast cancer case (exceptions III).

ID	S	R	Content
1	R	PCP	I could not do the imaging as the patient did not show up
2	PCP	P	Kindly visit the radiologist
3	P	PCP	Yes I will
4	PT	R	I cannot give the biopsy report as my lab machine is broken
5	R	PCP	PT cannot give the biopsy report
6	PCP	R	Ask another PT to perform the examination and give the report
7	R	PT_1	Will you prepare a biopsy report?
8	PT_1	R	Yes I will, if you give me the tissue specimen
9	R	PT_1	Here is the tissue specimen

Table 13: The breast cancer case (happy path IV).

ID	**S**	**R**	**Content**
1	P	PCP	I'll pay a, if you do a checkup
2	P	PCP	Here is the payment a for the checkup
3	PCP	P	Performs the checkup
4	PCP	P	If I find the lumps suspicious, I will order for an imaging
5	P	PCP	I agree for an imaging
6	PCP	P	The lumps are suspicious, ordering for the imaging
7	PCP	R	If you do the imaging, I will pay a_1
8	R	PCP	If you order a mammography, I will do mammography and provide you a report
9	PCP	R	Here is the order for a imaging with P's information
10	R	PCP	Here is the imaging report
11	PCP	P	The report suggests the tumor is malignant, hence, ordering for a biopsy
12	P	PCP	I agree for a biopsy
13	PCP	R	If you do a biopsy, I will pay a_2
14	R	PCP	If you order for a biopsy, I will provide you a biopsy report
15	R	PT	If I will do a biopsy, I will give you the tissue specimens
16	PT	R	If will get the tissue specimens, I will perform laboratory examinations and provide you a report
17	PCP	R	Here is the order for a biopsy with P's information
18	R	PT	Here are the P's tissue specimens
19	PT	R	If you hold a conference, I will share my biopsy report
20	R	PT	If you hold a conference, I will share my imaging report
21	PT	R	Here is the biopsy report
21	R	PT	Here is the imaging report
20	R	PCP	Here is the integrated report
21	PCP	P	Your results are positive and here are steps for the treatment
22	P	PCP	Agree for the treatment
23	PT	RG	Add P's information under cancer registry
24	RG	PT	Information added
25	PCP	R	Here is the payment a_1 for imaging report
26	PCP	R	Here is the payment a_2 for biopsy

Table 14: The breast cancer case (exceptions IV).

ID	**S**	**R**	**Content**
1	PT	R	I am busy and hence canceling the conference
2	R	PT	I agree
3	R	PT	Here is an another conference date, do you agree?
4	PT	R	Yes, I agree

Table 15: I: A successful scenario for the AGFIL example.

#	S	R	Content
1	AG	JD	I will insure your car when you pay p
2	JD	AG	Here is the premium p
3	AG	JD	EA will receive claims
4	JD	EA	Claim Insurance
5	EA	JD	Take the car to M
6	EA	AG	JD's car is with M
7	AG	LCS	Handle the claim and I will pay
8	LCS	M	Verify the repair estimate and I will pay
9	M	LCS	Here is the estimate for the repairs
10	LCS	AG	pay M for the estimation
11	LCS	M	If you repair, I will pay
12	M	LCS	Car is repaired
13	LCS	AG	pay M for the repair
14	AG	M	Here is the payment for estimates and repairs
15	AG	JD	Car is repaired

Table 16: I: A scenario of exceptions for the AGFIL example.

#	S	R	Content
1	EA	JD	I cannot receive the claims, I will delegate this to EA_1
2	EA	EA_1	Will you receive the claims?
3	EA_1	EA	Yes
4	EA	AG	EA_1 will receive the claims
5	EA	JD	EA_1 will receive the claims

Table 17: II: A successful scenario for the AGFIL example.

#	S	R	Content
1	JD	AG	If you insure my car, I will pay premium p
2	AG	JD	EA will respond to claims and resolve claims
3	JD	AG	OK, I accept EA as claims handle
4	JD	EA	Insurance claim
5	EA	JD	Take the car to M
6	EA	AG	JD's car is with M
7	AG	LCS	Handle the claim from JD
8	LCS	M	Verify the estimate of JD's car
9	M	LCS	Here is the estimate for the repairs
10	LCS	AG	LCS requests AG to pay
11	AG	LCS	AG agrees to pay
12	LCS	AG	Estimate is verified, OK
13	LCS	M	Repair the car
14	M	LCS	OK
15	M	LCS	Car is repaired
16	LCS	AG	Car is repaired
17	AG	M	Here is the payment for estimates and repairs
18	AG	JD	Car is repaired

Table 18: II: A scenario of exceptions for the AGFIL example.

#	S	R	Content
1	LCS	AG	I cannot handle the claims
2	AG	LCS_1	Will you handle the claims?
3	LCS_1	AG	Yes

Table 19: III: A successful scenario for the AGFIL example.

#	S	R	Content
1	AG	JD	If you pay premium p, I will insure your car
2	JD	AG	If you insure my car, I will pay premium p
3	JD	AG	Pay premium p
4	AG	JD	I will respond to claims and resolve claims
5	AG	JD	EA will receive claims
6	AG	EA	If you will receive claims, I will pay
7	EA	AG	OK, I will receive claims
8	JD	AG	OK, I accept EA as claims receiver
9	AG	EA	JD has agreed
10	AG	JD	EA has agreed
11	JD	EA	Insurance claim
12	EA	JD	Take the car to M
13	EA	AG	JD's car is with M
14	AG	LCS	If you handle the claim, I will pay
15	LCS	M	If you verify the estimate, I will pay
16	M	LCS	Here is the estimate for the repairs
17	LCS	AG	LCS requests AG to pay
18	AG	LCS	AG agrees to pay
19	LCS	AG	Estimate is verified, OK
20	LCS	M	If you repair, I will pay
21	M	LCS	OK
22	M	LCS	Car is repaired
23	LCS	AG	Car is repaired
24	AG	M	Here is the payment for estimates and repairs
25	AG	JD	Car is repaired
26	AG	EA	Here is payment for handling claims
27	AG	LCS	Here is payment for verifying claims

Table 20: III: A scenario of exceptions for the AGFIL example.

#	S	R	Content
1	LCS	M	Rejects the estimate from M
2	LCS	M2	LCS commits M2 to pay for the estimate
3	M2	LCS	Here is the estimate for the repairs
4	AG	M2	If you repair, I will pay
5	M2	AG	I cannot repair this damage