ABSTRACT

KALIA, ANUP KUMAR. Muon: Designing Multiagent Communication Protocols from Interaction Scenarios. (Under the direction of Munindar P. Singh.)

Designing a suitable communication protocol is a key challenge in engineering a multiagent system. This work 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 work 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) an algorithm to produce groups of causally related interactions from an interaction scenario; and (4) a methodology to synthesize specifications of communication protocols. This work reports on an empirical evaluation involving developers creating protocols from two real-life cases.

Muon: Designing Multiagent Communication Protocols from Interaction Scenarios

by
Anup Kumar Kalia

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2013

APPROVED BY:

John Doyle	Emerson Murphy-Hill

Munindar P. Singh
Chair of Advisory Committee

DEDICATION

To my mother Late Mrs. Kakali Kalia and my father Dr. Shiba Prasad Kalia for making endless sacrifices to provide me a great livelihood and showing me the righteous path to follow.

BIOGRAPHY

Anup Kalia was born and brought up in Rourkela, a town in the state of Odisha, India. He received his B.Tech in Computer Science and Engineering from College of Engineering and Technology, Bhubaneswar, India in 2008. After working for couple of years in Tata Consultancy Services, he joined the Master's Thesis program in Computer Science in North Carolina State University. In Spring 2012, he joined the PhD program in Computer Science. Anup has also done internships at HP Labs, Palo Alto, CA and Whodini, Los Altos, CA in Summer 2012 and 2013 respectively.

ACKNOWLEDGEMENTS

I am deeply indebted to my advisor Professor Munindar Singh for his endless guidance, motivation, and support during this thesis work. Through this work he introduced me to the world of research especially in the area of multiagent systems that inspired me to enroll in the PhD program and further carry on the research work.

I am also thankful to my committee members Professor Jon Doyle and Professor Emerson Murphy-Hill for discussing and providing useful feedback for this work.

I am thankful to my colleagues (in no particular order) Pankaj Telang, Scott Gerard, Dhanwant Singh Kang, Chung-Wei Hang, Zhe Zhang, Guangchao Yuan, Pradeep M, and Nirav Ajmeri who have supported me in my research work through discussions and meetings.

I am thankful to my friends (in no particular order) Arpit, Anant, Ankita, Gaurav, Srikant, Heena, Haritha, Abhignya, Dileep, Shubham, Shilpa, Ankur, Harsh, Divya, Vandit, Neeraj, Abhinav, and Sanujit for making my life wonderful here in Raleigh.

I am thankful to two of my best friends Narottam and Priyanka who have always inspired me to do the thing I wanted and pursue my dream. Without their support I did not even think of doing a Masters in the US.

Last but not least, I am thankful to my wonderful sisters Archana and Anima and their husbands Sandeep and Debabrata respectively for providing me all kind of support to come here to the US and conquer my dreams.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 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 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. Hence, they fail to provide an adequate basis for a participant to act flexibly and yet be assured of the correctness of the interaction. A participant should be able to act flexibly, exercising its autonomy where its partners are not adversely affected. For example, a customer may wish to pay earlier than absolutely necessary to take advantage of tax laws regarding business expenses.

The multiagent systems community has developed a family of approaches under the rubric of *commitment protocols* [Yolum and Singh, 2002b], which tackle the challenge of specifying communication protocols purely or largely in terms of the meanings of the messages. These works face two challenges.

- 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?

The **top-level research challenge** this paper addresses is *how may we go about specifying useful and robust protocols expressed in a traditional operational representation?* The Muon approach addresses this challenge by incorporating (1) a commitment-based *model* for protocols, (2) semantic *structures* and pragmatic patterns based on commitments, and (3) a *methodology* for producing sound specifications. For simplicity in understanding, we adopt the UML 2.0

sequence diagram notation as the output representation for protocols, but our approach would equally produce any other operational notation.

1.1 Highlights of our Approach

Our approach comprises two components: semantic structures and a methodology, both based on a conceptual model founded on agents creating and manipulating commitments to one another. An *agent* is an autonomous, active computational entity. A specification of 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). C(*debtor*, *creditor*, *antecedent*, *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.

Our proposed model of protocols builds on commitments. Specifically, we formalize communications in a way that captures the meanings of the messages along with any causal relationships among them. Importantly, commitments provide a basis for correctness: a sound protocol enactment must not violate any commitments. As in any approach to protocols, an autonomous agent may not comply: Nothing can force its compliance. Commitments provide a declarative basis for judging correctness, i.e., verifying compliance.

Further, commitments yield semantic (communication) **structures** that are reusable across many settings. A practically valuable generic structure is of how a commitment may be delegated. Continuing the above example, we can design a simple protocol wherein a merchant sends an offer to a customer specifying the goods to be sold and their price; the customer rejects or accepts the offer and, if he accepts the offer, pays the specified price; and, the merchant delivers the specified goods. Importantly, we can expand the above protocol to one wherein the customer pays via a bank and the merchant delivers via a shipper. The latter, more complex, protocol introduces additional roles and message types. Yet, in intuitive terms, it can readily be seen as being derived from the initial, simpler protocol via the application of semantic structures based on commitments.

Based on the above model and semantic structures, the Muon **methodology** helps specify protocols that address specific stakeholder needs. Early-stage specifications inherently require designer creativity, yet there are systematic ways to ensure that they capture the known needs of the stakeholders. 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.

1.2 Running Example

We explain Muon using a car insurance claim handling case study that was introduced by Browne and Kellet (1999) from an analysis for the real-life operations of the Assurance General France Irish Life (AGFIL) Holding Company, a subsidiary of Allianz. This case has been used often in previous research.

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 Contribution

Muon enhances and combines disparate bodies of work to provide a basis for designing multiagent communication protocols. Muon expands previous analyses to extract additional informa-

tion from annotated interaction scenarios, thus inducing cleaner specifications of roles and their interactions. In particular, Muon shows how to (1) ensure that stakeholder needs, expressed via illustrative scenarios by the stakeholders, are captured; (2) verify that such scenarios are completed with respect to events pertaining to any commitment, including potentially its violation; and (3) produce modular protocols that can be systematically combined.

1.4 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 most relevant literature, and important future directions.

Chapter 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

We consider agents as autonomous, reflecting the autonomy of the parties they represent. Commitments are publicly observable means of modeling their mutual expectations, which are key to their being able to cooperate. Because each agent is autonomous, it can apply its internal policies to decide whether to commit to another agent and, once committed, whether to satisfy a commitment. A commitment doesn't curtail an agent's autonomy but provides a standard for judging whether an agent is compliant.

A commitment $c = \mathsf{C}(\mathit{debtor}, \mathit{creditor}, \mathit{antecedent}, \mathit{consequent})$ is as defined in Section 1.1. 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 based on [Singh, 1999]. We extend these in Muon through the introduction of partial detach and discharge [Singh, 2008].

1. *Create*(c) creates the commitment c. This operation can be performed only by c's debtor. An example is the *quote* message described above.

2. *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 merchant fails to deliver the goods on time to a customer, we treat that failure as the merchant canceling a 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. The requisite compensations would be specified in a given protocol.

3. *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.

4. *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.

5. *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.

6. *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.

Notice that the underlying formal semantics of commitments is not in the scope of the present paper. Singh (2008) provides a formal model-theoretic semantics as well as inference rules that are sound and complete for that semantics. This semantics provides a basis for (partial or full) detach and (partial or full) discharge, and also supports partial assignment and partial 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 to the 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. Below, 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. An agent may violate a commitment; another agent may seek to penalize it for doing so. For example, if a debtor cancels a commitment, the creditor may complain about it. Such a complaint is not a required part of the commitment semantics and so we classify it as a pragmatic pattern. Section 3 discusses pragmatic patterns.

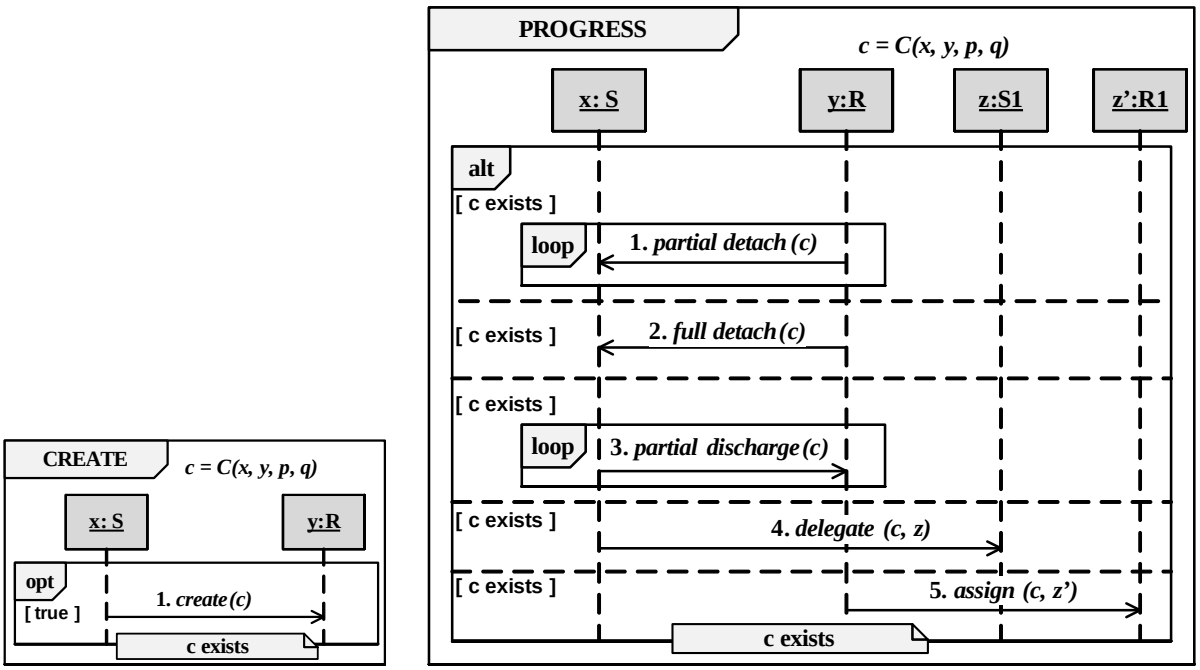
Figure 2.1a shows the elementary semantic structure of a commitment being created by an explicit *create*$(\cdot)$ message from its debtor. (We show the postcondition of each structure at the bottom of its diagram.) A commitment may also be created by the delegation or assignment of another commitment—we 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 2.1b 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* 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 this case, each such message in essence 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



(a) Create. (b) Progress.

Figure 2.1: Semantic structures: create and progress.

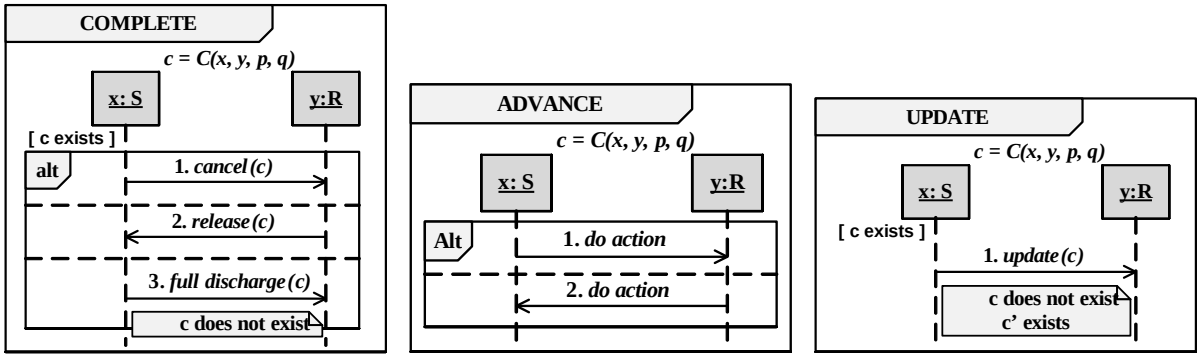
m_i, then m_j progresses m_i. Likewise, m_j progresses m_i if m_j delegates or assigns a commitment created by m_i.

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 2.2a.

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. 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.

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.



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

Figure 2.2: Semantic structures: complete, advance, and update.

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. Figure 2.2b 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 2.2c illustrates this structure. The debtor of a commitment can update its antecedent or consequent. In essence, an update is equivalent to canceling the existing commitment and creating a new one. In terms of Figure 2.3, 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 the progress of that commitment.

2.2.5 Grouping and Adequacy of the Semantic Structures

Figure 2.3 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;

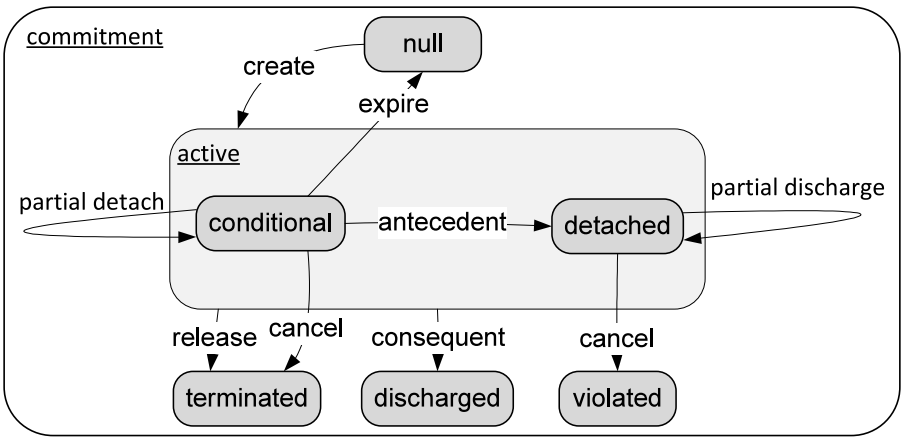


Figure 2.3: The life cycle of a commitment, enhanced from Singh et al. [2009]. Section 2.2.1 explains partial detach and partial discharge.

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.

Notice that our semantic relationships quite naturally help group messages. If a message progresses another, it must (partially or fully) detach, partially discharge, delegate, or assign it. If a message completes another, it must fully discharge, cancel, release, or update that message. If a message advances another, it could have any content. In particular, a creation of a commitment can only advance another message, if it relates to a prior message at all.

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.

2.2.6 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], which places related messages close to each other, and places unrelated messages in disconnected components. 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 and relationships based on meanings that are missing from Singh's approach.

The following terminology is important for understanding our approach. An *(interaction) scenario* is a sequence of messages corresponding to a specific instance or enactment of a protocol. Parunak used 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. Our scenarios pertain only to communications among autonomous roles. A designer annotates a scenario to capture any knowledge of the relationships among the various messages. Such knowledge may arise from a CRC exercise. A *role* is a participant in a scenario. A *character* is a part played by one role in a conversation. A *conversation* is a fragment of a scenario that reflects a causally self-contained logical unit.

Chapter 3

Pragmatic Patterns

The semantic structures capture ways in which we might manipulate commitments that are determined based on the commitment semantics. The pragmatic patterns, in contrast, represent the different optional or 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 it to a specified authority, (2) of a delegated commitment by complaining to the original debtor, and (3) of an assigned commitment by informing the original creditor. We can thus identify three protocol patterns above that the designers could apply in the given domain in different settings. In general, there is a huge variety of such patterns and nowhere should one assume that a particular pattern must be applied. Such 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 from different perspectives. 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. These operational details apply closer to the implementation than does Muon. 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 some patterns that highlight the 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 suitable 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, here we seek patterns that apply to protocols. Thus, the Muon patterns go beyond Singh et al.'s approach by describing how communications among the participants accomplish the corresponding business-level interaction.

We now introduce some important pragmatic patterns. Pragmatic patterns are potentially of unbounded variety here: disparate patterns appropriate for different domains. For each pattern, we provide six key attributes, loosely based on Gamma et al. (1995).

3.1 Accommodating Cancellation

A unilateral cancellation of a commitment would enable a debtor to simply exit midstream from an interaction. As an autonomous party, it can do so. However, since such an exit is usually not desirable, a protocol may include support for a jilted creditor to sanction a failing debtor, possibly by sending an *escalation* message to an appropriate authority, which might (or might not) proceed to sanction the errant debtor.

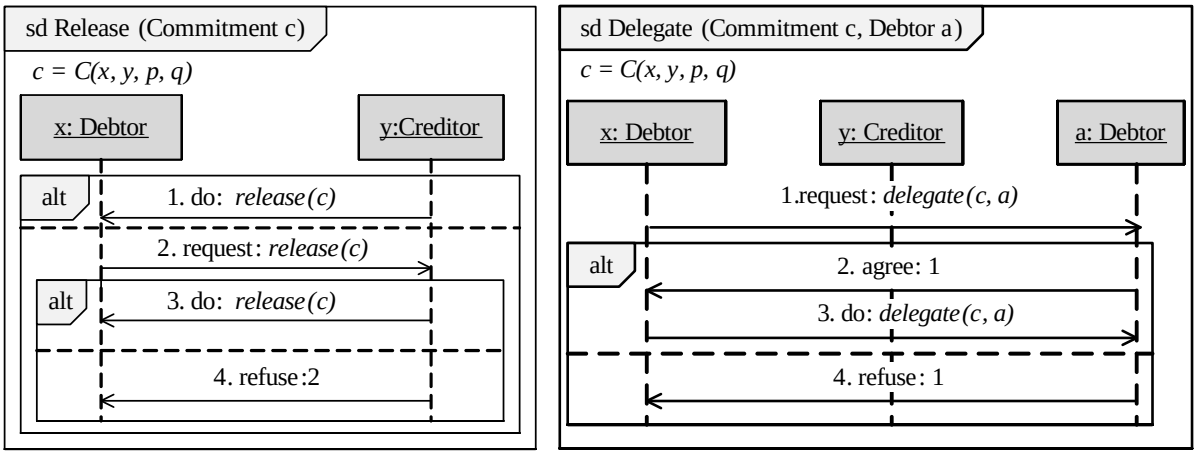
- *Intent*: The debtor no longer wishes to continue the commitment.
- *Motivation*: A buyer (debtor) cancels its commitment with its seller (creditor). On cancellation, the buyer returns any goods that the seller has already delivered.
- *Applicability*: When a buyer changes its decision of buying goods.
- *Consequences*: The debtor can cancel its commitment at will. The creditor may attempt to impose sanctions on the debtor, such as by escalating a complaint to a higher authority if such exists.
- *Implementation*: The commitment enters a terminal state in its life cycle.
- *Known uses*: RosettaNet's (2009) Request Shipping Order Cancellation (PIP 3B14), Request Purchase Order Cancellation (PIP 3A23).

3.2 Accommodating Release

A release is the converse of a cancel. As in Figure 3.1a, 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 creditor relieves the debtor from a commitment.

- *Motivation*: A buyer (debtor) rejects an offer from a seller or counters with another offer. The seller need no longer honor the original offer.
- *Applicability*: In negotiation, the parties may reject each other's offers. Further, in e-commerce, when a buyer finds that the goods received are damaged, it can return the goods and request the release of the commitment to pay for them.
- *Consequences*: The creditor can release its commitments at its will or when the debtor requests the creditor for releasing the commitments.
- *Implementation*: The commitment enters a terminal state in its life cycle.
- *Known uses*: HL7's [ISO/HL7, 2009] Cancel Admission (A11), Cancel Transfer (A12), and Cancel Discharge (A13) involve releasing a commitment.



(a) Release. (b) Delegate.

Figure 3.1: 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 3.1b. Here, the delegator requests the delegatee and only upon its agreement performs the delegation.

- *Intent*: When a debtor delegates its commitment to another debtor and stays responsible for the satisfaction of the commitment.
- *Motivation*: The seller (debtor) delegates its commitment of delivering goods to a shipper. However, the seller remains committed to delivering the goods to the buyer (creditor). When the shipper delivers the goods, its commitment as well as the seller's pending commitment are both discharged.
- *Applicability*: When a creditor does not have the capability to perform its committed task, it delegates its task to a third party, the delegatee.
- *Consequences*: This pattern presumes that suitable social relationships (other commitments) are in place between the delegator and the delegatee so that the delegation is successful.
- *Implementation*: When the creditor delegates its commitment to a delegatee, the original commitment becomes pending and a new commitment is created. Once the new commitment is discharged, the original commitment is also discharged.
- *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, RosettaNet's (2009) Request Shipping Order (PIP 3B12) and Notify of Shipping Order Confirmation (PIP 3B13) reflect the delegate pattern.

The explicit request would be redundant in cases where the delegator and delegatee have a prior organizational relationship that grants the delegator the power to perform the delegation. A useful variation of the above 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.

Notice that the above pattern does not describe the situation where the delegator requests permission from the creditor before proceeding with the delegation. Such a permission would be relevant only in cases where the delegator is additionally committed to the creditor to not perform the delegation. In such a case, the delegator may request the creditor to release it from the second commitment. Thus, we can accommodate the effect of seeking permission from a creditor simply via a composition of the above release and delegation pragmatic 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.

Chapter 4

The Muon Methodology

We now systematically describe the Muon methodology to guide designers of a cross-enterprise protocol. It presupposes expertise in the domain of the interaction being modeled, especially in capturing the meanings of a message. To illustrate our methodology, we adopt the AGFIL case introduced above. Figure 4.1 represents the flow of our overall methodology.

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.

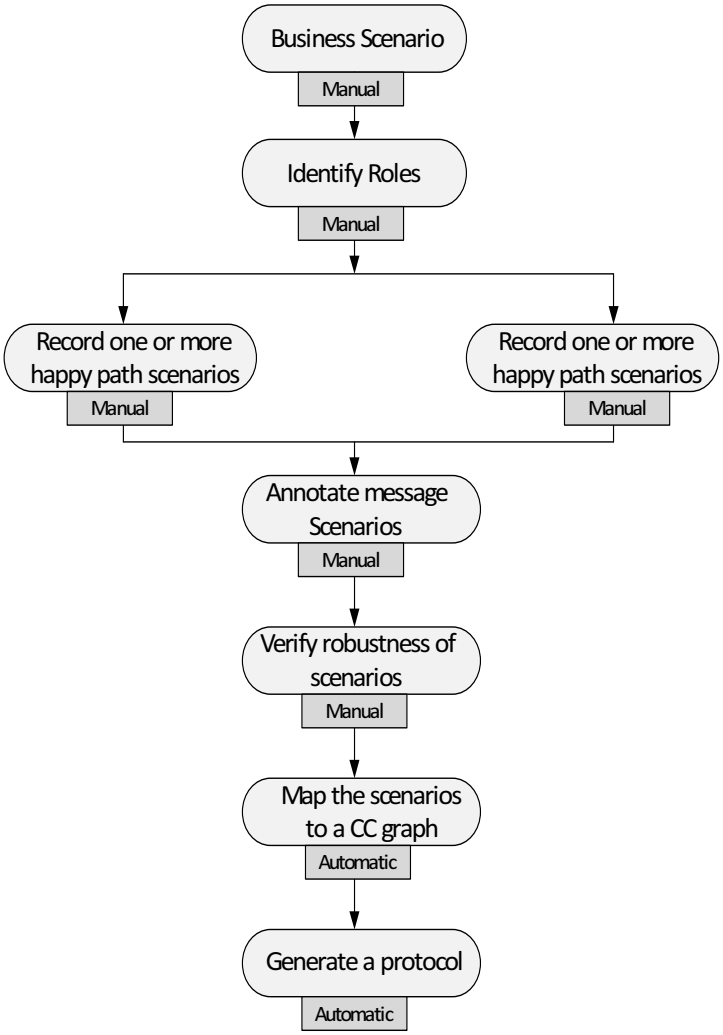


Figure 4.1: Our methodology summarized. The last two are automatic steps but we include them here to provide a complete picture of our approach.

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). This step follows the CRC methodology [Beck and Cunningham, 1989], wherein experts play act programming objects to be created. For example, an expert could play act a Bank Account object.

Our approach 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.

In this step, the experts record a suitable scenario.

- *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 primary scenario must be an exemplar. The role enactment must aim to reveal the most central instance of the interaction while also revealing important variations as secondary scenarios of reasonable complexity, as the following guidelines suggest:

- *Represent a positive outcome.* For example, we seek a scenario in which John Doe's car is repaired and all payments are made. Such a scenario serves to capture desirable enactments and eliminate trivial ones that might not provide sufficient information to engineer a protocol. Specifically, a scenario in which the mechanic simply refuses to repair the car is less useful at this stage.
- *Reflect the social or organizational relationships among the roles.* 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 messages that do not add to an observer's knowledge of the interaction.* In particular, we omit messages pertaining to other transactions, such as the mechanic repairing cars of customers unrelated to AGFIL. Doing so helps avoid spurious messages in the resulting protocol.
- *Omit roles that do not add to an observer's knowledge of the interaction.* Multiple instantiations of a role—two different mechanics, for instance—must not be introduced into

the enactment unless they add value to an observer's knowledge of the interaction. If Europ Assist getting estimates from multiple mechanics were relevant, we would show such mechanics, but otherwise not.

Table 4.1: 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	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

Table 4.1 shows a typical scenario, as might be recorded from an enactment session [Parunak, 1996]. The ID column represents the sequence in which the messages occur.

4.3 M3: Record One or More Scenarios Demonstrating Exceptions

Here, the domain experts identify one or more scenarios corresponding to exceptions.

- *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 4.2: A scenario demonstrating an exception in the AGFIL example.

#	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 4.2'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 4.1. The merged scenarios are provided in Table 4.4(I) and (II).

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.

Table 4.3: I-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.)

#	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)

Table 4.4: II-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.)

#	S	R	Content	A	P	U	C	Operation
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'	2a				
25	AG	JD	Take back the car from M2				2a, 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)

Let us illustrate the message annotations. 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 1 and 3, where AG provides the repaired car when JD claims it.

4.4.1 M5: Verify Robustness of Scenarios

- *Input:* The annotations of Step M4 along with 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.

Real-world business interactions often exhibit modularity in that they naturally map to multiple self-contained scenarios. Because our approach, groups messages that create, progress, or complete a given commitment, we can determine whether a given scenario includes all operations that affect a commitment logically. If not, we can help the designers come up with additional scenarios to ensure completeness.

Our approach lends itself to determining and addressing possible exceptions and opportunities based on the commitments of the parties involved. Representing commitments facilitates

verifying a target protocol for robustness. Specifically, when designing such a protocol, 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 arise naturally from the commitment life cycle we have adopted. They can be thought of as sanity checks on the given set of scenarios. Checks that are similar in spirit are applied in other representations as well: for example, one might check if a workflow handles the case when a condition may be false. Notice that this step does not capture advance messages because those are not constrained by the model.

In general, determining and addressing exceptions involves iterating 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 our approach does not demand whether and how designers address exceptions. In particular, determining a suitable sanction for a particular domain requires some imagination and application-specific knowledge.

In Table 4.4, JD never pays the insurance premium. Therefore, in Table 4.4, we need to add Message 2a, which completes a commitment between JD and AGFIL. Similarly, in Table 4.4, LCS never pays M for the estimates and repairs. Hence, we need to include Message 22a, which completes the commitment between LCS and M.

Table 4.5: Completeness check for the happy path for the AGFIL case.

#	S	R	Content	A	P	U	C	Operation
2a	JD	AG	Pay premium p		1		2	$\det(C_1)$, $\text{dis}(C_2)$
22a	LCS	M2	Here is the payment for performing the repairs				21	$\text{dis}(C_9)$

In Table 4.2, AG never pays for the estimate it obtains from M2. Therefore, we add Message 18a where AG makes the payment for the estimate, thereby discharging its commitment with M2 as well as discharging the commitment from LCS to M2.

In this step, we identify the incompleteness and iterate Steps M2 and M3. As Table 4.5 shows, Step M2 now yields Message 2a, which completes the commitment to pay the premium by discharging it, and Message 22a, which completes the detached commitment from AG to M. And, as Table 4.6 shows, Step M3 yields Message 18a, which completes the commitment by

Table 4.6: Completeness check for exceptions for the AGFIL case.

#	S	R	Content	A	P	U	C	Operation
18a	AG	M2	Here is the payment for the estimate				16, 15h	dis(C_8), dis(C_7)

discharging it.

4.5 M6: Map the Scenarios to a Commitment-Causal Graph

This step is automatic and not carried out manually by the designers.

- *Input:* The output of Step M4.
- *Output:* A commitment-causal (CC) graph of the scenario, identifying semantic structures in the interaction.
- *Description:* Generate a CC graph from the annotated scenario based on producing some intermediate graph representations.

- From Step M4, we obtain a complete scenario containing the happy path and exception scenarios as in Table A.2 (included in the appendix).
- Identify one or more *conversation graphs* from the scenario.
  - For convenience, interpret an annotated scenario as a graph whose vertices are messages (associating a sender (S), receiver (R), message (ID), and message meaning) 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_j to m_i.
  - A conversation graph G is a minimal graph (in its set of vertices) that contains at least one message and contains all the messages that relate via progress or completion (and hence also update) to any message that occurs in G. Additionally, the messages in a conversation graph are partially ordered such that if m_j progresses, completes, advances, or updates m_i, then m_j is ordered after m_i. In intuitive terms, a conversation graph is a logically coherent scenario fragment.

    For example, Message 1 forms a conversation graph with Messages 2a and 25. These are ordered Message 1, then Message 2a, then Message 25.

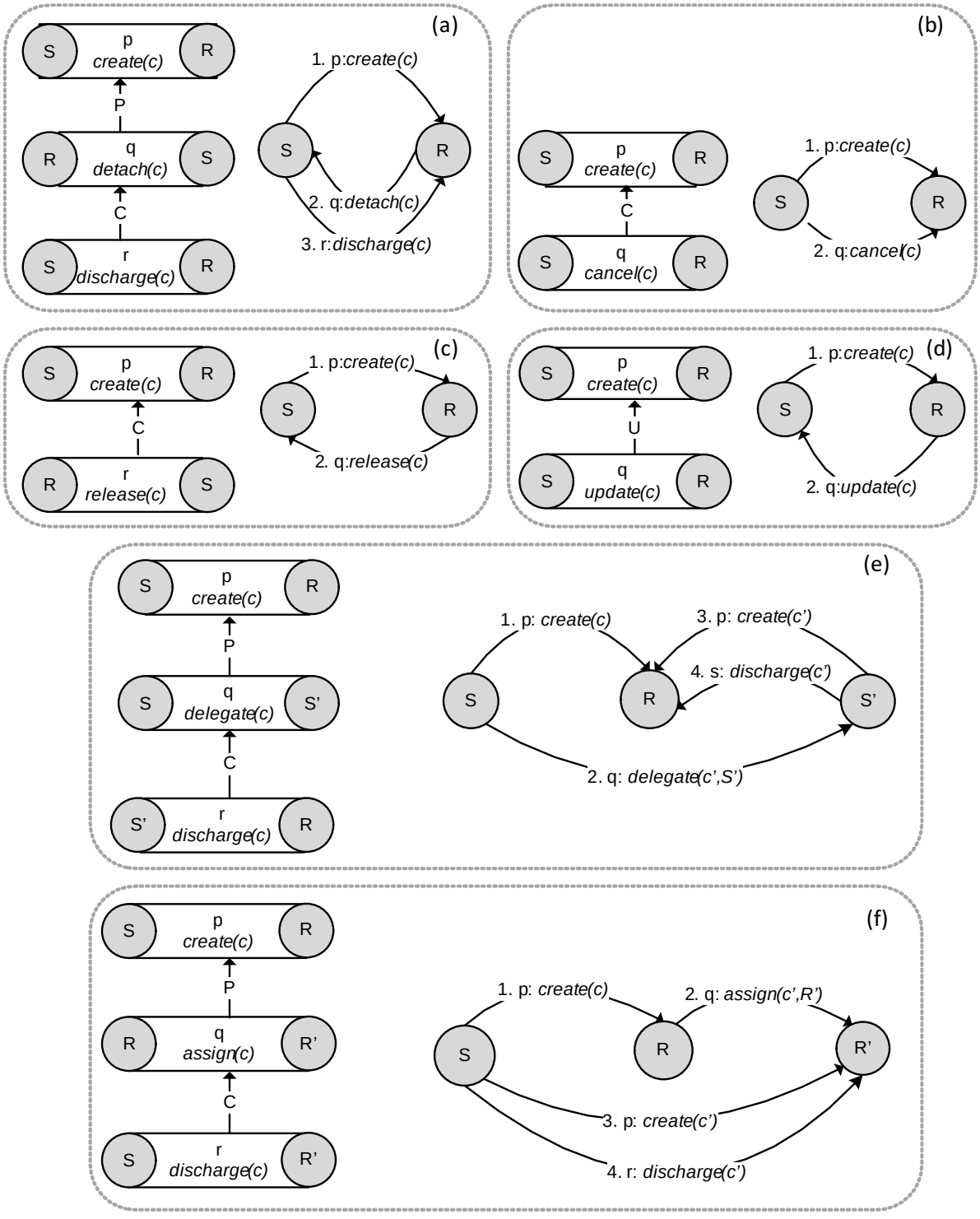


Figure 4.2: Algorithm: mapping conversation fragments to CC graph components. Each box shows a conversation fragment and the resulting CC graph component. The numbers on the CC graph components indicate ordering.

- Generate the CC graph of the scenario from its conversation graphs. The senders and receivers of messages become the vertices in the CC graph and the messages become the edges between the vertices. In a CC graph, the edges carry the meaning of commitment operations and are causally ordered. Figure 4.2 shows our modular algorithm for creating a CC graph, wherein it identifies the key roles involved in each component of a CC graph. Our algorithm applies to the appropriate fragments of a conversation graph and yields the corresponding CC component.

Importantly, we keep the roles identified in each conversation graph separate. Such split off roles are termed *characters* [Parunak, 1996]. Keeping the characters separate because we wish to modularize the protocol being produced. Thus a typical CC graph would contain multiple connected components. Each of the connected components originates from a conversation and yields a modular protocol. 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.

Figure 4.2 provides several patterns for the CC graph. For example, Figure 4.2(a) represents the pattern for commitment creation and discharge. Similarly, Figures 4.2(b), 4.2(c), and 4.2(d) represent how a commitment is canceled, released, or updated. Figures 4.2(e) and 4.2(f) represent how a commitment is delegated or assigned. One can verify by inspection that these patterns cover the life cycle of a commitment discussed in Section 2.2.5 and the pragmatic patterns introduced in Section 3. Therefore, any other pattern would merely be a variation on the above patterns. For example, canceling a commitment after it is detached is a variation of Figure 4.2(b) wherein the commitment is canceled after the commitment is created. Similarly, the cancellation of a commitment after it is delegated or assigned is a variation of Figures 4.2(e) and 4.2(f), respectively, wherein a commitment is discharged after it is delegated or assigned. Figure 4.3 shows the CC graph for the AGFIL case.

4.5.1 M7: Generate a Protocol

This step is automatic and not carried out manually by the designers.

- *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.

Identify semantic and pragmatic patterns that match each of the components of the CC graph. For instance, the graph consisting of Messages 1, 2a, and 25, Figure 4.3 corresponds to

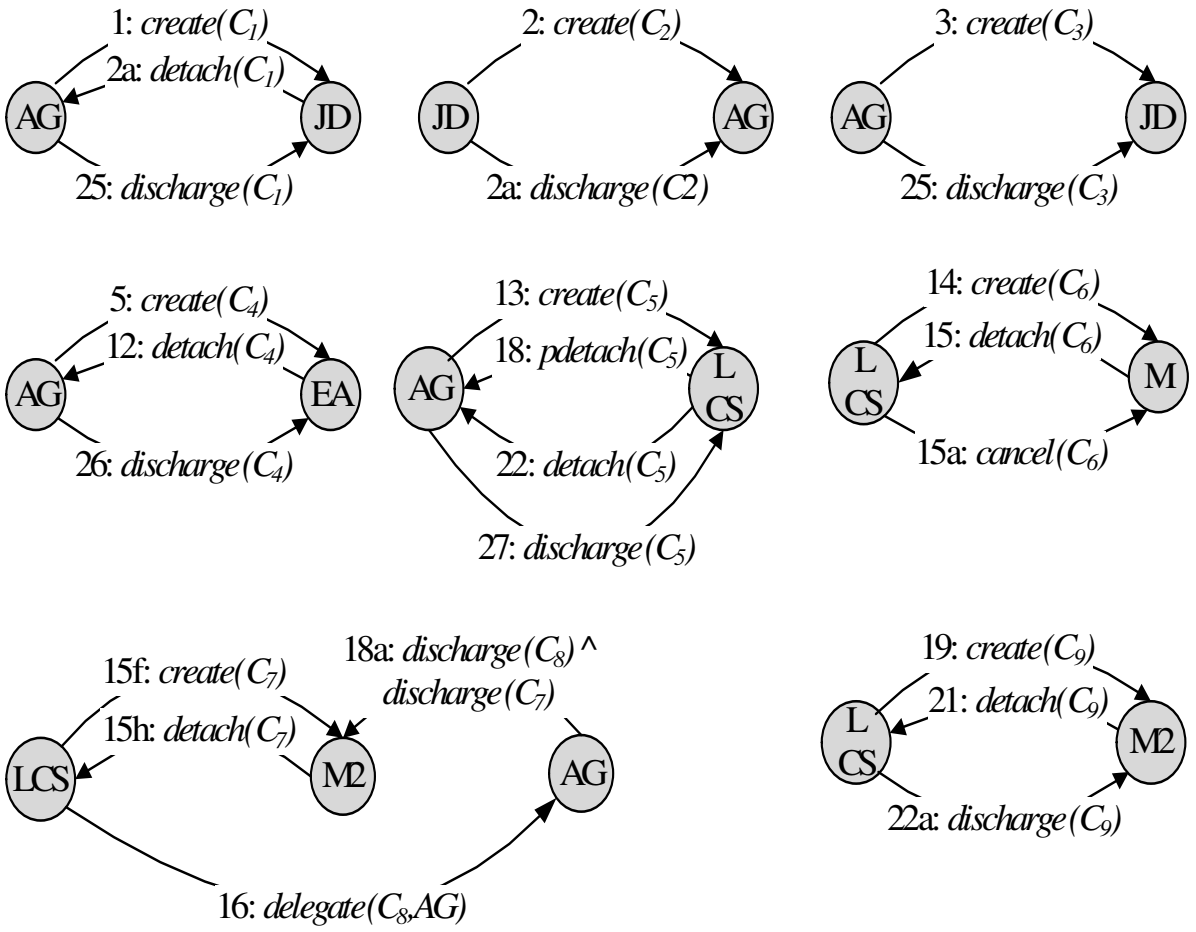


Figure 4.3: The CC graph for the example of Table A.2.

the conversation initiated by Message 1 and matches a composition of the Create, Progress, and Complete structures introduced in Section 2.2. Figure 4.4 shows the protocols generated based on the CC graphs shown in Figure 4.3.

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).

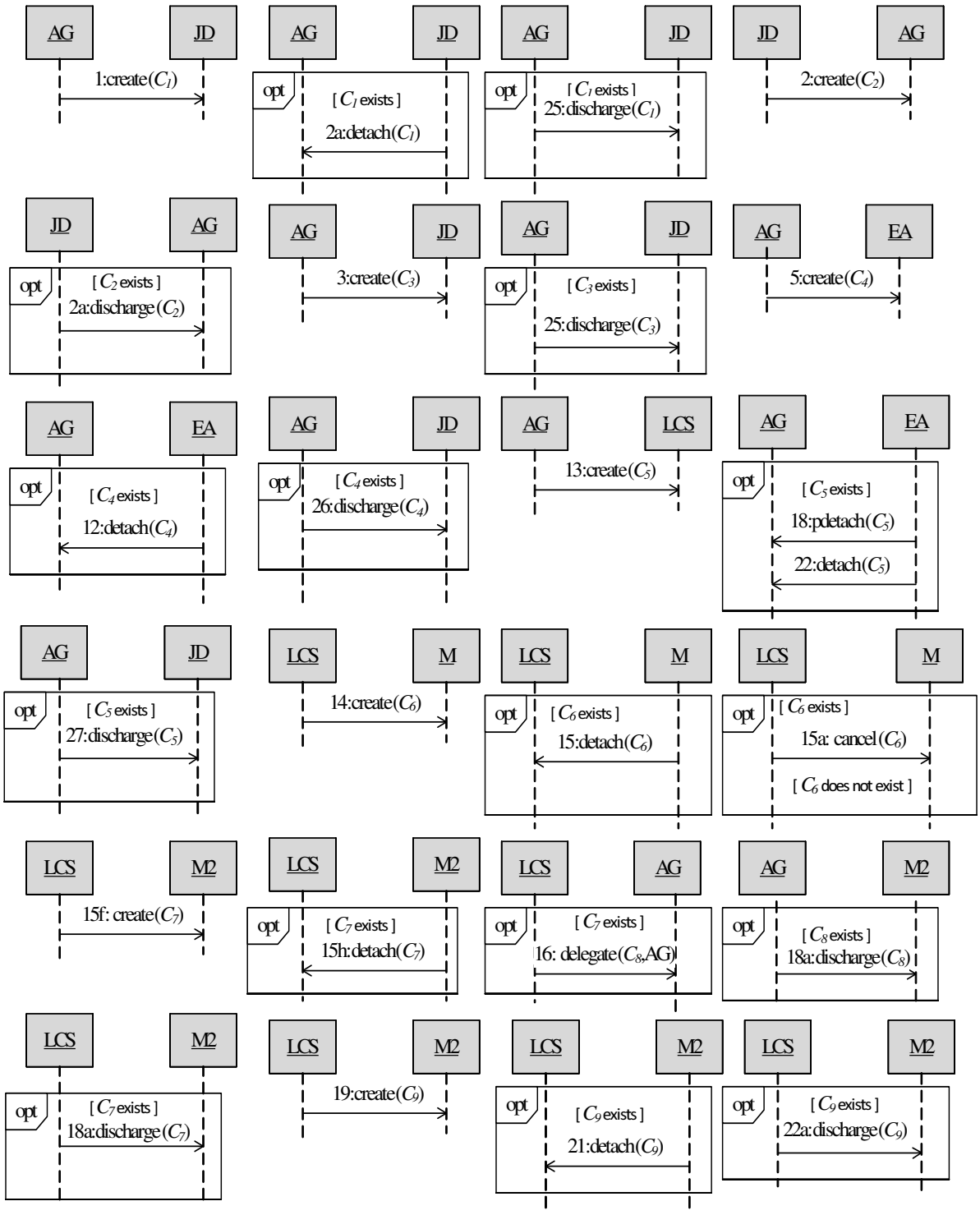


Figure 4.4: The generated protocol for the AGFIL case.

- Create a commitment from AG to M: C(AG, M, repair, pay).
- As needed, each of the above commitments may be delegated, assigned, canceled, released, or updated.

Chapter 5

Empirical Evaluation

We now describe a partial empirical evaluation that compared two approaches for creating a protocol: Muon and a traditional approach (dubbed *Trad* below) based on UML 2.0 Sequence Diagrams Object Management Group [2004]. AUML [Odell et al., 2001] introduced advanced sequence diagrams, adopted in UML 2.0, which match the notation we used for semantic structures above. However, AUML and UML 2.0 provide no account of meaning underlying such diagrams, which we define. Further, we provide a methodology for developing protocols.

The idea of the study was to compare the quality of the protocols produced by the two approaches. For each approach, the output was a protocol expressed in UML Sequence Diagrams restricted to messages (no method calls). And, the input was a textual description as well as happy path and exception scenarios. (The appendix shows the provided inputs.)

- **Trad:** We asked subjects to combine the scenarios to create a final scenario; check its completeness with respect to the provided description; create a protocol.
- **Muon:** We asked subjects to create a protocol using Muon, described as slides (http://research.csc.ncsu.edu/mas/code/chor/Methodology.pdf) outlining the steps.

The subjects were 58 computer science graduate students. 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 an equal mean industry experience, which we obtained via a survey conducted prior to initiating the study.
- **Learning effect**, wherein a subject learns the domain while applying the first method and therefore performs better while applying the second method. We considered two case

studies: (1) the AGFIL case and (2) the ASPE case of breast cancer diagnosis [ASPE, 2010]. First, Group A designed a protocol for AGFIL and Group B for ASPE using the traditional method. Next, the groups applied our methodology, though with cases switched (Group A on ASPE and Group B on AGFIL).

- **Instrumentation** refers to the effect of tooling for each method. All the subjects used IBM Rational Software Architect 8.0.2 for developing UML sequence diagrams.

Statistical Measures

We measured the following dependent variables for each approach, and compared them via statistical significance tests.

- **Flexibility** measures the number of possible executions.
  - *MSC count* indicates modularity (higher is better).
  - *Count of* ALT*,* OPT*, and* PAR fragments indicating the number of possible executions (higher is better).
- **Objective quality** measures model quality as assessed in objective terms.
  - *Missing number of guards* on UML Sequence Diagram fragment, which indicate spurious enactments (lower is better).
  - *Errors in MSCs*, indicating an attempt to tackle increased complexity of the MSCs (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.

5.1 Results

We write the traditional approach as Trad and our approach as Muon. From among the 58 subjects who took part in the study, 10 subjects were familiar with the notion of commitment from previous modeling exercises. 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.

5.1.1 Flexibility

Figure 5.1 shows boxplots of flexibility, as based on objective criteria. The median of the count of ALT, OPT, PAR in Muon is higher (11) than in Trad (6), indicating greater flexibility in Muon than Trad. The median number of MSCs 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, because of our experience in another settings wherein modelers tended to produce monolithic diagrams. Notice that the modularity of the Muon solution is encouraged by our method, without special attention by the developer. The median count for Trad-novice (6) is greater than Muon-novice. The median count for Trad-expert (4) is same as for Muon-expert (4).

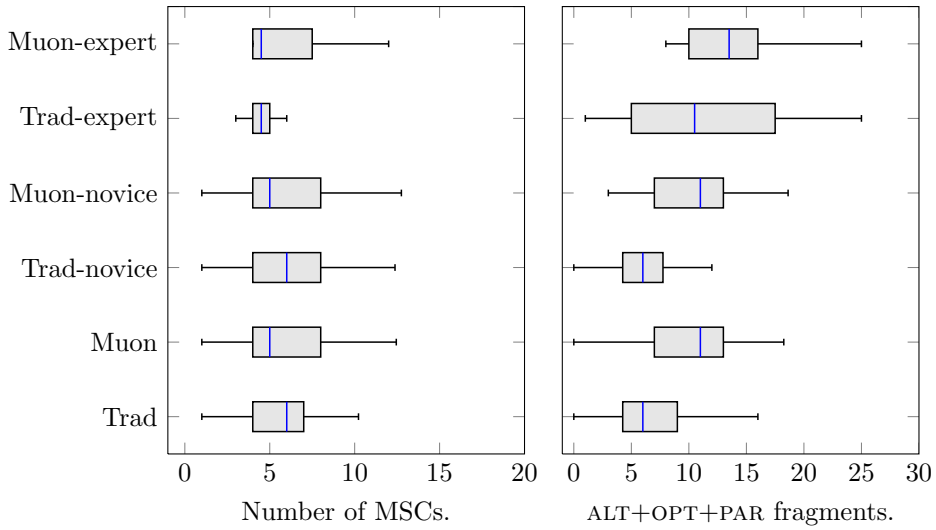


Figure 5.1: Flexibility, as judged based on objective criteria.

5.1.2 Objective Quality

Figure 5.2 (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 5.2 (right) shows the boxplots representing the numbers of errors in MSC 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.

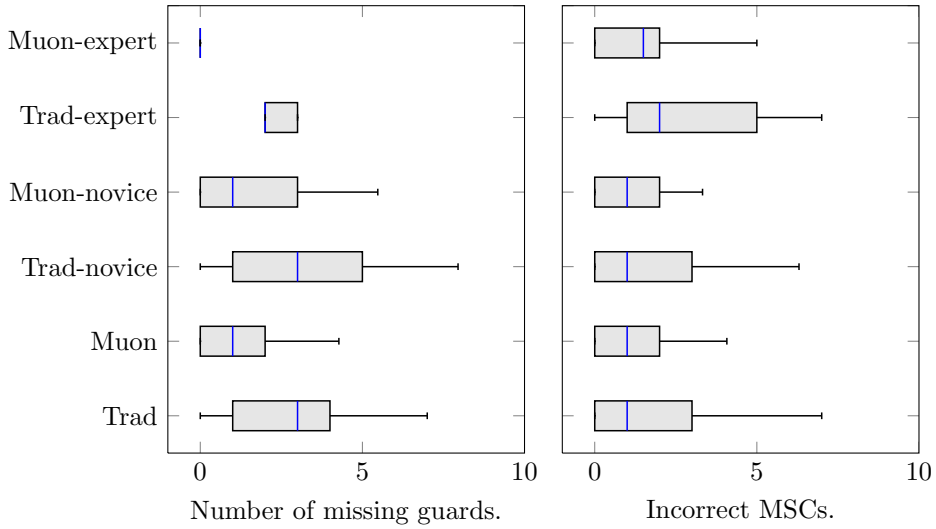


Figure 5.2: Objective measures of quality.

5.1.3 Subjective Quality

Figure 5.3 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.

5.1.4 Hypotheses

We now present the claims regarding the comparative effectiveness of Muon and Trad. The following are our alternative hypotheses, which claim that the means in the Muon and Trad distributions of the referenced variables are different.

- *Muon yields higher mean MSC count.* The mean number of MSCs generated using Muon is higher than for Trad. That is, Muon produces protocols of greater modularity than does Trad.
- *Muon yields higher mean count of* ALT*,* OPT*,* PAR *fragments.* The mean sum of ALT, OPT, and PAR fragments in Muon MSCs is higher than for Trad. That is, Muon yields more flexible MSCs than Trad.
- *Muon yields lower mean count of missing guards.* The mean number of missing guards in a MSC using Muon is smaller than using Trad. That is, Muon produces better structured

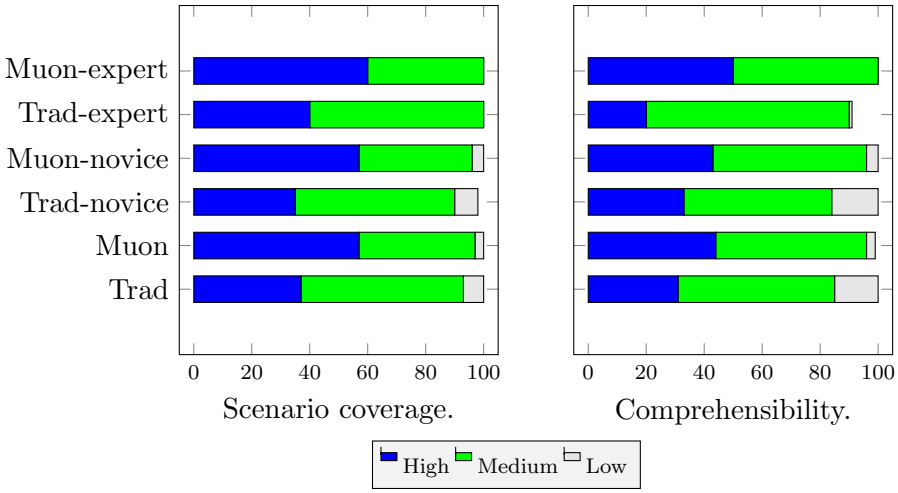


Figure 5.3: Distributions of quality, as judged subjectively.

protocols than Trad.

- *Muon yields lower mean count of errors in MSCs.* The mean number of errors using Muon is smaller than using Trad. That is, Muon produces fewer incorrect protocols than Trad.
- *Muon yields higher mean percentage of scenario coverage.* The mean percentage of scenario coverage is higher using Muon than using Trad. That is, subjects model different aspects of a business scenario more in Muon and Trad.
- *Muon yields higher mean percentage of comprehensibility.* The mean percentage of comprehensibility is higher using Muon than using Trad. That is, subjects models are more comprehensible for Muon than Trad.

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 5.1 shows the p-values for the above hypotheses. It shows we reject all null hypotheses for ALT, OPT, PAR fragments count, missing guards count, and comprehensibility percentage. That is, we find that subjects employing Muon produce more flexible MSCs, with fewer missing guards, and comprehensible MSCs than subjects employing Trad. We accept null hypothesis for MSC count, incorrect MSCs count, and for the percentage of scenario coverage. That is, the number of MSCs and the percentage of scenario coverage are not significantly higher and the count of incorrect MSCs is not significantly lower among the Muon subjects.

In addition, we performed the chi-squared test on the percentage of scenario coverage and comprehensibility. We found that the p-values for the percentage of scenario coverage

Table 5.1: Hypothesis testing.

Hypothesis	**Muon Mean** (μ_m)	**Trad Mean** (μ_t)	**Alternative Hypothesis**	**Null Hypothesis** [$\mu_m = \mu_t$] **p-value**	**Accepted at p-value 5%?**
Muon yields higher mean MSC count	5.94	5.35	$\mu_m > \mu_t$	p=0.14	✓
Muon yields higher mean count of ALT, OPT, PAR fragments	10.21	6.93	$\mu_m > \mu_t$	p=0.001	×
Muon yields lower mean count of missing guards	1.26	2.89	$\mu_m < \mu_t$	p=0.001	×
Muon yields lower mean count of incorrect MSCs	1.21	1.16	$\mu_m < \mu_t$	p=0.81	✓
Muon yields higher mean percentage of scenario coverage	84.03	77.36	$\mu_m > \mu_t$	p=0.24	✓
Muon yields higher mean percentage of comprehensibility	82.59	77.41	$\mu_m > \mu_t$	p=0.02	×

($5.1100e^{-10}$) and for the percentage of comprehensibility ($6.4482e^{-04}$) are extremely low. That is, the percentage of scenario coverage and comprehensibility both in Muon and Trad are not independent. This means that the measure of subjective quality for the subjects are not entirely different in Muon and Trad.

5.1.5 Threats to Validity

We briefly review the threats to validity of the above study. 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 developers: most have less work experience and were required to try out Muon in this study.

To control for the fact that the subjects are not experts in the chosen domain, we provide our subjects multiple scenarios corresponding to the 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.

Chapter 6

Discussion

Protocols have been used in cross-organizational settings for years. For example, RosettaNet (2009) originated in the 1990s and is a widely deployed standard that describes interactions for realizing supply chains. This work 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. These domains are architecturally well-suited to multiagent systems, though advances such as Muon are needed to help bridge the gap in a way to bring the benefits of multiagent systems to such domains.

Muon is based on semantic structures that, unlike other approaches, relate naturally to meanings captured via commitments. Our high-level representation facilitates the partners flexibly implementing their internal policies and provides a meaningful basis for judging the correctness of a protocol. We go beyond previous work on commitments in expressing a core set of meaning-based and causal relationships among messages. As a result, Muon emphasizes meaning and deemphasizes operational details such as message ordering.

Capturing scenarios is a natural way to elicit stakeholder requirements and to provide a design rationale for the protocol. This is based on our intuition, reflected also in work on scenarios [Filippidou, 1998], that a bottom up approach is simpler: the designers first imagine some messages in a scenario; make the meanings of those messages explicit as commitments; use the robustness criteria to expand the scenarios; capture the causal structures, eventually producing a protocol. In other words, the concreteness of the scenarios helps in coming up with the commitments.

Notice that the primary work done by the designers is in creatively coming up with the scenarios; understanding and semantically annotating them; applying sanity checks and deciding how to expand scenarios; and converting them into a protocol. For each pragmatic pattern, we provide six attributes of which intent, motivation, applicability, and consequences go toward

helping decide which pattern to apply. The semantic structures are used to generate modular protocols and are also used in Step M4 as a way for performing a sanity check.

Muon respects the intuition that mutually independent or nonoverlapping scenarios map to separate protocols. An interaction scenario I_1 that is causally affected by a message in another scenario I_2 would yield a protocol that hangs off the protocol for I_2. Thus, the advance relationship would lead to such causally connected protocols, which 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, which might lead the recipient of the message to create a new commitment, reflecting compensation for the creditor. There is thus a natural causal (and hence temporal) relationship between the cancellation and the escalation messages.

Protocols, in general, do not arise in a vacuum. The partners who specify and adopt a protocol would often have relevant internal processes already in place. Thus the choice of a protocol may be influenced by existing internal processes. However, the key idea is for a protocol not to expose such internal processes of one partner to another—doing so couples the partners' implementations and makes it harder to evolve any of them independently of the others.

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].

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 some kind of synchronization or some kind of an authoritative party that can adjudicate such matters. 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. Notice that our strong causal approach to patterns provides a natural way to create protocols that would avoid problems such as race conditions.

6.1 Assessment of our Approach

A benefit of Muon is that the protocol it yields is modular in terms of the commitments involved in the given case. This protocol supports flexible interactions among the partners. It can be readily observed that Muon's structured protocol, as in Figure 4.3, is superior to a monolithic solution, as produced by the CrossFlow project [Browne and Kellett, 1999], from which the AGFIL case is taken. The CrossFlow model, is in essence the same as a BPMN [OMG, 2010]

model: it shows where the various internal flows interact with one another but does not cleanly separate the internal implementation from the interactions. Further, it does not represent the meanings of the interactions and does not structure them. As a result, the CrossFlow model for AGFIL is neither robust nor extensible because it does not handle exceptions (e.g., cancellations) and opportunities (e.g., delegations).

More generally, a language such as BPMN whose representation is at the operational level (e.g., tasks and messages) does not natively represent social relationships among the partners. Using such a language, a designer would be able to deal with exceptions and opportunities in BPMN, but only by reducing them to the correct operational details. Such reductions are nontrivial to produce and difficult to maintain. Designers sometimes become over-cautious in such settings and capture nearly serial execution paths.

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 it does not highlight the causal and meaning-based relationships we have emphasized here. 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.

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.

A second limitation is that annotating scenarios requires manual effort. 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.

6.2 Related Work

We now review the works most relevant to the design of multiagent protocols from scenarios.

Conversation-Based Approaches

Muon extends Parunak's (1996) work on deriving Dooley graphs from a scenario. Wan and Singh (2003) relate communications to a framework for manipulating commitments through Commitment Causality Diagrams (CCDs), which depict the causal relationships between operations on various commitments created in a scenario. Their treatment, however, creates processes for specific scenarios, not reusable protocols that can be recombined in novel ways to

demonstrate valuable alternative interactions. Further, Wan and Singh assume that operations on commitments are preceded by explicit proposals, which restricts the autonomy of agents.

Muon differs from the CRC methodology [Beck and Cunningham, 1989] and scenario-based design [Filippidou, 1998] in an important respect. In these approaches, a stakeholder might act as a business object, such as a bank account. By contrast, in Muon, the stakeholders play roles that correspond to autonomous entities, e.g., real-life business partners, not to programming objects. In conceptual terms, the main difference between partners and programming objects is that partners are autonomous whereas objects are not. Partners can decide how to behave internally and can initiate interactions proactively. They can ignore or deny requests whereas objects must respond to any requests they receive from their clients. Thus programming objects are precisely the wrong metaphor for describing autonomous partners as they carry out their service engagements. An additional distinct facet of Muon is in how it uses the scenarios, which is to create annotations based on the semantic structures based on commitments.

6.2.1 Commitments

The notion of commitments has drawn attention from several researchers. We adopt the specific proposals that emphasize the autonomy of the participants and support both the logical (declarative) and the operational nature of commitments [Fornara and Colombetti, 2003; Singh, 2008].

Commitments have previously been applied in understanding communications, singly or in protocols [Verdicchio and Colombetti, 2003; Fornara and Colombetti, 2003; Flores et al., 2007]. In broad terms, the works of Colombetti and colleagues share our intuitions. A key difference is that 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, 2002a], rules [Desai et al., 2005], UML's object constraint language (OCL) [Robinson and Purao, 2009], and nonmonotonic causal logic [Desai et al., 2007].

Baldoni et al. (2010) relate commitments to organizational constraints or *regulations*. One key difference between such constraints and commitments is that the commitments make the responsible party explicit as the debtor, whereas constraints presume some sort of an architectural entity that would enforce constraints. Such an entity is not always clear in a decentralized system. Commitment protocols provide a flexible representation of interaction among agents. They are either operationalized directly via a logic-based or rule-based approach such as those listed above or via logic-based commitment machines, which can be translated into finite-state machines [Winikoff et al., 2005]. Carabelea and Boissier (2006) show how to design a model that represents the expected behavior of an agent using social commitments. The model can

be used as a basis for rewarding or punishing an agent if it fulfills or violates its commitments, respectively.

Winikoff (2006) proposes an approach for designing agent interactions using commitment machines Yolum and Singh [2002b]. His approach 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. Winikoff further inserts conditions and effects on the steps and their sequence to ensure "reasonable" executions 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.

Other work on commitments has considered formalizing existing protocols, but not a means to develop commitment protocols specific to an application. Flores and Kremer (2004) apply the *protocol for proposals* to propose every commitment operation, thus reducing the autonomy of the participants. They formalize the protocol using the Z notation, and use obligations to denote which agents' turn it is to perform some action in response to some another action. Such obligations, however, are inimical to autonomy. The above works address runtime state maintenance, not the creation of a specification. Further, they ignore three-party operations, such as delegate and assign.

Chopra and Singh (2011) propose a number of commitment-based business patterns. However, they are mainly concerned with identifying the patterns and antipatterns corresponding respectively to appropriate or inappropriate uses of commitments. They do not provide a methodology for eliciting protocol requirements and do not relate them to the design patterns, as we have sought to do here.

6.2.2 Other Frameworks

There are several agent-oriented software methodologies and frameworks, of which some leading ones include Bresciani et al. [2004]; Cossentino and Potts [2002]; DeLoach [2004]; Dignum et al. [2005]; Padgham and Winikoff [2005]; Wooldridge et al. [2000]. These approaches model agents and multiagent systems using concepts such as goals, roles, organizations, and norms. They do not provide specific guidelines for capturing stakeholder requirements based on explicit scenarios as we have shown in this work. In principle, Muon could be expanded to accommodate more of the constructs that the above approaches use. However, to do so, one would need to surmount two challenges. One, Muon is focused on interactions and so internal notions such as a goals would need to be provided separately. Two, Muon relies upon the commitment life cycle and

semantics: something analogous would be needed for each new construct to which Muon is expanded.

The following approaches involve concepts that are somewhat related to commitments. McCarthy (1982) introduced the Resource-Event-Agent (REA) model as an accounting framework. He treats commitments as economic objects that correspond to agreements between agents. Geerts and McCarthy's (2002) treatment of commitments as a core construct and their descriptions of it appear to agree with our intuitions. However, they do no provide the technical details that can lead to operational patterns, such as we present here. Gordijn and Akkermans (2003) introduce the e^3-value approach for modeling a business requirement as a collaboration between multiple actors where they exchange economic value with each other. However, their model is stated at a conceptual level and does not support formalization and operationalization into protocols.

6.3 Directions

This work opens up some important research directions. Specifically, one, it would be important to support Muon from within the tool for commitment-based modeling [Telang and Singh, 2012] that we have been developing. And, two, it would help to develop a more extensive pattern library corresponding to the situations that arise in business applications, especially those dealing with exceptions.

In some settings, it is possible to automatically annotate textual communications between participants in human-intensive service engagements, such as help-desk interactions, with operations on commitments [Kalia et al., 2013]. It would be an interesting challenge to take advantage of such automatic annotations to check whether a particular enactment violated any commitments and whether it can be incorporated into a protocol for flexible interactions.

Another general theme is the incorporation of a wider variety of norms than just commitments. The ideas of several research efforts on formalizing norms, e.g., [Aldewereld et al., 2010; Álvarez-Napagao et al., 2009; Artikis et al., 2009; Fornara and Colombetti, 2009], could be absorbed into our methodology by creating a suitable life cycle, identifying concomitant semantic structures, identifying useful pragmatic patterns, and formulating completeness checks and causal dependencies. We defer such work to the future.

REFERENCES

Huib Aldewereld, Sergio Álvarez-Napagao, Frank Dignum, and Javier Vázquez-Salceda. Making norms concrete. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 807–814, Toronto, 2010. IFAAMAS.

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

Alexander Artikis, Marek J. Sergot, and Jeremy V. Pitt. Specifying norm-governed computational societies. *ACM Transactions on Computational Logic*, 10(1):1:1–1:42, January 2009.

ASPE. The importance of radiology and pathology communication in the diagnosis and staging of cancer: Mammography as a case study, November 2010. 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.

Matteo Baldoni, Cristina Baroglio, and Elisa Marengo. Behavior-oriented commitment-based protocols. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*, pages 137–142, 2010.

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

Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, May 2004.

Sinead Browne and Michael Kellett. Insurance (motor damage claims) scenario. Document D1.a, CrossFlow Consortium, 1999.

Cosmin Carabelea and Olivier Boissier. Coordinating agents in organizations using social commitments. *Electronic Notes in Theoretical Computer Science*, 150(3):73–91, May 2006. ISSN 1571-0661.

Federico Chesani, Paola Mello, Marco Montali, and Paolo Torroni. Commitment tracking via the reactive event calculus. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 91–96, Pasadena, California, 2009. IJCAI.

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

Massimo Cossentino and Colin Potts. A CASE tool supported methodology for the design of multi-agent systems. June 2002.

Scott A. DeLoach. The MaSE methodology. In Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli, editors, *Methodologies and Software Engineering for Agent Systems*, chapter 6, pages 107–126. Kluwer, Boston, 2004.

Nirmit Desai, Ashok U. Mallya, Amit K. Chopra, and Munindar P. Singh. Interaction protocols as design abstractions for business processes. *IEEE Transactions on Software Engineering*, 31(12):1015–1027, December 2005.

Nirmit Desai, Amit K. Chopra, and Munindar P. Singh. Representing and reasoning about commitments in business processes. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI)*, pages 1328–1333, Vancouver, July 2007. AAAI Press.

Nirmit Desai, Amit K. Chopra, and Munindar P. Singh. Amoeba: A methodology for modeling and evolution of cross-organizational business processes. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 19(2):6:1–6:45, October 2009.

Virginia Dignum, Javier Vázquez-Salceda, and Frank Dignum. OMNI: Introducing social structure, norms and ontologies into agent organizations. In *Programming Multi-Agent Systems*, volume 3346 of *Lecture Notes in Computer Science*, pages 181–198. 2005.

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

Roberto A. Flores and Robert C. Kremer. A principled modular approach to construct flexible conversation protocols. In *Proceedings of the 17th Canadian Conference on Artificial Intelligence*, volume 3060 of *LNCS*, pages 1–15, London, Ontario, 2004. Springer.

Roberto A. Flores, Philippe Pasquier, and Brahim Chaib-draa. Conversational semantics sustained by commitments. *Autonomous Agents and Multi-Agent Systems*, 14(2):165–186, April 2007.

Nicoletta Fornara and Marco Colombetti. 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)*, pages 520–527, Melbourne, July 2003. ACM Press.

Nicoletta Fornara and Marco Colombetti. Specifying and enforcing norms in artificial institutions. In *Declarative Agent Languages and Technologies VI, Revised Selected and Invited Papers*, volume 5397 of *LNCS*, pages 1–17, Berlin, 2009. Springer.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Professional Computing Series. Addison-Wesley, Reading, MA, 1995.

Guido L. Geerts and William E. McCarthy. An ontological analysis of the economic primitives of the extended-REA enterprise information architecture. *International Journal of Accounting Information Systems*, 2(1):1–16, March 2002.

Jaap Gordijn and J.M. Akkermans. Value-based requirements engineering: Exploring innovative e-commerce ideas. *Requirements Engineering*, 8(2):114–134, July 2003. ISSN 0947-3602.

Benjamin N. Grosof and Terrence C. Poon. 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*, pages 340–349, 2003.

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

Anup K. Kalia, Hamid R. Motahari Nezhad, Claudio Bartolini, and Munindar P. Singh. Monitoring commitments in people-driven service engagements. In *Proceedings of the 10th IEEE International Conference on Services Computing (SCC)*, pages 1–8, Santa Clara, California, 2013. IEEE Computer Society.

Thomas W. Malone, Kevin Crowston, and George A. Herman, editors. *Organizing Business Knowledge: The MIT Process Handbook*. MIT Press, Cambridge, MA, 2003.

William E. McCarthy. The REA accounting model: A generalized framework for accounting systems in a shared data environment. *Accounting Review*, 57(3):554–578, July 1982. ISSN 0001-4826.

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

James Odell, H. Van Dyke Parunak, and Bernhard Bauer. Representing agent interaction protocols in UML. In *Proceedings of the 1st International Workshop on Agent-Oriented Software Engineering (AOSE 2000)*, volume 1957 of *LNCS*, pages 121–140, Toronto, 2001. Springer.

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

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

Lin Padgham and Michael Winikoff. Prometheus: A practical agent-oriented methodology. In Brian Henderson-Sellers and Paolo Giorgini, editors, *Agent-Oriented Methodologies*, chapter 5, pages 107–135. Idea Group, Hershey, PA, 2005.

H. Van Dyke Parunak. Visualizing agent conversations: Using enhanced Dooley graphs for agent design and analysis. In *Proceedings of the 2nd International Conference on Multiagent Systems*, pages 275–282, Kyoto, 1996. AAAI Press.

Asif Qumer and Brian Henderson-Sellers. 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, March 2008.

William N. Robinson and Sandeep Purao. Specifying and monitoring interactions and commitments in open business processes. *IEEE Software*, 26(2):72–79, March 2009.

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

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

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

Munindar P. Singh. Semantical considerations on dialectical and practical commitments. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*, pages 176–181, Chicago, July 2008. AAAI Press.

Munindar P. Singh. Commitments in multiagent systems: Some history, some confusions, some controversies, some prospects. In Fabio Paglieri, Luca Tummolini, Rino Falcone, and Maria Miceli, editors, *The Goals of Cognition: Essays in Honor of Cristiano Castelfranchi*, chapter 31, pages 601–626. College Publications, London, 2012. Available at http://www.csc.ncsu.edu/faculty/mpsingh/papers.

Munindar P. Singh, Amit K. Chopra, and Nirmit Desai. Commitment-based service-oriented architecture. *IEEE Computer*, 42(11):72–79, November 2009.

Pankaj R. Telang and Munindar P. Singh. Specifying and verifying cross-organizational business models: An agent-oriented approach. *IEEE Transactions on Services Computing*, 5(3):305–318, July 2012.

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

Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, July 2003.

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

Feng Wan and Munindar P. Singh. Commitments and causality for multiagent design. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 749–756, Melbourne, July 2003. ACM Press.

Michael Winikoff. Designing commitment-based agent interactions. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 363–370, 2006.

Michael Winikoff, Wei Liu, and James Harland. Enhancing commitment machines. In *Proceedings of the 2nd International Workshop on Declarative Agent Languages and Technologies (DALT)*, volume 3476 of *LNAI*, pages 198–220, Berlin, 2005. Springer.

Michael Wooldridge, Nicholas R. Jennings, and David Kinny. The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

Pınar Yolum and Munindar P. Singh. 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)*, pages 527–534, Bologna, July 2002a. ACM Press.

Pınar Yolum and Munindar P. Singh. Commitment machines. In *Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages (ATAL 2001)*, volume 2333 of *LNAI*, pages 235–247, Seattle, 2002b. Springer.

APPENDIX

Appendix A

Appendix

A.1 Appendix: A Complete Scenario

Step M4 yields a complete scenario containing the happy path and exception scenarios. Table A.2(I-A) and (II-A) show this complete scenario.

A.2 Appendix: Empirical Study Details

A.2.1 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 4-6 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.

Table A.1: I-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.)

#	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				

Table A.2: II-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.)

#	S	R	Content	A	P	U	C	Operation
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'			1		
24	JD	AG	Here is (the updated premium) p'	2a				
25	AG	JD	Take back the car from M2				2a, 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)

A.2.2 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.

A.2.3 Scenarios for the ASPE Case

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

Table A.3: 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 A.4: 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 A.5: 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 A.6: 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

A.2.4 Scenarios for the AGFIL Case

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

Table A.7: 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 A.8: 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 A.9: 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 A.10: 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 A.11: 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 A.12: 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 A.13: 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 A.14: 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 A.15: 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 A.16: 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