# Commitment Alignment as a Basis for Interoperability in Service Engagements

Amit K. Chopra and Munindar P. Singh, *Fellow, IEEE*

**Abstract**—The expanding interest in the modeling and enactment of *service engagements* creates fresh challenges for software engineering. Existing work on "technical" services, such as on web and grid services, bears little relationship to real-life "business" services that we consider here. Conventional software engineering is ill-equipped to deal with distributed information systems comprised of autonomous entities representing the business partners involved in a typical service engagement.

Although many have recognized the importance of communication in the modeling and enactment of distributed systems, existing approaches treat communication at a low level. Specifically, existing messaging-based approaches consider the flow, but not the meanings, of the messages. Thus they can facilitate interoperation only at a correspondingly low level. In contrast, we give primacy to the meanings of the messages exchanged, based on the notion of *commitments*. Doing so yields a new high-level notion of interoperability, which we term *alignment*. We propose a distributed (locally executable) method that guarantees the alignment of business partners despite asynchrony.

## 1 Introduction

Interoperation literally means working together. Since practical software systems inevitably comprise multiple components, software engineering is naturally concerned with their interoperability. Classically, software engineering has been concerned with inactive components resident within a single administrative domain. Of late, software engineering has trended toward active components. However, the notion of interoperability in most current research has not kept pace with these trends.

This paper addresses software systems whose components represent business partners. We treat such active components as autonomous *agents* who interoperate to realize a *service engagement*. For example, purchasing a book may involve a customer, a merchant, a shipper, a payment agency such as a credit card company, and two or more banks. Each may be modeled as an agent. As in any distributed system, the agents communicate via messaging. Although message transmission and reception are crucial, a purely syntactic treatment of messages cannot adequately express the inherently semantic business considerations that apply in a service engagement.

When considering service engagements, it is natural that interoperability would involve business-level considerations centered on how each partner performs on its *contracts* with the others. This understanding of interoperability conflicts with the traditional notion that components interoperate merely if each would receive messages that another would send it. For example, the above purchase engagement would fail if the customer doesn't pay, even if everyone receives all messages.

In simple terms, the state of the art on interoperability may be described as follows. Whereas traditional software engineering approaches disregard business considerations, approaches that incorporate business considerations disregard the challenges of distributed computing. In contrast, this paper motivates (1) *alignment* as a business-level notion of interoperability and (2) grounds alignment in messaging.

We consider software development as involving conceptual modeling, formalization, and methodology. Our program of research addresses all three with respect to service engagements: this paper focuses on the formalization and operationalization of commitments; our recent (and ongoing) work addresses modeling and methodology [8], [37].

### 1.1 Business-Level Interoperability

Any software architecture must be undergirded by a matching notion of interoperability. We advocate the fundamental intuition expressed by Parnas [28] that architectural connectors be treated not as control or data flow constructs but as *assumptions* made by each component about the others. Much of the subsequent work on architecture has regressed from Parnas' insight by primarily considering connectors at the level of control and data flow—or equivalently message order and occurrence, e.g., [15].

In contrast, our approach models the assumptions the agents make of each other in terms of their commitments toward one another [35]. Commitments yield a simple, declarative characterization of business interoperability that proves sufficiently expressive for the cases of interest. Several researchers recognize the value of commitments in modeling business applications and are developing tools supporting their use [22], [30], [40]. Singh et al. [37] describe additional aspects of commitments with respect to business service engagements.

Formally, a commitment is an expression of the form $\mathsf{C}(\mathsf{debtor}, \mathsf{creditor}, \mathsf{antecedent}, \mathsf{consequent})$, where debtor and creditor are agents, and antecedent and consequent are propositions. This expression means that the debtor commits (to the creditor) to bringing about the consequent if the antecedent holds. For example, $\mathsf{C}(\mathsf{EBook}, \mathsf{Alice}, \$12, \mathsf{BNW})$ means that EBook commits to Alice that if she pays \$12, EBook will send her a copy of *Brave New World (BNW)*.

Agents interact by sending each other messages. We model a physical interaction such as sending a package as a message. A distinguishing feature of our approach is that it assigns a meaning to each message in terms of how it affects the state of a conversation, in particular, including how it affects the commitments among the agents involved. For example, *offer* from EBook to Alice may activate the above commitment.

Now imagine that, at some point during their interaction, Alice infers that EBook is committed to sending her the book she paid for, but EBook infers no such commitment. Thus Alice and EBook fail to work together and are thus not interoperable. In general, a key requirement for successful interaction is that the interacting agents remain *aligned*, meaning that any commitment that a creditor infers the debtor infers as well.

### 1.2 Challenges

Our challenge is to guarantee alignment in asynchronous settings. Asynchrony promotes loose coupling among the components, and is natural in systems with significant communication delays. A robust solution that accommodates asynchrony would also apply in synchronous settings. Guaranteeing alignment in asynchronous settings is nontrivial: the agents may not observe the same messages in the same order. A naïve approach to computing commitments in such settings would lead to misalignments, as the following examples illustrate.

*Example 1:* (Fig. 1(A)). EBook sends Alice (a message that expresses) an *offer* that if she pays \$12, then EBook will deliver to her a copy of the book *Brave New World*. Alice sends EBook a rejection of the offer. Upon receiving the rejection, EBook resends the *offer*. ▌

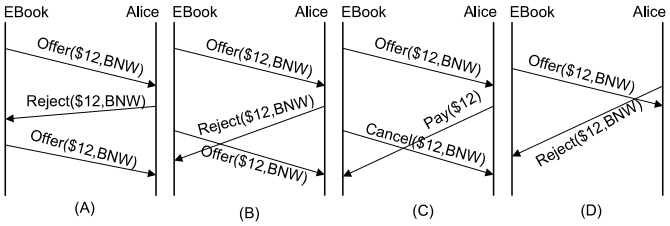


Fig. 1. Scenarios (B), (C), and (D) end in misalignment.

EBook's *offer* creates a commitment from EBook to Alice for *BNW* in return for \$12. In Example 1, Alice and EBook remain aligned by observing the messages in the same order.

*Example 2:* (Fig. 1(B)). EBook makes Alice an offer. Not seeing a response from Alice, EBook resends the offer. Suppose, in the meantime, Alice rejects EBook's offer. Then the *rejection* crosses EBook's repetition of the *offer*. ▌

What ought EBook and Alice to infer about the offer at the end of Fig. 1(B)? Upon Alice's rejection, EBook may infer that its offer no longer exists. But, having seen an offer last, Alice may infer that the offer holds. That is, Alice infers a commitment from EBook whereas EBook does not infer that commitment. This misalignment occurs because Alice's rejection and EBook's offer crossed in transit. Notice that Fig. 1(B) indicates a race condition: the mutual order of offer and rejection matters for EBook but is invisible to Alice.

*Example 3:* (Fig. 1(C)). EBook makes an offer. Alice accepts and pays for it. EBook cancels the offer by sending a message that crosses Alice's payment. ▌

In Example 3, upon sending the payment, Alice infers that EBook is committed to sending her a copy of the book. Because Alice accepted EBook's offer, she treats EBook's cancellation as spurious. But EBook sees the payment after it has canceled, so it considers the payment late. Thus Alice infers EBook's commitment for *BNW*, but EBook does not. Hence we have misalignment.

*Example 4:* (Fig. 1(D)) Here, EBook sends an offer, but in the meantime Alice sends a rejection. ▌

In Example 4, EBook sees the rejection last and infers that its offer was rejected, whereas Alice infers the offer exists she sees the offer last. We allow Alice to reject an offer before receiving it for generality. This helps us handle scenarios where messages arrive unexpectedly, analogous to when one receives a group reply to an email before the original email.

Distribution can cause misalignment by having some agents be better informed about some events than others.

*Example 5:* Alice commits to Rob that if the sky is clear at 5PM, she will meet him at the lake. At 5PM, Rob observes (as a message from the environment) that the sky is clear, and infers that Alice is committed to meeting him at the lake. However, Alice cannot see the sky, and thus does not infer the same commitment. Rob and Alice are thus misaligned. ▌

Most services engineering approaches consider only synchronous interactions as underlying their methodologies. Thus their results do not apply in real-life distributed settings.

### 1.3 Contributions and Organization

This paper proposes an approach that ensures alignment among agents. More specifically, this paper

- Motivates *(commitment) alignment* as a key notion of business-level interoperability suited to service engagements; building on commitment reasoning [36], it characterizes alignment in a rigorous manner.
- Proposes an operational semantics for computing snapshots from messages. Each agent proceeds independently of others, based solely on the messages it observes.
- Guarantees alignment under realistic assumptions of asynchrony with reliable pairwise-ordered queues.

We evaluate the contributions of this paper by modeling a number of realistic scenarios. In contrast with existing approaches, we obtain intuitive results for these scenarios.

Section 2 introduces our technical framework. Section 3 develops the background to specify the meanings of service engagements. Section 4 informally introduces the principles underlying alignment. Section 5 formalizes multiagent communication and alignment. Section 6 proves that our method guarantees alignment. Section 7 discusses the relevant literature. Section 8 discusses some important directions.

## 2 Technical Framework

We assume all (positive) atoms are stable: if an atomic proposition becomes true, it stays true forever. In English, stability

corresponds to sentences such as *the book has been delivered* and *the payment has been made* [36] or sentences that include an explicit timestamp, such as *the book is delivered by 3PM*. However, a commitment is inherently not stable, because it may be discharged, canceled, or released.

Below, $x$, $y$, $z$ are agents; $\alpha$, $\beta$ are atomic propositions; $p$, $q$ are conjunctions of atomic propositions, $r$, $s$, $t$, $u$, $v$, $w$ are (possibly nonatomic) propositions; $\vee$, $\wedge$, $\neg$, $\rightarrow$ are the usual connectives; $\top$ and $\bot$ are the constants for truth and falsity, respectively; $\vdash$ is the usual propositional inference symbol. To simplify notation, we treat $\top$ and $\bot$ as atomic propositions.

The nonterminals Snapshot (of an agent's state), Commitment, and Message are the most important ones.

$L_1$. Snapshot $\longrightarrow$ {Base}
$L_2$. Base $\longrightarrow$ Commitment | Atom | Stative(DNF, CNF)
$L_3$. Commitment $\longrightarrow$ C(Agent, Agent, DNF, CNF)
$L_4$. Content $\longrightarrow$ Atom | $\neg$ Atom | Stative(DNF, CNF)
$L_5$. Stative $\longrightarrow$ created | released | canceled
$L_6$. DNF $\longrightarrow$ And | And $\vee$ DNF
$L_7$. CNF $\longrightarrow$ Or | Or $\wedge$ CNF
$L_8$. And $\longrightarrow$ Content | Content $\wedge$ And
$L_9$. Or $\longrightarrow$ Content | Content $\vee$ Or
$L_{10}$. Message $\longrightarrow$ Declare(Agent, Agent, News)
$L_{11}$. News $\longrightarrow$ Atom | Atom $\wedge$ News
$L_{12}$. Message $\longrightarrow$ Bi(Agent, Agent, DNF, CNF)
$L_{13}$. Message $\longrightarrow$ Tri(Agent, Agent, Agent, DNF, CNF)
$L_{14}$. Bi $\longrightarrow$ Create | Cancel | Release
$L_{15}$. Tri $\longrightarrow$ Assign | Delegate

### 2.1 Reasoning about Commitments

An expression $\mathsf{C}(x, y, r, u)$ denotes an active commitment [36]. When its consequent $u$ holds, the commitment is *discharged* and does not hold any longer. If its antecedent $r$ holds, then $\mathsf{C}(x, y, r, u)$ is *detached*, and the commitment $\mathsf{C}(x, y, \top, u)$ holds. $\mathsf{C}(x, y, \top, u)$ entails $\mathsf{C}(x, y, r, u)$ so the latter continues to hold as well [36]. An unconditional commitment is merely a special case where the antecedent equals $\top$. Table 1 enumerates the commitments in our examples.

TABLE 1
Commitments used as running examples in this paper

| Name | Commitment |
|---|---|
| $c_{\mathsf{A}}$ | C(Alice, EBook, BNW, \$12) |
| $c_{\mathsf{B}}$ | C(EBook, Alice, \$12, BNW) |
| $c_{\mathsf{UA}}$ | C(Alice, EBook, $\top$, \$12) |
| $c_{\mathsf{UB}}$ | C(EBook, Alice, $\top$, BNW) |
| $c_{\mathsf{G}}$ | C(EBook, Alice, \$12, GamblingTips) |
| $c_0$ | C(EBook, Alice, \$12, BNW $\wedge$ GamblingTips) |
| $c_1$ | C(EBook, Alice, \$12 $\vee$ coupon, BNW) |
| $c_2$ | C(EBook, Alice, \$12 $\wedge$ coupon, BNW) |

Postulates $B_1$–$B_8$ are from Singh [36]. The remaining postulates are original. For brevity, we omit agents when they can be understood. When a postulate uniformly involves the debtor $x$ and creditor $y$, we write $\mathsf{C}(r, u)$ instead of $\mathsf{C}(x, y, r, u)$.

$B_1$. DISCHARGE. $u \rightarrow \neg\mathsf{C}(r, u)$
$B_2$. DETACH. $\mathsf{C}(r \wedge s, u) \wedge r \rightarrow \mathsf{C}(s, u)$
$B_3$. AUGMENT. From $\mathsf{C}(r, u)$, $s \vdash r$, $s \nvdash u$ infer $\mathsf{C}(s, u)$
$B_4$. L-DISJOIN. $\mathsf{C}(r, u) \wedge \mathsf{C}(s, u) \rightarrow \mathsf{C}(r \vee s, u)$
$B_5$. R-CONJOIN. $\mathsf{C}(r, u) \wedge \mathsf{C}(r, v) \rightarrow \mathsf{C}(r, u \wedge v)$
$B_6$. CONSISTENCY. $\neg\mathsf{C}(r, \bot)$
$B_7$. NONVACUITY. From $r \vdash u$ infer $\neg\mathsf{C}(r, u)$
$B_8$. WEAKEN. $\mathsf{C}(r, u \wedge v) \wedge \neg u \rightarrow \mathsf{C}(r, u)$

Consider Table 1. Intuitively, $c_0$ is stronger than $c_{\mathsf{B}}$ (an additional book for the same price); $c_1$ is stronger than $c_{\mathsf{B}}$ (two ways to obtain a book instead of one); $c_{\mathsf{B}}$ is stronger than $c_2$ (fewer conditions need to be satisfied in order to obtain a book). Definition 1 captures this intuition.

*Definition 1:* $\mathsf{C}(x, y, r, u)$ is *stronger than* $\mathsf{C}(x, y, s, v)$, or $\mathsf{C}(x, y, r, u) \succeq \mathsf{C}(x, y, s, v)$, if and only if $s \vdash r$ and $u \vdash v$.

For example, $c_0 \succeq c_{\mathsf{B}}$. If $\mathsf{C}(x, y, r, u) \succeq \mathsf{C}(x, y, s, v)$ but $\mathsf{C}(x, y, s, v) \not\succeq \mathsf{C}(x, y, r, u)$, $\mathsf{C}(x, y, r, u)$ is *strictly stronger than* $\mathsf{C}(x, y, s, v)$ or $\mathsf{C}(x, y, r, u) \succ \mathsf{C}(x, y, s, v)$. $B_3$ and $B_8$ capture strength: if $c_1$ holds, then by $B_3$, $c_{\mathsf{B}}$ holds as well. Similarly, if $c_0$ holds, then by $B_8$, $c_{\mathsf{B}}$ holds—unless BNW holds already, in which case according to $B_1$, $c_{\mathsf{B}}$ cannot hold.

### 2.2 Message Types for Commitment Operations

Table 2 introduces the message types by which agents can bring about the well-known operations on commitments [35].

TABLE 2
Messages and their effects

| Message | Sender | Receiver | Effect |
|---|---|---|---|
| Create$(x, y, r, u)$ | $x$ | $y$ | $\mathsf{C}(x, y, r, u)$ |
| Cancel$(x, y, r, u)$ | $x$ | $y$ | $\neg\mathsf{C}(x, y, r, u)$ |
| Release$(x, y, r, u)$ | $y$ | $x$ | $\neg\mathsf{C}(x, y, r, u)$ |
| Delegate$(x, y, z, r, u)$ | $x$ | $z$ | $\mathsf{C}(z, y, r, u)$ |
| Assign$(x, y, z, r, u)$ | $y$ | $x$ | $\mathsf{C}(x, z, r, u)$ |
| Declare$(x, y, p)$ | $x$ | $y$ | $p$ |

Table 2 also introduces Declare, by which a suitably empowered agent [16] brings about facts relevant to a given service engagement, such as the antecedent or consequent of a commitment. For example, making a payment or delivery would cause the detach or discharge of some commitments in Table 1. A forwarded or copied Declare carries the same weight as the original. An observation is a Declare from Env, the environment agent.

Below let $c = \mathsf{C}(x, y, r, u)$. To reduce clutter, the examples below use Create$(c)$ instead of Create$(x, y, r, u)$. And, similarly, Delegate$(c, z)$, Assign$(c, z)$, Release$(c)$, and Cancel$(c)$.

## 3 Understanding Service Engagements via Commitments

Traditional approaches describe a service engagement in terms of the occurrence and relative order of specific messages.

The engagement of Fig. 2 begins with EBook sending Alice an offer. Alice may either accept or reject the offer. If she rejects it, the engagement ends; if she accepts it, EBook sends her the book. Next, Alice sends EBook the payment. Because an FSM ignore the meanings of the messages, it defines compliance on low-level considerations, sometimes

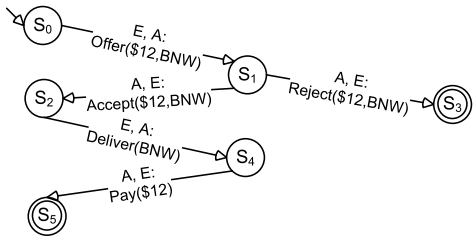


Fig. 2. A purchase engagement as a finite state machine. Each message is tagged with its sender and receiver (here and below, E is EBook; A is Alice).

unintuitive: Alice fails to comply if she sends the payment before she receives the book. In contrast, we build on *commitment protocols* [43], which describe messages along with their business *meanings*. Table 3 shows our specification.

TABLE 3
A purchase engagement expressed in terms of commitments

| Domain-Specific Message | Commitment-Oriented Message |
|---|---|
| *Offer*$(E, A, \$12, \mathsf{BNW})$ | Create$(E, A, \$12, \mathsf{BNW})$ |
| *Accept*$(A, E, \mathsf{BNW}, \$12)$ | Create$(A, E, \mathsf{BNW}, \$12)$ |
| *Reject*$(E, A, \$12, \mathsf{BNW})$ | Release$(E, A, \$12, \mathsf{BNW})$ |
| *Deliver*$(E, A, \mathsf{BNW})$ | Declare$(E, A, \mathsf{BNW})$ |
| *Pay*$(A, E, \$12)$ | Declare$(A, E, \$12)$ |

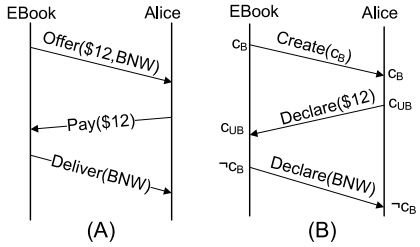


Fig. 3. An enactment of the engagement of Table 3 in terms of (A) domain-specific messages and (B) commitments. We show only the strongest commitments at each point. For example, because $c_{\mathsf{UB}} \succeq c_{\mathsf{B}}$, $c_{\mathsf{UB}}$ is sufficient.

Let us walk through the interaction of Fig. 3. Upon sending Create$(c_{\mathsf{B}})$, EBook infers $c_{\mathsf{B}}$; upon receiving the message Alice infers $c_{\mathsf{B}}$. Upon sending Declare($12), Alice infers that $12 holds. Consequently, she infers that $c_{\mathsf{B}}$ is detached (by $\mathsf{B}_2$), yielding $c_{\mathsf{UB}}$. When EBook receives Declare($12), it infers $c_{\mathsf{UB}}$. EBook finally sends Declare(BNW), upon which it concludes that its commitment is discharged (by $\mathsf{B}_1$). When Alice receives Declare(BNW), she draws the same inference.

Notice that Table 3 does not specify any ordering constraints on messages. In effect, *each party can send messages in any order*. Fig. 4 shows some additional enactments of the purchase engagement between Alice and EBook. The enactments of Figs. 4(B) and 4(C) correspond to the paths that include the dashed transitions in Fig. 5.

So when is an agent compliant with an engagement? The answer is simple: an agent complies if its commitments are

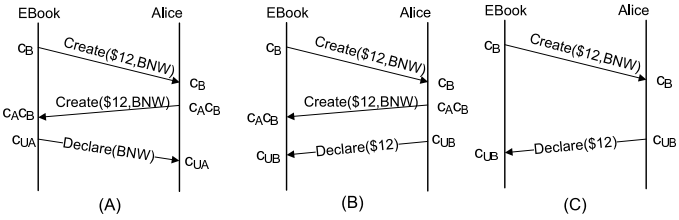


Fig. 4. Three *possible* enactments of the engagement of Table 3.

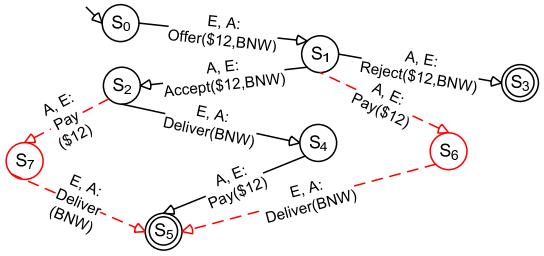


Fig. 5. Expanded set of execution paths for the engagement of Fig. 2.

discharged, no matter if delegated or otherwise manipulated. Traditional approaches force a tradeoff: checking compliance is simple with rigid automaton-based representations and difficult with flexible reasoning. Engagements specified using commitments find the happy middle, promoting flexibility by constraining interactions at the business level, yet providing a rigorous notion of compliance.

Clearly, agents could be misaligned if they interpret messages incompatibly. In our examples, we treat *offer* as a commitment from EBook to ship a book in return for payment. Let's imagine that Alice follows this interpretation but EBook treats an *offer* as Create$(E, A, \mathsf{ack} \wedge \$12, \mathsf{BNW})$ or as Create$(E, A, \mathsf{created}(A, E, \top, \$12), \mathsf{BNW})$. Then, naturally, Alice and EBook may end up misaligned. Here we assume the agents agree on the meanings of their messages. Chopra and Singh [6] address the complementary problem of checking whether agents' interpretations of messages are compatible.

## 4 Principles of Alignment

The misalignments in Fig. 1 arise due to each agent inferring naïvely upon observing a message: inferring $\mathsf{C}(r, u)$ from Create$(r, u)$ and $\neg\mathsf{C}(r, u)$ from Release$(r, u)$ or Cancel$(r, u)$.

Fig. 6 labels each state with the commitment that hold there, and each transition with the message observed. The dashed lines show the transitions for Alice (*A.i*) and EBook (*E.i*). Although they begin from the same state, $\neg c_{\mathsf{B}}$, they end up in different states at the end of *A.2* and *E.2*, respectively.

The traditional way to avoid misalignment would be to synchronize the agents, but synchronization is not viable for real-life service engagements. Instead, we formulate *five* principles of service engagements that are informed by commitments and distributed systems; show that the principles yield intuitively correct enactments; and show how to operationalize them. We begin with three principles that address Fig. 1.

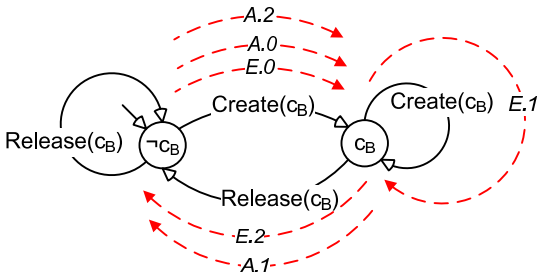


Fig. 6. State machine annotated with the scenario of Fig. 1(B). EBook and Alice follow the same machine, but end up misaligned.

**Principle 1: NOVEL CREATION.** Create$(r, u)$ has no effect if a stronger commitment C$(s, v)$ has held before.

**Principle 2: COMPLETE ERASURE.** Consistency demands that Release$(r, u)$ or Cancel$(r, u)$ removes all commitments weaker than C$(r, u)$ provided no C$(s, v)$ strictly stronger than C$(r, u)$ holds; otherwise it is a noop.

**Principle 3: ACCOMMODATION.** From Release$(r, u)$ and Cancel$(r, u)$, infer that each weaker C$(s, v)$ held before.

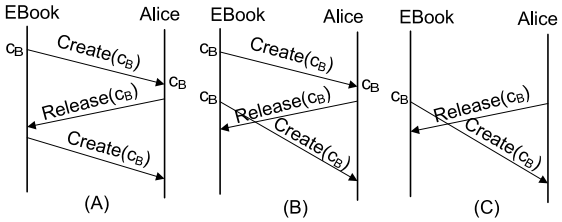


Fig. 7. Proposed approach.

Fig. 7 shows how these principles restore alignment to Fig. 1 (here *offer* is Create$(c_\mathsf{B})$ and *reject* is Release$(c_\mathsf{B})$). In Fig. 1(B), Alice infers $c_\mathsf{B}$ at the end, and EBook does not. But in Fig. 7(B), neither Alice nor EBook infers $c_\mathsf{B}$ at the end. Upon receiving the reject, by COMPLETE ERASURE, EBook infers being released from the offer. When Alice receives the offer again, it is a noop because of NOVEL CREATION.

Similarly, in Fig. 1(D), Alice infers $c_\mathsf{B}$ at the end, and EBook does not. But in Fig. 7(C), however, neither Alice nor EBook infers $c_\mathsf{B}$ at the end. Upon receiving the reject, by COMPLETE ERASURE, EBook infers being released from the offer. When Alice receives an offer she already rejected, ACCOMMODATION ensures that from Alice's release we can presume the offer had been made before. And, thus NOVEL CREATION ensures EBook's actual offer has no effect. Note that Figs. 1(A) and 7(A) yield *differing* outcomes. Fig. 1(A) ends with EBook and Alice both inferring $c_\mathsf{B}$, but Fig. 7(A) ends with both inferring $\neg c_\mathsf{B}$.

NOVEL CREATION means that resending a Create has no effect. But we can use identifiers to distinguish offers. To avoid interference with reasoning about commitments, we apply identifiers not on commitments but on conditions. In Example 6, at the end, both Alice and EBook infer that the $\$12(i_1)$ commitment holds and the $\$12(i_0)$ commitment does not. Section 7 returns to this point as *semanticity*.

*Example 6:* EBook sends Create(E, A, $\$12(i_0)$, BNW$(i_0)$). Alice sends Release(E, A, $\$12(i_0)$, BNW$(i_0)$). To repeat the offer, EBook sends Create(E, A, $\$12(i_1)$, BNW$(i_1)$). ▌

Note that NOVEL CREATION does *not* prevent resurrecting a commitment. A commitment may come to hold again because of a Create message for a *stronger* commitment. It is common practice for a seller to improve its offers, effectively making stronger commitments, as in Example 7.

*Example 7:* EBook makes Alice the offer $c_\mathsf{B}$. Alice rejects the offer thus releasing EBook from $c_\mathsf{B}$. However, EBook is persistent, and it makes Alice the stronger offer $c_0$ (two books for the same price). This automatically resurrects $c_\mathsf{B}$ to ensure consistency. Say, now Alice rejects EBook's improved offer. ▌

When Alice sends Release$(c_0)$, COMPLETE ERASURE means that Alice's rejection also removes $c_\mathsf{B}$ and $c_\mathsf{G}$. In contrast, Release$(c_\mathsf{B})$ has no effect when $c_0$ holds. In that case, $c_0$ continues to hold and re-creates $c_\mathsf{B}$.

**Principle 4: NOTIFICATION.** Whenever a creditor of a commitment learns of a condition that features in the antecedent, it notifies the debtor and whenever a debtor learns of a condition that features in the consequent, it notifies the creditor.

The motivation is to make sure that we test for the alignment of two agents only when both or neither have received vital information. Thus, a creditor must notify relevant debtors of (partial) detaches, and a debtor must notify relevant creditors of (partial) discharges.

Consider Fig. 8(A). Initially, $c_\mathsf{R} = \mathsf{C}(\mathsf{Alice}, \mathsf{Rob}, \mathsf{clear}, \mathsf{lake})$ holds meaning that Alice commits to Rob that if the sky is clear, she will meet him at the lake. Rob observes that the sky is clear—as a message from Env, the environment. Now, Rob infers $c_\mathsf{UR} = \mathsf{C}(\mathsf{Alice}, \mathsf{Rob}, \top, \mathsf{lake})$ but Alice does not (maybe because she is in a basement and cannot look at the sky). Thus, Rob and Alice would appear misaligned but only because Rob has received vital information that Alice has not.

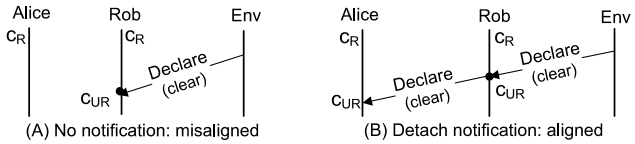


Fig. 8. Notifying about detaches.

In Fig. 8(B), the bold dot on Rob's lifeline indicates that Rob must notify Alice of clear. The state at the dot, where Rob infers $c_\mathsf{UR}$ but Alice does not, is deemed irrelevant for judging alignment. Hence, we avoid a false negative claim about alignment. The case where a debtor notifies a creditor of a discharge is analogous.

Note that if none of the parties learn of the relevant conditions, they might remain aligned without making progress. For example, if Rob never learns about clear, he would not infer $c_\mathsf{UR}$. But he and Alice would remain aligned. We defer the problem of ensuring progress to future work.

**Principle 5: PRIORITY.** When two agents may take conflicting actions, they must agree ahead of time as to whose action has priority.

It is possible that a debtor cancels a commitment concurrently with the creditor detaching it. Recall Example 3 where Alice's payment crosses EBook's cancellation. Fig. 9(A) annotates that example with commitments. If EBook's cancellation and Alice's payment cross, they become misaligned—Alice infers $c_{\mathsf{U}}$ whereas EBook does not. The reason is that receiving $\mathsf{Cancel}(c_{\mathsf{B}})$ has no effect on Alice because she already infers $c_{\mathsf{U}}$, and $c_{\mathsf{U}} \succ c_{\mathsf{B}}$. Receiving Alice's payment has no effect on EBook because it has no commitment to detach.

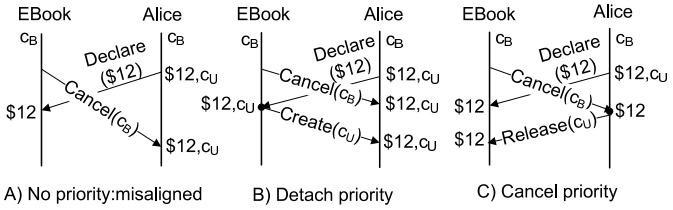


Fig. 9. Race between cancel and detach.

There is no fundamental reason to prefer either the creditor's or the debtor's viewpoint. They simply must agree on what takes priority: cancel over detach, or detach over cancel. The choice may depend upon the domain, e.g., in keeping with the practices of a specific industry. Detach priority means that the debtor considers its cancellation to be overridden by the detach. Cancel priority means that the creditor considers its detach to be overridden by the cancellation.

Our theory can handle whichever alternative a service engagement specifies. Consider $\mathsf{C}(r \wedge s, u)$. Under detach priority, if the debtor observes $\mathsf{Declare}(s)$ (a detach) after it has canceled $\mathsf{C}(r \wedge s, u)$, it must send $\mathsf{Create}(\mathsf{r}, \mathsf{u})$. In Fig. 9(B), EBook creates $c_{\mathsf{U}}$. Alternatively, under cancel priority, if the creditor detaches $\mathsf{C}(r \wedge s, u)$ by sending $\mathsf{Declare}(s)$, and then observes a cancellation for $\mathsf{C}(r \wedge s, u)$, the creditor must send $\mathsf{Release}(r, u)$. In Fig. 9(C), Alice releases EBook from $c_{\mathsf{U}}$.

## 5 Formalizing Alignment

Agents communicate by messaging. Below $m$, $m'$, $m_0$, ... are variables over messages. Assumptions A1–A5 model reliable, asynchronous communication. Distributed systems algorithms routinely make Assumptions A1–A4, e.g., [20]. Assumption A5 simplifies our technical development.

- A1. Communication is *point-to-point*. Below $m(x, y)$ indicates a message $m$ from $x$ to $y$.
- A2. An agent observes all and only those messages that it sends or receives. Observations are ordered *serially*.
- A3. Messaging is *reliable* and *noncreative*. Messages are neither created nor destroyed by the infrastructure.
- A4. Messaging is *ordered*. Any two messages sent by an agent to the same recipient are received in order.
- A5. All observations pertain to messages. Observations of the environment are treated as messages from Env.

An agent $x$'s *observation sequence* $\langle m_0, \ldots, m_n \rangle_x$ describes the sequence of messages $x$ observes in a particular execution. For an observation sequence $o$ of the form $\langle \ldots, m \rangle$ and message $m'$, we define the *concatenation* of $o$ with $m'$ as $\langle \ldots, m, m' \rangle$ and write it as $o; m'$.

Let $\mathcal{A}$ be a system of $k$ agents. Then, $O = [O_0, \ldots, O_{k-1}]$ is an *observation vector* over $\mathcal{A}$, where the $O_i$s are the observation sequences, one for each of the $k$ agents. In other words, an observation vector is a snapshot of the system. Below, $o$ is a variable over observation vectors; $o_x$, etc. are variables over a particular agent's observation sequence. A3 and A4 impose validity requirements on vectors.

*Definition 2:* An observation vector $O$ is *valid* if and only if (1) if $m(x, y)$ occurs in $O_y$, then $m(x, y)$ occurs in $O_x$; and (2) if $m_1(x, y)$ occurs in $O_y$ and $m_0(x, y)$ precedes $m_1(x, y)$ in $O_x$, then $m_0(x, y)$ precedes $m_1(x, y)$ in $O_y$.

Conditions (1) and (2) in Definition 2 capture A3 and A4, respectively. In other words, a valid observation vector is a valid snapshot of the system meaning that a message is received only if (previously) sent and messages between a sender and a receiver are received in the order in which they were sent. This paper considers only valid observation vectors.

An agent's observation sequence represents the agent's snapshot at the granularity of the engagement (i.e., ignoring low-level aspects of the agent's state). Then, in essence, an observation vector corresponds to a consistent set of snapshots. $\mathcal{O}_{\mathcal{A}}$, the set of all possible observation vectors for system $\mathcal{A}$, corresponds to the set of all possible executions of $\mathcal{A}$.

### 5.1 Quiescence and Integrity

Our definition of alignment exploits the inherent asymmetry of a commitment. Specifically, two agents are aligned if and only if whenever one infers a commitment in which it is the creditor, the putative debtor also infers the same commitment. Alignment requires agreement only from a creditor to a debtor: a debtor inferring a commitment does not mean the creditor must infer the same commitment.

In a distributed system with asynchrony, message transmission is asymmetric: the sender is aware of a message before the receiver is. When a message and a commitment "flow" in the same direction, the situation is particularly simple. For example, Alice infers a commitment when she receives an offer from EBook. But at that point, EBook is already inferring the commitment, so Alice and EBook remain aligned. As we explain next, we need to finesse our definition with *quiescence* and *integrity*. Definition 3 states that an observation vector is *quiescent* if and only if every sent message has been received.

*Definition 3:* An observation vector $O \in \mathcal{O}_{\mathcal{A}}$ is quiescent if and only if $\forall x, y \in \mathcal{A}$, if $m(x, y)$ occurs in $O_x$, then $m(x, y)$ occurs in $O_y$.

We verify alignment only at quiescent observation vectors to ensure the agents are "synced" up when we do so. For example, as in Fig. 3(B), suppose Alice pays in response to EBook's offer, thereby creating a stronger (unconditional) commitment from EBook. EBook learns of this only when it receives the payment from Alice. But while the payment is in transit, Alice and EBook would appear to be misaligned. Quiescence avoids this false negative. If even at quiescence, Alice and EBook were to disagree, we would have a problem on our hands.

We verify alignment only at snapshots of agents where each has forwarded all vital information to the relevant parties. In

Fig. 8(B), it would be premature to consider alignment before Rob notifies Alice of clear. In this sense, Rob's notifying Alice of clear is integral with receiving Declare(clear) from Env. Similarly, in Fig. 9(B), EBook sending the Create is integral with receiving Declare(\$12). We disregard intervening observations from the point of view of alignment because they are not integral.

We now show how to state integrity constraints on observations in general. Section 6 specifies the integrity constraints relevant to alignment. Let $\mathcal{S}(O_x)$ be the snapshot of agent $x$ immediately upon $x$ observing $O_x$. (Section 6 formalizes the computation of $\mathcal{S}(o)$.)

$\lfloor m[P]m' \rfloor_x$ is an *integrity constraint* on the observations of agent $x$. Here, $m$ is the *trigger* (message received), $P$ is the *precondition*, and $m'$ is the *effect* (message sent). The interpretation is that when a *trigger* message $m$ arrives if the *precondition* holds, then the receiving agent must send the *effect* message $m'$. Message $m'$ is an *enabled* effect of $m$ with respect to an observation sequence $o$ and a constraint $\lfloor m[P]m' \rfloor_x$ if and only if $P \in \mathcal{S}(o)$.

*Definition 4:* An observation sequence $O_x$ is *integral* with respect to a set of constraints if and only if for any prefix $o; m$ of $O_x$, there exists a prefix $o'$ of $O_x$ such that $o; m$ is a prefix of $o'$ and $o'$ includes an interleaving of all of the enabled effects of $m$ with respect to $o$ and the stated set of constraints.

An observation vector is *integral* with respect to a set of constraints if and only if each observation sequence in it is integral with respect to the set of constraints.

Definition 4 identifies enabled messages as those the agent must send based on the integrity constraints. An observation sequence is not integral unless all messages enabled in it have been observed. Notice that, to be integral, $O_x$ simply must contain the enabled effects (for every prefix of $O_x$); there is no restriction that the enabled effects must occur immediately after the trigger. This means that $x$ may make extraneous observations between the trigger and its enabled effects that are directly relevant for the purposes of integrity.

### 5.2 Agent Snapshots and their Local Maintenance

This section may safely be skipped on a first reading. It formalizes the treatment of agent snapshots, and describes an algorithm for an agent to locally maintain its snapshot.

The snapshot of an agent reflects its local state corresponding to an observation sequence. In syntactic terms, a snapshot is a finite data structure. In semantic terms, a snapshot yields a potentially infinite set of propositions that an agent may logically infer from its observations. To enable how an agent's snapshot progresses based on its observations, we introduce two *update operators* called *addition* $\oplus$ and *subtraction* $\ominus$, which respectively "add" or "subtract" a proposition to or its snapshot. These operators ensure that the resulting representation is consistent according to the postulates of this paper, and that only the essential (i.e., minimal) changes are made due to each message. We now describe one way in which to implement the update operators.

As Section 2 describes, let $S$ be a set of base propositions, i.e., atoms, commitments, or statives. Thus $S$ represents the snapshot of an agent arising from a sequence of observations. In semantic terms, the state of the agent is $[\![S]\!]$, the deductive closure of its snapshot. We extend $\vdash$ to apply to snapshots: $S \vdash p$ means $p \in [\![S]\!]$. We assume that $\vdash$ respects all of the postulates asserted in this paper: $B_1$ to $B_{13}$.

Let $\alpha$ be an atom and $\beta$ be a base proposition. Then we define the above operators logically to capture that they modify a snapshot minimally while forcing the introduction of newly received observations: (1) $S \oplus \beta \vdash \beta$; (2) From $S \vdash \gamma$ and $\beta \wedge \gamma \nvdash \bot$ infer $S \oplus \beta \vdash \gamma$; (3) From $\mathsf{C}(r,u) \prec \mathsf{C}(s,v)$ and $S \vdash \mathsf{C}(r,u)$ infer $S \ominus \mathsf{C}(s,v) \vdash \mathsf{C}(r,u)$; (4) From $\mathsf{C}(r,u) \prec \mathsf{C}(s,v)$ and $S \nvdash \mathsf{C}(r,u)$ infer $S \ominus \mathsf{C}(s,v) \nvdash \mathsf{C}(s,v)$.

Operationally, these operators can be realized as follows:

- $S \oplus (\alpha_1 \wedge \ldots \wedge \alpha_k) = (S \oplus \alpha_1) \ldots \oplus \alpha_k$
- $S \oplus \beta = (S \cup \{\beta\} \setminus \chi(S)) \cup \sigma(S, \beta, \top)$, where
  - $\chi(S)$ is the set of commitments in $S$
  - $\sigma(S, \beta, \alpha)$ is the set of commitments in $S$ that result from $\beta$ being substituted by $\alpha$, and is given by $\{\mathsf{C}(r,u) : r = s|^{\beta}_{\alpha},\ u = v|^{\beta}_{\alpha},\ \mathsf{C}(s,v) \in S,\ u \not\equiv \top, u \not\equiv \bot, r \nvdash u\}$
- $S \ominus \mathsf{C}(s,v) = S$ if $(\exists \mathsf{C}(r,u) \in S$ and $\mathsf{C}(r,u) \succ \mathsf{C}(s,v))$
- $S \ominus \mathsf{C}(s,v) = S \setminus \{\mathsf{C}(t,w) : \mathsf{C}(s,v) \succeq \mathsf{C}(t,w)\}$ if $(\nexists \mathsf{C}(r,u) \in S$ and $\mathsf{C}(r,u) \succ \mathsf{C}(s,v))$

### 5.3 Alignment

Now we are in a position to formalize alignment. Definition 5 considers the observations of creditors and debtors from the same integral and quiescent observation vectors. It says that if a creditor infers a commitment from its observations, then the debtor must infer the same commitment from its own observations. As Section 5 explains, when a debtor infers a commitment that the creditor does not, no harm is done, and alignment is unaffected.

*Definition 5:* A multiagent system $\mathcal{A}$ is *aligned*, written $[\![\mathcal{A}]\!]$, if and only if $\forall O \in \mathcal{O}_{\mathcal{A}}$ such that $O$ is quiescent and integral, for all agents $x$ and $y$ that belong to $\mathcal{A}$, we have that if $\mathsf{C}(x,y,r,u) \in \mathcal{S}(O_y)$ then $\mathsf{C}(x,y,r,u) \in \mathcal{S}(O_x)$.

## 6 Formalizing the Principles

Let $\mathcal{S}(o_x)$ be the current snapshot of $x$. Then we compute $\mathcal{S}(o_x; m)$ from $\mathcal{S}(o_x)$ via one or more applications of the above update operators. The specific applications of the operators depend upon the semantics of the message $m$.

As the base case, constraint $S_1$ tells us that at the outset no specific proposition holds other than $\top$.

$S_1$. $\mathcal{S}(\langle\ \rangle) = \{\top\}$

Below, $S_2$ expresses the semantics of Declare($p$), namely, that $p$ holds upon observing Declare($p$). Notice that by the syntax of Declare($p$), $p$ is a conjunction of one or more atoms. A well-formed Declare($p$) must be consistent, meaning that for no atom $\beta$ must both $\beta$ and $\neg\beta$ occur within $p$.

$S_2$. $\mathcal{S}(o; \mathsf{Declare}((\alpha_1 \wedge \ldots \wedge \alpha_k))) = \mathcal{S}(o) \oplus (\alpha_1 \wedge \ldots \wedge \alpha_k)$

We introduce three stative propositions—created$(x,y,r,u)$, released$(x,y,r,u)$, and canceled$(x,y,r,u)$—each corresponding to the occurrence of the eponymous commitment operation. The above propositions are *stable*,

meaning that once they become true, they remain true forever. Our formalization does not require propositions corresponding to the occurrence of DELEGATE and ASSIGN. We adopt postulates $B_9$–$B_{13}$ in addition to $B_1$–$B_8$.

$B_9$. From $\mathsf{created}(r,u)$ and $\mathsf{C}(r,u) \succeq \mathsf{C}(s,v)$ infer $\mathsf{created}(s,v)$

$B_{10}$. From $\mathsf{released}(r,u)$ and $\mathsf{C}(r,u) \succeq \mathsf{C}(s,v)$ infer $\mathsf{released}(s,v)$

$B_{11}$. From $\mathsf{canceled}(r,u)$ and $\mathsf{C}(r,u) \succeq \mathsf{C}(s,v)$ infer $\mathsf{canceled}(s,v)$

$B_{12}$. $\mathsf{released}(r,u) \rightarrow \mathsf{created}(r,u)$

$B_{13}$. $\mathsf{canceled}(r,u) \rightarrow \mathsf{created}(r,u)$

Let's consider some examples based on the commitments introduced in Table 1 to see how $B_9$–$B_{13}$ apply. Suppose $\mathsf{created}(c_0)$ holds; by $B_9$, $\mathsf{created}(c_\mathsf{B})$ and $\mathsf{created}(c_\mathsf{G})$ hold. Suppose $\mathsf{released}(c_0)$ holds; by $B_{12}$, $\mathsf{created}(c_0)$ holds too; by $B_{10}$, $\mathsf{released}(c_\mathsf{B})$ and $\mathsf{released}(c_\mathsf{G})$ hold; by $B_{12}$, $\mathsf{created}(c_\mathsf{B})$ and $\mathsf{created}(c_\mathsf{G})$ hold. We treat canceled commitments on par with released commitments.

Let's see how $B_9$–$B_{13}$ relate to the principles we introduced earlier. $B_9$ relates to NOVEL CREATION. It ensures that once $\mathsf{created}(r,u)$ holds, all commitments weaker than $\mathsf{C}(r,u)$ are also considered created. $B_{10}$ and $B_{11}$ relate to COMPLETE ERASURE. If a commitment is released or canceled, all weaker commitments are simultaneously released or canceled. $B_{12}$ and $B_{13}$ (together with $B_{10}$ and $B_{11}$) capture ACCOMMODATION: if a commitment has been canceled or released, treat all weaker commitments as if they had held.

### 6.1 Commitment Operations

The messages $\mathsf{Create}(r,u)$, $\mathsf{Release}(r,u)$, and $\mathsf{Cancel}(r,u)$ realize the corresponding two-party operations (eliding the creditor and debtor since those are understood here).

$S_3$, $S_4$, and $S_5$ give the semantics of $\mathsf{Create}(r,u)$. $S_3$ states that if $\mathsf{created}(r,u)$ already holds, then upon observing $\mathsf{Create}(r,u)$, there is no change. $S_4$ states that if the consequent $u$ already holds, then upon observing $\mathsf{Create}(r,u)$, we simply insert $\mathsf{created}(r,u)$. We do not insert $\mathsf{C}(r,u)$, because according to $B_1$, $\mathsf{C}(r,u)$ does not hold—in effect, it is immediately discharged. Conversely, $S_5$ states that if neither $\mathsf{created}(r,u)$ nor $u$ holds in the current state, then upon observing $\mathsf{Create}(r,u)$, we insert both $\mathsf{C}(r,u)$ and $\mathsf{created}(r,u)$.

$S_3$. $\mathcal{S}(o;\mathsf{Create}(r,u)) = \mathcal{S}(o)$, if $\mathsf{created}(r,u) \in \mathcal{S}(o)$

$S_4$. $\mathcal{S}(o;\mathsf{Create}(r,u)) = \mathcal{S}(o) \oplus \mathsf{created}(r,u)$, if $u \in \mathcal{S}(o)$

$S_5$. $\mathcal{S}(o;\mathsf{Create}(r,u)) = (\mathcal{S}(o) \oplus \mathsf{created}(r,u)) \oplus \mathsf{C}(r,u)$, if $\mathsf{created}(r,u) \notin \mathcal{S}(o)$ and $u \notin \mathcal{S}(o)$

According to $S_6$, upon observing $\mathsf{Release}(r,u)$, we remove all commitments weaker than $\mathsf{C}(r,u)$, and insert $\mathsf{released}(r,u)$. $S_7$ analogously gives the semantics of $\mathsf{Cancel}(r,u)$.

$S_6$. $\mathcal{S}(o;\mathsf{Release}(r,u)) = (\mathcal{S}(o) \ominus \mathsf{C}(r,u)) \oplus \mathsf{released}(r,u)$

$S_7$. $\mathcal{S}(o;\mathsf{Cancel}(r,u)) = (\mathcal{S}(o) \ominus \mathsf{C}(r,u)) \oplus \mathsf{canceled}(r,u)$

Based on the foregoing, we see that $S_3$–$S_7$ capture NOVEL CREATION, COMPLETE ERASURE, and ACCOMMODATION.

Clearly, any implementation of DELEGATE and ASSIGN must involve at least two messages. Fig. 10(A) exemplifies the message pattern for delegating a commitment with EBook as delegator and Charlie as delegatee. EBook sends $\mathsf{Delegate}(c_\mathsf{B}, \mathsf{Charlie})$ to Charlie. Let $d\_c_\mathsf{B} = \mathsf{C}(\mathsf{Charlie}, \mathsf{Alice}, \$12, \mathsf{BNW})$. Upon its receipt, Charlie sends $\mathsf{Create}(d\_c_\mathsf{B})$ to Alice, thus realizing the delegation. Fig. 10(B) exemplifies the message pattern for assigning a commitment with Alice as assigner and Rob as assignee. Alice sends $\mathsf{Assign}(c_\mathsf{B}, \mathsf{Rob})$ to EBook. Let $a\_c_\mathsf{B} = \mathsf{C}(\mathsf{EBook}, \mathsf{Rob}, \$12, \mathsf{BNW})$. Upon its receipt, EBook sends $\mathsf{Create}(a\_c_\mathsf{B})$ to Rob, thus realizing the assignment.

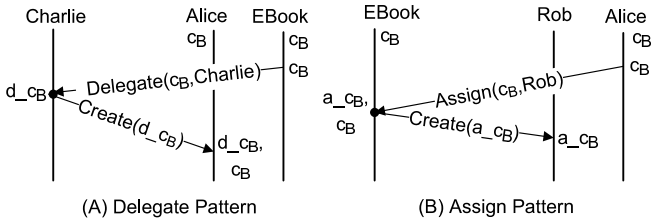


Fig. 10. The delegate and assign patterns.

As we define delegation, it does not involve a notification from the delegator to the creditor. Such notifications could be valuable in some practical scenarios [39], but our aim here is to delineate the core patterns on top of which additional patterns, such as those involving a notification to the creditor, may be built. Similarly, assignment does not involve a notification from the assigner to the assignee.

$S_8$ and $S_9$ treat Delegate and Assign as noops; the subsequent Create messages in the above patterns do all the work. Thus Delegate and Assign are handled through the integrity mechanism motivated for notifications, and introduced next.

$S_8$. $\mathcal{S}(o;\mathsf{Delegate}(x,y,z,r,u)) = \mathcal{S}(o)$

$S_9$. $\mathcal{S}(o;\mathsf{Assign}(x,y,z,r,u)) = \mathcal{S}(o)$

### 6.2 Integrity of Observations

Each row in Table 4 expresses an integrity constraint on agent behavior of the form (referring to column headers):

$$\lfloor \mathsf{Trigger}[\mathsf{Precondition}]\mathsf{Effect} \rfloor_{\mathsf{Who}}$$

For example, the Delegate Rule

$$\lfloor \mathsf{Delegate}(x,y,z,r,u)[\top]\mathsf{Create}(z,y,r,u) \rfloor_z$$

states that upon receiving a Delegate, the delegatee must send the specified Create to the creditor.

Recall that NOTIFICATION states that creditors must notify debtors of detaches, and debtors must notify creditors of discharges. Two cases arise for each kind.

**Detach Posterior.** Initially, $y$ infers $\mathsf{C}(x,y,r \wedge s,u)$ and $\neg\mathsf{C}(x,y,r,u) \wedge \neg s$, meaning that the commitment is not detached yet. Next $y$ observes $\mathsf{Declare}(s)$ from some $z$, thereby detaching $\mathsf{C}(x,y,r \wedge s,u)$, and $y$ inferring $\mathsf{C}(x,y,r,u)$. Hence, $y$ must now inform $x$ about the detach by sending $\mathsf{Declare}(y,x,s)$.

**Detach Prior.** Initially, $y$ infers $s$ and $\neg\mathsf{C}(x,y,r,u)$. Subsequently, $y$ observes $\mathsf{Create}(x,y,r \wedge s,u)$. Therefore, $y$ infers $\mathsf{C}(x,y,r \wedge s,u)$. $\mathsf{C}(x,y,r \wedge s,u)$ is immediately detached because of $s$, thereby yielding $\mathsf{C}(x,y,r,u)$.

TABLE 4
Constraints on agent behavior to ensure integrity of observation vectors

| Name | Who | Trigger | Precondition | Effect |
|---|---|---|---|---|
| Delegate | $z$ | $\mathsf{Delegate}(x, y, z, r, u)$ | $\top$ | $\mathsf{Create}(z, y, r, u)$ |
| Assign | $x$ | $\mathsf{Assign}(x, y, z, r, u)$ | $\top$ | $\mathsf{Create}(x, z, r, u)$ |
| Detach Posterior | $y$ | $\mathsf{Declare}(z, y, s)$ | $\mathsf{C}(x, y, r \wedge s, u) \wedge \neg\mathsf{C}(x, y, r, u) \wedge \neg s$ | $\mathsf{Declare}(y, x, s)$ |
| Detach Prior | $y$ | $\mathsf{Create}(x, y, r \wedge s, u)$ | $\neg\mathsf{C}(x, y, r, u) \wedge s$ | $\mathsf{Declare}(y, x, s)$ |
| Discharge Posterior | $x$ | $\mathsf{Declare}(z, x, u)$ | $\mathsf{C}(x, y, r, u) \wedge \neg u$ | $\mathsf{Declare}(x, y, u)$ |
| Discharge Prior | $x$ | $\mathsf{Create}(x, y, r, u)$ | $\neg\mathsf{C}(x, y, r, u) \wedge v$ where $v$ is the strongest $u'$ that holds where $u \vdash u'$ | $\mathsf{Declare}(x, y, v)$ |
| Creditor Priority | $x$ | $\mathsf{Declare}(z, x, s)$ | $\mathsf{canceled}(x, y, r \wedge s, u) \wedge \neg\mathsf{C}(x, y, r \wedge s, u) \wedge \neg s$ | $\mathsf{Create}(x, y, r, u)$ |
| Debtor Priority | $y$ | $\mathsf{Cancel}(x, y, r \wedge s, u)$ | $s \wedge \mathsf{C}(x, y, r \wedge s, u) \wedge \neg\mathsf{C}(x, y, r', u')$ such that $\mathsf{C}(x, y, r', u') \succ \mathsf{C}(x, y, r, u)$ | $\mathsf{Release}(y, x, r, u)$ |

Hence, $y$ must now inform $x$ about the detach by sending $\mathsf{Declare}(y, x, s)$.

Notice that $\mathsf{Create}(x, y, r, u)$ is merely a special case of $y$ observing $\mathsf{Create}(x, y, r \wedge s, u)$, namely, $\mathsf{Create}(x, y, r \wedge \top, u)$.

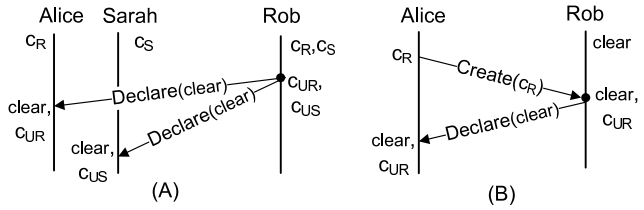


Fig. 11. Detach notifications.

Fig. 11(A) demonstrates Detach Posterior. Here Alice and Sarah are committed to meeting Rob at the lake if the sky is clear ($c_{\mathsf{R}}$ and $c_{\mathsf{S}}$, respectively). Rob learns the sky is clear and thus infers that both Alice and Sarah are now unconditionally committed to meeting him ($c_{\mathsf{UR}}$ and $c_{\mathsf{US}}$, respectively). Rob must notify both Alice and Sarah, thus preserving his alignment with each of them. Fig. 11(B) illustrates Detach Prior. Here, Rob already infers clear. So when Rob receives $\mathsf{Create}(c_{\mathsf{R}})$, Rob infers that Alice is unconditionally committed ($c_{\mathsf{UR}}$). Then, Detach Prior ensures that Rob notifies Alice.

**Discharge Posterior.** Initially, $x$ infers $\mathsf{C}(x, y, r, u)$ and $\neg u$. Next, $x$ observes $\mathsf{Declare}(u)$ from some $z$. As a result, $\mathsf{C}(x, y, r, u)$ is discharged. Hence, $x$ must now inform the creditor $y$ of the discharge by sending $\mathsf{Declare}(x, y, u)$.

**Discharge Prior.** Initially, $x$ infers $v$. Next, $x$ sends $\mathsf{Create}(x, y, r, u)$ such that $u \vdash v$. Hence, $x$ will not infer $\mathsf{C}(x, y, r, v)$ because $v$ holds. But $y$ may yet infer $\mathsf{C}(x, y, r, v)$. Hence, $x$ must send $\mathsf{Declare}(x, y, v)$ to prevent $y$ from inferring $\mathsf{C}(x, y, r, v)$.

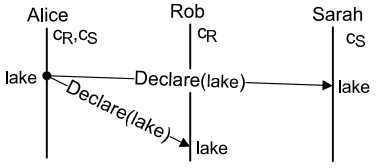


Fig. 12. Discharge notification.

Fig. 12 illustrates Discharge Posterior. Alice is committed to both Rob and Sarah to be at the lake ($c_{\mathsf{R}}$ and $c_{\mathsf{S}}$, respectively). When Alice gets to the lake, she discharges those commitments. Discharge Posterior ensures that Alice informs both Rob and Sarah accordingly, so that they each may consider their respective commitments discharged.

Below, we formalize the implications of detach priority and cancel priority for a commitment $\mathsf{C}(x, y, r \wedge s, u)$.

**Creditor Priority.** Initially, $x$ infers $\mathsf{canceled}(x, y, r \wedge s, u)$ and $\neg\mathsf{C}(x, y, r \wedge s, u) \wedge \neg s$. Note that $\mathsf{canceled}(x, y, r \wedge s, u)$ does not entail $\neg\mathsf{C}(x, y, r \wedge s, u)$ because a commitment once canceled may come to hold again implicitly if a stronger commitment is subsequently created. Next, $x$ receives $\mathsf{Declare}(s)$ from some agent $z$. If $\mathsf{C}(x, y, r \wedge s, u)$ had not been canceled, it would have been detached. But $y$ may not know about the cancellation yet. Therefore, the debtor must act as if the commitment has been detached, in essence giving priority to the creditor. Hence, the debtor $x$ must now send $\mathsf{Create}(x, y, r, u)$.

**Debtor Priority.** Initially, $y$ (as a creditor) infers $s$ and $\mathsf{C}(x, y, r \wedge s, u)$. Therefore, it also infers $\mathsf{C}(x, y, r, u)$. Next, $y$ observes $\mathsf{Cancel}(x, y, r \wedge s, u)$. It could be that $x$ sent $\mathsf{Cancel}(x, y, r \wedge s, u)$ without knowing that $s$ holds, and therefore $x$ may not infer $\mathsf{C}(x, y, r, u)$. To avoid this possible misalignment, $y$ must now send $\mathsf{Release}(y, x, r, u)$. However, $y$ need not send the release if a commitment strictly stronger than $\mathsf{C}(x, y, r, u)$ holds. Sending the release would then be ineffective because of COMPLETE ERASURE.

Fig. 9(B) illustrates the case of detach priority to fix the misalignment of Fig. 9(A), whereas Fig. 9(C) illustrates the case of cancel priority.

It could be that in the case of detach priority, Alice cheats by sending the payment even after receiving the cancellation. Analogously, in the case of cancel priority, EBook could cheat and get away with it. In settings where the parties are mutually untrustworthy, we can imagine the use of techniques such as secure mediators to ensure that neither party deceives the other.

It is important to realize that the above rules are weak and *locally executable* constraints on an agent's behavior because they only call for an agent to *send* messages. Specifically, they involve neither receiving a message nor synchronizing with another agent.

### 6.3 Proof of Correctness

Now it remains to show that under the assumptions we have made, the formalization of commitment operations we have

proposed guarantees that any multiagent system is aligned. Notice that a commitment is strengthened only through a Create or through a Declare that serves to detach. A commitment is removed or weakened only through a Release or a Cancel, or through a Declare that serves to discharge.

*Theorem 1:* For any multiagent system $\mathcal{A}$, under (i) assumptions A1–A4 and (ii) the snapshot computation given by $S_1$–$S_9$, the notification rules of Table 4 guarantee alignment, that is, $[\![\mathcal{A}]\!]$.

*Proof:* $\mathcal{A}$ is aligned at the outset, i.e., in the observation vector of empty sequences, when no agent has made any observations. Our proof is by mathematical induction on the height of a commitment expression (defined as the height of the nested statives it encloses) and by mathematical induction on the size of an observation vector (defined as the sum of the lengths of the observation sequences in it).

First, consider a commitment expression of height zero, i.e., an expression containing no stative expression. Inductively, assume that $\mathcal{A}$ is aligned up to a quiescent, integral observation vector $O$. Consider two agents, $x$ and $y$ in $\mathcal{A}$.

Next, expand $O$ to a quiescent, integral observation vector $O' = O; O^{\Delta}$. There are two possible threats to alignment: (1) if $y$ infers a new commitment as creditor that its debtor does not; and (2) if $y$ continues to infer a commitment as creditor that it previously inferred, but its debtor no longer does.

For (1), consider a commitment added by $y$, i.e., $\mathsf{C}(x, y, r, u) \in \mathcal{S}(O'_y) \setminus \mathcal{S}(O_y)$. Without loss of generality, assume $\mathsf{C}(x, y, r, u)$ is maximally strong, i.e., no other commitment added by $y$ is strictly stronger than $\mathsf{C}(x, y, r, u)$. This means $O^{\Delta}_y$ includes receiving a (partial or total) detach (Declare) or a Create. For a detach, by integrity, $y$ would have sent a message $\mathsf{Declare}((\alpha_1 \wedge \ldots \wedge \alpha_k))$ to $x$, which would have landed within $O^{\Delta}_x$ to ensure the quiescence of $O^{\Delta}_x$. A Create would have originated from $x$. In either case, the quiescence of $O'$ ensures that $O'_x \vdash \mathsf{C}(x, y, r, u)$.

For (2), consider a commitment not added by $y$ but removed by $x$, i.e., $\mathsf{C}(x, y, r, u) \in \mathcal{S}(O_y)$ and $\mathsf{C}(x, y, r, u) \in \mathcal{S}(O_x) \setminus \mathcal{S}(O'_x)$. Without loss of generality, assume $\mathsf{C}(x, y, r, u)$ is maximally strong, i.e., no other commitment removed by $x$ is strictly stronger than $\mathsf{C}(x, y, r, u)$.

Because $\mathsf{C}(x, y, r, u) \in \mathcal{S}(O_y)$, by our inductive hypothesis, $\mathsf{C}(x, y, r, u) \in \mathcal{S}(O_x)$. Hence, if $\mathsf{C}(x, y, r, u) \notin \mathcal{S}(O'_x)$, this means $O^{\Delta}_x$ includes receiving a (partial or total) discharge (Declare) or Release, or sending a Cancel. The Release would be sent by $y$, thus $\mathsf{C}(x, y, r, u) \notin \mathcal{S}(O'_y)$. The Cancel would be sent to $y$ and the discharge would be propagated to $y$ to ensure integrity. Therefore, by quiescence, $\mathsf{C}(x, y, r, u) \notin \mathcal{S}(O'_y)$.

The above establishes the base case for commitment expressions. The inductive step on commitment expressions of height $h + 1$ follows trivially by, in essence, repeating the above argument where one or both of $r$ and $u$ may contain a stative expression of height $h$. □

# 7 Relevant Literature

This paper synthesizes ideas from software engineering in distributed computing and multiagent systems for formalizing service engagements at a high level. Below, we review the related literature from these fields.

## 7.1 Distributed Systems

Existing work on interoperability treats the assumptions each component makes of another solely at the level of messages [1], [5], [13]. Such approaches—inspired by the software components literature [7], [42]—typically check for two properties, namely, that each service is ready to receive any messages that may be sent to it by others (safety), and that there are no deadlocks (liveness). Such considerations are undoubtedly valuable, but are far from adequate in properly enacting service engagements among autonomous parties.

Commitment alignment captures a form of business-level interoperability that is appropriate for service engagements. Determining both business-level (in terms of alignment) and message-level (in terms of flow) interoperability are important. For example, EBook and Alice may be aligned *but* deadlocked if EBook expects to receive the message corresponding to payment first and Alice expects to receive the message corresponding to the delivery of the book first. Conversely, EBook and Alice may never deadlock, but may nonetheless become misaligned. Business-level interoperability pertains to the meaning of an interaction, whereas message-level interoperability pertains to how the agents are implemented.

The correct execution of a distributed application often requires strong guarantees, such as *causal delivery* [33], from the communication substrate. In comparison, we make weak assumptions that are readily supported by existing systems.

Much attention has been devoted to abstractions, e.g., [27], to facilitate programming distributed applications. Our approach falls into this theme but focuses on service engagements. It frees the programmer from the burden of reasoning about commitments and constitutes the essential elements of a business-level middleware, as Section 8 describes.

Detecting global properties from local states is an important problem. The challenge lies in identifying globally consistent states [20]. We ensure that agents can act locally in a manner that ensure alignment, thus *obviating the need to maintain a global representation*.

Joshi and Misra [18] motivate *maximality*. A program is maximal with respect to a specification if it can produce any execution that satisfies the specification. Our approach reflects a similar intuition. By capturing and enacting business-level meanings, we can support a maximal set of enactments of a service engagement.

## 7.2 Multiagent Systems

We now motivate two important criteria by which to evaluate approaches for alignment. Current approaches fall into two main categories, each violating at least one of these criteria.

**Autonomy** means that no agent should have to obtain approval from another agent in order to effect a change in its commitments. An acknowledgment-based approach requires a debtor to seek the creditor's approval to effect a cancellation or discharge. Acknowledgments simplify analysis because they ensure the agents observe the relevant messages in the same order [22]. But the price in terms of tight coupling and violation of autonomy is too high in our view.

**Semanticity** means respecting the meaning of commitments. In particular, inserting identifiers into commitments means that we cannot reason about them freely. For example, from $\mathsf{C}(i_0, x, y, r, u)$ and $\mathsf{C}(i_1, x, y, r, v)$ we have no basis to infer $\mathsf{C}(i_0, x, y, r, u \wedge v)$ or $\mathsf{C}(i_1, x, y, r, u \wedge v)$. We would end up tracking dependencies in some ad hoc manner. Approaches that require identifiers for commitments [11], [32] thus violate semanticity.

Commitments help formalize agent communications [12] for business protocols [11] and argumentation [2]. Existing works ignore the challenges posed by distributed settings. Many proposals employ a shared commitment-store and rely on simplified dialog approaches based on turn taking or acknowledgments [22], [32]. These are incompatible with autonomy. Our results could lead to more flexible business and dialog protocols.

Winikoff [40] studies how commitments may be implemented in a distributed setting. However, his solution only allows for a monotonically increasing set of commitments, and does not support operations such as DISCHARGE, RELEASE, and CANCEL.

Whereas we map commitment operations to messages, some others consider success conditions in greater depth. Norman and Reed [25] discuss delegation and Jones and Sergot [16] study institutional power formally, but they don't address distribution as we do. We defer synthesizing the above researchers' insights with our approach to future work.

Paurobally et al. [29] address alignment, but in terms of beliefs. But commitment and beliefs are fundamentally different: commitments are public artifacts whereas beliefs are private, which makes them inappropriate for open settings such as in service engagements [34]. Further, in contrast with commitment alignment, belief consistency is symmetric, which makes it difficult to achieve in an asynchronous setting. Further, Paurobally et al. rely on synchronization, which too is inappropriate in service engagements.

### 7.3 Software Engineering for Business Processes

Osterweil [26] advocates expanding "software" to include artifacts such as regulations. We agree: our specifications of service engagements are higher level than method invocations or message exchanges, but very much software nevertheless. To this end, they need—and we provide—both a semantics of interoperability and a means for realizing it.

Zirpins and Emmerich [44] study *production networks* (of services). They formulate patterns that characterize the business meanings of the interactions independently of specific orchestrations. Our work has the same motivation; we go beyond it in addressing interoperability in a formal manner.

Jonquet et al. [17] propose an integration of grid computing and multiagent systems. Like us, they take an expansive view of services, and propose a specification language centered on interactions among services. Our intuitions are similar, but we go further in formalizing an approach for interoperability.

Robinson and Purao [30] build on our previous work on specifying, composing, and operationalizing commitment protocols [9]. They show how to monitor the discharge of a commitment, and conclude how commitments simplify requirements gathering and monitoring, thereby supporting error recovery. Our approach goes further in operationalizing service engagements and ensuring alignment.

Cross-organizational business processes are often specified as workflows over services. Kona et al. [19] provide a general approach for automatically composing services into a workflow. In contrast to our approach, workflows make no allowance for the fact that services are provided by autonomous agents. Thus workflows apply at the level of technical services—below the level of service engagements.

Bhattacharya et al. [4] and Narendra et al. [24] propose an artifact-centric approach for modeling business processes. Each artifact is associated with a workflow that describes how it may be manipulated. The dependencies between artifacts are also modeled. However, there is no explicit characterization of meaning, as in our approach. In principle, we specify artifacts and their updates declaratively through the use of commitments.

Choreographies are increasingly used to express cross-organizational business processes [14], [31], [38]. A choreography specifies constraints on message occurrence and ordering. Benatallah et al. [3] formalize similarity and replaceability relations between choreographies. Foster et al. [13] check for the correctness of service implementations with respect to choreographies. Properties analogous to the above would apply to service engagements and agents, although they would need to be formalized in a declarative, meaning-driven manner.

Further, choreography-based approaches typically address state alignment via synchronization or its surrogate, explicit acknowledgments. For instance, Molina-Jimenez et al. [23] use lockstep synchronization to ensure consistency among participants in a business conversation. However, synchronization and acknowledgments preclude flexible service enactments.

Mahfouz et al. [21] tie organizational requirements, expressed as goal models, to message choreographies. They complement our approach, which provides a sound basis enacting organizational requirements, expressed via commitments.

## 8 Conclusions and Future Work

Commitment alignment provides a notion of interoperability that supports the flexibility of service engagements. It obviates the need for tight coupling or synchronization, which would be impractical anyway. Our formalization satisfies Section 7's properties of autonomy and semanticity.

This paper opens up some important directions for future work. First, we will study additional forms of alignment, which would largely be subject to the same principles as above. For example, delegator-delegatee alignment would ensure that the delegator could keep tabs on its delegatees, treating them as subcontractors in a complex service engagements.

Second, we will incorporate the *context* of a commitment, which arises in conceptual treatments of service engagements [37]. A context sets up the rules of encounter, and may serve as an arbiter or enforcement authority. For example, we can model eBay as the context for auctions that happen on its site. A commitment, as envisaged by Singh [35], includes an

explicit parameter for a context. Aligning with the context would be critical in ensuring felicitous interactions.

Third, we will identify additional business patterns. For example, we might create a delegation pattern where the delegator explicitly cancels its commitments, thus relinquishing all responsibility to the delegatee. Singh et al. describe some such patterns for service engagements [37].

Fourth, this paper leads to a new kind of middleware centered on commitments. This middleware computes commitments from messages; maintains the snapshot for each agent; and notifies the appropriate participants automatically based on the integrity constraints.

## REFERENCES


[1] M. Baldoni, C. Baroglio, A. K. Chopra, N. Desai, V. Patti, M. P. Singh. Choice, interoperability, and conformance in interaction protocols and service choreographies. In *Proc. 8th Intl. Conf. Auton. Agents & MultiAgent Systems (AAMAS)*, pp. 843–850, 2009.

[2] J. Bentahar, B. Moulin, J.-J. C. Meyer, B. Chaib-draa. A logical model for commitment and argument network for agent communication. In *Proc. 3rd Intl. Conf. Auton. Agents & Multiagent Systems*, pp. 792–799, 2004.

[3] B. Benatallah, F. Casati, and F. Toumani. Representing, analysing and managing web service protocols. *Data and Knowledge Engineering*, 58(3):327–357, 2006.

[4] K. Bhattacharya, R. Hull, J. Su. A data-centric design methodology for business processes. In J. Cardoso, W. van der Aalst, eds., *Handbook of Research on Business Process Modeling*, chapter 23, pp. 503–531. IGI-Global, 2009.

[5] M. Bravetti, G. Zavattaro. A theory for strong service compliance. In *Proceedings of 9th Intl. Conf. Coordination Models & Languages (Coordination'07)*, LNCS 4467, pp. 96–112, 2007.

[6] A. K. Chopra, M. P. Singh. Constitutive interoperability. In *Proc. 7th Intl. Conf. Auton. Agents & MultiAgent Systems (AAMAS)*, pp. 797–804, 2008.

[7] L. de Alfaro, T. A. Henzinger. Interface automata. In *Proc. Joint 8th European Soft. Eng. Conf. & 9th ACM SIGSOFT Symp. Found. Soft. Eng.* , pp. 109–120, 2001.

[8] N. Desai, A. K. Chopra, M. P. Singh. Amoeba: A methodology for modeling and evolution of cross-organizational business processes. *ACM Trans. Soft. Eng. & Method. (TOSEM)*, 19(2):6:1–6:45, 2009.

[9] N. Desai, A. U. Mallya, A. K. Chopra, M. P. Singh. Interaction protocols as design abstractions for business processes. *IEEE Trans. Soft. Eng.*, 31(12):1015–1027, Dec. 2005.

[10] ebBP. Electronic business extensible markup language business process specification schema v2.0.4, Dec. 2006. docs.oasis-open.org/ebxml-bp/2.0.4/OS/.

[11] R. A. Flores, P. Pasquier, B. Chaib-draa. Conversational semantics with social commitments. In R. M. van Eijk, M.-P. Huget, F. Dignum, editors, *Agent Communication*, *LNCS* 3396, pp. 18–32. Springer, 2004.

[12] N. Fornara, M. Colombetti. A commitment-based approach to agent communication. *Applied Artif. Intelligence*, 18(9-10):853–866, 2004.

[13] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-based analysis of obligations in web service choreography. In *Proc. of IEEE Intl. Conf. on Internet and Web Appl. and Services*, 2006.

[14] HL7 reference information model, version 1.19. www.hl7.org/ Library/ data-model/ RIM/ C30119/ Graphics/ RIM_billboard.pdf, 2002.

[15] G. Hohpe, B. Woolf. *Enterprise Integration Patterns*. 2004.

[16] A. J. I. Jones, M. Sergot. A formal characterisation of institutionalized power. *J. IGPL*, 4(3):429–445, 1996.

[17] C. Jonquet, P. Dugenie, S. A. Cerri. Service-based integration of grid and multi-agent systems models. In *Service-Oriented Computing: Agents, Semantics, & Eng.* , *LNCS* 5006, pp. 56–68. Springer, 2008.

[18] R. Joshi, J. Misra. Maximally concurrent programs. *Formal Aspects Comput.*, 12(2):100–119, 2000.

[19] S. Kona, A. Bansal, M. B. Blake, G. Gupta. Generalized semantics-based service composition. In *Proc. IEEE Intl. Conf. Web Services (ICWS)*, pp. 219–227, 2008.

[20] A. D. Kshemkalyani, M. Raynal, M. Singhal. An introduction to snapshot algorithms in distributed computing. *Dist. Syst. Eng.*, 2(4):224–233, 1995.

[21] A. Mahfouz, L. Barroca, R. Laney, B. Nuseibeh. Requirements-driven collaborative choreography customization. In *Proc. 7th Intl. Joint Conf. Service-Oriented Comput.*, 2009.

[22] P. McBurney, S. Parsons. Posit spaces: a performative model of e-commerce. In *Proc. 2nd Intl. Joint Conf. Auton. Agents & Multiagent Systems*, pp. 624–631, 2003.

[23] C. Molina-Jimenez, S. Shrivastava, N. Cook. Implementing business conversations with consistency guarantees using message-oriented middleware. In *Proc. 11th IEEE Intl. Enterprise Dist. Object Computing Conf.*, pp. 51–62, 2007.

[24] N. C. Narendra, Y. Badr, P. Thiran, and Z. Maamar. Towards a unified approach for business process modeling using context-based artifacts and web services. In *IEEE Intl. Conf. on Services Computing*, pages 332–339, 2009.

[25] T. J. Norman, C. Reed. Delegation and responsibility. In *Proc. 7th Intl. Workshop Agent Theories Architectures & Languages, VII*, pp. 136–149, 2001.

[26] L. J. Osterweil. What is software? *Auto. Soft. Eng.*, 15(3–4):261–273, 2008.

[27] K. Ostrowski, K. Birman, D. Dolev. Live distributed objects: Enabling the active web. *IEEE Internet Computing*, 11(6):72–78, 2007.

[28] D. L. Parnas. Information distribution aspects of design methodology. In *Proc. Intl. Federation for Information Processing Congress*, TA-3, pp. 26–30, Amsterdam, 1971. North Holland.

[29] S. Paurobally, J. Cunningham, N. R. Jennings. Ensuring consistency in the joint beliefs of interacting agents. In *Proc. 2nd Intl. Joint Conf. on Auton. Agents & Multiagent Systems*, pp. 662–669, 2003.

[30] W. N. Robinson, S. Purao. Specifying and monitoring interactions and commitments in open business processes. *IEEE Soft.*, 26(2):72–79, 2009.

[31] RosettaNet. Home page, 1998. www.rosettanet.org.

[32] M. Rovatsos. Dynamic semantics for agent communication languages. In *Proc. 6th Intl. Conf. Auton. Agents & Multiagent Systems (AAMAS)*, pp. 100–107, 2007.

[33] A. Schiper, K. Birman, P. Stephenson. Lightweight causal and atomic group multicast. *ACM Trans. Computer Systems*, 9(3):272–314, 1991.

[34] M. P. Singh. Agent communication languages. *IEEE Computer*, 31(12):40–47, 1998.

[35] M. P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. & Law*, 7(1):97–113, Mar. 1999.

[36] M. P. Singh. Semantical considerations dialectical and practical commitments. In *Proc. 23rd Conf. Artif. Intell. (AAAI)*, pp. 176–181, 2008.

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

[38] Transaction workflow innovation standards team, 2006. http://www.twiststandards.org.

[39] M. Venkatraman, M. P. Singh. Verifying compliance with commitment protocols. *Auton. Agents & Multi-Agent Systems*, 2(3):217–236, 1999.

[40] M. Winikoff. Implementing commitment-based interaction. In *Proc. 6th Intl. Joint Conf. on Auton. Agents & MultiAgent Systems (AAMAS)*, pp. 868–875, 2007.

[41] WS-CDL. Web services choreography description language version 1.0, Nov. 2005. www.w3.org/TR/ws-cdl-10/.

[42] D. M. Yellin, R. E. Strom. Protocol specifications and component adaptors. *ACM Trans. Programming Languages & Systems*, 19(2):292–333, 1997.

[43] P. Yolum, M. P. Singh. Flexible protocol specification and execution. In *Proc. 1st Intl. Joint Conf. Auton. Agents & MultiAgent Systems*, pp. 527–534. 2002.

[44] C. Zirpins, W. Emmerich. A reference model of virtual service production networks. *Service Oriented Comput. & Apps.*, 2(2–3):145–166, 2008.