Colaba: Collaborative Design of Cross-Organizational Processes

Amit K. Chopra
University of Trento
chopra@disi.unitn.it

Munindar P. Singh
North Carolina State University
singh@ncsu.edu

Abstract—**Cross-organizational processes naturally involve multiple stakeholders with distinct business interests. Yet, current process modeling approaches are conceptually centralized. This paper develops *Colaba*, a novel approach for the design of a cross-organizational process. Colaba naturally accommodates the requirements of multiple stakeholders. It is realized as a tool and methodology that uses and maintains a repository of business protocols, each describing a business function to the desired level of nuance. Design in Colaba is thus simply a matter of selecting, refining, extending, and composing protocols.**

Colaba is based on the concepts of *Issue-Based Information Systems*. It naturally supports stakeholders proposing process models, raising issues about them, and evaluating alternative proposals. Colaba contributes argumentation primitives and a representation relevant to business processes that accommodates the diverse perspectives of the roles in a protocol. This paper includes an evaluation of Colaba via a case study.

Index Terms—**Requirements, Argumentation, Interaction, Protocols, Commitments**

I. Introduction

We focus our attention on cross-organizational business processes, which span organizational boundaries, and thus involve multiple stakeholders that are mutually *autonomous* (as in independent business entities) or function as if they were autonomous (as in the business units of the same enterprise). Autonomy here means that the stakeholders potentially have conflicting motivations and interests, and thus conflicting perspectives on the business process they would jointly conduct. We are not concerned directly with the local implementations that together constitute a cross-organizational business process. A participating organization could have well-defined processes of its own, apply hard-coded functionality, or use highly flexible agents.

What concern us instead are the specifically cross-organizational aspects of cross-organizational business processes, namely, how the participants interact with one another. The importance of properly capturing such interactions is now widely recognized, for example, in leading standardization efforts such as Health Level 7 (better known as HL7), RosettaNet, and TWIST (for foreign exchange and other financial transactions). The interest in interactions has led to the attention paid to choreography specification languages, such as WS-CDL and BPMN. However, prevalent choreography approaches focus primarily on the occurrence and ordering of messages.

In contrast, we adopt an approach centered on *(business) protocols*, which have been advocated by several researchers [1], [2], [3]. Our approach explicitly captures the business meanings of the messages that the parties exchange. This approach yields greater confidence in the correctness of the interaction being designed as well as increased adaptability both to changing requirements (at design time) and exceptional events (at run time).

Current cross-organizational processes suffer from two shortcomings, which can be attributed to how they are designed. First, these processes tend to be rigid and over-specified. For example, an industry implementation of Oracle's Quote to Cash process ends up with just one legal execution path, in which the participants exchange messages in sequence. Second, existing processes do not properly accommodate the preferences of their multiple stakeholders. Even when one is not interested in designing a process from scratch, it is a nontrivial matter for a prospective participant to be able to identify a process that would meet its requirements.

Both of the above shortcomings trace back to two culprits: (1) today's conceptual metamodels for processes and (2) today's design activities in specifying processes. First, today's metamodels require an emphasis on message ordering and occurrence because that is primarily what they specify (besides message formats). Not only is such a low-level representation difficult to reason about, it also precludes discussions about the business aspects of greatest relevance to the participants. As a result, process designers whether working for one organization or several organizations find it simplest to over-constrain the specification—so they can obtain the requisite confidence in its correctness. Second, the prevalent design methodologies naturally take the perspective of one party. As a result, one party may impose its local constraints or even historical ways of conducting business upon the cross-organizational process as a whole. Moreover, that one party would naturally have a simplified view of how the others interact, and this too disregards relevant possibilities for the others. Thus existing design methodologies tend to unnecessarily constrain the specifications.

In contrast, this paper proposes *Colaba*, a framework, vocabulary, and tool for collaborative business process design. Colaba enables each stakeholder to motivate her own proposals, articulate her concerns about alternative proposals, and suggest suitable solutions. Further, Colaba stores the proposals and the arguments about them, pro or con, in a repository to facilitate reuse in subsequent design engagements.

Colaba exploits the natural match between protocols and collaborative business process design: stakeholders are the loci of autonomy *both* during design and enactment. Thus stakeholders use Colaba to review existing protocols along with the issues raised about them by previous stakeholders playing a desired role, compose protocols in novel ways, and express and evaluate their concerns with the process being designed.

To understand Colaba's new contribution, it helps to consider the main elements of an argumentation-based design exercise.

Argumentation scheme that the stakeholders must follow. The argumentation scheme defines how an argument is initiated, progresses, and terminates. The argument proceeds as the stakeholders raise or address *issues*, each issue being an evaluation of a proposal with respect to some criterion of interest to a stakeholder.

Bases for argumentation, that is, the content of the arguments. The bases correspond to the ontology using which stakeholders may raise issues. In Colaba, this ontology is specific to collaborative design of cross-organizational process. Several sections of it are independent of the business domain, such as manufacturing or IT. Additionally, Colaba supports creating repositories of issues raised by the participants that are specific to a business domain.

Both the argumentation scheme and bases are necessary to ensure coherent, well-focused interactions among the stakeholders. Identifying the bases is particularly challenging. Too many bases, and the argumentation is likely to be complex, time-consuming, and most likely incoherent. Too few bases, and the argumentation is unlikely to unearth any interesting issues, and thus unlikely to help reconcile conflicting perspectives.

Contributions: We make the following contributions. First, our main contribution is that we delineate the bases of argumentation geared for facilitating cross-organizational business process design. The bases serve as guidance to the stakeholders regarding the points they must argue about to arrive at a *well-considered* process design, and how they ought to go about arguing. Second, we have realized our approach in a tool wherein the bases are reflected in suitable widgets and templates. Third, Colaba features a repository: stakeholders may use existing protocols and recorded past experience to inform their choices in creating new protocols from scratch or (ideally) by creating new variants of existing protocols, or by composing protocols from existing protocols. Fourth, Colaba supports a variety of natural queries through which a designer may identify protocols, the issues that have been raised about them, and how such issues have been resolved.

Organization: The rest of the paper is organized as follows. Section II introduces the essential background on argumentation and business protocols. Section III describes the main elements and usage of the Colaba tool. Section IV outlines a methodology for collaborative process design, delineating the bases of protocol-centric argumentation, a classification of issues, the kinds of reasoning one can perform about in Colaba about its repository of protocols. Section V discusses how Colaba applies to a real-world example. Section VI summarizes our contributions with respect to the relevant literature and indicates some directions for future research.

II. Background: Commitments and Protocols

A commitment C(*debtor*, *creditor*, *antecedent*, *consequent*) means that if the antecedent holds, then the creditor commits to bringing about the consequent. A commitment is *detached* when its antecedent holds, and *discharged* when its consequent holds. A detached commitment is, in effect, unconditional—meaning that it is fully in force. To a first approximation, an active commitment corresponds to a directed conditional obligation from the debtor to the creditor. However, a commitment differs from an obligation in that it arises within an organizational context, which constrains its applicability [4]. Moreover, a commitment may be manipulated by performing a variety of natural operations. These are CREATE, DISCHARGE (satisfy and terminate, nominally by the debtor), CANCEL (terminate, possibly in failure, by the debtor), and RELEASE (terminate by nullifying, by the creditor), DELEGATE (change debtor, performed by the current debtor) and ASSIGN (change creditor, performed by the current creditor).

Commitments and their manipulations are central to declaratively encoding the business meaning of a process. Specifically, commitments help capture the business meanings of the messages that the stakeholders exchange in the process of interest. For example, we can formalize an offer message as creating a commitment from a merchant to a customer that if the customer agrees to paying the specified amount, the merchant will deliver the specified goods (the payment amount and goods being expressed by the contents of the message). And, we can formalize a purchase order message from a customer to a merchant as a commitment from the customer to pay the merchant the specified amount provided the merchant will deliver the specified goods. Reasoning about commitments enables flexible enactment and modeling of business processes, because it draws attention to the meanings of the processes and away from arbitrary execution constraints.

In particular, the above formalization leaves open the possibility that a customer may make its payment through a bank, in essence, delegating its commitment to make a payment to the bank. Likewise, the merchant may delegate its delivery commitment to a shipping company. In this manner, commitments facilitate specifying a multiparty cross-organizational process without prematurely focusing on the operational details. However, operational details that are relevant to one or more of the participants can be captured through the notion of protocols.

A. Business Protocols

A business protocol is a grouping of interactions related to a specific business purpose. For example, one can easily imagine a protocol *Offering* for ordering items from a catalog, a protocol *Payment* for processing payments, and a protocol

Shipping for shipping goods. A protocol declares the roles involved in the interactions, and the messages that agents playing the roles may send to each other; for example, *Offering* may declare the messages *rfq* (from customer to merchant), *quote* (from merchant to customer), and *accept* (from customer to merchant).

Thus, the message *quote* may have the meaning that the customer commits to paying for the goods if the merchant delivers the goods. A protocol may specify an ordering constraint between messages—for example, *Offering* may state that a *quote* for goods may be sent only if an *rfq* for the specified goods has been received.

B. Protocol Composition

What makes protocols interesting is that they are reusable, modular specifications of interaction, and may be readily composed. For example, *Offering*, *Payment*, and *Shipping* protocols may be suitably composed to produce a *Purchase* protocol. Composition is achieved by using the so-called *composition axioms*. Colaba supports the following axiom schemas (based on those specified by Desai et al. [5]).

Role identification. Each role in the composed protocol must be identified with roles in the constituent protocols. For example, the role purchaser in *Purchase* is identified with the roles customer in *Offering*, payer in *Payment*, and shippee in *Shipping*.

Term alignment. Independently specified protocols are likely to use different terms to means the same thing. In such cases, the composer must suitably align the terms. For example, let *quote* in *Offering* mean $\mathsf{C}(\mathit{merchant}, \mathit{customer}, \mathit{pay}, \mathit{goods})$. Now pay is expected to be brought about in the *Payment* protocol. However, *Payment* may refer to that event by the term *paymentMade*. Then, in *Purchase*, we use a term alignment axiom to say that *pay* is said to occur when *paymentMade* occurs.

Event order. A composition might state that an event in one protocol must happen before another in another protocol. For example, in *Purchase*, it might be reasonable to state that *goodsShipped* in *Shipping* occurs before *paymentMade* in *Payment*. In essence, the composer asserts a constraint on the allowed enactments of the composed protocol.

III. The Colaba Tool

Colaba is built on top of Shum's Compendium tool [6], a general graphing tool for facilitating discussions among stakeholders. Colaba is set up as an application of Compendium with node and edge types specialized to collaborative process design based on protocols.

We now show a screenshot of Colaba followed by the concepts that the screenshot demonstrates. Figure 1 shows the *Offering* specification in the right pane as a graph using the constructs described above. The left pane shows a repository being browsed and edited, and shows the protocols in a hierarchy. The right pane also shows attributes of the selected protocol, for example, that the protocol is *mutual commitment* (via the *quote* and *accept* messages). These attributes may be used to search the repository for protocols.

With each node type we associate a *template* that describes a frequently used pattern. Dropping a node type into a workspace effectively introduces the graph corresponding to the template associated with the node type into the workspace. Colaba defines a *stencil*, akin to a palette of tools. A user may drag and drop a node type from the Colaba stencil into a workspace to quickly create process designs. The Colaba stencil contains the following node types (here, written in the form ⟨SMALL CAPS⟩).

- Protocol concepts: ⟨PROTOCOL⟩, ⟨ROLE⟩, and ⟨MESSAGE⟩.
- Commitment operations: ⟨CREATE⟩, ⟨DISCHARGE⟩, ⟨CANCEL⟩, ⟨RELEASE⟩, ⟨DELEGATE⟩, and ⟨ASSIGN⟩.

In addition, Colaba defines a *link group* consisting of labeled edge types customized for linking the node types from the Colaba stencil. The link group contains the following directed edge types corresponding to the above-mentioned protocol composition axioms (here, written in the form $\overrightarrow{\text{SMALL CAPS}}$).

$\overrightarrow{\text{MEANS}}$: from ⟨MESSAGE⟩ to a commitment operation node type. In addition, each message m carries the implicit meaning that the exchange of message m causes the proposition named m to hold (thus for example, the message PAY causes the proposition pay to hold). The holding of m corresponds to the exchange of m, which we treat as an atomic event at the level of protocols.

$\overrightarrow{\text{ORDER}}$: from ⟨MESSAGE⟩ to ⟨MESSAGE⟩. This edge type specifies that the source message must precede the sink message.

$\overrightarrow{\text{ROLEIDENTIFICATION}}$: from ⟨ROLE⟩ to ⟨ROLE⟩. This edge type specifies that the sink is identified with the source.

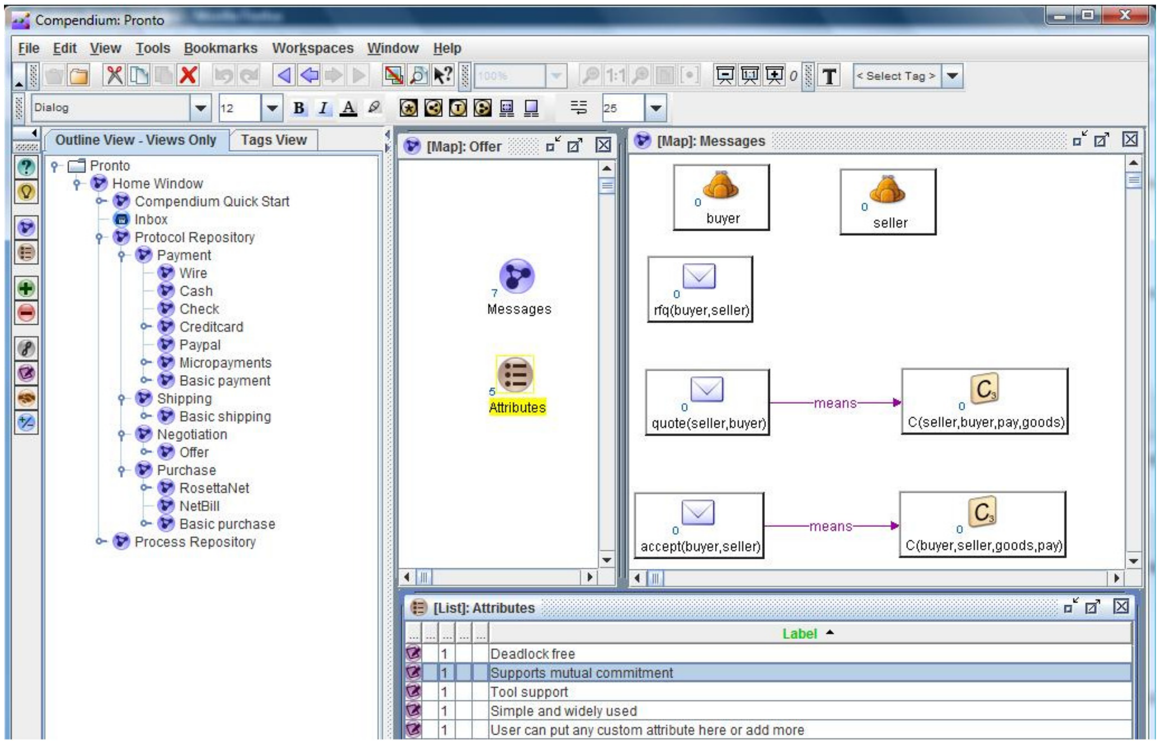
$\overrightarrow{\text{TERMALIGNMENT}}$: from ⟨MESSAGE⟩ to ⟨MESSAGE⟩.

Colaba users can explore or create and modify repositories of protocols. A protocol repository is structured by specialization, and organized as a tree. For example, *CreditCard* payment protocols and *Micropayment* protocols represent specializations of *Payment*, and hence appear as children of *Payment* in the repository. Some of the protocols may be abstract, which do not declare any messages or roles, and only serve to group their children.

Using Colaba, designers may create new protocols from scratch or by composing protocols that already exist in a repository. A composed protocol *refers* to existing protocols. The composition is achieved by using the edge types introduce above.

IV. Argumentation in Colaba

This section describes how the stakeholders in a business process arrive at a well-considered design for a business process. First, we describe how the argumentation is set up. Second, we introduce the argumentation schemes that the stakeholders follow. Third, we delineate the bases of argumentation employed in Colaba. The argumentation scheme



	Label
1	Deadlock free
1	Supports mutual commitment
1	Tool support
1	Simple and widely used
1	User can put any custom attribute here or add more

Fig. 1. Repository and the *Offering* protocol in detail

described here constitutes an outline of the proposed Colaba methodology for cross-organizational process design.

A. *Steps 1 and 2: Setting up Argumentation*

S1. Define a new collaborative workspace for the target business process. Identify the stakeholders. State the purpose of the desired business process. Identify the roles in the business process.

For example, let the desired business process *Purchase-BP* be one to facilitate the sale of silicon wafers; it has two roles, seller and buyer. The stakeholders in the process are a wafer producer Ace Semiconductors and a chip manufacturer NanoCorp. The process as an abstract entity is a protocol that describes the interactions of the stakeholders.

S2. The stakeholders adopt roles in the target process.

Ace Semiconductors acts as seller and NanoCorp as buyer. The stakeholders carry out the argumentation while wearing their respective *role hats*. Colaba associates the arguments with the process roles of the stakeholders who make those arguments. This knowledge is crucial to the value of a protocol repository. A future stakeholder who considers the arguments pro and con for a particular proposal can do so from the perspective of an appropriate role: the role it is contemplating adopting or the role of a counterparty. For example, a new seller might benefit from knowing the objections or support presented for a protocol by another seller as well as how a buyer may respond to its proposals.

B. *Steps 3 to 6: Argumentation Scheme*

Let us define the argumentation scheme by which stakeholders interact in Colaba. This scheme applies to an individual design episode. It is a feature of Colaba that it stores design rationales in the form of arguments in its repository. Thus a proposal that originates in a particular design episode may be applied in a subsequent episode; in the latter episode, a stakeholder may raise a new issue about a proposal that was settled in the earlier episode. Although the set of parties who participated in an earlier episode would in general not overlap with the set of parties in the current episode, the arguments created and stored in the prior episode might inform the current episode.

S3. A process role makes a *proposal* for realizing the process.

The proposal consists of a protocol that realizes the desired process, along with the protocol roles to be adopted by the stakeholders. Additionally, the proposing role may give justifications for her proposal. Let's say seller proposes using the protocol *Purchase* for the purchase process, and that it itself should adopt the producer role in *Purchase* whereas the buyer should adopt the consumer role in *Purchase*.

S4. A stakeholder may either accept an open proposal or raise an issue with it. If all stakeholders accept the proposal, then we have a winner and no more argumentation is needed.

S5. If a role raises an issue in Step S4, then the stakeholders recursively argue about the issue raised.

S6. The participants may repeat Steps S3, S4, and S5 as often as they desire.

Colaba does not impose any other restrictions upon the argumentation scheme, for example, it does not limit who may make an argumentation move when. The steps above are quite high-level; in general, stakeholders may compare proposals, search the repository and create new protocols in the process of argumentation.

C. Bases of Argumentation

Issues that may be raised by a stakeholder are of two types: *micro* and *macro*. Micro issues are those which are raised with respect to some specific element of the proposal—for example, with a role or message in a protocol, or perhaps a composition axiom. Macro issues cut across elements, and have more to do with the overall nature of the proposal. Macro issues are high-level issues; micro issues are raised in the context of a macro issue. Thus the separation between the two facilitates a top-down issue-based argumentation.

1) Macro Issues: Let us first delineate the macro issues. We identify three classes of macro issues that form the basis for argumentation. These derive from Singh et al.'s [3] classification of the elements of a business service engagement: transactional, structural, and contextual. Figure 2 depicts how these requirements relate to the architectural elements of a business process.

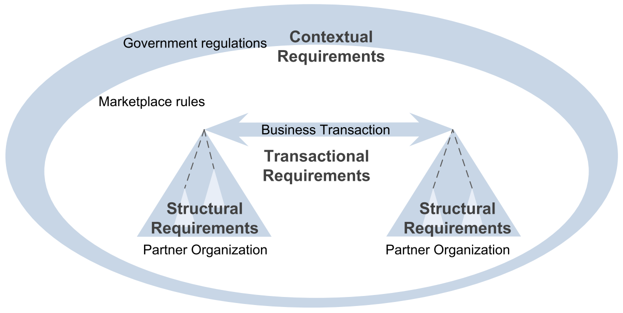


Fig. 2. Three elements of requirements of cross-organizational business processes [3]

Transactional or the *what* issues The purpose of a business process is to facilitate some business transaction. A process role may have an issue with the transaction that the proposal supports. For example, *Purchase-BP* facilitates the buying and selling of silicon wafers. The buyer may find issue with a proposal for *Purchase-BP* if the proposal does not support refunds for purchases that are damaged in transit.

Some transactional issues may be nonfunctional. These issues range from concerns such as efficiency, complexity, adoption by others, and backward compatibility, to formal properties such as termination and absence of deadlock, collectively termed *ilities* by Filman et al. [7]. For example, the buyer may raise the issue that in *Purchase-BP*, the message schemas of the *Purchase* protocol are not standardized.

Structural or the *who* and *how* issues These fall into two main categories. A process role may raise any or both of the two.

First, the plainly *who* issues concern which process roles may play which protocol roles. For example, suppose *Purchase-BP* also had a role escrow. The stakeholder who plays seller may have an issue with *Purchase-BP* if the same stakeholder plays both escrow and buyer.

Second, the *how* (and indirectly *who*) issues concern whether new stakeholders may be brought in through delegation or assignment of commitments. By delegating or assigning a commitment to another party, a stakeholder brings such a party into the transaction, and thus changes the structure of the business process. Another stakeholder may have an issue with such changes. For example, consider the outsourcing of shipping by the seller. Such outsourcing is naturally modeled in terms of delegating the commitment to deliver goods. However, outsourcing may not be acceptable to the buyer. An example of assignment commonly encountered is when a service provider Alltapped takes over service provider Veriphone—all the customers of Veriphone are assigned to Alltapped, meaning that the customers are now committed to Alltapped to pay for the service.

Contextual or the *where* issues The context of the interaction is the social or legal setting in which is the interactions are presumed to happen. It defines the rules of encounter to which the business process is subject.

For example, the seller may argue that the context be a marketplace such WaferValley, which the seller knows to have seller-favorable policies in the past. The buyer on the other hand, may argue for a more broadly accepted setting such as the Uniform Commercial Code (UCC). Some contexts are often implicit, such as an overarching legal one. Contexts are particularly relevant for commitments. As mentioned earlier, commitments exist in a certain context. This means that the context outlines policies that govern the eventualities where commitments are violated. These policies in turn can be modeled as commitments where the context is the debtor.

2) Micro Issues: We identify the following kinds of micro issues related to business processes.

Protocol Role-Specific. A process role has issue with the roles in the protocol. The role may propose fewer or additional protocol roles. For example, the buyer may argue that an escrow role is required.

Message Meaning. A process role has issue with the meaning of some message. For example, seller may argue that the message *quote* does not mean any commitment to provide goods on its part; the seller only considers itself committed upon an acceptance of *quote* from the buyer. In addition, a role can argue for a stronger or weaker commitment. For example, the buyer may argue that *quote* means the commitment C(seller, buyer, *pay*, *goods* $\wedge$ *futureDiscount*), whereas the merchant may argue that *quote* only means C(seller, buyer, *pay*, *goods*)—no discount—which is a weaker commitment.

Term Alignment. An argument that the term alignment is not what it should be.

Event Order. An argument that an event order axiom is either needed or not needed, or is not what it should be. For example, seller argues that payment must occur before the shipment does.

D. Reasoning about Arguments

A key advantage of organizing the protocols (our design artifacts) and arguments about such artifacts according to roles is to facilitate reasoning about the artifacts and arguments in a uniform framework. This is an important point of distinction between Colaba and existing approaches, so we describe it further here. Fundamentally, there is no limit to the kinds of reasoning one might perform. We give examples of some important types of reasoning.

- Based on organizational role. A stakeholder contemplating a particular role in a protocol may wish to understand the protocol in light of the contemplated role. And a stakeholder may wish to understand the perspective of its contemplated counterparties as well. To this end, queries such as the following are immediately of value:
  - *Find all issues raised by a role*. For example, find all issues raised by a buyer.
  - *Find all issues raised by a role that have been resolved (or not yet resolved)*. For example, find all issues raised by a buyer that are still open.
- A stakeholder may wish to identify past domain-independent issues with a view toward addressing them in the process being designed currently. Using Colaba's taxonomy of issues, the stakeholder may query for how issues of a particular type are resolved. Some examples include the following:
  - *Find all issues that pertain to the meaning of a specified message in the specified class of protocols*. For example, find all issues regarding whether a quote message indicates a commitment to sell the specified goods at the specified price.
  - *Find all issues dealing with event order*. For example, find all issues regarding whether goods should be delivered prior to payment.
  - *Find all issues dealing with conflict of interest or separation of duties*. For example, find all issues regarding whether a delivery confirmation can be given by the shipment agency itself.
- A stakeholder may wish to identify and address domain-specific issues that it has historical or other reasons to be concerned about. Using a separately created taxonomy of domain-specific issues, the stakeholder may query regarding issues of a particular type. Some examples include the following:
  - *Find all issues that pertain to security*. For example, find all issues about the potential loss of goods if they are delivered by simply dropping them off on the *recipient*'s doorstep. And, find all issues about the potential payment fraud when using a credit card for a foreign transaction.
  - *Find how all issue that pertain to security of patient information are resolved in protocols geared for health care laboratory processing*. For example, find all issues regarding security of patient information along with the resolution for each such issue.
- Combinations of the above are natural. Further, when we identify an issue, we also identify all the ways in which it was addressed and resolved the previous design episode.

Currently, Colaba supports both plain text queries and structured ones. We have developed our own schema for protocols and arguments, and have programmed some queries in XQuery to run against instances of this schema. The structured queries are not supported in the tool itself because Compendium does not support custom schemas. Integration of the schema with the tool remains a key future direction.

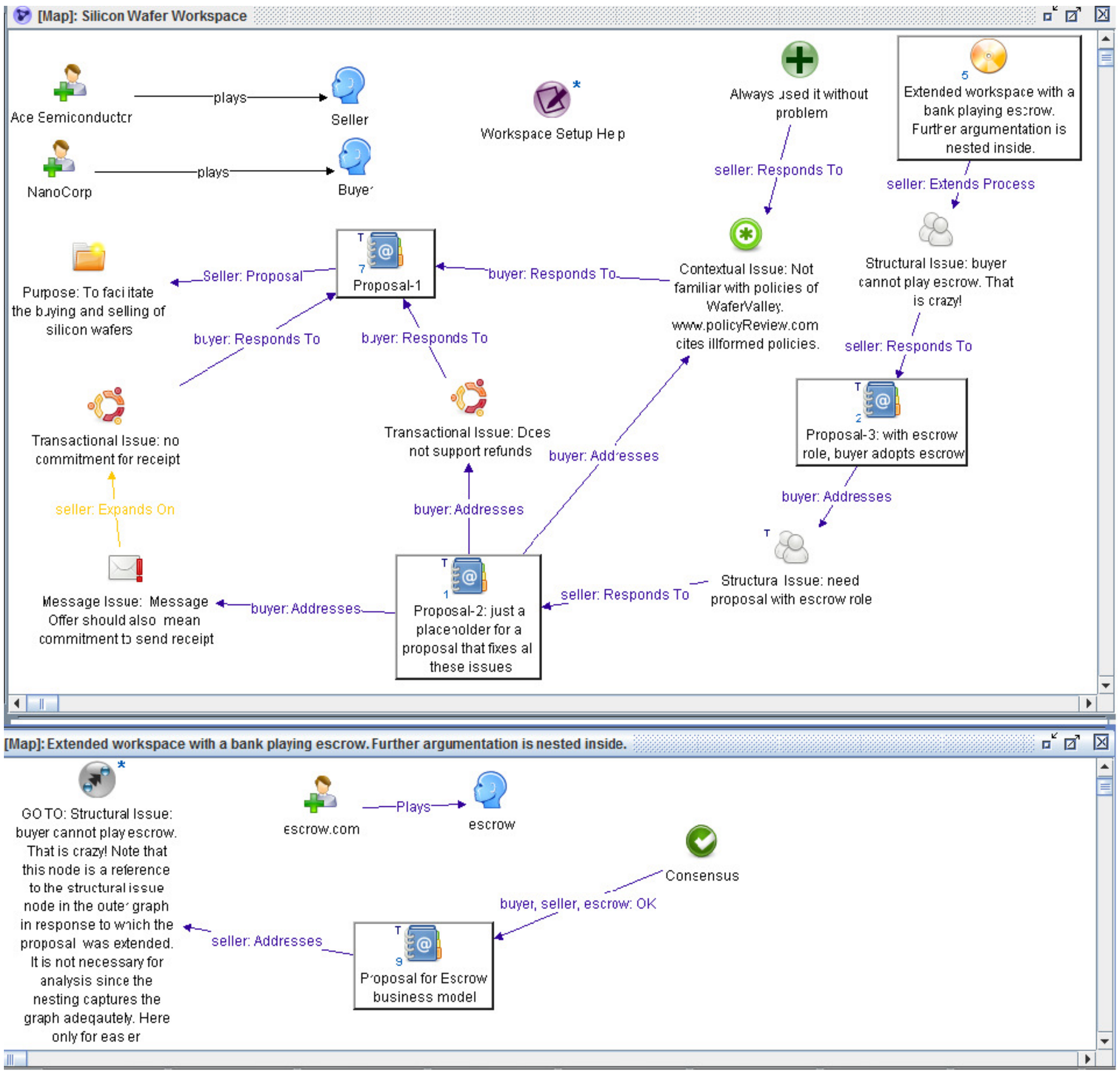
V. Application

We now present a more elaborate application of the argumentation related concepts introduced earlier. Figure 3 shows the argumentation between two organizations Ace Semiconductors and NanoCorp. Ace Semiconductors is a producer of silicon wafers used in the manufacturing of semiconductors. NanoCorp is manufacturer of semiconductors. The two organizations want to collaboratively design a business process for the buying and selling of silicon wafers—as the node labeled Purpose in Figure 3 shows. Ace Semiconductors and NanoCorp adopt the role of seller and buyer respectively in the argumentation. The argumentation in Figure 3 then proceeds as follows.

The seller makes a proposal labeled *Proposal-1* that (presumably) addresses the purpose of the argumentation. The proposal itself is a composite structure consisting of some protocol from the repository, the context, and the roles in the protocol that seller and buyer would adopt in the protocol. The node *Proposal-1* may be elaborated further by clicking upon it; for reasons of space, we do not show the details here. In this particular case, the selected protocol is *Purchase* and the context is *WaferValley*, presumably an organization with expertise in such silicon wafer transactions.

The buyer responds to the proposal with three issues. One, a *transactional* issue that in the proposed protocol the seller makes no commitment to provide a receipt of the transaction. Two, a *transactional* issue that the protocol does not support refunds. Three, a *contextual* issue that she is not familiar with the policies of *WaferValley*. The buyer even provides reference to a web site that cites *WaferValley* for having ill-formed business policies.

The buyer then goes on to address the issues by making a new proposal *Proposal-2*. However, the seller now brings up a *structural* issue with the buyer's proposal, namely, that the protocol therein does not support escrow.


[Map]: Silicon Wafer Workspace

Ace Semiconductor

NanoCorp

Seller

Buyer

Workspace Setup Help

Always used it without problem

Extended workspace with a bank playing escrow. Further argumentation is nested inside.

Purpose: To facilitate the buying and selling of silicon wafers

Proposal-1

Contextual Issue: Not familiar with policies of WaferValley. www.policyReview.com cites illformed policies.

Structural Issue: buyer cannot play escrow. That is crazy!

Transactional Issue: no commitment for receipt

Transactional Issue: Does not support refunds

Proposal-3: with escrow role, buyer adopts escrow

Structural Issue: need proposal with escrow role

Message Issue: Message Offer should also mean commitment to send receipt

Proposal-2: just a placeholder for a proposal that fixes all these issues

[Map]: Extended workspace with a bank playing escrow. Further argumentation is nested inside.

GO TO: Structural Issue: buyer cannot play escrow. That is crazy! Note that this node is a reference to the structural issue node in the outer graph in response to which the proposal was extended. It is not necessary for analysis since the nesting captures the graph adeqautely. Here only for easer

escrow.com

escrow

Consensus

Proposal for Escrow business model

Fig. 3. Ace Semiconductors and NanoCorp engage in issue-based argumentation over the design of a business process for buying and selling silicon wafers

To address the issue of escrow, the buyer makes a new proposal, *Proposal-3*. The protocol proposed therein contains an escrow role, and the buyer proposes that she herself play the role of escrow. The seller responds to this proposal with another *structural* issue, namely, that the buyer and the escrow roles cannot be played by the same party in any business transaction. The seller then makes a proposal of his own that addresses this issue. In the figure, though, the details are nested inside a new extended workspace (top right corner, shown expanded in the bottom window)—a feature for argumentation management.

In this particular example, the seller and the buyer eventually arrive at a mutually acceptable proposal; in another instance, it may happen that they cannot. Nonetheless, stakeholders would be better off at resolving conflicts armed with Colaba than without.

This argumentation instance, like all others, becomes part of the repository, and is available for future reference. Subsequently, new organizations who have a similar purpose in mind may exploit this knowledge for better designing their business processes, especially when guided by past practice and design rationales.

VI. Discussion

This paper presents a tool-supported collaborative approach for designing business processes. Our approach features a repository of protocols, the artifacts upon which business processes are based. The stakeholders in a business process argue about the merits of proposed business processes, and exploit knowledge stored in the repository about prior argumentation instances. In employing stakeholder oriented argumentation, our approach reflects the openness of the settings in which business interactions happen.

Collaborative business process design is garnering active research attention from various standpoints. Important examples include Seshasai et al. [8] for knowledge engineering and Decker et al. [9] for conflict resolution (and achieving consensus) among stakeholders. Balasubramaniam and Dhar's [10] and Shum et al.'s [11] point on the importance of capturing stakeholder views for requirement analysis is now commonly accepted as crucial. Colaba brings together all of the above considerations in a uniform framework. In particular, Colaba supports gathering and applying business knowledge and best practices by providing a structured repository of protocols and process designs. Colaba not only helps in settings where one party dominates but also facilitates the expansion of process technologies where there is no clear-cut dominant participant. Although we apply argumentation to commitment protocols, the methodological elements are general enough to be applied to other forms of business process specification such as BPMN.

Above all Colaba emphasizes the role of communication in software engineering. The arguments are the communications of the stakeholders, and the software artifact being designed, here the business protocol, evolves with the communications. Traditionally RE approaches have been more concerned with an analysis of the requirements model—the product of communication—than communication among stakeholders and how it affects requirements. Argumentation has previously been considered to be an important part of business process engineering, e.g., due to Yu and Mylopoulos [12], but in newer newer work (for example, Bresciani et al. [13]) takes a back seat to modeling and analysis.

Mahfouz et al. [14] describe a methodology for collaborative choreography design, in which too the participants make proposals for changes to the choreography being designed. Whereas their methodology focuses on the goal models of participants, our approach focuses solely upon the public aspects of collaborative design, that is, the raising of issues and argumentation, and does so in a more comprehensive manner.

Recent work has sought to apply commitments in conjunction with the kinds of goal models used in RE. Chopra et al. [15] model service-oriented systems as a combination of agents (whose rationales are expressed in terms of goals) and business protocols. They consider agents and protocols as independent software engineering artifacts. Ali et al. [16] make the case for replacing intentional dependencies of Tropos with commitments. Colaba is complementary to them in that it informs the specification of protocols.

In modeling the communication among stakeholders via argumentation, Colaba treats software engineering as an application—just the same as healthcare, insurance, and other applications that involve autonomous interacting parties. The communication itself could be given a semantics in terms of dialectical commitments on the part of the stakeholders [17]. Thus, for example, when a stakeholders says "the purchase protocol must support escrow because transactions involve big transfers of money between untrusted parties", he is committing dialectically (to the other stakeholders) to that position. An interesting direction would be to explore the connection between such commitments and the artifact being designed in order to support conformance against the requirements communicated by the stakeholders.

Acknowledgments

This research was partially supported by a grant from the Intel Research Council at NCSU and by a Marie Curie Trentino fellowship at Trento. We are indebted to Mark Klein, Eleanor Wynn, and Nirmit Desai for valuable discussions early in this effort.

References

[1] M. Baldoni, C. Baroglio, and E. Marengo, "Commitment-based protocols with behavioral rules and correctness properties of MAS," in *Proceedings of the International Workshop on Declarative Agent Languages and Technologies (DALT)*, 2010, pp. 66–83.

[2] C. Cheong and M. P. Winikoff, "Hermes: Designing flexible and robust agent interactions," in *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, V. Dignum, Ed. Hershey, PA: IGI Global, 2009, ch. 5, pp. 105–139.

[3] M. P. Singh, A. K. Chopra, and N. Desai, "Commitment-based service-oriented architecture," *IEEE Computer*, vol. 42, no. 11, pp. 72–79, Nov. 2009.

[4] M. P. Singh, "An ontology for commitments in multiagent systems: Toward a unification of normative concepts," *Artificial Intelligence and Law*, vol. 7, no. 1, pp. 97–113, Mar. 1999.

[5] N. Desai, A. K. Chopra, and M. P. Singh, "Amoeba: A methodology for modeling and evolution of cross-organizational business processes," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 19, no. 2, pp. 6:1–6:45, Oct. 2009.

[6] Http://compendium.open.ac.uk/institute/index.htm.

[7] R. E. Filman, S. Barrett, D. D. Lee, and T. Linden, "Inserting ilities by controlling communications," *Communications of the ACM*, vol. 45, no. 1, pp. 116–122, 2002.

[8] S. Seshasai, A. Gupta, and A. Kumar, "An integrated and collaborative framework for business design: A knowledge engineering approach," *Data and Knowledge Engineering*, vol. 52, no. 1, pp. 157–179, Jan. 2005.

[9] B. Decker, J. Rech, K.-D. Althoff, A. Klotz, E. Leopold, and A. Voss, "eParticipative process learning—process-oriented experience management and conflict solving," *Data and Knowledge Engineering*, vol. 52, no. 1, pp. 5–31, Jan. 2005.

[10] B. Ramesh and V. Dhar, "Supporting systems development by capturing deliberations during requirements engineering," *IEEE Transactions on Software Engineering*, vol. 18, no. 6, pp. 498–510, Jun. 1992.

[11] S. J. B. Shum, A. M. Selvin, M. Sierhuis, J. Conklin, C. B. Haley, and B. Nuseibeh, "Hypermedia support for argumentation-based rationale," in *Rationale Management in Software Engineering*, A. H. Dutoit, R. McCall, I. Mistrik, and B. Paech, Eds. Berlin: Springer, 2006, ch. 5, pp. 111–132.

[12] E. S. K. Yu and J. Mylopoulos, "Using goals, rules and methods to support reasoning in business process reengineering," *International Journal of Intelligent Systems in Accounting Finance and Management*, vol. 1, no. 5, pp. 1–14, 1996.

[13] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004.

[14] A. Mahfouz, L. Barroca, R. Laney, and B. Nuseibeh, "Requirements-driven collaborative choreography customization," in *Proceedings of the Seventh International Joint Conference on Service Oriented Computing*, 2009, pp. 144–158.

[15] A. K. Chopra, F. Dalpiaz, P. Giorgini, and J. Mylopoulos, "Modeling and reasoning about service-oriented applications via goals and commitments," University of Trento, TR DISI-09-068, dec 2009.

[16] R. Ali, A. K. Chopra, F. Dalpiaz, P. Giorgini, J. Mylopoulos, and V. E. S. Souza, "The evolution of Tropos: Contexts, commitments and adaptivity," in *In Proceedings of the 4th International iStar Workshop*, ser. CEUR-WS, vol. 586, 2010, pp. 15–19.

[17] M. P. Singh, "Semantical considerations on dialectical and practical commitments," in *Proceedings of the 23rd Conference on Artificial Intelligence*, 2008, pp. 176–181.