A Semantic Approach for Designing Business Protocols [*]

Ashok U. Mallya
aumallya@ncsu.edu

Munindar P. Singh
singh@ncsu.edu

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7535, USA

ABSTRACT

Business processes involve interactions among autonomous partners. We propose that these interactions be specified modularly as protocols. Protocols can be published, enabling implementors to independently develop components that respect published protocols and yet serve diverse interests. A variety of business protocols would be needed to capture subtle business needs. We propose that the same kinds of conceptual abstractions be developed for protocols as for information models. Specifically, we consider (1) *refinement*: a subprotocol may satisfy the requirements of a superprotocol, but support additional properties; and (2) *aggregation*: a protocol may combine existing protocols. In support of the above, we develop a formal semantics for protocols, an operational characterization of them, and an algebra for protocol composition.

Categories and Subject Descriptors: D.1.3 [Software]: [distributed programming]; I.2.11 [Computing Methedologies]: [multiagent systems]

General Terms: Management, Design

Keywords: Business Process Composition, Web Services, Commitments

1. INTRODUCTION AND MOTIVATION

Business processes involve interactions among heterogeneous and autonomous components. For cross-enterprise processes, the components may represent mutually competitive interests. Web services provide a basis for realizing processes by standardizing component interfaces, but are too low a level of abstraction for encoding realistic business scenarios. This has led to an interest in technologies such as coordination and process flows, business process management, distributed transactions, and conversations. While these approaches have some benefits, they suffer from the major shortcoming that they do not enable the construction of complex processes. They mostly take a centralized perspective, akin to workflow technologies, viewing a process as a series of tasks to be performed. This proves too tedious for reliable modeling and too rigid for enactment, which is the reason workflow technologies have been considered a failure in many practical settings.

We propose a novel framework for thinking about processes. Simply put, a process instantiates one or more business protocols among designated parties. We take inspiration from popular networking and distributed systems protocols such as IP, TCP, HTTP, and SIP. These protocols provide published specifications of the roles of the parties interacting through them. Thus they enable implementors to independently create components that, if compliant, are guaranteed to work together. If we could specify *business protocols* precisely, we could enable the desired business processes in which the autonomy of the participants is maximized (limited only by the requirements of the protocols in effect) and their implementations are heterogeneous. Components could be plugged in or out with ease and processes could be readily adjusted so that the participants can handle exceptions and exploit emerging opportunities.

Whereas networking and distributed systems need only a handful of universal protocols, realistic business settings would need an endless variety of business protocols. We want general protocols to support flexibility, and specialized protocols to support efficiency, security, or risk management. Ultimately, each environment may have a set of desirable protocols. For example, we can imagine a generic payment protocol as well as specializations of it such as payment by cash, credit card, checks, travelers checks, bank draft, wire transfer, in foreign currencies, via vostro and nostro accounts, by Paypal, using hawala, and so on. Each of these would have any number of subtleties, e.g., to pay with a check you must show a government ID and a major credit card (which isn't used, but its number is noted). And these protocols could be combined in unusual ways, e.g., some restaurants accept credit card or travelers checks only; some only accept cash or check. Moreover, the protocols only specify the interactions, not the local policies of the participating entities, such as that they don't take cash after sunset. Protocols enable such policies to be inserted but are not directly concerned with the policies.

Protocols would be needed for every aspect of business processes that involves standardized interaction. Payment is just one aspect, but there are as many aspects as business interactions. Clearly, developing sophisticated protocols would not be a trivial undertaking. It needs abstraction support akin to information or object modeling in traditional systems. We develop two main classes of abstractions: *refinement* (like the subclass-superclass hierarchy) and *aggregation* (like the part-whole hierarchy). We develop a formal semantics to support the hierarchy and propose an algebra to facilitate reasoning about protocols.

2. DESIGN PRINCIPLES

We define a protocol as a specification of a logically related set of interactions. Protocols are interfaces, specifying only the key desired aspects of the interactive behavior and leaving the details to implementors. If new protocols are developed by composing existing protocols, integration and configuration is simplified, because you can put together the necessary software modules in a manner that parallels how the protocols are put together.

[*]We thank Amit Chopra and Nirmit Desai for valuable comments. This research was supported by the NSF under grant DST-0139037 and by a contract from DARPA.

Protocol Aggregation. The variety of protocols that realistic business processes require should be developed without the need for a serious integration and configuration effort every time a process changes. Such a requirement can be met by employing well-specified, published protocols [1] and by being able to combine these protocols with a clear understanding of the properties of the resulting process. In particular, we achieve aggregation via a characterization of how to *splice* a protocol into another to yield a hybrid protocol.

Protocol Refinement. A protocol is *refined* when it is made more specific. The most general protocols specify little, thus allowing a lot of flexibility. For example, a payment protocol that only specifies that the payer transfer funds to the payee can be refined into a variety of payment protocols, each with different tradeoffs of expense, speed, and convenience (for one or more of the parties involved), as described before.

Commitments. To determine whether a refinement of a protocol is legitimate, we must represent not just the behaviors of the participants but also the evolution of the contractual relationships among the participants. These contractual relationships are naturally represented through *commitments*, which have gained importance in the field of multiagent systems [3]. Commitments capture the obligations of one party to another. Using contractual properties of a protocol also enables us to readily detect and accommodate business exceptions and opportunities [4]. In principle, violations of commitments can be detected and, with the right infrastructure, commitments can be enforced.

3. EXAMPLE

The following example illustrates our approach. Technical details are available in an unpublished paper [2]. Consider a *purchase* interaction in which a customer wants to buy a book from an online bookstore. The bookstore obtains the customers' order for a book if the customer accepts the price quoted by the bookstore for that book. The book is then shipped to the customer, and the bookstore is then paid for the book. The actual execution of the process, however could involve many different scenarios such as: (1) the customer returns the book for a refund, and (2) the customer delegates the payment to its bank, and the bookstore uses a third party shipper to have the book delivered. Figures 1 and 2 illustrate these two processes.

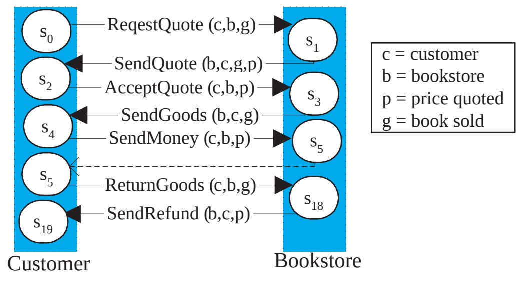

Figure 1: Refined purchase, with goods returned

Protocol Semantics. Our protocol semantics are based on the notion of similarity of contractual relationships that hold at various states of a protocols. For example, one might consider states to be similar if they have the same contracts regardless of which party is actually bound to the contracts. Based on such state comparisons, long runs *subsume* short ones, provided the longer run has states similar to and in the same relative temporal order as the shorter run. For example, scenario 1 subsumes the purchase protocol, since it contains all the states of a purchase run, in addition to two more states corresponding to the return of the book and the payment of the refund. This is an example of protocol refinement. A general protocol subsumes a more specific one, if every run of the specific protocol subsumes some run of the general protocol.

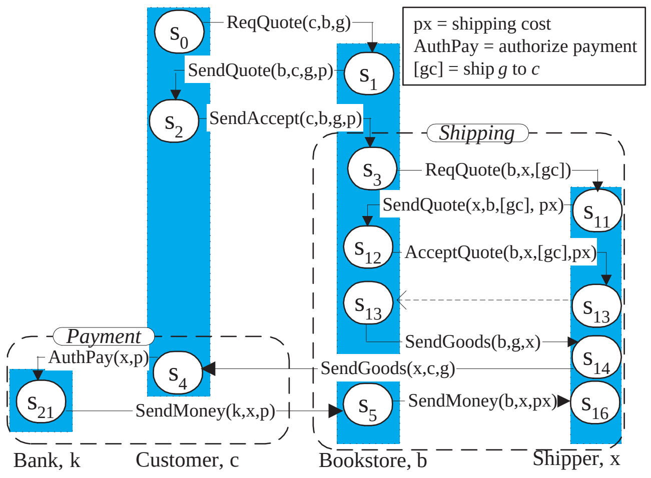

Figure 2: Aggregated purchase, by splicing in shipping and payment

We also define two operators over protocols, the *merge* and the *choice*. These allow us to splice and aggregate protocols as shown in Figure 2, where a shipping and a payment protocol are spliced into the simple purchase protocol.

4. DISCUSSION

Protocols are conceptually like conversations. However, current work on conversations (e.g., WSCI) merely specifies the constraints of a given party, and does not treat them as independent specifications. This conflates a protocol with the question of who is willing to play a role in the protocol. Further, traditional business process automation efforts like workflow technologies specify rigid sequences of steps to be followed, are a prone to failure in the face of exceptional curcumstances. The use of contractual relationships in protocol specification helps alleviate both these problems.

The algebra we have developed provides a basis for conceptual reasoning about protocols in terms of refinement and aggregation, which is essential if we are to engineer protocols the way other software systems are engineered. Such composition aids both design-time and run-time process engineering. To our knowledge, this work is unique in formulating the problem of process design at a conceptual level.

5. REFERENCES

[1] M. N. Huhns, L. M. Stephens, and N. Ivezic. Automating supply-chain management. In *AAMAS-2002*, pages 1017–1024. ACM Press, July 2002.

[2] A. U. Mallya and M. P. Singh. A semantic approach for designing e-business protocols, In *ICWS-2004, To appear*. July 2004.

[3] M. P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *AI and Law*, 7:97–113, 1999.

[4] P. Yolum and M. P. Singh. Flexible protocol specification and execution: Applying event calculus planning using commitments. In *AAMAS-2002*, pages 527–534. ACM Press, July 2002.