

# Forming Agents for Business Process Orchestration

M. Brian Blake  
*Department of Computer Science*  
*Georgetown University*  
*Washington, DC 20057*  
*blakeb@cs.georgetown.edu*

## Abstract

*Distributed component-based services and semantic web services are promising technologies for next generation inter-enterprise integration. The dynamic nature of this domain presents a complex problem for potential software agent-based approaches that support this cross-organizational integration. Currently, there are few studies that measure the impact of the dynamic environmental effects on service composition. On an on-going basis, composite services or workflow processes of web services may be constantly changing in terms of responsiveness of services, accessibility of services and their meta-information, business process schema changes, etc. This paper describes an approach, model, and supporting software toward the efficient formation of agent teams and interaction protocols for business process orchestration in response to certain environmental conditions.*

## 1. Introduction

The term, *services-based cross-organizational workflow* (SCW), can be used to describe the workflow interaction that occurs when one business incorporates the services of another within its own processes (also described as business-to-business (B2B electronic commerce [1] and similar to the notion of virtual enterprises [2][3]). Though, we introduce the new term, SCW, the ideas are similar to the research conducted in the related areas of component-based software engineering. In fact, SCW is a natural extension of the area of component composition. In traditional component composition research [4], components and their interfaces are modeled using formal (text and visual) languages and implemented with technologies such as CORBA, COM+, J2EE, Enterprise Java Beans, or .NET.

The use of web services [5] for functional specification and interactions has attained a great deal of attention currently. The Simple Object Access Protocol (SOAP) [6] is a protocol concerned with the specification of service-based messages. The Web Services Description Language (WSDL) [7] allows the specification of the services that use these messages. The robustness of this approach is realized when distributed registries such as Universal Description, Discovery, and Integration (UDDI) [8] architectures make distributed services available, universally. Other Extensible Markup Language (XML)-based specification, such as the Web Services Flow Language (WSFL), Business Process Execution for Web Services (BPEL4WS), and the Business Process Modeling Language (BPML) [9][10][11] specify the business processes of services and the composition of these services.

### 1.1. Agents in the SCW Environment

With respect to the web service paradigms and the SCW environment, businesses can advertise their offerings in UDDI registry servers. Since these registries have relatively straight-forward access methods (i.e. the *find\_service* and *get\_serviceDetail* methods in the UDDI specification). Intelligent or reasoning mechanisms can be developed which access specific service descriptions and message descriptions in the form of WSDL and SOAP documents, respectively. These mechanisms can act as brokers for the services represented in the WSDL/SOAP documents. Agent technologies represent a possible solution for the implementation of these brokers.

Software entities have been defined as agents when they exhibit certain characteristics. The grouping of these characteristics has been defined as levels of agency [12]. Agents that act as brokers in the SCW environment have the capability to reactively and proactively manage service executions

and process management using internal information about their environment. As such, these agents can be classified as having weak agency. In addition, these agents can react to both functional and nonfunctional conditions of the workflow and proactively effect change when negative nonfunctional events occur.

The agent-based SCW environment is illustrated in Figure 1. On-line businesses can initiate a configuration environment (Step 1) where broker agents can be instantiated and given a specific service requirement. With this requirement, agents

are enabled to proxy services from a distributed UDDI registry server. In addition to specifications of services that they proxy, these agents can advertise their own processing capabilities. Given both the available service and the capability of the agents, agent teams can be evaluated for specific conditions of the networked environments (Step 2). During business process enactment, these agents bind to specific services and through their coordination can manage the cross-organizational services (Steps 3 and 4).

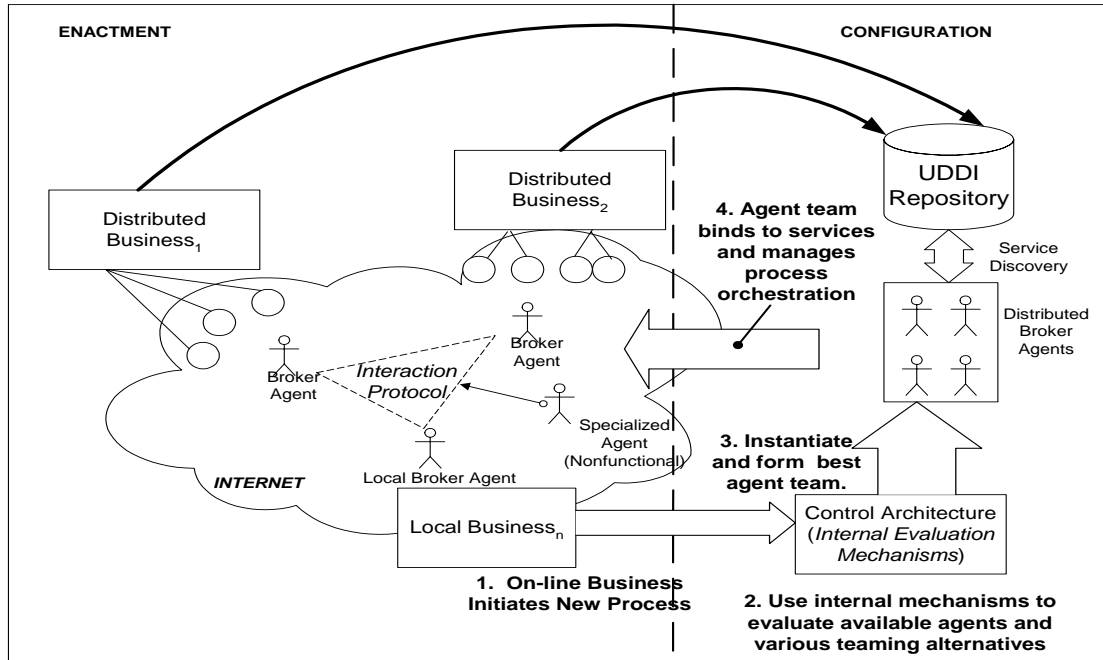


Figure 1. Agents Integrated into the SCW Environment.

## 1.2. Formation of Collaborative Agent Teams

In this work, several questions must be answered to support the creation of the SCW environment :

1. How are agents characterized in the SCW environment?
2. What are the alternative interaction protocols among these agents and how are they evaluated?
3. How can these evaluations consider the dynamism of the Internet environment?

In addressing the aforementioned questions, we introduce an approach called *Collaborative Organization of Agents for the Composition of Heterogeneous Electronic Services (COACHES)*. In this approach, we consider historical network activity, service execution statistics, statistics on agent capabilities, and the business process (the source for composition) as input to an *agent*

*realization* mechanism. This agent realization application (from COACHES suite of modules) uses this input information to determine the best formation of agents for the specific cross-organizational workflow process or routine. This approach is illustrated in Figure 2.

The paper continues in Section 2 and 3 with a discussion of interaction protocols based on atomic workflow paradigms and routines. In Section 4, we discuss methods to evaluate independent protocols based on both environmental attributes and specific attributes surrounding agent communication and coordination. We define the COACHES approach to and the simulation-based software for the evaluation of the agent-to-agent protocols in Section 5 and 6. In the final sections, we discuss the relation of our work to other related research and our future focuses.

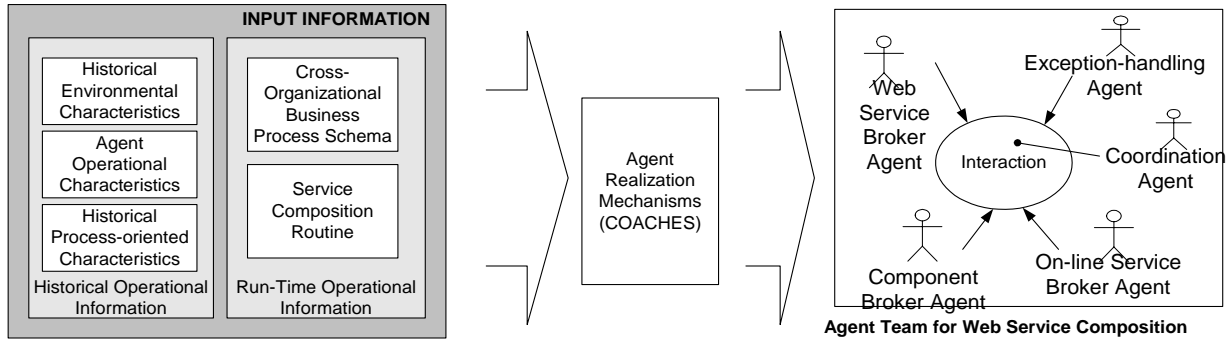


Figure 2. Agent Interactions for Workflow-based Service Composition.

## 2. Business Processes Composed of Workflow Patterns

Van der Aalst [13] maintains a repository of atomic workflow routines or *workflow patterns* that define common workflow operations used to implement business processes. In further work [14], workflow patterns have been used as a baseline to evaluate process languages such as BPEL4WS, XLANG, WSFL, BPML, and WSCI. We complement these earlier evaluation approaches by using workflow patterns to provide formal descriptions of common operational procedures underlying a business process. Our assumption in this work is that most business processes can be represented by the composition of workflow patterns.

### 2.1. Workflow Patterns for SCW

There are numerous workflow patterns introduced at [34], but, in the scope of this paper, we discuss several common to agent-based service composition. These workflow patterns are *normal sequence*, *parallel split*, *synchronization*, *exclusive choice*, and *simple merge*. In the normal sequence pattern, modelers must specify the basic sequence of activities (service executions). In parallel split, multiple activities are executed concurrently. At times, parallel threads must be synchronized (synchronization) or one path is chosen from many alternative choices (exclusive choice). Finally, business process modelers can specify when two paths are merged (simple merge). It is not in the scope of this work to show all possible workflow patterns. However, in Figure 3, there is an illustration of sample exclusive choice and deferred-choice workflow patterns.

In the exclusive choice pattern, after the completion of Service A, the supporting system must choose between two services (Service B and Service C) to continue operation. In the deferred-choice pattern, a parallel split occurs after the completion of Service A

that allows both Service B and Service C to be executed. Based on some criteria or later system event, either Service B or Service C is allowed to continue.

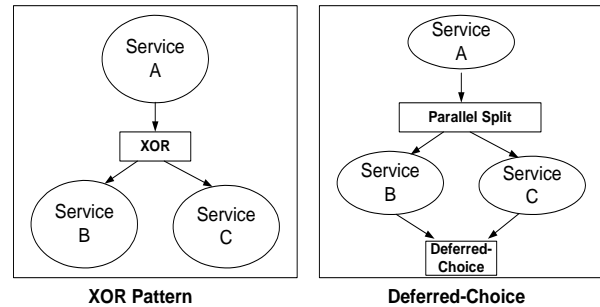


Figure 3. The Exclusive Choice and Deferred Choice Workflow Patterns.

### 2.2. Business Processes & Workflow Patterns

In this work, business processes can be decomposed into workflow patterns. In earlier work, business processes were defined using Unified Model Language (UML) activity diagrams [15]. In Figure 4, a simple business process is illustrated using an activity diagram. This business process shows the composition of services that perform a simple travel agency scenario for reserving a motel or hotel room, reserving a rental car, and receiving an e-mail delivered itinerary. In this simple scenario, three workflow patterns (as discussed in Section 2.1) can be extracted. These patterns are stipulated with the <<PARALLEL SPLIT>>, <<DEFERRED-CHOICE>>, and <<SIMPLE-MERGE>> stereotypes.

In this business process, reserveRoom services are executed concurrently for both the MotelReservation and HotelReservation actors and the CarRental's reserveCar service. The deferred-choice pattern represents the operation where some criteria are used to determine which reserveRoom service will remain active. Finally, the resulting services are merged before the initiation of the publishItinerary service.

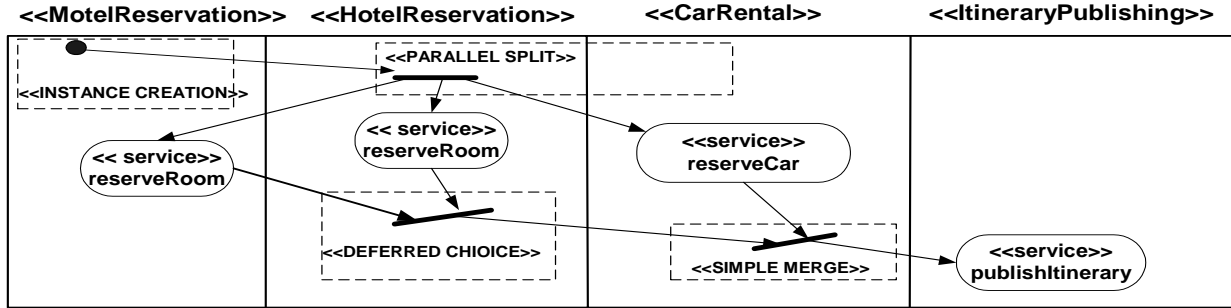


Figure 4. The Travel Reservation Business Process Composed of Workflow Patterns.

In order to find the most efficient configuration of an agent-supported business process management system, the underlying agent interactions must be evaluated. A broker agent, as in Figure 1, may have specific operational parameters based on its delay in sending messages, accessing data, or executing the underlying service. These types of operational measures can impact the performance of a set of agents that interact to fulfill a specific workflow pattern. These operational factors affect the process as a whole and consequently affect the business process. In the following section, we show how these agent interactions are modeled.

### 3. Evaluating Agent Interactions for the Deferred-Choice Workflow Pattern

Similar to the exclusive-or pattern, the deferred-choice pattern occurs when there must be a choice of two activities in a workflow process. The exclusive-or workflow pattern designates one of many activities to execute while the deferred-choice allows multiple activities to execute concurrently. However, for deferred-choice, at some point prior to completion, one activity is allowed to continue while all other activities are disabled. This pattern is particularly applicable to cases when some subset of the resultant information from the activity can be used as the criteria for choosing which service continues.

#### 3.1 Generating Agent Interaction Protocols

A challenge in this research is determining the differing variations of agent teams to support workflow patterns. An assumption in this work is that software engineers are capable of conceptualizing the proper set of interaction protocols that realize the workflow patterns. This assumption may not be practical considering that there may be some very efficient interactions that may not be obvious to software engineers that develop such orchestration systems. As such, in future work, we intend to investigate mechanisms that randomly generate interaction protocols as an extension to the COACHES

suite. At this point, agent interaction protocol choices are human-generated.

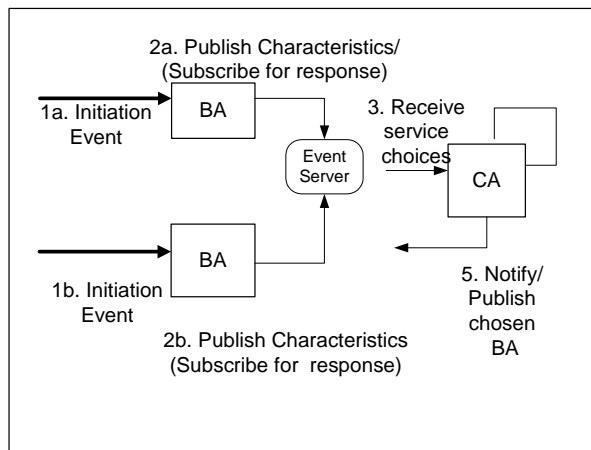
In generating agent interaction protocols, we adhere to an event-based paradigm for communication among agents [16]. We also assume that agents interact using a shared dataspace coordination paradigm as first described in Linda-based coordination architectures [17] implemented in applications such as IBM's TupleSpace, Sun Microsystems' JavaSpaces, or Java Messaging System (JMS). JMS however has additional functionality including a publish/subscribe paradigm [18]. In addition, we define a set of basic roles for agents to achieve, such as brokering services, initiating and/or coordinating activities, handling exceptions, monitoring and/or proactively enhancing performance, and other specialized management activities.

Considering web service composition domain, a deferred-choice pattern occurs when a business process is specified with an option of two workflow tasks (services) for one particular step. The choice of which task can be made based completely on an attribute such as performance, or the choice can rely on some other criteria. There is an option of the type of agent interaction that must take place to perform this pattern. In the interaction protocol for deferred choice, we identify two types of agents, a broker agent (BA) and a coordination agent (CA). The main task of the broker agent is to execute the underlying web service. However, the (workflow) coordination agent has the responsibility to encapsulate evaluation criteria that designates the specific representative broker agent from a group of qualified broker agents.

In Figure 5a and Figure 5b, we define two different interaction protocols that can be used to implement the deferred-choice pattern. In Figure 5a, an initiation event is sent to two or more broker agents. All applicable broker agents receive the initiation event as a deferred-choice request and immediately publish to the event server the characteristics (such as service time, reliability, etc.) of their underlying services that will be used to fulfill the workflow step. The coordination agent is notified of these postings in the

form of non-functional instructions. The coordination agent then evaluates each set of characteristics based on some criteria. The coordination agent then posts a decision to the event server. The correct broker agent is notified and continues the workflow process with its underlying service.

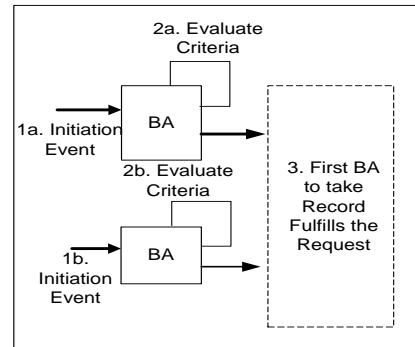
In a slightly different interaction, the pattern in Figure 5b differs from traditional deferred-choice and allows the broker-level agents to control their own termination. Broker agents are required to contain copies of the criteria, internally. When a deferred-choice request is captured by two or more broker agents, each broker agent acts as if it has been chosen. However, before taking the record, the broker agent must complete an internal evaluation in parallel with performing the task. Each agent uses pre-configured internal criteria to evaluate its characteristics. The first broker agent that completes the task and determines that it meets the criteria takes the record from the event-server. Once this record is taken, other broker agents that fulfill the criteria (but operating at a slower pace) will try to take the record and find that the request has already been fulfilled. Consequently these broker agents will terminate themselves. The author acknowledges that there are several other variations of this interaction.



**Figure 5a.** Third-Party Control-Oriented Interactions for Deferred-Choice Pattern

The interaction protocol in Figure 5b follows more of a peer-to-peer (P2P) interaction style [18] among the broker agents, while the interaction protocol in Figure 5a delegates control responsibility to a third-party coordination agent. In this scenario, it is evident that the COACHES approach has the fundamental capability to not only evaluate atomic interaction protocols, but also the potential capability to evaluate full operational paradigms (such as P2P style, call-and-

return style, etc.). In the following section, we discuss how these interaction protocols can be evaluated based on historical statistics, agent capabilities, and dynamic environmental concerns.



**Figure 5b.** Peer-to-Peer Oriented Interactions for Deferred-Choice Pattern

#### 4. Chosen Evaluation Attributes

There are numerous attributes and characteristics that can be used to evaluate the effectiveness of one interaction protocol with that of another. These attributes are grouped into three categories, historical networked environment attributes, historical process attributes, and agent operational attributes (historical and real-time). In this section, we briefly summarize many of the attributes, both environmental and internal system-related. The attributes were chosen based on their importance to earlier agent prototype studies [19] and are described in detail in previous work [20].

The historical network attributes reflect the dynamism of the Internet. In this work, we consider three attributes, service changes, new service additions, and past network responsiveness. **Service changes and additions** will have increasing impact on business process management systems. With the acceptance of web services on the Internet, it is feasible to expect that new services will be constantly added to the registries as pre-existing services become obsolete and are disabled. The rate and variability of service changes must be considered when establishing bindings between agents and available services. Also, the **network responsiveness** may vary when accessing services housed in different locations. Additional specialized agents (i.e. performance modeling or performance enhancement agents) may need to be deployed for a particular business process.

Historical process attributes describes trends on how business processes are accessed and utilized. Using these trends as predictors of future utilizations of services, we consider historical business process

changes, average service execution time, average meta-information response time (from registries), and frequency of the concurrent process requests/enactments. **Business process changes** can be driven by certain daily occurrences. In such cases, new business process definitions will cause the initiation of new service bindings. Systems that necessitate frequent reconfiguration may be more efficient with agent teams that have efficient protocols to reconfigure themselves at the initiation of every job, while more static systems can have more efficient run-time procedures and less efficient reconfiguration protocols. Moreover, the **expected service execution time** or variability of such time can be important in determining which nonfunctional agents to deploy. The **meta-information response time** is related to the time that it takes to access a service registry (i.e. UDDI). If this time is excessive, interaction protocols should be designed that minimize the number of times the registry is accessed. Finally, understanding **frequency of the concurrent process requests/enactments** can help in forming agents to efficiently handle services that support such processes.

Agent operational attributes relate specifically to internal system concerns. These operational attributes consist of concerns such as **agent-to-agent communication time**, **agent performance based on load**, and **agent response to data queries**. The execution time for an agent to communicate and access information while operating under variable loads is important to the selection of interactions. These agent characteristics are the subject of further evaluations in the following sections.

## 5. Simulation-Based Evaluation Approach and Application

A goal of this research is toward the evaluation of agents that manage service composition of business processes by systematically evaluating the agent interactions of the underlying workflow patterns. In the scope of this paper, we evaluate the deferred-choice pattern particularly with respect to one set of the aforementioned attributes (agent operational attributes described in the earlier section). There are three operational measures related to these attributes. These measures are related to data retrieval, processing due to concurrency load, and agent communication.

### 5.1 Describing Operational Delay for Agent Interactions

Formally, we use the term operational delay,  $O_D$ , to designate the sum of the agent operational attributes, described in the previous section. Operational delay

can further be decomposed into the three different delays, each as a function of the interaction protocol,  $i$ , that is used for the workflow execution. The three delays are the data management delay,  $D_D$ , the general agent execution delay,  $E_D$ , and the network communication delay,  $N_D$ . Therefore, operational delay can be defined as:

$$O_D(i) = D_D(i) + E_D(i) + N_D(i), \text{ where:}$$

- $D_D$  is the function that determines the time all data requests to a data repository. This function considers weights based on the expense of querying different types/locations of data.
- $E_D$  is the function that determines the agent execution time. A major parameter of this function is the load imposed on the agent.
- $N_D$  is the function that determines the network communication delay.

### 5.2 Evaluating the Interactions for the Deferred-Choice Pattern

A basic example can be used to demonstrate the approach to evaluating interactions using operational delay. In this example, we consider the deferred-choice interaction. To further simplify the example, initially we consider that the deferred-choice interaction is strictly coordination-based. As such, there are no data queries or data management delays. Therefore, the interactions are only evaluated on execution time and network communication times. For the purpose of comparing different operations (execution time and communication time), we set a baseline,  $x$ . This baseline is exactly the time for sending one agent-to-agent message. In previous work [19], we discovered that the execution time for BAs was approximately  $\frac{1}{4}$  of such a baseline,  $x$ . These measures were taken over an extended period of time. Even though this ratio may fluctuate in live systems, for this initial study we assume that the magnitudes will remain the same. Since the BAs are considered to be distributed across the Internet, we also assume no additional system delay for having a large number attempt to accept the same job. Again, it may be reasonable to evaluate, in the future work, that communication mechanisms may incur additional delay. Since we compare two interactions for the same scenario, we also assume that CPU and network speeds would be consistent across both, and, as such, any associated delays are neglected.

The CAs, however, have to search a large number of criteria to locate the correct one, therefore the delay based on this evaluation for these types of agents is slightly more (approximately equivalent to the baseline,  $x$ ). A listing of all delays for this interaction for both paradigms is in shown Table 1a and 1b.

**Table 1a** 3<sup>rd</sup> Party Control for Deferred-Choice Processing (Total = 4x)

Agent Action	Execution Delay ( $E_D$ )	Network Cost ( $N_D$ )
BA(2..n) Post Criteria	-----	x
CA Evaluates the Criteria for all BAs	x	-----
CA Posts Decision	-----	x
Chosen BA is notified	-----	x

**Table 1b** P2P coordination for Deferred-Choice Processing (Total = 1.25x)

Agent Action	Execution Cost ( $E_D$ )	Network Cost ( $N_D$ )
BA(2..n) Evaluate Criteria	.25x	-----
BA (2..n) Take the Initiation Record	-----	x

As aforementioned, we consider execution times that remain the same with increased load. This is practical since the idea of agents is toward distribution. Considering this distribution, the agent-based communication delay and execution times does not increase when multiple BAs work in parallel. By inspecting Table 1a and 1b, it is clear that the P2P coordination paradigm is more than 3 times more efficient based on the numbers provided from the WARP prototype [19]. Though the performance is better in the P2P paradigm, there are situations where the third-party control paradigm may be optimal. If the criteria require frequent changes, it is clear that the 3rd party control scenario would be more efficient (i.e. the criteria would only be changed in one place (CA) as opposed to a change to all relevant BAs). In previous work, a variation of this simulation approach was used to determine the threshold where changes can affect the choice of interaction [21].

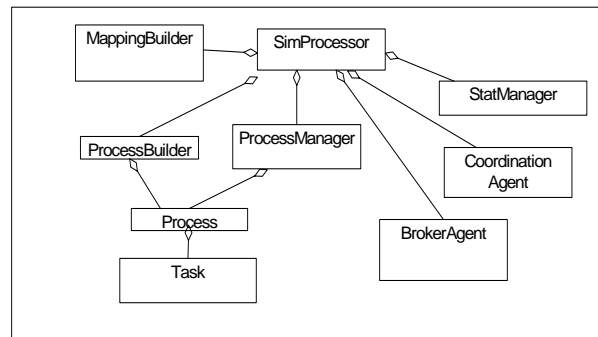
## 6. A Simulation-Based Evaluation Approach and Application

In Section 5, there is a simple case for evaluating the deferred-choice interactions. We neglected several important factors that we feel are best evaluated using discrete-time simulation. One factor is the variability of execution times with the increase or decrease of load on the independent agents. Secondly, data management delays can be associated with accessing the selection criteria for both the BA and CA. The data management delays can also vary based on the type of information accessed or the location of the information. Finally, communication delay can vary based on the location of the destination agent. In the following sections, we discuss the design of a

simulation application that handles these types of factors and initial experimental results discovered through the use of this application.

### 6.1 COACHES Evaluation Tool

We have designed and developed a simulation-based approach to evaluating the aforementioned factors in addition to the agent operational factors. The COACHES evaluation tool consists of 9 classes, SimProcessor, ProcessManager, ProcessBuilder, Process, Task, BrokerAgent, CoordinationAgent, MappingBuilder, and StatManager as shown in Figure 6. The SimProcessor object is the main control of the simulation. It is composed of a ProcessBuilder object that encapsulates different process traffic flows based on the containment of multiple Process objects (consisting of Task objects). These traffic flows can be created to emulate different composition of traffic thus incorporating the historical network attributes described in Section 4. The combination of these objects can assist in the evaluation of agent interactions with respect to service changes and concurrent processes. The ProcessManager object feeds Process objects to the CoordinationAgent objects and the BrokerAgent objects. Independent CoordinationAgent objects and BrokerAgent objects represent real-world agents. As such, these objects can be configured to perform differently as their load is increased. These BrokerAgent and CoordinationAgent objects report delays and operational conditions using the StatManager object. The StatManager object creates a report based on delay and queue size. The SimProcessor object can be used to make a final decision on which interaction is most efficient and for what purposes. In enabling the flexibility of the system, the MappingBuilder binds broker agents to services and coordination agents to processes at run-time.



**Figure 6.** OO Design for Simulation Application.

## 6.2 Simulation-Based Evaluation of the Deferred-Choice Workflow Pattern

In simulated experiments, we evaluated the effect that different flows of traffic have on the performance of agent interactions for the deferred-choice workflow pattern. In these experiments, broker agents have decreased performance when the number of concurrent tasks is increased. However, the coordination agents' performance can remain reasonably the same, since checking the criteria is its only major task. The broker agent is configured to add 1 second of delay for each 10 additional concurrent processes with a limit of 4 seconds. We evaluated the two deferred-choice workflow patterns to determine what type of traffic was most effective for each. For the purpose of these experiments, we set the baseline,  $x$ , to 4 seconds. We chose four types of traffic. Traffic files have a maximum of 100 job requests. Each type of traffic was evenly spaced over 100 cycles. The four types of traffic are listed below:

- ◆ 10 Job Requests every 10 cycles (Batch 10)
- ◆ 2 Job Requests every 2 Cycles (Batch 2)
- ◆ Starting from 1 increasing the amount of jobs by 1 every 7 cycles (Climb 14)
- ◆ 1 Job Request per Cycle (Trickle)

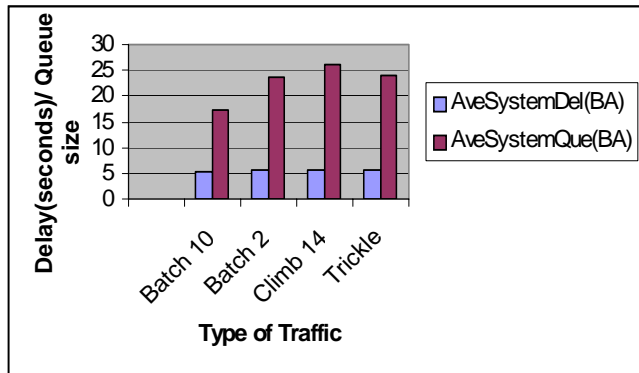
In these experiments, as mentioned earlier, the average system delay for the 3rd party control scenario was expected to remain consistent, since there was no variance in the execution time based on agent load. This expectation was consistent with the results as the average system delay of the 3rd party control scenario was consistent at 16 seconds shown in Table 2b. The operation of the P2P-oriented scenario is more efficient with the current traffic options. We discovered in further experiments that a sustained queue of 200 tasks was the threshold where the 3rd party control scenario becomes more efficient [21].

**Table 2a.** Delay for P2P-Oriented Interaction

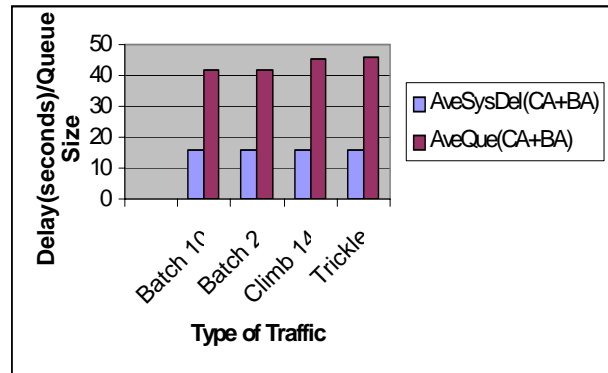
Traffic Type	AveDel (BA)	AveSystemQue (BA)
Batch 10	5.29966	17.4411
Batch 2	5.82	23.7267
Climb 14	5.79861	26.1771
Trickle	5.82333	23.9333

**Table 2b.** Delay for 3<sup>rd</sup> Party Control Interaction

Traffic Type	AveSysDel (CA+BA)	AveQue (CA+BA)
Batch 10	16	41.2649
Batch 2	16	41.6835
Climb 14	16	45.3933
Trickle	16	45.59



**Figure 7a.** P2P-Oriented Measures



**Figure 7b.** 3<sup>rd</sup> Party Control Measures

Both scenarios shown in Figure 7a and Figure 7b have the best performance when requests are sent in larger batches (Batch 10) in evenly-spaced time intervals. This result is logical since the time intervals are less than or equal to the service time of the tasks. With the number of requests being less than the capacity that can be serviced in one cycle, all services are completed

with the lowest estimation of delay. For both scenarios, the queue size varies with the composition of the network traffic. Other operational concerns (not discussed here) can be a function of the queue size. When this is the case, monitoring the queue size would be important to the evaluation of agent interactions.

In these experiments, we expected the average system delay to be a function of the average queue size. However, an interesting result is that the average system delay does not appear to be a function of the average queue size. However, in the P2P-oriented scenario, the average system delay varies based on the traffic. Though the difference is small for these experiments, this small variance can become significantly larger as the network traffic increases in magnitude. With the system delay varying based on the type and composition of traffic, agent teams need to be evaluated based on the business case and situation. We believe this result demonstrates the true need for real-time tools to propagate simulated traffic, where agent interactions for service composition can be evaluated at run-time.

## 7. Discussion and Related Work

This work is closely related to the areas of agent collaboration, workflow, and service composition [22][23][24]. There are several other projects that use agent theories for the workflow composition of services. Helal [25] uses an agent architecture for workflow enactment with consideration to web services using SOAP. In fact, later work [26] considers UDDI-registered services. Chen and Griss [27] also consider the use of agents for workflow with semi-structured specification languages. Finally, Singh [28] discusses the workflow composition of services as a community of services. The major emphasis in this Singh's work is an approach to the discovery of services.

Though research in forums for agent communication languages [29][30] and collaboration protocols investigate protocols from a general data exchange point of view (conversational), none of the studies, as known by this author, investigate the efficiency of agent interaction protocols specifically for the workflow composition of services. This work emphasizes the evaluation of agent interaction protocols using concepts from well-established specifications of workflow patterns and operations. Low-level studies of this sort are not included in aforementioned work.

In using object sharing paradigms for agent interactions, our work builds on our earlier work [16] and related work [31] [32] that endorse the use of such paradigms for business collaboration. We extend the state of the art in this area by taking an applied investigation of how the delay of this type messaging affects system operation.

The most closely-related agent-oriented research to our approaches is the studies performed by the SELF-SERV project [33]. Benatallah [34] evaluates his

composition environment for both P2P coordination and centralized control based on the aspects of reliability, monetary price, and execution costs. In our work, we perform studies that further decompose execution costs into the aforementioned three operational delays of data management, execution time based on parallel processing, and network communication delay. These studies are performed on a stand-alone simulation application developed specifically for the purpose of evaluation. An innovation in our work is the fact that we consider the dynamics of the Internet in our evaluation approach. In addition, these simulated evaluations can fully execute in just a few seconds, which suggests this approach to be viable for the real-time evaluation of agent interactions in operational service composition environments.

## 8. Conclusion

This work is toward automated mechanisms for evaluating the best formation of software entities to control the composition of web services. As composition languages and web services become more mature, these peripheral technologies will be important to discovering how control functionality should be organized and distributed. In this work, we developed applied approaches to investigating how operational delays can affect these choices. In addition, we have shown that the deferred-choice workflow routines are more efficient when broker-level control entities are allowed to interact in a P2P-style. We also discussed development of a simulation approach to evaluate these interactions in real-time. This is the first approach to evaluating the control architecture and mechanisms for each service composition routine using empirical methods, particularly in the web services domain. In addition, we explore the effects on this architecture with respect to factors as a result of the nature of the Internet. In future work, we plan to model other agent-to-agent interactions that implement other workflow patterns, such as synchronization and parallel-splits and joins. In this work, we evaluated the interactions based on relatively evenly-spaced traffic. In future work, the interactions will be evaluated using a Poisson distribution of traffic, which, in some cases, better simulates network activity. In addition, we plan to investigate the feasibility of automatic, tool-based approaches to modeling agent interactions for workflow patterns.

## References

- [1] M.B. Blake, "B2B Electronic Commerce: Where Do Agents Fit In? ", Proceedings of the AAAI-2002 Workshop on Agent Technologies for B2B E-Commerce, Edmonton, Alberta, Canada, July 28, 2002
- [2] C. Petrie, and C. Bussler, "Service Agents and Virtual Enterprises: A Survey" IEEE Internet Computing, August 2003, pp 1-12
- [3] J.W.J. Gijsen, N.B. Szirbik., and G. Wagner, "Agent Technologies for Virtual Enterprises in the One-of-a-Kind-Production Industry," International Journal of Electronic Commerce: Special Section on Agent-Based Approaches to B2B Electronic Commerce, Eds. M. Brian Blake and Maria Gini, Vol. 7, No. 1 pp 9-34, October 2002
- [4] Heineman, G. and W. Council, Component-Based Software Engineering Putting the Pieces Together, Reading, MA: Addison-Wesley, 2001.
- [5] Web Services (2002) <http://www.w3.org/2002/ws/desc/>
- [6] SOAP (2003) <http://www.w3.org/TR/soap12-part0/>
- [7] WSDL (2003) <http://www.w3.org/TR/wsdl>
- [8] UDDI (2003) <http://www.uddi.org/>
- [9] WSFL (2003): <http://www.ebpml.org/wsfl.htm>
- [10] BPEL4WS (2003): <http://www.ebpml.org/bpel4ws.htm>
- [11] S. Thatte, "XLANG: Web Services for Business Process Design". Microsoft, 2001
- [12] M. Wooldridge and N.R. Jennings, "Intelligent Agents: Theory and Practice," In Knowledge Engineering Review 10(2), 1995.
- [13] Workflow Patterns (2003): <http://tmitwww.tm.tue.nl/research/patterns/patterns.htm>
- [14] W.M.P. Van der Aalst, "Don't go with the flow: Web Services composition standards exposed", IEEE Intelligent, February 2003
- [15] M. B. Blake, *Agent-based Workflow Modeling for Distributed Component Configuration and Coordination*, Ph.D Dissertation, George Mason University, 2000 online at: <http://www.cs.georgetown.edu/~blakeb/pubs/diss.zip>
- [16] M. B. Blake, "Agent-Based Communication for Distributed Workflow Management using Jini Technologies" , International Journal on Artificial Intelligence Tools (IJAIT), Vol. 12, No. 1, March 2003, World Scientific Publishers
- [17] N. Busi, and G. Zavattaro, "Publish/Subscribe vs. Shared Dataspace Coordination Infrastructure: Is it Just a Matter of Taste", 10<sup>th</sup> IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Boston, Massachusetts, 2001.
- [18] K. Aberer, "P-Grid: A self-organizing access structure for P2P information systems," Proc. Coop. Info. Sys. (CoopIS), pp. 179-194, 2001.
- [19] M. B. Blake, " WARP: Workflow Automation through Agent-Based Reflective Processes ", Proceedings at the 5th Int. Conf. on Autonomous Agents/ACM Press, Montreal, Canada, May 2001 (software demonstration)
- [20] M. B. Blake, "Forming Agents for Workflow-Oriented Process Orchestration" Workshop on Electronic Commerce, Agents, and Semantic Web Services in conjunction with the International Conference on Electronic Commerce (ICEC2003), Pittsburgh, PA 2003
- [21] M. B. Blake, "Coordinating Multiple Agents for Business Process Orchestration", Information Systems and E-Business Management (to appear)
- [22] F. Casati, L. Jin, S. Ilnicki, and M.C. Shan, "An Open, Flexible, and Configurable System for Service Composition". HPL Technical Report HPL-2000-41.
- [23] P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig, "CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises," International Journal of Computer Systems Science & Engineering, Vol. 15, No. 5, 2000; pp. 277-290
- [24] L. Zeng, A. Ngu, B. Benatallah, and M. O'Dell, "An agent-based approach for supporting cross-enterprise workflows," In Proceedings of the 12<sup>th</sup> Australasian Conference on Database Technologies, 123-130, Queensland, Australia 2001
- [25] A. Helal, M. Wang, A. Jagatheesan, and R. Krithivasan, "Brokering Based Self Organizing E-Service Communities". Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems (ISADS) With an Emphasis on Electronic Commerce, March 26-28, 2001 Dallas, Texas
- [26] A. Jagatheesan, and A. Helal, "Sangam: Universal Interop Protocols for E-service Brokering Communities using Private UDDI Nodes" (under review)
- [27] Q. Chen, U. Dayal, M. Hsu, and M.L. Griss, Dynamic Agents, Workflow and XML for E-Commerce Automation. EC-Web 2000: 314-323, London, UK
- [28] M. P. Singh, B. Yu, and M. Venkatraman, "Community-based service location," CACM 44(4): 49-54 (2001)
- [29] B. Chaib-draa and F. Dignum, "Trends in Agent Communication Languages", Computation Intelligence, Vol. 18, No. 2, pp 89-101, 2002
- [30] FIPA Interaction Protocol Specification (2002), <http://www.fipa.org/repository/ips.html>
- [31] J. Nickerson, "Event-based Workflow and the Management Interface," in the Proceedings of the 36th Annual Hawaii International Conference on System Sciences, Jan 6-9.
- [32] B.M. Oki, M. Pfluegl, M., A. D. Siegel, and D. Skeen, D. "The Information Bus-An Architecture for Extensible Distributed systems.," In Proc of the Fourteenth ACM Symposium on Operating Systems Principles, 1993.
- [33] B. Benatallah, Q. Sheng, and M. Dumas, "The Self-Serv Environment for Web Services Composition," IEEE Internet Computing, Jan/Feb 2003, 7,1, pp. 40-48
- [34] B. Benatallah, M. Dumas, Q. Sheng, and A. Ngu, Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In *18th International Conference on Data Engineering*, February 2002