

Coordinating multiple agents for workflow-oriented process orchestration

M. Brian Blake

Department of Computer Science, Georgetown University, 234 Reiss Science Building,
Washington, DC 20057, USA (e-mail: 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 tools that intend to support this cross-organizational integration. However, the autonomy and adaptation of software agents represent a viable solution for the composition and enactment of cross-organizational services. 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. These conditions impact what interactions a team of agents must undergo to achieve a specific process derived of composite web services. This paper describes an approach, model, and supporting software toward the efficient design of interaction protocols for coordinating agent teams in the business process orchestration domain. This approach considers several environmental conditions related to the dynamism of the Internet.

Key words: Workflow, agent architectures, business process execution

1 Introduction

There are new opportunities, involving developmental paradigms, where high-level component-based services and semantic web services [35] will be sufficient in modularity and autonomy to fulfill the requirements of other businesses. 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)* H Blake 2002). This term is also associated with the idea of a third-party organization that composes the

services of multiple businesses (similar to the notion of *virtual enterprises*; Petrie and Bussler 2003).

1.1 Background on distributed services

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 (Heineman and Council 2001), components and their interfaces are modeled using formal (text and visual) languages. Consequently, these specifications can be used for automated component composition. In addition, these specifications can be used for simulations that help to analyze and/or validate such component-based architectures prior to development (Aksit and Bergmans ■; The CHAIMS Project). The importance of this area is further emphasized with the new developments of sophisticated component-based service environments such as CORBA, COM+, J2EE, Enterprise Java Beans, .NET, etc. In fact, we believe the specification of such services using semantic web service technologies will lead to large-scale electronic market interoperability.

The use of web services for functional specification and interactions has attained a great deal of attention currently. The Simple Object Access Protocol (SOAP; 2003) is a protocol that contains a framework that allows the specification of the composition of messages and their responses. This protocol is specific mostly to web services over HTTP but not limited to the specification of component-based services, as previously discussed. The Web Services Description Language (WSDL; 2003) 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; 2003) architectures make distributed services available, universally. To date, the web services technologies are mostly toward the specification of interfaces and communication. However, there are also other Extensible Markup Language (XML)-based languages, such as the Web Services Flow Language (WSFL), Business Process Execution for Web Services (BPEL4WS), and the Business Process Modeling Language (BPML), that allow the process specification or composition of these services (*only a representative list*) (Thatte 2001; BPEL4WS 2003; WSFL 2003).

1.2 Integrating agents into the SCW environment

With respect to the web services paradigms and the SCW environment, businesses can advertise their offerings in UDDI registry servers. These registries have relatively straight-forward access methods (i.e. the *find_service* and *get_service Detail* methods in the UDDI specification). Intelligent or reasoning mechanisms can be developed which access specific service descriptions in the form of WSDL and SOAP documents. These mechanisms can act as *brokers* for the services represented in the WSDL/SOAP

documents. Agent technologies represent a promising solution for the implementation of these brokers.

Agent-oriented programming approaches can be used to realize the workflow composition and management of services. Software entities have been defined as agents when they possess certain characteristics. The grouping of these characteristics has been defined as levels of *agency* (Wooldridge and Jennings 1995, Jennings et al. 1998). Entities classified as having weak agency tend to possess characteristics such as autonomy, social ability, reactivity, and/or proactivity. However, strong agency dictates that, in addition to the weak agency characteristics, software entities must also possess mentalistic abilities, even emotions, usually attributed to human behavior.

Agents that act as brokers in the SCW environment can be classified as possessing weak agency. These agents can be defined as autonomous software entities that have knowledge of their environment to reactively and proactively proxy service executions and process management. In addition, these agents have independent assignments to invoke specific services or manage specific business processes. In addition to reacting to both functional and nonfunctional conditions of the workflow, these agents also are programmed with general proactive abilities to affect change when negative nonfunctional events occur.

In a configuration environment, these broker agents can be instantiated and given a specific service requirement enabling them to search through a distributed UDDI registry server (populated with business service offerings). Agents can discover multiple applicable services to be brokered in future business process executions. During business process enactment, these agents can collectively manage the composition of the cross-organizational services. This environment is illustrated in Fig. 1.

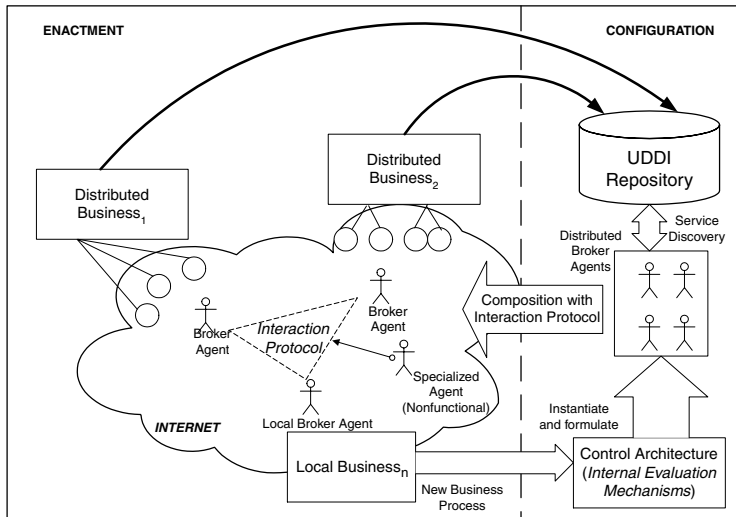


Fig. 1. Agents integrated into the SCW environment

1.3 Efficient formation of collaborative agent teams for service composition

Given the SCW environment described in Fig. 1, we have determined a need for an approach for the formation of collaborative agent teams for the business process enactment of web services. In developing this approach, several general questions must be answered:

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 to evaluate interaction protocols for particular composition routines considering 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 E-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 Fig. 2.

The work here is toward the first tool in the COACHES suite for the evaluation of agent interactions. As such, the paper continues in Sect. 2 and 3 with a discussion of interaction protocols based on atomic workflow paradigms and routines. In Sect. 4, we discuss methods to evaluate the independent protocols based on environmental attributes and specific attributes surrounding agent communication and coordination. We define the COACHES approach to evaluating agent-to-agent interactions and the supporting simulation-based methods in Sect. 5 and 6. In the final sections, we discuss our work with relation to other related research and our plan for future work.

2 Business processes composed of workflow patterns

The requirements of a business process can be realized by understanding the underlying workflow actions that define that process. Van der Aalst et al. (2002) maintains a repository of atomic workflow routine descriptions or workflow patterns that define common workflow operations used to

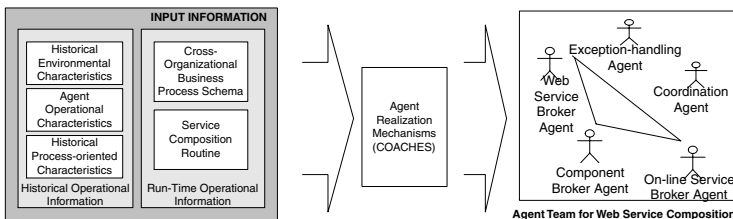


Fig. 2. Agent interactions for workflow-based service composition

implement business processes. A sample of the most common workflow patterns are sequence, parallel split, synchronization, exclusive choice, simple merge, deferred-choice, cancel activity/case, etc. In further work (Van der Aalst 2003), he evaluates the semantic effectiveness of common composition languages such as BPEL4WS, XLANG, WSFL, BPML, and WSCI to support this set of fundamental workflow patterns. The workflow patterns formally describe workflow routines that should be common to most researchers involved in business process engineering research, and, as such, are not described in detail here. Fundamentally, the design of workflow processes and complex interactions have been investigated in great detail. Also these workflow patterns have been used as a baseline to evaluate languages and systems for business process management (Van der Aalst 2003). We complement these earlier evaluation approaches by using workflow patterns as formal descriptions of common operational procedures that underlie business processes. In fact, we believe that business processes can be represented by the composition of workflow patterns, in very general cases.

2.1 Defining workflow patterns for service composition

There are numerous workflow patterns introduced in Van der Aalst et al. (2002), but, in the scope of this paper, we discuss the workflow patterns that have been fundamental in our service composition research. 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 Fig. 3, there is an illustration of a sample exclusive choice workflow pattern. In this illustration, after the completion of Service A, the supporting system must choose between two services (Service B and Service C) to continue operation.

In this work, we also devise other *system-oriented* workflow patterns, such as re-configuration and workflow instance creation. Re-configuration occurs when the business process schema must change at run-time. Instance creation is the run-time operation that occurs when a new business process of services is initiated.

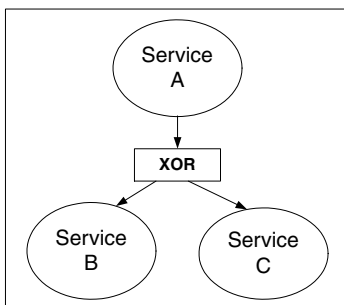


Fig. 3. The exclusive choice workflow pattern

These patterns describe the technical operations of the business process management systems. As an example, in Fig. 4, the instance creation pattern, that was developed in this work, is illustrated. The orchestration system must first select from a list of available and accessible services. Secondly, the system must assign tasks to the services and thirdly create and assign a unique identifier for the process instance.

2.2 Business processes composed of workflow patterns

In this work, workflow patterns represent the atomic operations that can be combined to support a fully-specified business process definition. In earlier work, business processes were defined using Unified Model Language (UML) activity diagrams (Blake 2000; 2003) as a process modeling language. In Fig. 5, a simple business process is illustrated using this approach illustrated as 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, five workflow patterns can be extracted, which are illustrated by the shaded regions. In first shaded region, the instance creation pattern would be relevant to the realization of a new job request (designated by the <<INSTANCE-CREATION>> stereotype). Likewise, the other shaded regions show other workflow patterns, that are designated by the <<SYNCHRONIZATION>>, <<PARALLEL-

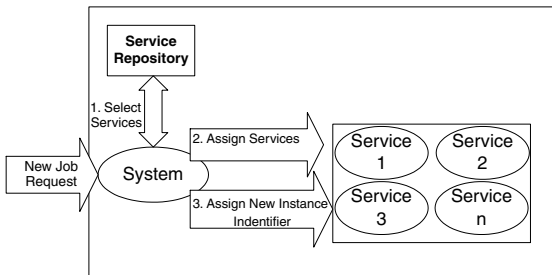


Fig. 4. The instance creation workflow pattern

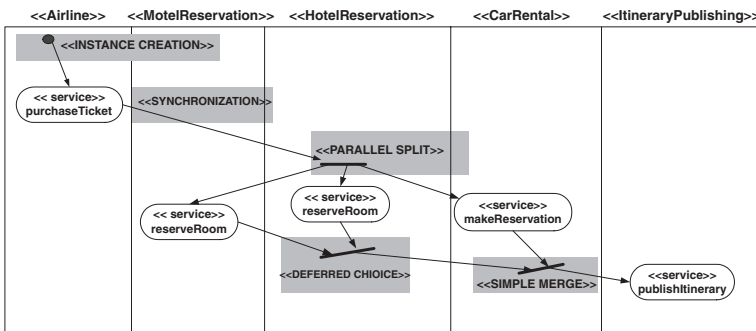


Fig. 5. The travel reservation business process composed of workflow patterns

SPLIT>>, <<DEFERRED-CHOICE>>, and <<SIMPLE-MERGE>> stereotypes.

In this business process, the first service is the airline ticket purchasing service. Subsequent services are dependent on the successful completion of this service (<<SYNCHRONIZATION>> pattern). Next, the reserveRoom services are executed concurrently for both the Motel Reservation and Hotel Reservation actors and the CarRental's makeReservation service. The deferred-choice pattern represents the operation where some criteria is used to determine which reserve Room service will remain active. Finally, the resulting services are merged before the initiation of the publish Itinerary service.

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 Fig. 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. As such, these operational factors can affect the business process, as a whole. In the following section, we show how contracts are composed of these agent interactions.

3 Evaluating agents for the deferred-choice workflow pattern

Deferred-choice occurs when there must be a choice of two activities in a workflow process. The deferred-choice is similar to the exclusive-or workflow pattern. 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 there is a subset of the resultant information from the activity that can be used as the criteria for choosing which service continues.

3.1 *Generating agent interaction protocols for the deferred-choice pattern*

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 always 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. We also assume that agents interact using a *publish/subscribe* paradigm as described in Linda-based coordination architectures (Gelernter 1991) implemented in applications such as IBM's Tuple Space, Sun Microsystems' Java Spaces, or the Java

Messaging System (JMS). This is an extension of an earlier work (Blake 2003). 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 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 protocols 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 the evaluation criteria that designates the specific representative broker agent from a group of qualified broker agents.

In Fig. 6a, b, we define two different interaction protocols that can be used to implement the deferred-choice pattern. In Fig. 6a, 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 various 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 (Fig. 6b), 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 uses pre-configured criteria to evaluate its own internal characteristics. The first broker agent to determine 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 disable themselves.

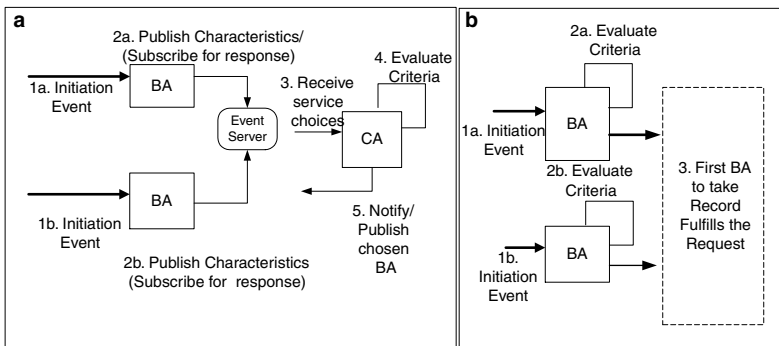


Fig. 6a. Third-party control-oriented interactions. **b.** Peer-to-peer oriented interactions

The interaction protocol in Fig. 6b follows more of a peer-to-peer (P2P) interaction style (Aberer 2001) among the broker agents, while the interaction protocol in Fig. 6a 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, centralized control, 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.

4 Attributes important to the evaluation of agent interaction protocols

There are numerous attributes and characteristics that can be used to evaluate the effectiveness of one interaction protocol with that of another. As in Fig. 2, we group these attributes into three categories, historical networked environment attributes, historical process attributes, and agent operational attributes (historical and real-time). In this section, we describe many of the attributes, both environmental and internal system-related.

The historical network attributes reflect the dynamism of the Internet. In these studies, we consider attributes such as service changes, new service additions, and past network responsiveness. However, we acknowledge that other relevant attributes may exist. The chosen attributes are described in further detail below.

- **Service changes and additions.** 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. This rate may vary considerably across different services based on the popularity of a service. In the COACHES approach, we consider the rate of service changes, since this rate impacts performance when service bindings must be re-established between agents and available services.
- **Network responsiveness.** The ability to access services in different locations may vary based on the type of service requested. Particularly, geographical locations may impact the response time when binding to specific services. The COACHES approach considers historical network connectivity in determining if additional specialized agents (i.e. performance modeling or performance enhancement agents) should 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.

- **Historical business process changes.** The processes or business cases of certain online business may be driven by certain daily occurrences. For example, a stock brokerage company may suggest different investment strategies depending on the state of the stock market. These types of companies may have different software services based on the type of advisement necessary (i.e. stock-purchasing, mutual funds, long-term

savings bonds, etc.) In such cases, new business process definitions will cause the initiation of new service bindings. Systems that necessitate frequent reconfiguration may favor 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.

- **Average service execution time.** The amount of latency in the atomic services can affect interaction protocols. More importantly, the variability in the latency can be important in determining which nonfunctional agents to deploy.
- **Average meta-information response time.** The time that it takes to access a service registry (i.e UDDI) to get information about a desired service may vary based on its geographical location or the physical network connectivity of that registry. In the case of particularly poor response times to service registries, it may be necessary to choose coordination protocols that limit the number of times such registries are accessed.
- **Frequency of the concurrent process requests/enactments.** Business processes that are particularly popular may require the support architecture to concurrently handle many process instances. Understanding the magnitude of concurrency can be helpful 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.

- **Agent-to-agent communication overhead.** In the COACHES approach, we use event-based communication among agents using publish/subscribe principles. In our studies, the cost of this type communication tends to be relatively low. However, knowledge of this expense can be helpful in determining the efficiency of an interaction protocol that may be messaging-intensive.
- **Agent performance.** Agents' performance may degrade with the increasing load or number of concurrent processes being managed. If this degradation is severe, then interaction protocols must be devised that provide for the additional distribution of functionality.
- **Agent response to data queries.** Agents may use both internal and external data-banks to assist in their operations. The amount of times an agent accesses these data-banks and the overhead expense can be used to determine protocols that either reduce or increase the amount of information that agents maintain in their internal memory. This is particularly important when an increase in the amount of memory maintained by an agent has negative effects on that agent's overall performance.

5 Using agent operations to evaluate the deferred-choice pattern

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 the aforementioned agent operational attributes and several historical process 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 latency for agent interactions

Formally, we use the term operational latency, O_L , to designate the delay associated with the agent operational attributes, described in the previous section. Operational latency can further be decomposed into the three operational costs, each as a function of the interaction protocol, i , that is used for the workflow execution. The three costs are the data management cost, D_c , the general agent execution cost, E_c , and the network communication cost, N_c . Therefore, operational latency can be defined as:

$$O_L(i) = D_C(i) + E_C(i) + N_C(i), \text{ where:} \quad (1)$$

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

The functions of data management, execution, and communication are formally described in greater detail in other work [9].

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 latency. 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 costs. Therefore, the interactions are only evaluated on execution costs and network communication costs. For the purpose of comparing different operations (execution costs and communication costs), we set a baseline, x . This baseline is exactly the cost for sending one agent-to-agent message. In previous work (Blake 2003), we discovered that the execution cost for BAs was approximately $\frac{1}{4}$ of the baseline, x . Since the BAs are distributed, there are no additional system costs if a large number of BAs attempt to accept the same job. The CAs, however, have to search a large number of criteria to locate the correct one, therefore the cost of evaluation for these type agents is slightly more (approximately equivalent to the baseline, x). A listing of all delays in the interactions (both paradigms) is shown Table 1a, b.

As aforementioned, we consider execution costs 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

Table 1a. 3rd Party control for deferred-choice processing (Total = 4x)

Agent action	Execution cost (E_c)	Network cost (N_c)
BA (2..n) post criteria	–	x
CA evaluates criteria of all BAs	x	–
CA posts decision	–	x
Chosen CA is notified	–	x

Table 1b. P2P coordination for deferred-choice processing (Total = 1.25x)

Agent action	Execution cost (E_c)	Network cost (N_c)
BA (2..n) evaluate criteria	0.25x	–
BA (2..n) take the initiation record	–	x

costs and execution costs are not additive when multiple BAs work in parallel. By inspecting Table 1a, b, it is clear that the P2P coordination paradigm is greater than 3 times more efficient based on the numbers provided from the WARP prototype (Blake 2001). Though the performance is better in the P2P paradigm, there are situations where the third-party control paradigm may be optimal. If the criteria needs to be changed frequently, 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).

6 A simulation-based evaluation approach and application

In Sect. 5, there is a basic 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 costs with the increase or decrease of load on the independent agents. Secondly, data management costs can be associated with accessing the selection criteria for both the BA and CA. The data management costs can also vary based on the type of information accessed or the location of the information. Finally, communication costs can vary based on the location of the destination agent. In the following sections, we discuss the design of a simulation software that handles these factors and initial experimental results using 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 as shown in Fig. 7. This software was developed in C++ and contains approximately 2,350 lines of code. Although there are many freely available

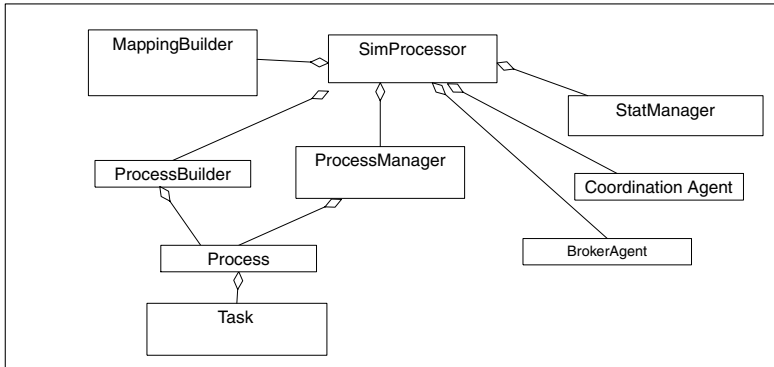


Fig. 7. Object-oriented design for COACHES simulation-based evaluation application

simulation tools on the market, we decided to develop a new simulation application based on our belief that a specialized application would facilitate the integration with a process modeling tool in future studies.

The COACHES evaluation tool consists of 9 classes, Sim Processor, Process Manager, Process Builder, Process, Task, Broker Agent, Coordination Agent, Mapping Builder, and Stat Manager. The Sim Processor object is the main control of the simulation. It is composed of a Process Builder object that encapsulates different process traffic flows based on a container of multiple Process objects (consisting of Task objects). These traffic flows can be created to emulate different compositions of traffic thus incorporating the historical network attributes described in Sect. 4. The combination of these objects can assist in the evaluation of agent interactions with respect to service changes and concurrent processes. The Process Manager object feeds Process objects to the Coordination Agent objects and the BrokerAgent objects. Independent Coordination Agent objects and Broker Agent objects represent real-world agents.

As such, these objects can be configured to perform differently as their load is increased. These Broker Agent and Coordination Agent objects report delays and operational conditions using the Stat Manager object. The Stat Manager object creates a report based on delay and queue size. The Sim Processor 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 Mapping Builder binds broker agents to services and coordination agents to processes at run-time.

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 s of delay for each 10 additional concurrent processes with a limit of

4 s for delay. 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 s. 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 expected, the average system delay of the 3rd party control scenario remained consistent at 16 s. Therefore in these experiments shown in Table 2, the operation of the P2P-oriented scenario is more efficient considering the current traffic options. This is reasonable since the current traffic is well within normal operational parameters. In other experiments, we discovered that a sustained queue of 200 tasks was the threshold where the 3rd party control scenario becomes more efficient (Blake 2003b) based on the same set of operational parameters.

As shown in Fig. 8a, b, both scenarios operate the best 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. Since the number of requests is less than the capacity that can be serviced in one cycle, then all services are completed with the lowest estimation of delay. For both scenarios, the queue size varies with different compositions of 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.

Table 2. P2P-oriented measures (*left*) and 3rd party control measures (*right*)

Traffic Type	AveDel (BA)	AveSystemQue (BA)	AveSysDel (CA + BA)	AveQue (CA + BA)
Batch 10	5.29966	17.4411	16	41.2649
Batch 2	5.82	23.7267	16	41.6835
Climb 14	5.79861	26.1771	16	45.3933
Trickle	5.82333	23.9333	16	45.59

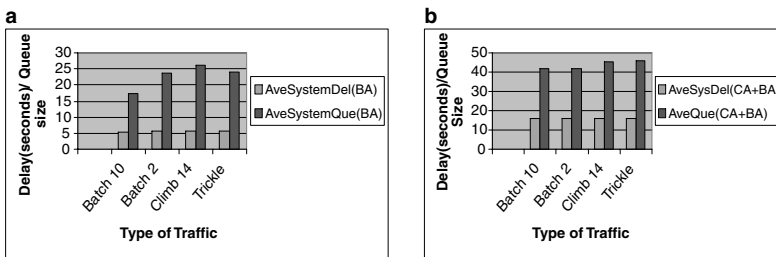


Fig. 8. a P2P-oriented measures. b. 3rd party control measures

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. Another interesting result is that average system delay does not appear to be a function of the average queue size. We believe this result demonstrates the true need for real-time tools of this sort to evaluate agent interactions for service composition at run-time.

7 Discussion and related work

This work is closely related to the areas of agent collaboration, workflow, and service composition (Casati et al. 2000; Grefen et al. 2000; Zeng et al. 2001). There are several other projects that use agent theories for the workflow composition of services. Helal et al. (2001) uses an agent architecture for workflow enactment with consideration to web services using SOAP. In fact, later work (Jagatheesan and Helal 2003) considers UDDI-registered services. Chen et al (2000) also consider the use of agents for workflow with semi-structured specification languages. Finally, Singh et al. (2001; Yolum and Singh 2002) 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 (Chaib-draa and Dignum 2002; Dignum and Greaves 2000; FIPA 2002) 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 investigates and 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 publish/subscribe paradigms for agent interactions, our work builds on our earlier work (Blake 2003) and related work (Moro and Viroli 2001; Oki et al. 1993; Nickerson 2003) 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 (Benatallah et al. 2003). Benatallah et al. (2002) 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 costs of data management, execution cost based on parallel processing, and network communication costs. These studies are performed on a stand-alone simulation application developed specifically for the purpose of evaluation. An enhancement 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 executed in just a few seconds, which suggests this approach to be viable for the real-time evaluation of agent interactions for service composition.

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 architectures should be organized and distributed. In this work, we developed applied approaches to investigating how operational costs 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. A future investigation, in order to gain more precision, would require the modeling of heterogeneous agents. This study used peer-level agents that have the same operational costs. A more realistic setting would modeling agents of varying algorithmic and computational capabilities, thus considering computational costs in addition to the current operational latency studies.

We also discussed the development of a simulation approach to evaluate these interactions in real-time. This is the first approach, known to the author, toward the evaluation of control architectures and mechanisms for service composition routines 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 additional workflow patterns. In addition, we plan to investigate the feasibility of automated approaches to conceptualizing agent interactions for particular workflow patterns. Finally, we plan to incorporate variations of computational costs among peer-level agents to evaluate system-level impact.

References

- Aberer K (2001) P-Grid: A self-organizing access structure for P2P information systems. *Proc Coop Info Sys (CoopIS2001)*, pp 179–194
- Aksit M, Bergmans L (✉) Guidelines for Identifying Obstacles when Composing Distributed Systems from Components, In: Aksit M (ed) *Software Architectures and Component Technology: The State of the Art in Research and Practice*, Kluwer Academic Publishers, pp 29–56
- Benatallah B, Dumas M, Sheng Q, Ngu A (2002) Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In *18th International Conference on Data Engineering*, San Jose, CA
- Benatallah B, Sheng Q, Dumas M (2003) “The Self-Serv Environment for Web Services Composition.” *IEEE Internet Computing*, 7(1): 40–48
- Blake MB (2000) *Agent-based Workflow Modeling for Distributed Component Configuration and Coordination*, Ph.D Dissertation, George Mason University, online at: <http://www.cs.georgetown.edu/~blakeb/pubs/diss.zip>
- Blake MB (2001) “WARP: Workflow Automation through Agent-Based Reflective Processes”, *Proceedings at the 5th International Conference on Autonomous Agents*/ACM Press, Montreal, Canada (software demonstration)
- Blake MB (2002) B2B Electronic Commerce: Where Do Agents Fit In?, *Proceedings of the AAAI-2002 Workshop on Agent Technologies for B2B E-Commerce*, Edmonton, Alberta, Canada

- Blake MB (2003a) "Agent-Based Communication for Distributed Workflow Management using Jini Technologies", *International Journal on Artificial Intelligence Tools (IJAIT)*, vol 12, No. 1, World Scientific Publishers
- Blake MB (2003b) "Evaluating Agent-to-Agent Interactions for Workflow-Oriented Service Composition: Third-Party Control or P2P?", http://www.cs.georgetown.edu/~blakeb/newPapers/blake_JAAMAS2003.pdf (under review)
- Blake MB (2003c) "Agent-Oriented Compositional Approaches to Services-Based Cross-Organizational Workflow" http://www.cs.georgetown.edu/~blakeb/newPapers/blake_DSSJournal2003.pdf (under review)
- BPEL4WS (2003): <http://www.ebpm1.org/bpel4ws.htm>
- Casati F, Jin L, Ilnicki S, Shan MC (2000) An Open, Flexible, and Configurable System for Service Composition. HPL technical report HPL-2000-41
- Chaib-draa B, Dignum F (2002) Trends in Agent Communication Languages, *Computation Intelligence*, 18(2): 89–101
- Chen Q, Dayal U, Hsu M, Griss ML (2000) Dynamic-Agents, Workflow and XML for E-Commerce Automation. *EC-Web 2000*: 314–323, London, UK
- Dignum F, Greaves M (2000) *Issues in Agent Communication*, Springer Vol. 1916
- FIPA Interaction Protocol Specification (2002) <http://www.fipa.org/repository/ips.html>
- Gelernter D (1991) Current Research on Linda. *Research Directions in High-Level Parallel Programming Languages*: pp. 74–76
- Gijzen JWJ, Szirbik NB, Wagner G (2002) 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*, Ed. M. Brian Blake and Maria Gini 7(1): 9–34
- Grefen P, Aberer K, Hoffner Y, Ludwig H (2000) CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises; *International Journal of Computer Systems Science & Engineering*, 15(5): 277–290
- Heineman G, Council W (2001) *Component-Based Software Engineering Putting the Pieces Together*, Reading, MA: Addison-Wesley
- Helal A, Wang M, Jagatheesan A, Krithivasan R (2001) Brokering Based Self Organizing E-Service Communities". *Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems (ISADS2001)* With an Emphasis on Electronic Commerce, Dallas, Texas
- Jagatheesan A, Helal A (2003) "Sangam Universal Interoperable Protocols for E-service Brokering Communities using Private UDDI Nodes". *Submitted to the IEEE Symposium on Computers and Communications - ISCC'2003*, Antalya, Turkey
- Jennings NR, Sycara KP, Wooldridge M, (1998) A Roadmap of Agent Research and Development *Journal of Autonomous Agents and Multi-Agent Systems*. 1(1): 7–36
- Moro G, Viroli M (2001) "Enabling business cooperation using a publish-subscribe architecture aware of transactions," *Proc of the 34th Hawaii International Conference on System Sciences*
- Nickerson JV (2003) Event-based Workflow and the Management Interface. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, Jan 6–9
- Oki BM, Pfluegl M, Siegel AD, Skeen D (1993) The information bus- an architecture for extensible distributed systems, *Proc of the Fourteenth ACM Symposium on Operating Systems Principles*
- Petrie C, Bussler C (2003) "Service Agents and Virtual Enterprises: A Survey" *IEEE Internet Computing*, (to appear)
- Singh MP, Yu B, Venkatraman M (2001) Community-based service location. *CACM* 44(4): 49–54
- SOAP (2003) <http://www.w3.org/TR/soap12-part0/>
- Thatte S (2001) "XLANG: Web Services for Business Process Design". Microsoft
- The CHAIMS Project (2002): <http://www-db.stanford.edu/CHAIMS/>.
- UDDI (2003) <http://www.uddi.org/>
- Van der Aalst WMP (2003) "Don't go with the flow: Web Services composition standards exposed", *IEEE Intelligent*

- Van der Aalst WMP, ter Hofstede AHM, Kiepuszewski B, Barros AP (2002) Workflow Patterns. QUT Technical report. FIT-TR-2002-02, Queensland University of Technology, Brisbane, 2002 (<http://tmitwww.tm.tue.nl/research/patterns/wfs-pat-2002.pdf>)
- Web Services (2002) <http://www.w3.org/2002/ws/desc/>
- Wooldridge M, Jennings NR (1995) Intelligent Agents: Theory and Practice. *Knowledge Engineering Review* 10(2)
- Workflow Patterns (2003): <http://tmitwww.tm.tue.nl/research/patterns/patterns.htm>
- WSDL (2003) <http://www.w3.org/TR/wsdl>
- WSFL (2003): <http://www.ebpml.org/wsfl.htm>
- Yolum P, Singh MP(2002) "Agent-Based Approach for Trustworthy Service Location". *Proceedings of the Workshop on Agents and Peer-to-Peer Computing (AP2PC 2002)* at AAMAS2002, Bolgona, Italy
- Zeng L, Ngu A, Benatallah B, O'Dell M (2001) An agent-based approach for supporting cross-enterprise workflows. *Proceedings of the 12th Australasian Conference on Database Technologies*, 123–130, Queensland, Australia