

Forming Agents for Workflow-Oriented Process Orchestration

M. Brian Blake

Department of Computer Science

Georgetown University

Washington, DC 20057

(202) 687-3084

blakeb@cs.georgetown.edu

ABSTRACT

The sophistication and maturity of distributed component-based services and semantic web services provide a promising environment for specification-driven service composition. Given the dynamic nature of this domain, 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 in constant flux as far as 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 evaluation of agent interactions, as an initial step, toward the efficient formation of agent teams. In the approaches presented in this paper, we evaluate interaction protocols for business process orchestration in response to certain environmental conditions.

Keywords

Agent Architecture, Service-Oriented Computing, Workflow

1. INTRODUCTION

On-line businesses have new opportunities involving developmental paradigms where high-level component-based services and semantic web services [16] 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)). 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[10]).

1.1 Background on Distributed Services

Though, we introduce the new term, SCW, the ideas are not unlike 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 [7], 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 [9]. 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) [12] 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) [19] 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)[14] 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) [13][5][20].

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_serviceDetail` methods in the UDDI specification [14]). 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. We assert that agent technologies represent the best 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 [17][8]. Entities classified as 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 effect change when negative nonfunctional events occur.

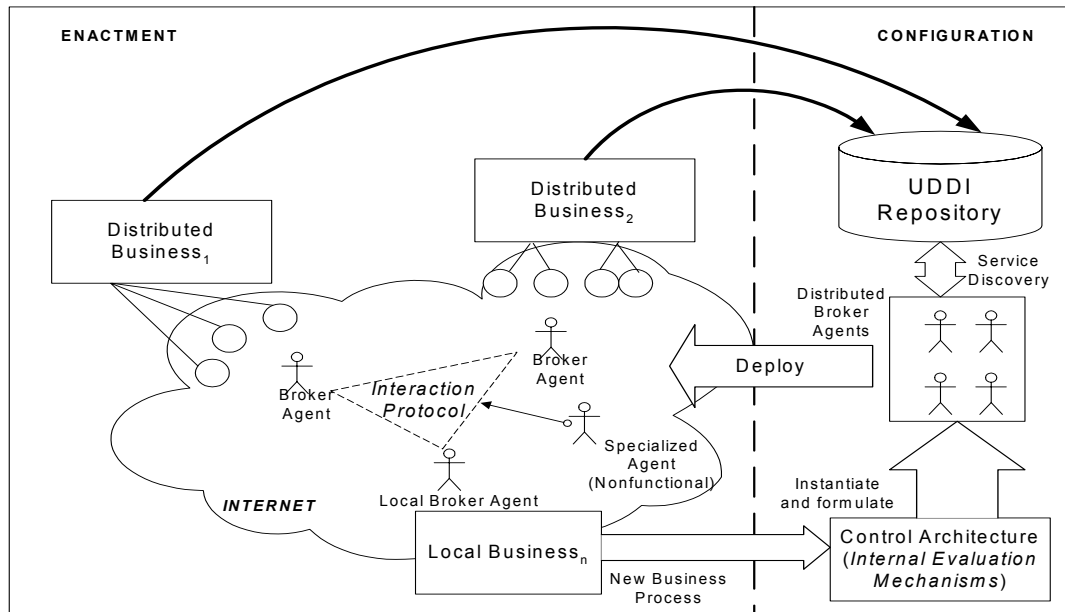


Figure 1. Agents Integrated into the SCW Environment

In a configuration environment, these broker agents can be instantiated and given a specific service requirement as a basis 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 coordinate to compose the cross-organizational services. This environment is illustrated in Figure 1.

1.3 Efficient Formation of Collaborative Agent Teams for Service Composition

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

- What are the roles of agents in the SCW environment?
- What are the alternative interaction protocols among these agents and how are they evaluated?
- Can the various interaction protocols be evaluated to determine their efficiency for a particular case while incorporating 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 Figure 2.

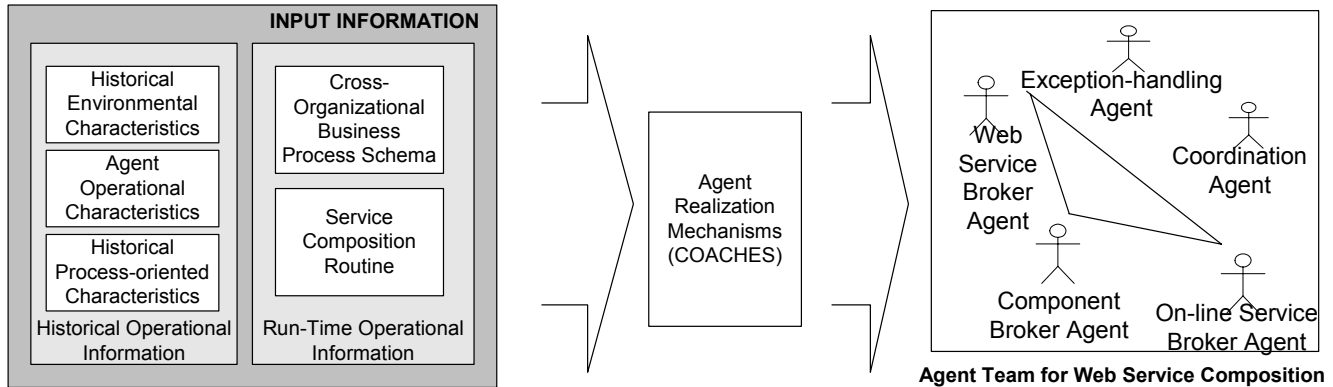


Figure 2. Agent Interactions for Workflow-based Service Composition.

The paper continues in Section 2 with a discussion of multiple interaction protocols based on atomic workflow paradigms and routines. In Section 3, 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 agent team realization and the need for real-time simulation approaches for the evaluation of the agent-to-agent protocols in Section 4. In the final section, we discuss our experiments and 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 supports that process. Van der Aalst maintains a repository of atomic workflow routines descriptions or workflow patterns that define common workflow operations used to 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 [15], he evaluates the semantic effectiveness of common composition languages such as BPEL4WS, XLANG, WSFL, BPML, and WSCI against 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 describe in detail here.

2.1 Using COACHES for the Deferred-Choice Workflow Pattern

In the COACHES environment, we decompose a business process into a set of workflow routines which we describe using Van der Aalst's workflow patterns. By evaluating the efficiency of a specific agent team to handle each workflow pattern contained in a business process, then automated mechanisms can decide the composition of an agent team best suited for the job. In this work, we have conceptualized multiple interaction protocols that realize certain workflow patterns. In our initial studies, we consider the set of workflow patterns for exclusive-or processing, synchronization, and deferred-choice, re-configuration, workflow instance creation, and exception-handling. The reader should understand that the schema of a business process may specify any number or combination of workflow patterns. Therefore, to

evaluate such a business process, each pattern must be evaluated against a particular agent team. However, in the scope of this paper, we describe the application of the COACHES approach to just one workflow pattern, deferred-choice.

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 some sub-set of the resultant information from the activity can be used as the criteria for choosing which service continues.

2.2 Generating Agent Interaction Protocols: Implementing the Deferred-Choice Pattern

A challenge in this research is determining variations in agent teams to support workflow patterns. In addition, 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 implemented in applications such as IBM's TupleSpace or Sun Microsystems' JavaSpaces. More detail is given in earlier work [3]. In addition, we define a set of basic roles for agents to achieve. In initial studies, we investigate agent-based roles such as brokering services, initiating and/or coordinating activities, handling exceptions, monitoring and/or proactively enhancing performance, and other specialized management activities.

As aforementioned, a deferred-choice scenario occurs when a business process specifies that there is the 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. In all cases, there is an

option of the type of interactions that must take place to perform this process. In an interaction protocol for deferred choice, we identify two types of for agents, a broker agent (BA) and a coordination agent (CA). The main task of the broker agent is to execute the underlying web service. The (workflow) coordination agent at this facility has the responsibility to encapsulate an evaluation criteria and which agent of many will complete the task.

In Figure 3a and Figure 3b, we define two different interaction protocols that can be used to implement the deferred-choice pattern. In Figure 3a, an initiation event is sent to two or more

broker agents. All applicable broker agents receive the initiation event as an exclusive-or request and immediately post to the event server 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 posting as 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.

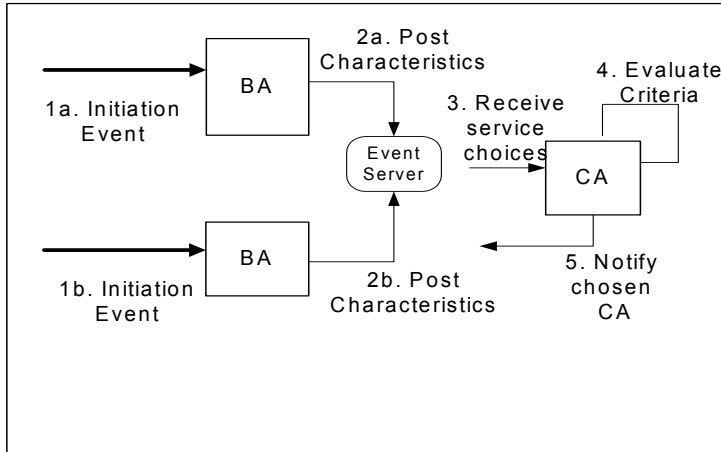


Figure 3a. Third-Party Control-Oriented Interactions for Deferred-Choice Pattern

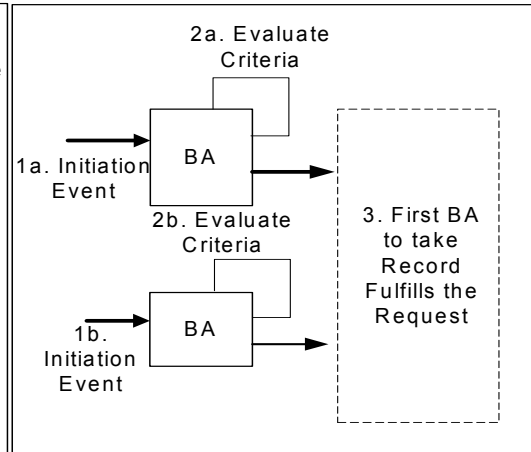


Figure 3b. Peer-to-Peer Oriented Interactions for Deferred-Choice Pattern

In a slightly different interaction (Figure 3b), broker agents are required to contain copies of the criteria, internally. When an deferred-choice request is captured by two or more broker agents, each broker agent processes its own internal characteristics against a pre-configured criteria. 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.

The interaction protocol in Figure 3b follows more of a peer-to-peer (P2P) interaction style among the broker agents, while the interaction protocol in Figure 3a 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, call-and-return, etc.). In the following section, we discuss how these interaction protocols can be evaluated based on historical statistics, agent capabilities, and dynamic environmental concerns.

3. 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 to another. 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 initial studies, we consider attributes such as service changes, new service additions, and past network responsiveness. These 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. The COACHES approach considers this rate based on the impact that service changes has on

re-establishing bindings 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 should be deployed for a particular business process.

Historical process attributes describes trends on how business processes have been operated in the past. 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 type 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 what 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. (Cut from other document)

- *Agent-to-Agent Communication Overhead.* In the COACHES approach, we use event-based communication among agents using Linda-based 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 communication-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 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 as increasing the amount of memory maintained by an agent can have negative effects on that agent's overall performance.

4. Using Agent Operations to Evaluate the Deferred-Choice Pattern

In the scope of this paper, we evaluate the deferred-choice pattern with respect to the agent operational attributes described in the aforementioned section (*Future work is toward integrating historical network and process attributes*). From the aforementioned attributes, we extract three operational measures related to data retrieval, processing due to concurrency load, and agent communication.

4.1 Describing Operation Latency for Agent Interactions

Formally, we use the term operational latency, O_L , to designate the sum of 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:

$$O_L(i) = D_c(i) + E_c(i) + N_c(i), \text{ where:}$$

- 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.

4.2 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 that the deferred-choice interaction. To further simplify the example, we consider that the deferred-choice interaction to be 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 [4], 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 attempt to accept the same job. The CAs, however, have to search a large number of criterias to pinpoint the correct one, therefore the cost of evaluation for these type agents is equivalent to the baseline. A listing of costs for each step for both paradigms is shown in Table 4a and 4b.

Agent Action	Execution Cost (E_c)	Network Cost (N_c)
BA(2..n) Post Characteristics	-----	x
CA Evaluates All BAs based on Criteria	x	-----
CA Posts Decision	-----	x

Table 4a 3rd Party Control for Deferred-Choice Processing (Total = 4x)

Agent Action	Execution Cost (E_c)	Network Cost (N_c)
BA(2..n) Self-Evaluation based on Criteria	.25x	-----
BA (2..n) Take the Initiation Record	-----	x

Table 4b P2P coordination for Deferred-Choice Processing (Total = 1.25x)

Another assumption in this example is that the execution cost does not vary considerably; therefore their execution costs in this example remain the same even with increased load. In addition, since the idea of agents is toward distribution then the communication costs and execution costs are not additive when multiple BAs work in parallel. It is clear that the P2P coordination paradigm is more than 3 times more efficient based on the numbers provided from the WARP prototype. 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 have to be changed in one place as opposed to a change to all relevant BAs).

4.3 A Simulation-Based Evaluation Approach and Application

In the example in Section 4.2, 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 costs with the increase or decrease of load on the independent agents. Secondly, the 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 type of information accessed or the location of the information. Finally, communication costs can vary based on the location of the destination agent.

We have designed and developed a simulation-based approach to evaluating these factors in addition to the agent operational factors. Although we examined several available simulation packages, we decided a specialized tool would allow us more flexibility in the future to integrate with modeling tools such as Rational Rose. In the scope of this paper, we only give an overview of this simulation approach and application.

The SimProcessor object is the main control of the simulation. It is composed of a ProcessBuilder object, a MappingBuilder object, a StatManager Object, and several Agent-proxy objects. The ProcessBuilder object encapsulates different process traffic flows based on the multiple Process objects (consisting of Task objects). The Process and Task objects are loaded from a configuration file at run-time. This configuration file can be changed to model different forms of traffic. 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 agent-representing objects report delays and operational conditions to the StatManager, which sends a report to the SimProcessor. Finally the SimProcessor object makes the final decision on which interaction is most efficient and for what purposes.

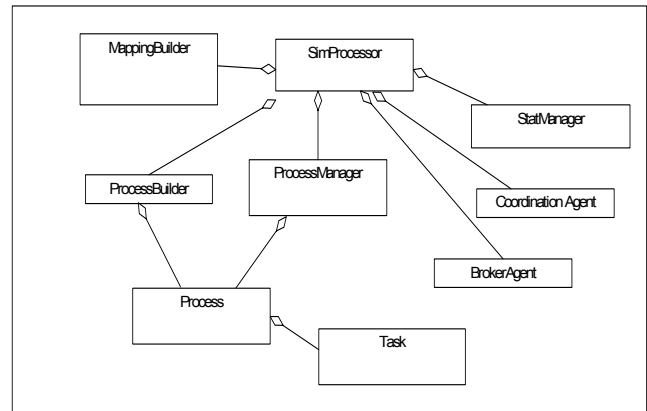


Figure 4. Object-Oriented Design for COACHES Simulation-based Evaluation Application.

5. Discussion and Related Work

Investigations in this research are 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 investigate formal 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. Though these results were determined using manual evaluation, we believe this approach is promising in the creation of automated tools. As such, we also discussed our initial development of a simulation approach to evaluate these interactions in real-time.

The innovations of this work are toward the empirical examination of the underlying protocols. Though research in forums for agent communication languages 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. Low-level studies of this sort are not included in aforementioned work. The work, that is most related to these approaches, is the studies performed by the SELF-SERV project [2]. Benatallah [1] 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 decompose execution costs into the aforementioned three operational costs of data management, execution cost based on parallel processing, and network communication costs. These performance measures are based on numerical figures from the earlier agent systems prototype [3], but the additional extended studies are performed on a stand-alone simulation applications developed specifically for the purpose of evaluation. Other work investigates service composition with concentration on the service-oriented aspects [6].

In the COACHES approach, we evaluate the control architecture and mechanisms for the specific service composition routines using empirical methods, particularly in the web services domain. In addition, we intend to explore the effects on this architecture with respect to factors as a result of the nature of the Internet. A weakness of our approach is the concentration on latency. An improvement to these the COACHES approach would be an additional emphasis on computational cost. In future work, we plan to model other agent-to-agent interactions that implement more of the set of workflow patterns. In addition, we plan to develop several sets of process traffic that test the impact of concurrent processes and service changes as specified in Section 3.

6. Conclusion

In this paper, we discussed an approach to evaluating agent interactions when performing the management of business processes. We evaluated this approach using the deferred-choice workflow pattern. In future work, we plan to evaluate several other workflow patterns as we strengthen automated tool support.

7. References

- [1] Benatallah, B., Dumas, M., Sheng, Q., and Ngu, A. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In *18th International Conference on Data Engineering*, February 2002
- [2] Benatallah, B., Sheng, Q., and Dumas, M. "The Self-Serv Environment for Web Services Composition," *IEEE Internet Computing*, Jan/Feb 2003, 7,1, pp. 40-48
- [3] Blake, M.B., "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
- [4] Blake, M.B., "Agent-Oriented Compositional Approaches to Services-Based Cross-Organizational Workflow" (under review)
http://www.cs.georgetown.edu/~blakeb/newPapers/blake_A_AMASJournal2003_extended.pdf
- [5] BPEL4WS (2003): <http://www.ebpm.org/bpel4ws.htm>
- [6] Casati, F., Jin, L., Ilnicki, S. and Shan, M.C. An Open, Flexible, and Configurable System for Service Composition. HPL technical report HPL-2000-41.
- [7] Heineman, G. & Council, W. *Component-Based Software Engineering Putting the Pieces Together*, Reading, MA: Addison-Wesley, 2001.
- [8] Jennings, N.R., Sycara, K. P., and Wooldridge, M., A Roadmap of Agent Research and Development In *Journal of Autonomous Agents and Multi-Agent Systems*. 1(1), pages 7-36. July 1998.
- [9] M. Aksit and L. Bergmans, Guidelines for Identifying Obstacles when Composing Distributed Systems from Components, in *Software Architectures and Component Technology: The State of the Art in Research and Practice*, M. Aksit (Ed.), Kluwer Academic Publishers, pp. 29 – 56
- [10] Petrie, C. and Bussler, C., "Service Agents and Virtual Enterprises: A Survey" *IEEE Internet Computing*, (to appear)
- [11] Singh, M.P., Yu, B., and Venkatraman, M.: Community-based service location. *CACM* 44(4): 49-54 (2001)
- [12] SOAP (2003) <http://www.w3.org/TR/soap12-part0/>
- [13] Thatte, S. "XLANG: Web Services for Business Process Design". Microsoft, 2001
- [14] UDDI (2003) <http://www.uddi.org/>
- [15] Van der Aalst, W.M.P. "Don't go with the flow: Web Services composition standards exposed", *IEEE Intelligent*, February 2003
- [16] Web Services (2002) <http://www.w3.org/2002/ws/desc/>
- [17] Wooldridge, M. and Jennings, N. R.. *Intelligent Agents: Theory and Practice*. In *Knowledge Engineering Review* 10(2), 1995.
- [18] *Workflow Patterns* (2003):
<http://tmitwww.tm.tue.nl/research/patterns/patterns.htm>
- [19] WSDL (2003) <http://www.w3.org/TR/wsdl>