

Object-Oriented Modeling Approaches to Agent-Based Cross-Organizational Workflow Systems

M. Brian Blake

Department of Computer Science
Georgetown University
Washington, DC 20057
(202) 687-3084

blakeb@cs.georgetown.edu

Hassan Gomaa

Department of Information and Software Engineering
George Mason University
Fairfax, VA 22031
(703) 993-1652

hgomaa@gmu.edu

ABSTRACT

With the increasing popularity of component-based services and semantic web services, the idea of specification-driven service composition is becoming a reality. With the distribution of these autonomous services, a realizable goal will be the transformation of the Internet into a *universal service repository*. In such an environment, intelligent agents may play a significant role in configuring and enacting the workflow composition of the atomic distributed services to create entirely new higher-level services. In this work, there is a large-scale agent-based architecture to support such a distributed service environment. Furthermore, we introduce an object-oriented modeling approach and software engineering developmental approach towards the programming, configuration, and operational control of the agents that manage processes in this cross-organizational workflow environment.

General Terms

Design, Experimentation, Languages, Theory

Keywords

Agent architectures, Software Process, Workflow, UML

1. INTRODUCTION

On-line businesses are beginning to adopt a developmental paradigm where high-level component-based services and semantic web services [15] will be sufficient in modularity and autonomy to fulfill the requirements of other businesses. We use the term, *services-based cross-organizational workflow* (SCW), 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) [3]). This term is also associated with the idea of a third-party organization that composes the services of multiple businesses (also described as virtual enterprise [8]). Though, there are many other related projects that define cross-organizational workflow, we further distinguish our work by defining an architecture that uses the autonomy of agent technologies.

The SCW environment, with respect to our work, incorporates the interoperability of general services. To the extent services can be represented using the web services-oriented paradigm or using local invocation, this environment contains general services, web-based services, component-oriented services, or invocation-based services. In Figure 1.1, there is a SCW environment of multiple travel-related businesses. The initiating business is the travel agency company. The Travel

Agency has internal services for managing customers' accounts and credit card numbers. However, the travel agency uses other third-party vendors to realize the hotel reservation and car rental reservations. The Hotel Reservation and Car Rental companies register their offerings as web services in a distributed registry, such as a UDDI registry [13]. The Travel Agency uses these registry services as a part of its internal workflow. In addition, the Travel Agency has a partnership with an on-line publishing company that publishes the finalized itineraries. In this case, the travel agency has a static connection with the partner organization and is able to access services directly over a shared network connection.

In the event that the Travel Agency has automated services to collect customer requests, local developers may desire to set up a SCW environment to automatically respond to these customer requests. To support this scenario and possibly other business-oriented routines, there is the need for a framework that supports some process and methodology for workflow-oriented service specification. The workflow developers should be able to specify the process sequence of the local and distributed services. In addition, the message exchange must be specified. The specification methodology must contain support for non-functional concerns. The combination of all of these specifications should be adequate to configure the SCW framework.

Agents in this environment are characterized as having *weak agency* [10]. Thus having proactive and reactive characteristics and knowledge of their environment. In this work, we assert that an agent-based architecture is appropriate for the realization of this framework. In addition, we suggest the use of standard software engineering processes and languages for the specification of the aforementioned workflow processes and control. In Section 2, there is a discussion of the approach for the realization of the SCW environment. In Section 3, we introduce a new approach to modeling agents for this environment. Next, there is a discussion of the software developmental process for developing agent systems in this context. Finally, we discuss the experience using these approaches

2. THE WARP APPROACH

The WARP approach [4] is based on the use of an agent-based middleware architecture. The WARP architecture consists of software agents that can be configured to control the workflow operation of distributed services. In this section, there is a description of the architecture that supports the WARP approach.

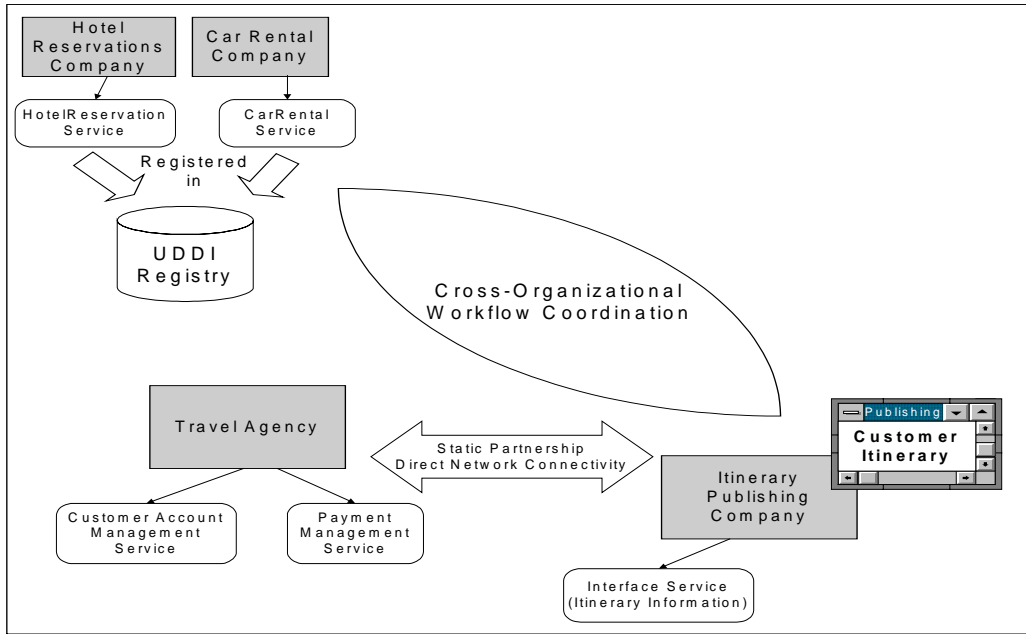


Figure 1.1 SCW Environment

The WARP architecture is divided into two layers, the application coordination layer and the automated configuration layer. The application coordination layer is the level in which the workflow instances are instantiated and the actual workflow execution occurs. The application coordination layer consists of two agents, the Role Manager Agent (RMA) and the Workflow Manager Agent (WMA). The RMAs have knowledge of a specific workflow role. The WMA has knowledge of the workflow policy and applicable roles. When a new process is configured, the workflow policy is saved in a centralized database, which is used as the agents' knowledge base. The RMA plays a role in the workflow execution by fulfilling one or more services as defined

by the workflow policy in the centralized database. The RMA registers for workflow step-level events in a centralized event server (based on tuple-space technology) based on its predefined role. When an initiation event is written into the event server, the RMA is notified. Subsequently based on its localized knowledge of services and its workflow role, the RMA invokes the correct service. The WMA has similar functionality, but instead registers for overall workflow level events (i.e. workflow initiation and nonfunctional concerns). The WMA does not control the workflow execution, but in some cases it adds events to bring about nonfunctional changes to the execution of the entire workflow.

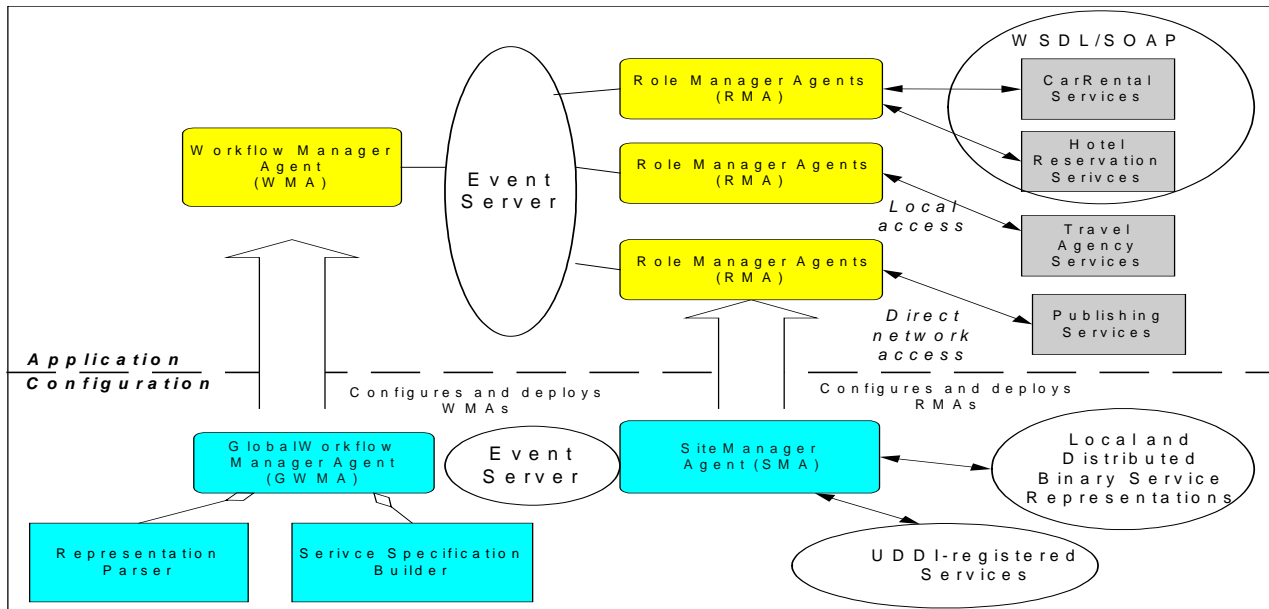


Figure 2.1 The WARP Architecture

At the automated configuration layer, agents accept new process specifications and deploy application coordination layer agents with the new corresponding policy. This layer consists of the Site Manager Agents (SMA) and the Global Workflow Manager Agent (GWMA). The GWMA accepts workflow representations/specifications from a workflow designer as input. The SMAs discover available services and provides service representations to the GWMA. This discovery can occur reflectively for local services or from a UDDI registry for distributed services. The GWMA accepts both of these inputs and writes the workflow policy to the centralized database. The GWMA then configures and deploys WMAs to play certain aspect-oriented roles. At the completion of workflow-level configuration, the SMA configures and deploys RMAs to play each of the roles specified in the workflow database. A general view of the WARP architecture is shown in Figure 2.1

3. RELATED PROJECTS

There are other projects that also investigate the workflow composition of the services. Casati [7] in the eFlow environment uses flowchart approaches to specify the workflow composition of services. Then, both the workflow and individual services are specified in an XML format. Benatallah [1] conducts research that uses UML-based state charts and formal methods for declarative peer-to-peer service composition. This research does not specifically claim to handle workflow, but the process-oriented specification is related to workflow. Also, in other work, agents are incorporated at a basic level. In addition, there is an architecture and formal methods for enacting the process enactment. This work is the most similar to work introduced in this paper and receive further comparison to our work in later sections.

There are also projects that combine agent theories for workflow enactment of services. Helal [9] uses an agent architecture for workflow enactment with consideration to web services. Chen and Griss [5] also consider the use of agents for workflow with semi-structured specification languages. Finally, Singh [12] discusses the workflow composition of services as a community of services. The major emphasis in this work is an approach to the discovery of services.

In the generalization of research in this SCW area, all projects seem to support the idea of multiple layers, a layer for configuration and an enactment layer. In some architectures, these layers are further decomposed. In most cases, there is an understood benefit of allowing services to remain on the providers' servers while composition occurs locally using distributed invocation. Another common approach is the use of some proxy component or agent that represents the services. These proxy components/agents encapsulate the knowledge of service capabilities and how to execute the services. With the use of proxy component/agents to wrap services, there must be a manager/coordination/process component or agent to control the composition. In all cases there is some language (either text-based or visual) that is used to allow the specification of the composition which is then later compiled or interpreted by the manager or coordination component.

Casati and Benatallah consider complex workflow-oriented interactions with visual and text-based specification approaches. Both adopt formal approaches to the service composition, but have fairly rudimentary, non-agent-oriented interactions in the

operation of their respective internal architectures. The work of Helal, Chen, and Singh all concentrate on the agent-oriented interactions for service composition. However, the service composition specification languages used to program their agents are neither visual nor do they seem capable of handling the complex workflow interactions that are supported in our work or the support given in the work of Casati and Benatallah.

The WARP approach follows closely to the state of the art as far as the layered approach to proxy agents, coordination agents, and specification language. In addition, our approach follows the state of the art of other related research (as Casati and Benatallah) by incorporating the use of both a visual language and a corresponding textual language for specification. We extend the approaches of Casati and Benatallah by considering both the complex workflow-oriented interactions among the services *and* the complex interactions of agents internal to our architecture. Prior related work has concentrated on either one or the other. Another innovation in the WARP approach is the software developmental process in the creation of these composite systems that respect workflow protocols. We believe this software development lifecycle, which will be discussed in later sections, is consistent with industry-standard software engineering lifecycles. In fact, the use of industry-standard modeling methodologies (as the UML) additionally tends to make the integration into industry processes more realizable. There are other variations and innovations in the WARP approach which will be covered in the technical details provided in the following sections.

4. OBJECT-ORIENTED MODELING FOR THE SCW ENVIRONMENT

Though there are numerous aspects of the WARP approach, in this paper, we highlight three major areas. Similar to related work (Benatallah and Singh), we adopt the ideas of elementary services and service communities. Elementary services, in the context of the WARP approach, are atomic component-based services, invocation-based services, and/or web services. In Section 4.1, we describe how agents characterize these services and store them in an agent-accessible repository for later composition. With respect to this work, a service community is the workflow composition of elementary services. In WARP, this service community is a virtual community, since services are distributed. Accessible to the agents is a data repository that maintains the information that defines the workflow-oriented composition specifications that make up this virtual community. In Section 4.2, there is a discussion of the agent specification approach, which uses industry-standard software approaches.

4.1 Modeling Elementary Services

Site Manager Agents (SMA) are responsible for the automated capturing of service characteristics in the WARP environment. The SMAs have two basic functions for gathering services, registry access and reflection. In registry access (UDDI), SMAs use the access methods provided by the registry. These access methods are relatively straight-forward (i.e. the *find_service* and *get_serviceDetail* methods in the UDDI specification [13]). The other function of the SMA is to gather service characteristics directly from the binary representations of the services. The SMA makes use of reflective architectures to discover the operational semantics of certain services. Reflection can be defined as a

programming approach that has both a base (integration) language as well as a meta-language that describes the base language. Two such reflective approaches for component-service development are the JavaBean API and .NET. Such reflective approaches allow developers to determine the characteristics of previously deployed software. The act of determining these characteristics at run-time is called introspection. The SMAs are able to reflectively gather operations, pre and post conditions directly from the previously compiled services, without the requirement of having initial source code or specifications. This reflective function is defined in previous work [4], thus not in the scope of this paper to discuss in greater detail. The overview of this automated population procedure is illustrated in Figure 4.1.

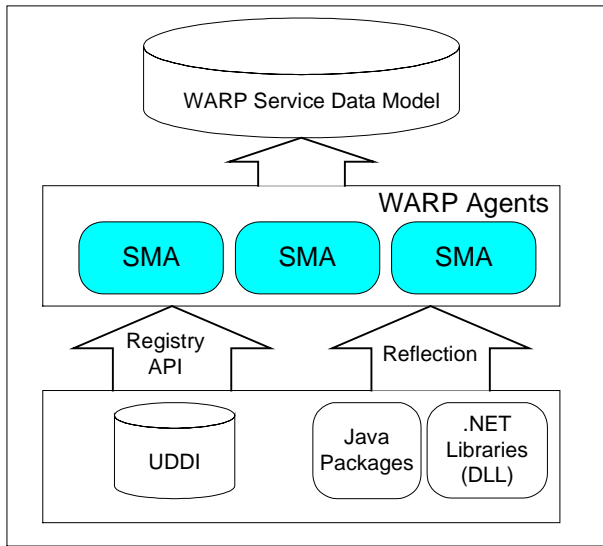


Figure 4.1 SMA Population Procedures

The SMAs operate on an object-oriented data model of service information that is gathered by the aforementioned functions. Similar to earlier discussion, this data model consists of 4 entities, Component, Operation, Precondition, and Postcondition. The Component entity defines the components that are available for composition. The component entity describes the component-oriented characteristics as location and network path. It is the assertion of this work that an individual component may contain multiple elementary services. Therefore an aggregation of the component entity is the Operation entity which specifies the independent elementary services. Each service consists of a number of Preconditions and Postconditions as defined by the respective entities. In these entities, the Pre/Postconditions are further defined by their data type. The Pre/Postconditions contain a *Value_Name* field that stores the data. Since message information is heterogeneous across different types of services, to normalize the data we create this agent-adapted field using ontological approaches. This approach is presented in other work [16], thus we do not discuss it in further detail in this paper. Finally, the unique identification numbers are used to distinguish components, services, and pre/postconditions that may be named similarly. An illustration of this data model is shown in Figure 4.2.

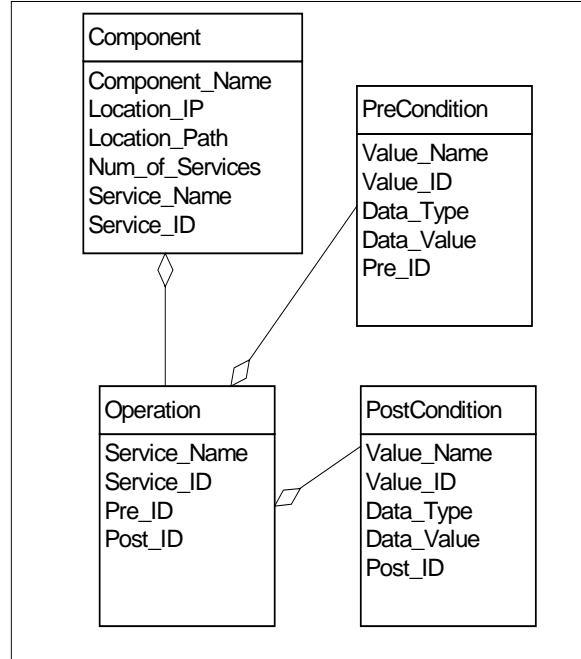


Figure 4.2 Object-Oriented Representation of SMA Data Model

4.2 Modeling the Service Community

Considering the fact that, in previous steps, elementary services have been discovered, captured, and stored by the SMAs, the next step in the WARP approach is where humans intervene to model the workflow composition of these services. A service community can be defined as a repository of these compositions. In the specification of this service community, the WARP approach incorporates industry-standard modeling approaches, such as the use of UML. In this section, a subset of the workflow modeling semantics are described in detail.

The workflow language here follows workflow terminology used commonly by researchers (as in Lei and Singh [5]). In order to set the nomenclature for further discussion, the following set of definitions are adhered to throughout this paper.

- A *task* is the atomic work item that is a part of a process.
- A task can be implemented with a *service*. (In complex cases, it may take multiple services to fulfill one task)
- An *actor* or resource is a person or machine that performs a task by fulfilling a service.
- A *role* abstracts a set of tasks into a logical grouping of activities.
- A *process* is a customer-defined business process represented as a list of tasks.
- A *workflow model* depicts a group of processes with interdependence.
- A *workflow* (instance) is a process that is bound to particular resources that fulfill the process.

The WARP approach separates the semantic modeling of the workflow from the specification of services that implement the model. The *task*, *role*, *process*, and *workflow model* terms are

used for workflow process specification. However, the terms, *service*, *actor*, and *workflow instance*, specify implementation-oriented information. This is not a new approach, but one that is necessary to assure that the services and the workflow modeling can evolve separately.

In the WARP approach, workflow processes are specified using UML activity diagrams and class diagrams. Several investigations have undertaken that show that UML activity diagrams do not have the appropriate semantics for complex workflow modeling [6]. At this point in investigations, we are unable to formally invalidate those findings. However, in the WARP approach, we introduce an approach not investigated in prior work where the workflow process is specified with one diagram. In this approach, workflow processes are specified with multiple diagrams. This is a new approach to modeling workflow, because it promotes the *separation of concerns* in workflow specification. It is the belief of the authors that this approach will allow for greater and more efficient specificity. In addition, multiple agents can be deployed to implement the workflow process with respect to individual concerns. Currently, this approach has been investigated using the basic *workflow patterns* as defined by van der Aaslt [14]. These patterns include *normal sequence*, *parallel split*, *synchronization*, *exclusive choice*, and *simple merge*.

There are three major concerns that are specified in WARP-based models, structural, dynamic, and nonfunctional concerns. The structural concerns deal with specification of workflow roles and how those roles are associated with specific workflow processes. From an implementation aspect, the description of underlying services must be considered in addition to their binding to workflow roles. The dynamic concerns incorporate control flow and message flow for normal workflow operation. Nonfunctional concerns consider such things as performance, atomicity, and error-handling. Nonfunctional concerns tend to be peripheral concerns important to the operation of the workflow. We adopt two groupings as specified by Kamath [11], *failure atomicity* and *execution atomicity*. In the failure atomicity grouping, models must define sequence actions that must take place when errors occur. The execution atomicity grouping includes specification of services and groups of services that are incompatible in the same workflow instance or across instances.

Using the WARP approach, the aforementioned three workflow concerns can be modeled using multiple models and views in UML. These models and views are summarized in Figure 4.3. The two central models are the Control Model and Role Collaboration Model. These models are described using control-based and information-based activity diagrams. This approach is similar to previously mentioned related work that uses activity diagrams. One distinguishing feature of the WARP approach is the separation of control flow and message flow into two different activity diagrams. The remainder of the specifications are views that further describe the models. These views are the *Service Representation View*, the *Role Association View*, the *Workflow Structural View*, the *Failure Atomicity Views*, and *Execution Atomicity Views*.

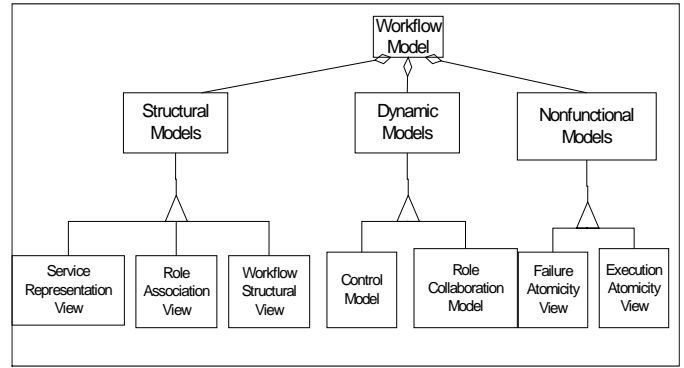


Figure 4.3 Overview of WARP Models

4.3 Service Representation View

Since services are captured autonomously by the SMAs, there must be some representation such that the available services can be presented to the human workflow modeler. In the WARP approach, visual representations are presented to the human designer prior to modeling. The Service Representation View allows the GWMA to represent the services available to be modeled in context of the organization housing them. This view is represented in a UML class diagram. The class name represents a unique identifier of the organization. Class operations are used to depict the independent services available within that organization. Service composition can occur independent of organizational boundaries, however workflow modelers typically need knowledge of available organizations. In addition, this organizational name is later associated with the routing to the services. This view is represented in Figure 4.4.

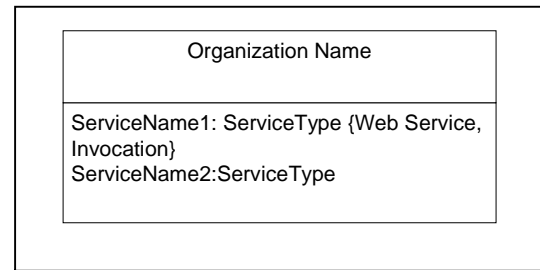


Figure 4.4. Service Representation View

4.4 Role Association View

The Role Association View allows a modeler to define workflow roles based on the distributed services that the particular role will encapsulate. This view is also illustrated in a class diagram. This view builds on the Service Representation View such that additional classes are added, that represent each of the roles. These classes are then associated to the service-based classes. An unnamed association assumes that all services from that organization are used or available to that role. However, the association name can be used to declare a list the specific services used in the role. The Role Association View is illustrated in Figure 4.5

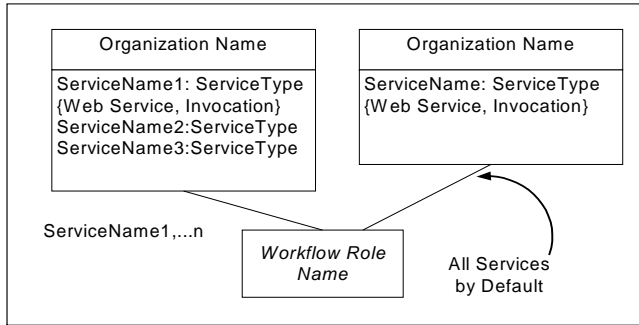


Figure 4.5 Role Association View

4.5 Workflow Structural View

The Workflow Structural View allows a modeler to specify which roles will be enacted in a particular process. Similarly, a class diagram is used that incorporates the workflow roles specified in the Role Association View. The views in the structural models are all tightly integrated. The Workflow Structural View is illustrated in Figure 4.6.

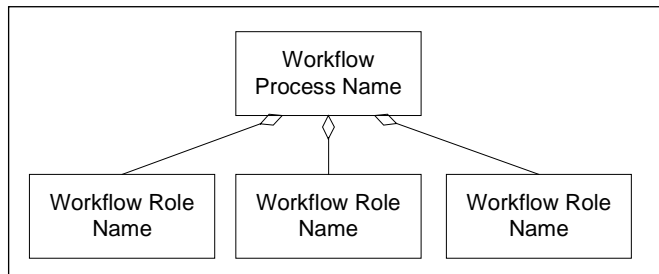


Figure 4.6 Workflow Structural View

4.6 Dynamic and Nonfunctional Models

The semantics for the Control Model and the Role Collaboration Model follow closely to related work using activity diagrams for workflow [6]. Each role can be illustrated with a new UML swimlane. Each time a role executes a specific service an activity state (oval) is placed in the swimlane. The major difference that distinguishes the WARP approach is the use of the Control Model as an activity diagram that describes the sequence of actions, in addition to the Role Collaboration Model that describes the exchange of messages. In the Control Model, standard fork and join notations are used and the transitions are illustrated with solid arrows. In the Role Collaboration Model, the dotted arrow notation is used between messages. Class notations are used between the services to illustrate the individual messages. The class name represents a message as defined by the modeler. The attributes of this class are a list of the Value_Names, that act as the set or subset of postconditions of one service that serve as the set of preconditions to the subsequent service. The notations for these models and basic exemplar model samples are illustrated in Figure 4.7

The workflow designer may also model common nonfunctional concerns. The first concern is the assurance of failure atomicity or recovery, when some domain-related problem occurs. The Failure Atomicity Views consists of a Control Model and the Role Collaboration Model. The major difference is that default values are stipulated with the Value_Name attributes. In addition, agents can parse Failure Atomicity Views that mix control flow and message flow in one diagram. This feature was allowed because the Failure Atomicity Views tend to consist mostly of control flows after the initial exception is realized.

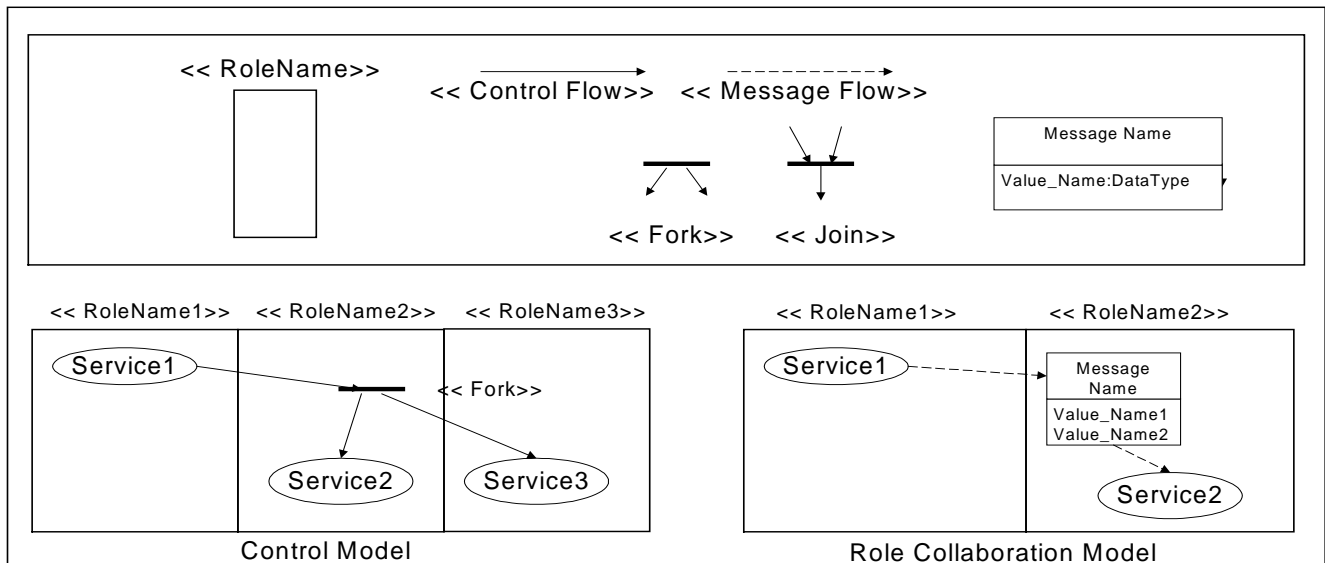


Figure 4.7. The Dynamic Models

Agents in the WARP architecture monitor messages for the existence of these exception-driven values. The existence of these values serve as a the trigger for the execution of the exception-handling-specific workflows (specified in the Failure Atomicity Views). Consequently, for each possible exception, there is a corresponding set of Failure Atomicity Views.

In addition, UML stereotypes are used to specify actions that must be taken to correct services that have been executed in the process of correcting the workflow state. Several common actions represented as stereotypes are <<abort>>, <<roll-back>>, <<roll-back and abort>>, <<re-execute>>, <<roll-back and re-execute>>, <<initiation>>. To illustrate this modeling approach in a concrete example, an incorrectly formatted customer identification scenario is shown in Figure 4.8.

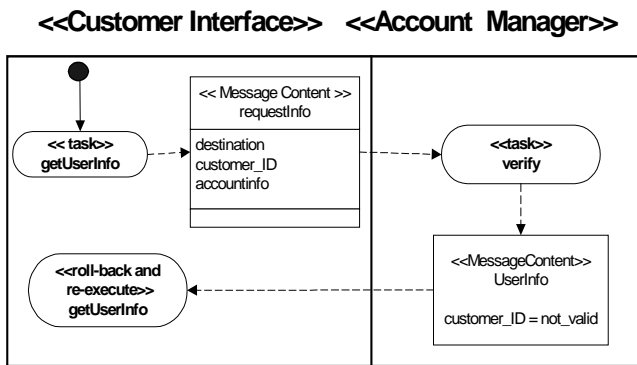


Figure 4.8 Concrete example of the Failure Atomicity View

There was one major assumption made in this approach. This assumption is that services contain basic error-handling functionality. Actions such as roll-back and re-execute have to be supported by the services internally. It is not unrealistic to assume that services will be constructed with internal error correcting capabilities. However, the WARP architecture also requires that these error-correcting actions must be implemented in a relatively uniform manner so that agents can invoke them universally. Though web service messages have support for this, this manner of development is clearly not the current state of heterogeneous services developed in different enterprises. This is a weakness in the WARP approach that serves as an avenue for future research. Execution Atomicity Views are used in the initial attempts for cross workflow instance modeling. Since this work is preliminary and in the consideration space, we choose not to illustrate these models in the scope of this paper.

4.7 Capturing the Process Models

In the WARP architecture, GWMA's operate on information extracted from the WARP models and views as represented in the data model in Figure 4.9. The main table for the process specification is the *Workflow Policy* table. Each record in this table defines a single process transition. A transition can be defined as the control flow between the completion of a service or group of services as a precondition of the initiation of a subsequent service or group of services. These services are grouped in the *EventGrouping* table. There is also a *Role* table that defines a role based on a group of services. The *ExecutionAtomicity* and *FailureAtomicity* tables are used to capture the nonfunctional concerns of exception-handling, atomicity, performance, and security. All tables represent the long-term storage in the run-time operation of the workflow.

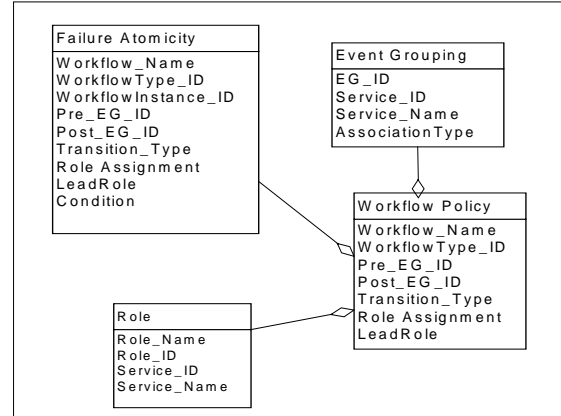


Figure 4.9 OO Model of the Process-Oriented Data

5. AGENT-SCW SOFTWARE PROCESS

Software design and configuration in the WARP environment, in a general sense, consists of five steps as illustrated in Figure 5.1. In the first step, Site Manager Agents are deployed locally or with access to distributed services that are required for the cross-organizational workflow composition. This discovery can occur on services in UDDI registry or locally registered component-based services. These SMAs search for relevant services, and, in the second step, save the service characteristics in the service-oriented data model as illustrated in Figure 4.2. Also in the second step, with help from the Global Workflow Manager Agent, these service characteristics are captured in WARP models. In the third step, a workflow designer accesses the available services as Service Representation Views. The workflow designer then performs the cross-organizational process models. Once the process modeling is complete, in the fourth step, the GWMA captures the WARP models and extracts the raw information. This raw process information is stored in the process-oriented data model as illustrated in Figure 4.10. Consequently, an integrated data model of both service and process data models is ready for agent access. In the final step, application-layer agents (Workflow Manager Agents and Role Manager Agents) access the integrated data model and configured themselves for workflow enactment in the SCW environment. The focus of this paper is on the modeling as the agent self-configuration operations are presented in other work [4]

6. DISCUSSION

The first major contribution of this work is a reusable agent-based architecture and software process that supports the composition of distributed services. This is one of few projects that develop approaches using industry standard developmental approaches such as the use of the UML. We introduce the use of multiple UML views to separate concerns in the specification of the agent system that supports this SCW environment. By using multiple views and software agents to extract the operational data, service evolution and process evolution can occur independently. Furthermore, a workflow designer can control this change by visually modeling the workflow design in available object-oriented software engineering tools. This approach has been applied to a WARP prototype [2] where several successful scenarios were implemented with encouraging performance results.

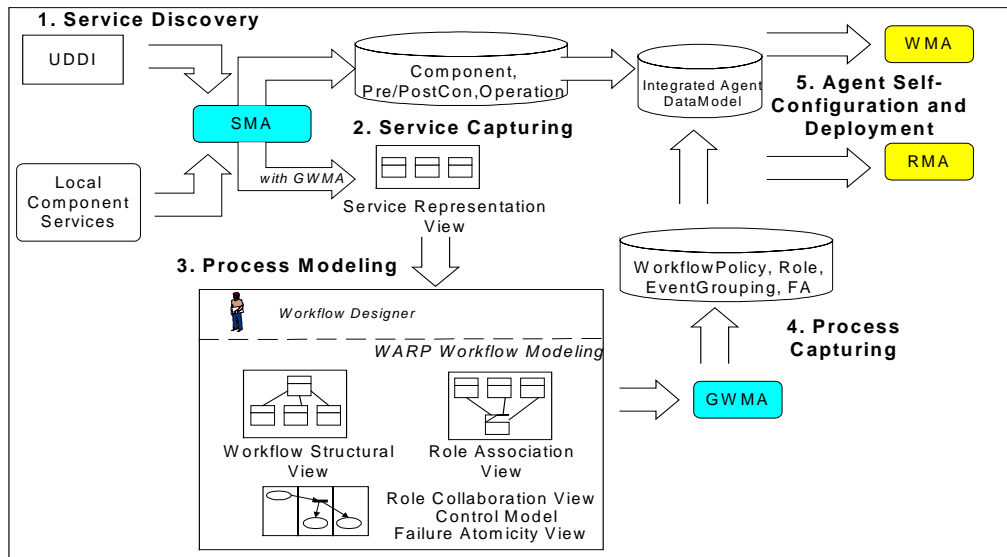


Figure 5.1 An Agent-Integrated Software Development Process to for the SCW Environment

There were several issues encountered in this approach that are the source for future investigations. As briefly discussed in earlier sections, we have started investigating agent approaches for the major problem of data integration and modeling [16]. Since pre/postconditions can be represented as heterogeneous formats, such as plain text, concrete objects, or XML, modeling approaches and agent support have a formidable challenge to create operational patterns to deal with these formats. Another similar problem is support for heterogeneous methods of error-handling. An inefficient approach taken in this work is to create new models for each error-handling case. We assert these cases can be more efficiently wrapped into the standard operational models. Finally, future work would consist of further evaluating this approach with advanced workflow patterns and interactions.

REFERENCES

- [1] Benatallah, B., Dumas, M., Sheng, Q., and Ngu, A. Declarative composition and peer-to-peer provisioning of dynamic web services. ICDE 2002, San Jose, CA Feb. 2002
- [2] Blake, M.B. " WARP: Workflow Automation through Agent-Based Reflective Processes ", Proceedings at the 5th International Conference on Autonomous Agents/ACM Press, Montreal, Canada, May 2001 (software demonstration)
- [3] Blake, M.B. "B2B Electronic Commerce: Where Do Agents Fit In?" In Proceedings of the AAAI-2002 Workshop on Agent-Based Approaches to B2B Electronic Commerce, Edmonton Canada, August 2002
- [4] Blake, M.B. *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>
- [5] Chen, Q. Dayal, U., Hsu, M. and Griss, M.L.: *Dynamic-Agents, Workflow and XML for E-Commerce Automation*. EC-Web 2000: 314-323, London, UK
- [6] Dumas, M. and ter Hofstede, A.H.M., "UML Activity Diagrams as Workflow Specification Language", UML 2001, Toronto, Canada, October 1-5, 2001, Proceedings. [Lecture Notes in Computer Science](#) 2185 Springer 2001, ISBN 3-540-42667-1
- [7] F. Casati, L. Jin, S. Ilnicki, M.C. Shan. An open, Flexible, and Configurable System for Service Composition. HPL technical report HPL-2000-41.
- [8] Gijsen, J.W.J., Szirbik, N.B., and Wagner, G. Agent Technologies for Virtual Enterprises in the One-of-a-Kind-Production Industry, International Journal of Electronic Commerce, Vol. 7, No.1 pp 9-34, October 2002
- [9] Helal, Wang, A.M., Jagatheesan, A., and Krithivasan, R. "Brokering Based Self Organizing E-Service Communities". In 5th International Symposium on Autonomous Decentralized Systems (ISADS) March 26-28, 2001 Dallas, Texas
- [10] 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.
- [11] Kamath, M. and Ramamrithan, K., "Correctness Issues in Workflow Management," Distributed Systems Engineering Journal-Special Issue on Workflow Systems, 3(4): 213-221, December 1996
- [12] Singh, M.P., Yu, B., and Venkatraman, M.: Community-based service location. CACM 44(4): 49-54 (2001)
- [13] UDDI Specification 2.04 (2002) : <http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.pdf>
- [14] Van der Aalst, W.M.P. "Don't go with the flow: Web Services composition standards exposed", IEEE Intelligent, February 2003
- [15] Web Services (2002) <http://www.w3.org/2002/ws/desc/>
- [16] Williams, A.B., Padmanabhan, A., and Blake, M.B., " Local Consensus Ontologies for B2B-Oriented Service Composition", Proceedings of the AAMAS2003, July 2003