

An Agent-Based Cross-Organizational Workflow Architecture in Support of Web Services

M. Brian Blake

*Department of Computer Science
Georgetown University
234 Reiss Science Building
Washington, DC 20057
blakeb@cs.georgetown.edu*

Abstract

The latest trend in system interoperability is the emerging concept of web services. Web services promote the use of the Extensible Markup Language (XML) to represent services (Web Service Description Language (WSDL)), their locations and interactions (Simple Object Access Protocol (SOAP)). Using the web services paradigm with the Internet as a medium has the potential of universal system interoperability and functional reuse. Workflow Automation through Agent-Based Reflective Processes or WARP is the initial work that has the same goals in mind. WARP uses reflection and tuple-space communication to coordinate a workflow of component-based services. The goal is toward the automatic configuration and management of low-level services (component-based). This work has the most potential with respect to business-to-business interaction (B2B). In this paper, we discuss our findings in the development of WARP and briefly discuss how these findings have relevance to future use of web services for business interactions.

Keywords : Agent architectures, workflow collaboration, tuple-space

1. Introduction

The use of web services for functional specification and interactions has attained a great deal of attention currently. SOAP [9] is a protocol that contains a framework where message composition and their responses can be specified. This protocol is specific mostly to web services over HTTP. WSDL [7] allows the specification of the services that use these messages. To date, the web services technologies are mostly toward the specification of interfaces and communication. To a certain extent, these technologies do not support the specification of complex interaction in the use of the web services. In addition, the use of XML-based specification allows for the specification of the interfaces through text-based representations. However, the use of text-based representations has proven to be less effective than visual representations. In the development of the WARP architecture, we have developed an approach that extends

the flexibility of XML by mapping it unto industry standard modeling (visual) techniques. In addition, WARP uses a tuple-space coordination-driven agent architecture [1] to manage the coordination of component-based services.

This paper will proceed with an overview of the WARP architecture. The following section explains the visual and XML-based representations that drive this architecture. Next, there is a discussion of the operational view of this architecture. Finally, there is a discussion of the aspects of WARP and its relation to the automatic management of web services.

2. WARP Overview

The WARP approach [2][3] is based on the use of an agent-based middleware architecture. This WARP architecture consists of software agents that can be configured to control the workflow operation of distributed services. 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. 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 the tuple-space 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 non-functional changes to the execution of the entire workflow.

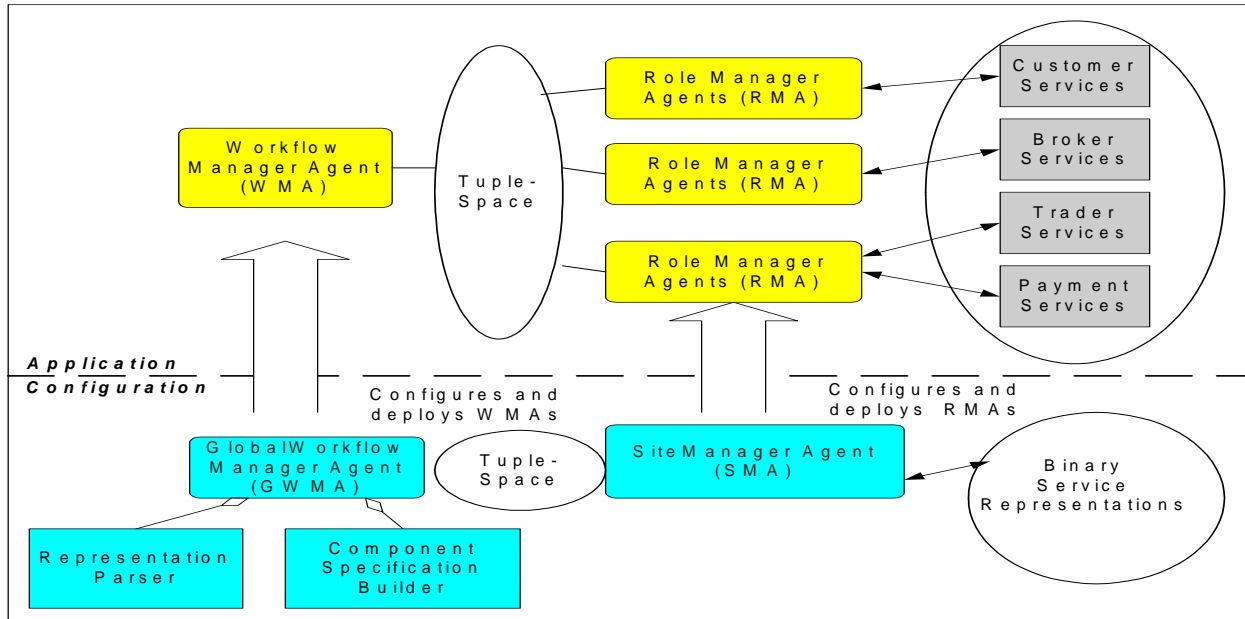


Figure 2.1. The WARP Architecture

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 from a workflow designer as input. The SMAs discover available services and provides service representations to the GWMA. The GWMA accepts both of these inputs and writes the workflow policy to the centralized database.

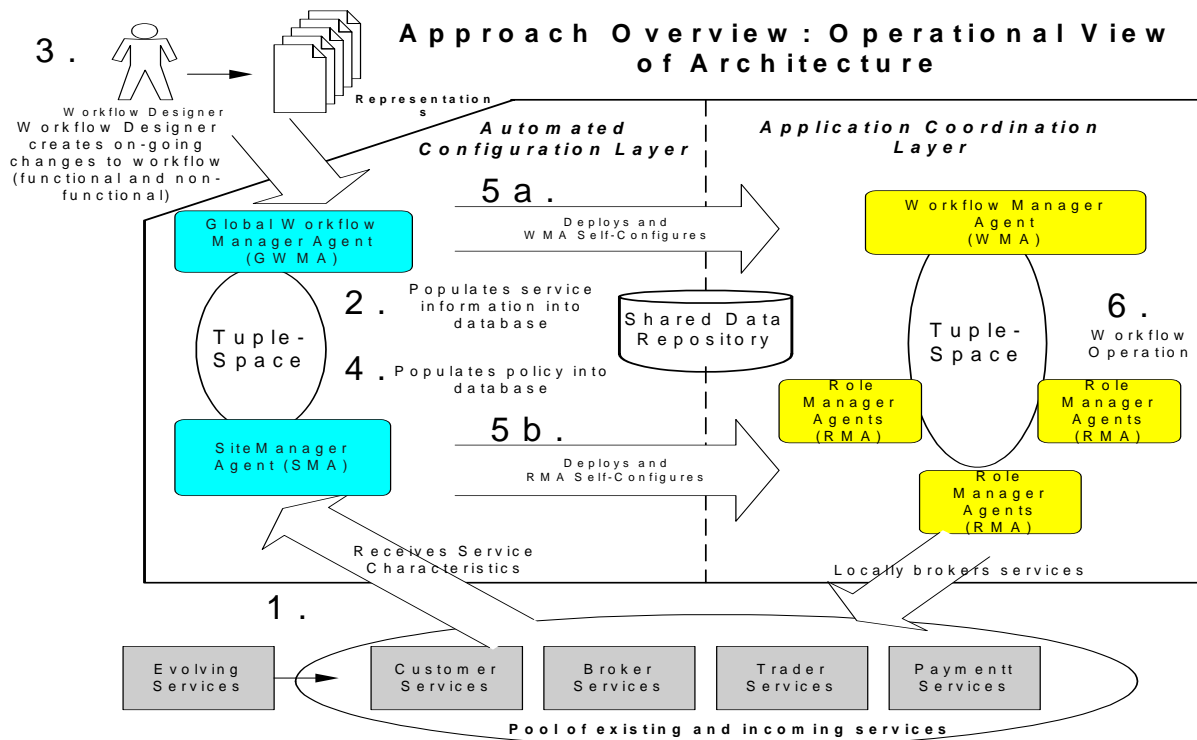


Figure 2.2 WARP Operational View

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.

In the context of a bottom-up compositional configuration, the WARP process is direct. The system takes workflow and service-based information and coordinates the operation of a workflow among previously deployed and new components. Any approach for compositional configuration has to be flexible enough to accept iterative programming. WARP architecture allows this iterative configuration with the input of workflow and service-based specifications. In bottom-up compositional configuration, there also must be mechanisms that can evolve as the services evolve. This evolution of concrete services drives the evolution of any support architecture. The WARP architecture has reflective processes that allow it to discover new service characteristics as system developers integrate new components. Finally, it is important for the support architecture to follow a process that can accurately and consistently coordinate new workflow specifications. The main focus of the WARP research is to define the components and formalize the method to achieve this semi-automated configuration.

The operational view of the WARP Architecture is illustrated in Figure 2.2. In Figure 2.2, step 1 shows how the SMA discovers service characteristics from the existing services. In step 2, the SMA populates this information into a workflow-based repository. In step 3 and 4, the GWMA gathers information from a Workflow Designer (human) and populates this information into the same workflow-based repository. In steps 5a and 5b, the application-layer agents are deployed. Step 6 consists of the real-time coordination of the workflow operations.

3. Service and Workflow Specification

Systems configuration using the WARP architecture essentially presents a new software engineering process for development. The WARP configuration process is a semi-automated process that requires the coordination of a human (workflow designer) and multiple software agents. This configuration process is split into two main parts, specification of the component-based services and specification of the workflow processes. This approach is similar to the web services approach. WARP representations specify distributed component-based services such as JavaBean components and .Net component for their part in workflow scenarios. Instead of a top-down approach where web services are developed and specifically documented with WSDL and interfaces specified with SOAP, WARP representations are introspected from pre-existing services while humans

enhance these representations with location and interaction-based information.

3.1 Workflow Terminology

The workflow language here follows workflow terminology used presently 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 a 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.

3.2 Three Perspective Modeling View

The WARP approach models services and their interactions from 3 perspectives. These models are captured visually, in message form, and in database form (for persistence). Web services paradigms basically model the messages and allow independent vendors to handle the other two perspectives. However, in the development of WARP, there was definitely some benefit in at least giving consideration to representations outside of XML-based depictions. In this approach, we use the industry standard Unified Modeling Language (UML) to model services and interactions visually. We use XML to model messages. Finally, there is a relational format for persistent storage of the models. In the WARP research, interactions are modeled for workflow interactions and paradigms as defined in Section 3.1.

3.3 Visual Modeling

The WARP approach uses UML to model services and their interactions. Using the workflow terminology, this requires the modeling of roles, workflow structure, workflow control flow and message flow, and exception-handling cases. We have investigated modeling for all of the above, but in the scope of the paper, there is only the modeling of roles, workflow structure, and operational flow models.

Lets consider an on-line stock trading workflow as the example domain. Prior to the first modeling step, the WARP architecture contains agents that are able to introspect components and create the initial models. In this case, agents start the initial modeling environment with just a few on-line stock-related services as depicted in Figure 3.3.1. These services are invocation-based services depicted as UML class diagrams. These services are a mixture of web services and local intranet-based services.

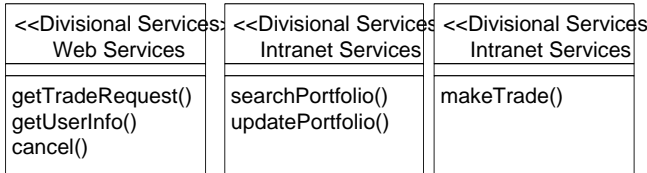


Figure 3.3.1 Service Representation View

The first human-enacted step in WARP modeling is to model all roles that are available. These roles must be associated with the services that realize them. The three roles in this domain are the Customer Interface, the Portfolio Manager, and Trading Manager. This model is shown in Figure 3.2.2. This model also is depicted in a UML class diagram, where roles are associated to their corresponding services.

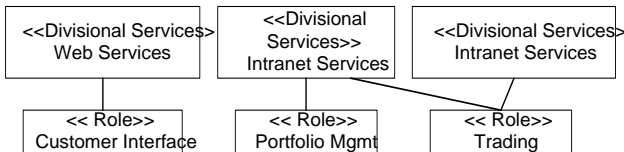


Figure 3.3.2 Role Association View

Roles are grouped into a workflow. A workflow is named and modeled as a UML class (“Stock Purchasing”). All roles associated to this workflow are shown in this view as depicted in Figure 3.3.3. In this figure, the workflow is an on-line stock purchasing workflow.

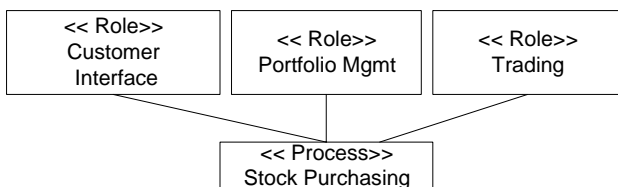


Figure 3.3.3 Workflow Structural View

The final two models are the control flow view and the message flow view. These models are the centerpieces of

the description of the workflow coordination. The models are depicted as UML activity diagrams, both for control flow and information flow. The Role Collaboration view in Figure 3.3.4 shows the sequence of roles and their services that must be executed throughout the workflow. It is typical in workflow to have branches of control flow. Activity diagrams allow for multiple paths in a workflow schema with forks and joins.

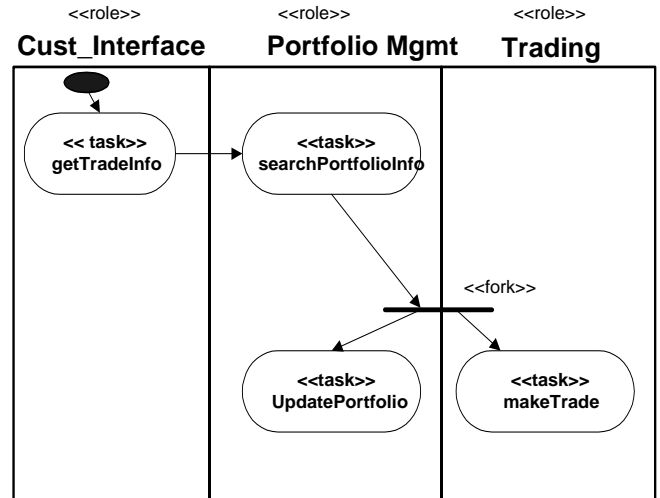


Figure 3.3.4 Role Collaboration View

The Message Collaboration View shows the exchange of information as the workflow executes. The classes that represent the data flow can map to other more complex XML schemas. This model is shown in Figure 3.3.5.

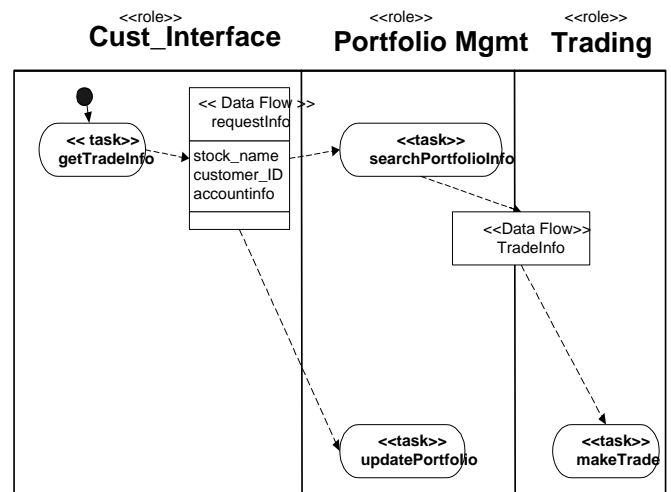


Figure 3.3.5 Message Collaboration View

The workflow models detailed in this section are just a subset of models needed for a production-level workflow and yet still only a subset of the models developed in the WARP research. However, this subset of models is a fair

representation of the minimal information needed to model a workflow of web services. Using visual modeling has been proven to be clearer with fewer multiple interpretations than their textual counterparts. The following sections map these representations onto XML messaging formats and the relational model.

3.4 Messaging Formats and Relational Models

The final two formats for workflow specification are XML based formats and the relational format. In the scope this paper, there are only brief discussions of both. The relational format in Figure 3.4.1 shows how the workflow terminology in Section 3.1 is captured. In addition, this relational format includes repositories for the workflow control flow and message flow as well as representation for the underlying services.

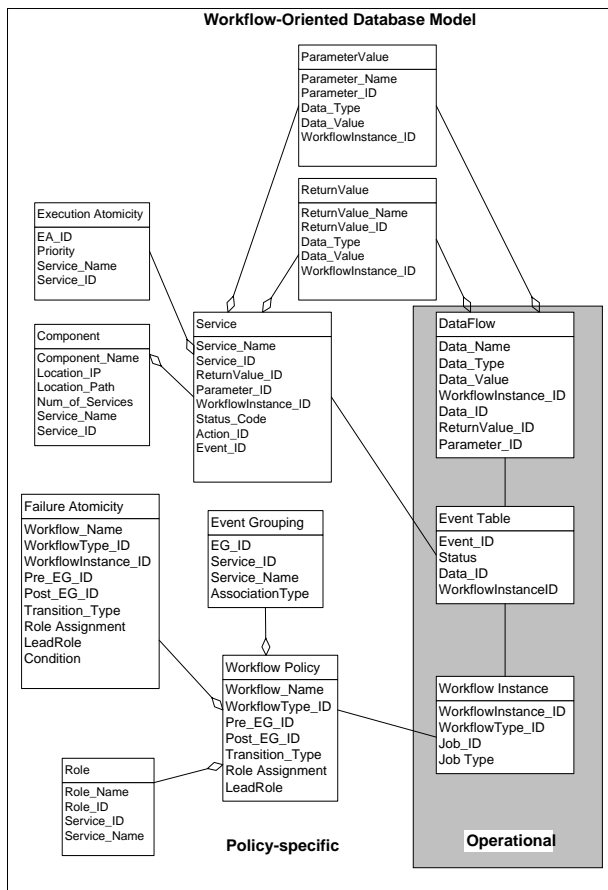


Figure 3.4.1 Relational Format for Workflow Models

There are several XML-based specifications in WARP that depict all workflow aspects. The XML schema is generic but workflow specific. The goal in this research was not to develop a proprietary XML schema, in fact, this approach maps with considerable ease, onto the latest SOAP and WSDL services. The culmination of all

models however is our current work. That culmination is weaving all models while maintaining the underlying workflow coordination. Figure 3.4.2 shows how all three models can be mapped for the on-line stock workflow policy where the schema is to verify a user's identity prior to the on-line stock purchase.

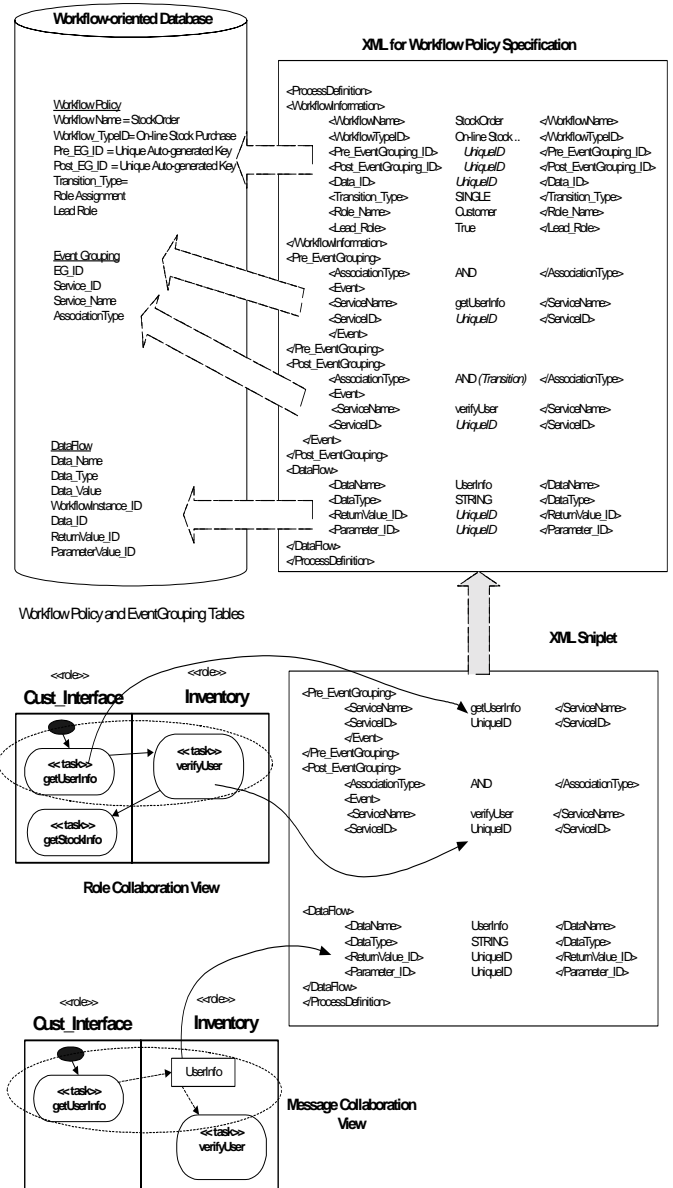
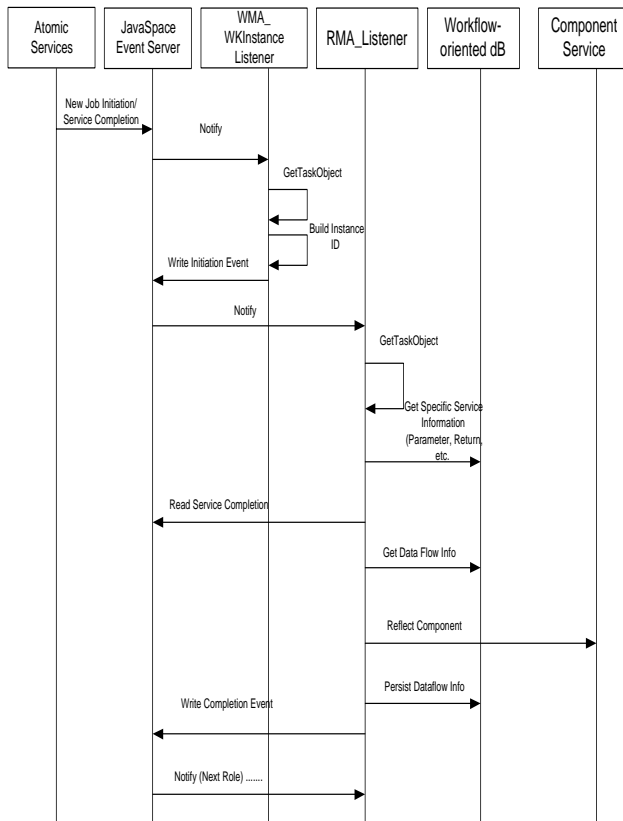


Figure 3.4.2 Integration of the Three-Model Perspective

4. Workflow Operation of On-line Services

The standard operation of the WARP agents occurs when a job initiation event or service completion is captured and the workflow process executes normally without error. In the spirit of other UML diagrams in this paper, this operational view is illustrated in Figure 4.1

using a UML sequence diagram. To summarize this standard operational flow, initially a job event or service completion initiates the workflow instance. The WMA captures this event and builds a unique ID based on the correct workflow policy as defined in its TaskObject (contain Event-Condition-Action rules for the agents). The WMA broadcasts an event that initiates the workflow. From this point on, the RMAs coordinate to complete the workflow. The RMAs have reflex engines that await certain service completions as registered in the tuple-space and perform the pertinent actions (reflectively invoking specified services). Once their workflow task in completed, they submit a completion event in the tuple-space that triggers subsequent services.



5. Discussion

The WARP approach is similar to the web services paradigm. The use of an agent-based middleware is comparable to the use of SOAP. In addition, the message format uses XML schemas, which can be mapped seamlessly to WSDL. There are other related works that uses tuple-space approaches [4][6] to coordination. However these approaches are not as specific to workflow and web services. In addition, these approaches do not adopt the three views of capturing interaction specifications.

Visual modeling has been proven in software engineering to reduce the number of interpretations in modeling requirements and interactions. Since the web services paradigm uses predominantly textual representations, the WARP representation approaches can be used to extend this approach. Using an industry standard software engineering modeling language like the UML only makes a stronger connection between the service and interaction modeling potentially for web services.

Continuing and future work is toward the extension of web services and their connection to business-to-business interactions with agents. The main thrust is integrating the modeling languages of WSDL, SOAP, and other business specific representation as ebXML and RosettaNet. Understanding system interoperability with these representations is key to developing run-time evolving applications in the B2B domain.

6. References

- [1] Blake, M.B. "KOJAC: Implementing KQML with Jini to Support Agent-Based Communications in Emarkets," *AAAI-2000 Workshop on Knowledge-based Electronic Markets (KBEM) at the 17th National Conference on Artificial Intelligence*, Austin, TX, August 2000
- [2] Blake, M.B. "Agent-based Workflow Configuration and Management of On-line Services", *Proceedings of the International Conference on Electronic Commerce Research (ICECR-4)*, pp 567-588, Dallas, TX, November 2001
- [3] Blake, M.B. "WARP: Workflow Automation through Agent-Based Reflective Processes", *Proceedings at the 5th International Conference on Autonomous Agents*, Montreal, Canada, May 2001 (software demonstration)
- [4] Ciancarini, P., Tolksdorf, R., Zambonelli, F., "Coordination Middleware for XML-Centric Applications", *Proceedings of the 16th ACM Symposium on Applied Computing*, Madrid (E), March 2002
- [5] Lei, Y. and Singh, M.P., "A Comparison of Workflow Metamodels", Workshop on Behavioral Models and Design Transformations: Issues and Opportunities in Conceptual Modeling at ER'97, Los Angeles, CA, November 1997
- [6] Tolksdorf, R. "Coordination Technology for Workflows on the Web: Workspaces. In *Proceedings of the Fourth International Conference on Coordination Models and Languages COORDINATION 2000*, LNCS, pages 36-50. Springer-Verlag, 2000.
- [7] Web Services (2002) <http://www.w3.org/2002/ws/desc/>
- [8] WSDL (2002) <http://www.w3.org/TR/wsdl>
- [9] SOAP (2002) <http://www.w3.org/TR/soap12-part0/>