QUALITY OF SERVICE AND SEMANTIC COMPOSITION OF WORKFLOWS

by

ANTONIO JORGE SILVA CARDOSO

(Under the Direction of AMIT SHETH)

ABSTRACT

Workflow management systems (WfMSs) have been used to support a variety of business processes. As organizations adopt new working models, such as e-commerce, new challenges arise for workflow systems. These challenges include support for the adequate management of quality of service (QoS) and the development of new solutions to facilitate the composition of workflow applications involving Web services. The good management of QoS directly impacts the success of organizations participating in e-commerce activities by better fulfilling customer expectations and achieving customer satisfaction. To enable adequate QoS management, research is required to develop mechanisms that specify, compute, monitor, and control the QoS of the products or services to be delivered. The composition of workflows to model e-service applications differs from the design of traditional workflows due to the number of Web services available during the composition process and to their heterogeneity. Two main problems need to be solved: how to efficiently discover Web services and how to facilitate their interoperability.

To enhance WfMSs with QoS management, we have developed a QoS model that allows for the description of nonfunctional aspects of workflow components, from a quality of service perspective. To automatically compute the overall QoS of a workflow, we have developed a mathematical model and implemented an algorithm (SWR algorithm). Our QoS model and mathematical model have been validated with the deployment and execution of a set of production workflows in the area of genetics. The analysis of the collected data proves that our models provide a suitable framework for estimating, predicting, and analyzing the QoS of production workflows.

To support, facilitate, and assist the composition of workflows involving Web services, we present a solution based on ontologies. We have developed an algorithm that workflow systems and discovery mechanisms can use to find Web services with desired interfaces and operational metrics, and to assist designers in resolving heterogeneity

issues among Web services. Our approach provides an important solution to enhance Web service discovery and interoperability.

INDEX WORDS: workflow management systems (WfMSs), quality of service (QoS), workflow composition, web services, business process management.

QUALITY OF SERVICE AND SEMANTIC COMPOSITION OF WORKFLOWS

by

ANTONIO JORGE SILVA CARDOSO

Licenciatura, University of Coimbra, Portugal, 1995

Mestrado, University of Coimbra, Portugal, 1998

A Dissertation Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2002

© 2002

Antonio Jorge Silva Cardoso

All Right Reserved

QUALITY OF SERVICE AND SEMANTIC COMPOSITION OF WORKFLOWS

by

ANTONIO JORGE SILVA CARDOSO

Approved:

Major Professor: Amit Sheth

Committee: Christoph Bussler
John Miller
Jonathan Arnold
Krys Kochut
Robert Bostrom

Electronic Version Approved:

Gordhan L. Patel
Dean of the Graduate School
The University of Georgia
August 2002

DEDICATION

To my parents and brothers.

ACKNOWLEDGMENTS

This dissertation is part of the group effort to enhance the METEOR workflow system at the Large Scale Distributed Information System (LSDIS) Laboratory of the Department of Computer Science at the University of Georgia. I wish to acknowledge several people who have been particularly helpful and supportive during my research. First, I would like to thank my major advisor Amit Sheth for his support of my studies. I also wish to thank John Miller for having played the devil's advocate so many times, advancing my research always a step further. I thank Arnold Jonathan for his help and valuable input on technical matters related to genetic processes. Thanks to Robert Bostrom and Krys Kochut for their advice, encouragement, and assistance. Thanks also to Christoph Bussler for joining my committee. Special thanks go to António Dias de Figueiredo for his support and encouragement during my Ph.D. program. I cannot close without a final word of thanks to my friends and colleagues at the LSDIS laboratory: Kemafor Anyanwu, Ketan Bhukanwala, Sonali Sheth, Zhongwei Luo, Zhongqian Li, Wil M. P. van der Aalst, David Hall, and Madalena Lordelo.

This work was supported by the European Social Fund (FSE), III Community Frame for Support (QCA), and by the Portuguese Ministry of Science and Technology (MCT).

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

Semantics are critical to support the next generation of the Web. The important contribution of the "Semantic Web", vis-à-vis the current Web, is the ability to represent and process descriptions of every resource on the Web. A resource description, informally called its "semantics", includes that information about the resource that can be used by computers – not just for display purposes, but for using it for automatic processing in various applications.

This dissertation focuses on two issues: semantic Web services and process composition. Semantic Web services are Web services with a formal description (semantics) that can enable a better discovery, selection, composition, monitoring, and interoperability. Processes are next steps to carrying out core business activities, such as e-commerce and e-services, and are created from the composition of Web Services or other components.

This dissertation is about associating semantics to Web Services, and exploiting it in process composition. The composition process involves a functional perspective and an operational perspective. The functional perspective involves Web Service Discovery, addressing semantic heterogeneity handling. The operational perspective takes form of the research on QoS specification for Web Services and Processes.

1.1 WORKFLOW MANAGEMENT SYSTEMS

Workflow management systems (WfMSs) are a key tool that can be employed by organizations motivated to improve their competitive advantage, customer service, productivity, and conformity with standards.

A few decades ago, work was carried on traditionally; work items passed from one participant or worker to another. Business processes were coordinated and managed by their participants, since they inherently knew their own business rules. With the introduction of workflow systems the process itself became automated and the system became responsible for the scheduling and execution of tasks associated with various processes.

WfMSs are based on the concept of a workflow; this is an abstraction of a business process. A workflow normally comprises a number of logical steps (known as tasks), dependencies among tasks, routing rules, and participants. A task can require human involvement, or it might be executed automatically by applications.

A workflow system reads, automates, processes, and manages workflows by coordinating the sharing and routing of information. During processing, tasks, information, and documents are passed from one participant to another in a manner governed by a set of rules, routes, and roles. Workflow instances run on one or more workflow engines which are able to interpret workflow definitions, interact with workflow participants, and, where required, invoke the use of external tools and applications.

The automation of work items increases process efficiency. Furthermore, the management and analysis of workflow instances provides an opportunity for measuring the execution of the parameters of business processes, in order that continuous improvements can be implemented (Cardoso, Miller *et al.* 2002).

1.2 WORKFLOW MANAGEMENT SYSTEMS EVOLUTION

The idea of workflow management systems arose in the 90's. Yet, there is some consensus that the office information systems (OIS) field, an important field in the 70's, is the predecessor of workflow systems (Edward and Zhao 2001). The first OIS prototypes were developed in the late seventies. Pioneer systems included the *SCOOP* project (Zisman 1977), which was oriented to the automation of office procedures, and *Officetalk* (Ellis 1979; Ellis and Bernal 1982), which provided a visual electronic desktop metaphor, a set of personal productivity tools for manipulating information, and a networked environment for sharing information.

In the 80's, due to several failures in office automation projects and installations (Hammer 1984; Suchman and Wynn 1984), the interest in office information systems declined, and work was directed towards researching flexible groupware systems and models (Ellis and Nutt 1996). Then, in the 90's, there was a resurgence of interest in OIS. New technological approaches, such as transaction processing, document image processing, and integrated office systems were investigated and developed. The fundamental support of these technologies have paved the way for the emergence of workflow management technology, which claimed to be one of the innovative applications of the 90's. Alonso *et al* (1996) point out that they are highly innovative; they also observe that workflow management systems have gained a high level of popularity. Nevertheless, these systems have not yet matured into well-proven and stable technologies.

1.3 MOTIVATION

The work described in this dissertation focuses on the enhancement of workflow systems to respond to current requirements. Two areas are studied: quality of service (QoS) management and workflow composition.

Recently, while modeling workflows to support genetic processes, we realized that a key aspect of workflows is to be able to anticipate their behavior prior to execution and then characterize their behavior during execution, according to their quality of service. Being able to carry out these two types of analysis allows organizations to better understand their workflow processes and therefore foresee the quality of services rendered to customers.

It is natural to think that if one of the most important goals of organizations is to continuously seek to raise their competitive position – through the improvement of services rendered to customers – workflow systems with quality of service support would take their place in the market. We estimate (Sheth, Aalst *et al.* 1999) that the number of readily available workflow systems is between 200 and 300; curiously, however, these systems lack the support of a comprehensible and computable QoS model.

The gap between the type of WfMSs available on the market and the type of systems actually needed reveals an interesting research problem. The workflow market offers a broad spectrum of WfMSs, but somehow the systems do not match the requirements of organizational managers. Tools and mechanisms that compute, estimate, and analyze workflow QoS are not present. The first objective of this dissertation is to eliminate this gap between technological supply and demand. The second area of research targets the development of mechanisms that facilitate and assist users during the workflow design process.

Emergent trading models, such as e-commerce, have evoked the development of systems and infrastructures to support the concept of Web services. An organization's functionality is encapsulated with an appropriate interface and advertised as Web services. While in some cases Web services may be utilized in an isolated form, it is normal to expect Web services will be integrated as a part of workflows. The composition of workflows that abstractly model e-commerce applications – such as business-to-business, business-to-customer, and customer-to-customer processes – differs

from the design of traditional workflows. The main differences are in terms of the number of tasks (Web services) available to the composition process, and in their autonomy and heterogeneity. Therefore, two problems need to be solved: how to efficiently discover Web services and how to facilitate the interoperability of heterogeneous Web services.

1.4 WORKFLOW QUALITY OF SERVICE

Organizations are constantly seeking new and innovative information systems to better fulfill their mission and strategic goals. With the advent and evolution of global-scale economies, organizations need to be more competitive, efficient, flexible, and integrated in the value chain at different levels, including the information system level. In the past decade, Workflow Management Systems (WfMSs) have been distinguished by their significance and impact on organizations. WfMSs allow organizations to streamline and automate business processes and reengineer their structure; in addition, they increase efficiency and reduce costs.

Several researchers have identified workflows as the computing model that enables a standard method of building Web services applications and processes to connect and exchange information over the Web (Chen, Dayal *et al.* 2000; German Shegalov, Michael Gillmann *et al.* 2001; Leymann 2001; Fensel and Bussler 2002). The new advances and developments in e-services and Web services set new requirements and challenges for workflow systems.

Our past research has involved the development of fully distributed enactment services for workflow management. Our infrastructure, the METEOR system, and specifically its OrbWork (Kochut, Sheth *et al.* 1999) and WebWork (Miller, Palaniswami *et al.* 1998) enactment services have been used in prototyping and deploying applications to various domains, such as bio-informatics (Hall, Miller *et al.* 2000), healthcare

(Anyanwu, Sheth *et al.* 1999), telecommunications (Luo 2000), the military (Kang, Froscher *et al.* 1999), and university administration (CAPA 1997).

Our experience with real-world applications has made us aware that existing workflow systems, both products and research prototypes, provide a set of indispensable functionalities that manage and streamline business processes. Yet, organizations operating in e-commerce and in global economies that include competitive and constantly changing markets have a new set of requirements that have not been answered by current workflow technologies. One important missing requirement is the management of quality of service. Organizations operating in modern markets, such as e-commerce, require quality of service management. Products and services with well-defined specifications must be made available to customers. An appropriate control of quality leads to the creation of quality products and services; these, in turn, fulfill customer expectations and achieve customer satisfaction.

While QoS has been a major concern in the areas of networking (Cruz 1995; Georgiadis, Guerin *et al.* 1996), real-time applications (Clark, Shenker *et al.* 1992) and middleware (Zinky, Bakken *et al.* 1997; Frolund and Koistinen 1998; Hiltunen, Schlichting *et al.* 2000), few research groups have concentrated their efforts on enhancing workflow systems to support workflow QoS management.

For organizations, being able to characterize workflows based on QoS has four distinct advantages. First, it allows organizations to translate their vision into their business processes more efficiently, since workflow can be designed according to QoS specifications. Second, it allows for the selection and execution of workflows based on their QoS, to better fulfill customer expectations. As workflow systems carry out more complex and mission-critical applications, QoS analysis serves to ensure that each application meets user requirements. For e-commerce processes, it is important to know the QoS an application will exhibit before making the service available to customers. Third, it makes possible the monitoring of workflows based on QoS. Workflows must be

rigorously and constantly monitored throughout their life cycles to assure compliance both with initial QoS requirements and targeted objectives. QoS monitoring allows adaptation strategies to be triggered when undesired metrics are identified or when threshold values are reached. Fourth, it allows for the evaluation of alternative strategies when adaptation becomes necessary. The unpredictable nature of the surrounding environment has an important impact on the strategies, methodologies, and structure of business processes. Thus, in order to complete a workflow according to initial QoS requirements, it is necessary to expect to adapt, replan, and reschedule a workflow in response to unexpected progress, delays, or technical conditions. When adaptation is necessary, a set of potential alternatives is generated, with the objective of changing a workflow as its QoS continues to meet initial requirements. For each alternative, prior to actually carrying out the adaptation in a running workflow, it is necessary to estimate its impact on the workflow QoS. For example, when a workflow managing e-commerce services becomes unavailable due to the malfunction of its components, it is essential to evaluate adaptive strategies that can be applied to correct the process.

It is furthermore essential that the services rendered follow customer specifications, in order to meet their expectations and ensure satisfaction. Customer expectations and satisfaction can be translated into the quality of service rendered. Organizations have realized that quality of service management is an important factor in their operations. Quality models, such as ISO9000 (ISO9000 2002) have been created to help organizations and their individual performers meet customer needs.

Workflow QoS is composed of different dimensions that are used to characterize workflow schema and instances. The effort of developing a comprehensive QoS model specification and its computation, covering various quality dimensions, is innovative. Most of the research carried out in order to extend workflow system capabilities to include project management features has mainly been done for the time dimension (Kao and GarciaMolina 1993; Bussler 1998; Eder, Panagos *et al.* 1999; Marjanovic and

Orlowska 1999; Dadam, Reichert *et al.* 2000; Sadiq, Marjanovic *et al.* 2000; Son, Kim *et al.* 2001); this is only one of the dimensions under the workflow QoS umbrella. Even though some WfMSs currently offer time management support, the technology available is still rudimentary (Eder, Panagos *et al.* 1999). Research on workflow reliability issues has also been conducted, but the work has mostly concerned system implementation (Kamath, Alonso *et al.* 1996; Tang and Veijalainen 1999; Wheater and Shrivastava 2000). The Crossflow project (Klingemann, Wäsch *et al.* 1999; Damen, Derks *et al.* 2000; Grefen, Aberer *et al.* 2000) is the one that most closely resembles our work. Not only is time considered, but also the cost associated with workflow executions is taken into account. In Crossflow, the information about past workflow execution is collected in a log. From this information, a continuous-time Markov chain (CTMC) is derived. Since Markov chains do not directly support the concept of parallel executions introduced by the *and*-split/*and*-join structure, the power set of the parallel activities of the tasks inside an *and*-split/*and*-join structure needs to be constructed. While for small workflows the computation of a power set is affordable, this may not be the case for large workflows with a parallel nature, for which the power set can reach millions of states. Our approach uses a different concept to compute quality of service dimensions, one which does not suffer from exponential complexity.

Our goal is to develop both a model for the specification of workflow QoS and methods for computing, estimating, and analyzing QoS. We investigate the relevant quality of service dimensions which are necessary to correctly characterize workflows. We not only target the time dimension, but also investigate other dimensions required to develop a real and usable workflow QoS model. Once the QoS and associated dimensions are selected, it is necessary to develop methods and algorithms to compute workflow QoS. In workflows, quality metrics are associated with tasks, and tasks, in turn, compose workflows. The computation of workflow QoS is done based on the QoS of the tasks that compose a workflow.

1.5 SEMANTIC WORKFLOW COMPOSITION

E-services have been heralded as the next wave of Internet-based business applications that will dramatically change the use of the Internet (Fabio Casati, Ming-Chien Shan *et al*. 2001). With the development and maturation of infrastructures and solutions that support e-services, we expect organizations to incorporate Web services into their business processes. Workflow management systems are capable of integrating business objects for setting up e-services in an amazingly short time and with impressively little cost (Shegalov, Gillmann *et al*. 2001). Workflows play a major role in architectures such as dynamic trading processes (Sheth, Aalst *et al*. 1999), dynamic value chains (Lee and Whang 2001), virtual organizations, and virtual web organizations (Ulrich 2001).

The emergent need of workflows to model e-service applications makes it indispensable that workflow tasks be associated with Web services. As a result, workflows are currently being enhanced to also include and manage Web services (Shegalov, Gillmann *et al*. 2001).

The modeling of e-services using workflows raises two challenges for workflow systems. First, Web services must be located that might contain (a) the desired functionality and (b) operational requirements needed to carry out the realization of a given task. It is necessary to efficiently discover Web services from the potentially thousands of services available on the Internet. Second, once the desired Web services have been found, mechanisms are needed to (c) facilitate the resolution of structural and semantic differences. This is because the heterogeneous Web services found in the first step need to interoperate with other components present in a workflow host.

The design of traditional workflow applications involves the selection of appropriate tasks to compose a workflow and the establishment of connections among tasks (control and data flow). Tasks are selected from a workflow repository (I.B.Arpinar, Miller *et al*. 2001; Song 2001) which typically contains only a few hundred of tasks. Since the

number of tasks to choose from is modest, the search process is humanly manageable; it does not require sophisticated discovery mechanisms. When a workflow is employed to model e-services, the number of Web services available on the Internet for the composition process can be extremely large. Thus, we are no longer searching for a task from a set of a few hundred, but we are searching for a service from a set that can potentially contain millions of Web services. One cannot expect a designer to manually browse through all the Web services available.

The interoperability problems that the composition of workflows involving Web services face are already well known within the distributed database systems community (Kashyap and Sheth 1996). When Web services are put together, their interface (inputs and outputs) need to interoperate; therefore, structural and semantic heterogeneity need to be resolved. Structural heterogeneity exists because Web services use different data structures and class hierarchies to define their interfaces. Semantic heterogeneity considers the intended meaning of the terms employed to label input and output variables. The data that is interchanged among Web services has to be understood. Semantic conflicts occur when a Web service output connected to another service input does not use the same interpretation of the information being transferred. To achieve interoperability it is necessary to address the problem of semantic integration – the identification of semantically similar objects that belong to different systems and the resolution of their schematic differences (Kashyap and Sheth 1996). The general approach to semantic integration has been to map the local terms onto a shared ontology. Even though a shared ontology ensures total integration, constructing such an ontology is costly, if not impractical, because autonomous systems are required to commit to a shared ontology and compromises are difficult to maintain when new concepts are added (Rodríguez and Egenhofer 2002). Therefore, this approach cannot be employed to assist and facilitate the resolution of semantic and schematic differences when Web services are involved.

The main motivation for this work is the need to enhance workflow systems with better mechanisms for Web service discovery and integration. Our approach relies on the use of ontologies that describe Web service interfaces. Ontology-based approaches have proved to be an important solution for information integration in order to achieve interoperability (Uschold and Gruninger 1996). Our main objective is to assess the similarity of terms and concepts that are employed to specify Web service interfaces. Our solution relies on Tversky's feature-based similarity model (Tversky 1977), which is arguably the most powerful similarity model to date (Richardson and Smeaton 1995). The model is based on the idea that common features tend to increase the perceived similarity of two concepts, and that feature differences tend to diminish perceived similarity. Tversky's model states that feature commonalities tend to increase perceived similarity more than feature differences can diminish it.

1.6 MAJOR RESULTS

This dissertation represents one of the earliest comprehensive studies on quality of service and semantic workflow composition. We propose a detailed QoS framework and its implementation in a workflow prototype system; further, we show the experimental results obtained from executing a real-world application. For the workflow composition, we describe a methodology and an algorithm based on semantics to assist users during the workflow design phase. This includes the development of mechanisms to discover and assist workflow designers in resolving interoperability issues among Web services.

Our efforts on workflow QoS management have resulted in the following advances:

a) Development of a comprehensive and predictive QoS model for workflows.
b) Development of a QoS mathematical model.
c) Development of an algorithm (the SWR algorithm) to automatically compute and estimate workflow QoS.
d) Implementation of the above elements in the METEOR workflow system.

In the area of process composition, our research has resulted in the following advances:

a) Development of a methodology for semantic workflow composition.
b) Development of an algorithm to compute the syntactic, operational, and semantic similarity of Web services and to assist designers in resolving interoperability issues among Web services.
c) Development of a prototype incorporating the above concepts.

1.7 Intended Audience

The intended audience of this dissertation is any person interested in workflow systems in general and in quality of service and the semantic composition of workflows in particular. This dissertation is of particular interest to workflow systems architects and workflow designers, as well as to researchers from the fields of business process re-engineering, e-service applications, and the interoperability of Web services.

1.8 Dissertation Organization

This dissertation is structured as follows. The first chapter presents our QoS model for workflows. To better explain the emergent need for QoS management, we describe a real-word scenario (from the genetics area) from which we enumerate the new challenges for workflow systems and portray their current limitations. The chapter describes a mathematical model which computes the quality of service of workflows based on atomic elements (tasks). We present an algorithm and describe how simulation techniques can be used to automatically compute and estimate workflow QoS. The chapter ends with an example, based on the initial scenario, which illustrates the computation of workflow QoS.

The second chapter describes the implementation of the concepts introduced in the first chapter to the METEOR workflow management system. The support of QoS management requires the modification and extension of most workflow system

components. This includes the enactment system, the workflow builder, the monitor, the code generator, the repository, the workflow model, and the task model.

Finally, the last chapter discusses the composition of workflows involving Web services. We present an algorithm that workflow systems and discovery mechanisms can use to find Web services with desired interfaces and to assist designers in resolving heterogeneity issues among Web services. The algorithm proposed uses syntactic, operational, and semantic information to establish the degree of similarity between a Web service template and Web service objects. A prototype is also described to illustrate how the discovery mechanism and interoperability are achieved.

1.9 REFERENCES

Alonso, G. (1996). Advanced Transaction Models in Workflow Contexts. Proceedings of the International Conference on Data Engineering. pp. 574-581.

Anyanwu, K., A. P. Sheth, J. A. Miller, K. J. Kochut and K. Bhukhanwala (1999). "Healthcare Enterprise Process Development and Integration.," LSDIS Lab, Department of Computer Science, University of Georgia, Athens, GA, Technical Report.

Bussler, C. (1998). Workflow Instance Scheduling with Project Management Tools. 9th Workshop on Database and Expert Systems Applications DEXA'98, Vienna, Austria, IEEE Computer Society Press. pp. 753-758.

CAPA (1997). "Course Approval Process Automation (CAPA)," LSDIS Lab, Department of Computer Science, University of Georgia, Athens, GA. July 1, 1996 - June 30, 1997.

Cardoso, J., A. Sheth and J. Miller (2002). Workflow Quality of Service. International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference (ICEIMT/IEMC'02), Valencia, Spain, Kluwer Publishers.

Chen, Q., U. Dayal, M. Hsu and M. L. Griss (2000). Dynamic-Agents, Workflow and XML for E-Commerce Automation. EC-Web. pp. 314-323.

Clark, D., S. Shenker and L. Zhang (1992). Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. Proceedings of ACM SIGCOMM. pp. 14-26.

Cruz, R. L. (1995). "Quality of service guarantees in virtual circuit switched networks." IEEE J. Select. Areas Commun. **13**(6): 1048-1056.

Dadam, P., M. Reichert and K. Kuhn (2000). Clinical Workflows: the Killer Application for Process Oriented Information Systems. 4th International Conference on Business Information Systems (BIS 2000), Poznan, Poland. pp. 36-59.

Damen, Z., W. Derks, M. Duitshof and H. Ensing (2000). Business-to-business E-Commerce in a Logistics Domain. The CAiSE*00 Workshop on Infrastructures for Dynamic Business-to-Business Service Outsourcing, Stockholm, Sweden.

Eder, J., E. Panagos, H. Pozewaunig and M. Rabinovich (1999). Time Management in Workflow Systems. BIS'99 3rd International Conference on Business Information Systems, Poznan, Poland, Springer Verlag. pp. 265-280.

Edward, A. S. and J. L. Zhao (2001). "Workflow Automation: Overview and Research Issues." Information Systems Frontiers **3**(3): 281-196.

Ellis, C. A. (1979). Information Control Nets: A Mathematical Model of Office Information Flow. Conference on Simulation, Measurement and Modelling of Computer Systems, ACM, New York. pp. 225-239.

Ellis, C. A. and M. Bernal (1982). "OfficeTalk-D: An experimental office information system." SIGOA Newsletter **3**(1): 131-140.

Ellis, C. A. and G. J. Nutt (1996). Workflow: The Process Spectrum. NSF Workshop on Workflow and Process Automation in Information Systems, Athens, Georgia. pp. 140-145.

Fensel, D. and C. Bussler (2002). The Web Service Modeling Framework. Vrije Universiteit Amsterdam (VU) and Oracle Corporation, http://www.cs.vu.nl/~dieter/ftp/paper/wsmf.pdf.

Frlund, S. and J. Koistinen (1998). "Quality-of-Service Specification in Distributed Object Systems." Distributed Systems Engineering Journal **5**(4).

Georgiadis, L., R. Guerin, V. Peris and K. Sivarajan (1996). "Efficient Network QoS Provisioning Based on Per Node Traffic Shaping." IEEE ACM Transactions on Networking **4**(4): 482-501.

German Shegalov, Michael Gillmann and G. Weikum (2001). "XML-enabled workflow management for e-services across heterogeneous platforms." The VLDB Joumal **10**: 91-103.

Grefen, P., K. Aberer, Y. Hoffner and H. Ludwig (2000). "CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises." International Journal of Computer Systems Science & Engineering **15**(5): 227-290.

Hall, D., J. A. Miller, J. Arnold, K. J. Kochut, A. P. Sheth and M. J. Weise (2000). "Using Workflow to Build an Information Management System for a Geographically Distributed Genome Sequence Initiative," LSDIS Lab, Department of Computer Science, University of Georgia, Athens, GA, Technical Report.

Hammer, M. (1984). The OA Mirage. Datamation. **30:** 36-46.

Hiltunen, M. A., R. D. Schlichting, C. A. Ugarte and G. T. Wong. (2000). Survivability through Customization and Adaptability: The Cactus Approach. DARPA Information Survivability Conference and Exposition (DISCEX 2000). pp. 294-307.

ISO9000 (2002). ISO9000. International Organization for Standardization, http://www.iso.ch/iso/en/iso9000-14000/iso9000/iso9000index.html.

Kamath, M., G. Alonso, R. Guenthor and C. Mohan (1996). Providing High Availability in Very Large Workflow Management Systems. Proceedings of the 5th

International Conference on Extending Database Technology, Avignon. pp. 427-442.

Kang, M. H., J. N. Froscher, A. P. Sheth, K. J. Kochut and J. A. Miller (1999). A Multilevel Secure Workflow Management System. Proceedings of the 11th Conference on Advanced Information Systems Engineering, Heidelberg, Germany, Springer. pp. 271-285.

Kao, B. and H. GarciaMolina (1993). Deadline assignment in a distributed soft realtime system. Proceedings of the 13th International Conference on Distributed Computing Systems. pp. 428-437.

Klingemann, J., J. Wäsch and K. Aberer (1999). Deriving Service Models in Cross-Organizational Workflows. Proceedings of RIDE - Information Technology for Virtual Enterprises (RIDE-VE '99), Sydney, Australia. pp. 100-107.

Kochut, K. J., A. P. Sheth and J. A. Miller (1999). "ORBWork: A CORBA-Based Fully Distributed, Scalable and Dynamic Workflow Enactment Service for METEOR," Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia, Athens, GA.

Leymann, F. (2001). Web Services Flow Language (WSFL 1.0). IBM Corporation, http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

Luo, Z. (2000). Knowledge Sharing, Coordinated Exception Handling, and Intelligent Problem Solving to Support Cross-Organizational Business Processes. Ph.D. Dissertation. Department of Computer Science, University of Georgia, Athens, GA.

Marjanovic, O. and M. Orlowska (1999). "On modeling and verification of temporal constraints in production workflows." Knowledge and Information Systems **1**(2): 157-192.

Miller, J. A., D. Palaniswami, A. P. Sheth, K. J. Kochut and H. Singh (1998). "WebWork: METEOR2's Web-based Workflow Management System." Journal of Intelligence Information Management Systems: Integrating Artificial Intelligence and Database Technologies (JIIS) **10**(2): 185-215.

Sadiq, S., O. Marjanovic and M. E. Orlowska (2000). "Managing Change and Time in Dynamic Workflow Processes." The International Journal of Cooperative Information Systems **9**(1, 2): 93-116.

Shegalov, G., M. Gillmann and G. Weikum (2001). "XML-enabled workflow management for e-services across heterogeneous platforms." The VLDB Joumal **10**(1): 91-103.

Sheth, A. P., W. v. d. Aalst and I. B. Arpinar (1999). "Processes Driving the Networked Economy." IEEE Concurrency **7**(3): 18-31.

Son, J. H., J. H. Kim and M. H. Kim (2001). "Deadline Allocation in a Time-Constrained Workflow." International Journal of Cooperative Information Systems (IJCIS) **10**(4): 509-530.

Suchman, L. and E. Wynn (1984). "Procedures and Problems in the Office." Office: Technology and People **2**(2): 133-154.

Tang, J. and J. Veijalainen (1999). "Using Fragmentation To Increase Reliability For Workflow Systems." Society for Design and Process Science **3**(2): 33-48.

Wheater, S. M. and S. K. Shrivastava (2000). "Reliability Mechanisms in the OPENflow Distributed Workflow System," Department of Computing Science, University of Newcastle upon Tyne Technical Report 31, Esprit LTR Project No. 24962 - C3DS First year Report, pp. 269-288.

Zinky, J., D. Bakken and R. Schantz (1997). "Architectural Support for Quality of Service for CORBA Objects." Theory and Practice of Object Systems **3**(1): 1-20.

Zisman, M. (1977). Representation, Specification and Automation of Office Procedures. PhD Dissertation. Department of Business Administration, Wharton School, University of Pennsylvania, Philadelphia, PA.

CHAPTER 2

MODELING QUALITY OF SERVICE FOR WORKFLOWS AND WEB SERVICE PROCESSES[1]

[1] Cardoso, J.S., J. Miller, A. Sheth, and J. Arnold. Submitted to the Very Large Data Bases Journal (05/27/2002).

2.1 ABSTRACT

Workflow management systems (WfMSs) have been used to support various types of business processes for more than a decade now. In workflows for e-commerce and Web services applications, suppliers and customers define a binding agreement or contract between the two parties, specifying Quality of Service (QoS) items such as products or services to be delivered, deadlines, quality of products, and cost of services. The management of QoS metrics directly impacts the success of organizations participating in e-commerce. Therefore, when services or products are created or managed using workflows, the underlying workflow system must accept the specifications and be able to estimate, monitor, and control the QoS rendered to customers. In this paper, we present a predictive QoS model that makes it possible to compute the quality of service for workflows automatically based on atomic task QoS attributes. To this end, we present a model that specifies QoS and describe an algorithm and a simulation system in order to compute, analyze and monitor workflow QoS metrics.

2.2 INTRODUCTION

Organizations are constantly seeking new and innovative information systems to better fulfill their missions and strategic goals. With the advent and evolution of global scale economies, organizations need to be more competitive, efficient, flexible, and integrated in the value chain at different levels, including the information system level. In the past decade, Workflow Management Systems (WfMSs) have been distinguished due to their significance and their impact on organizations. WfMSs allow organizations to streamline and automate business processes and reengineer their structure; in addition, they increase efficiency and reduce costs.

Several researchers have identified workflows as the computing model that enables a standard method of building Web services applications and processes to connect and

exchange information over the Web (Chen, Dayal *et al.* 2000; German Shegalov, Michael Gillmann *et al.* 2001; Leymann 2001; Fensel and Bussler 2002). The new advances and developments in e-services and Web services set new requirements and challenges for workflow systems.

Our past research has involved the development of fully distributed enactment services for workflow management. Our infrastructure, the METEOR system, and specifically its OrbWork (Kochut, Sheth *et al.* 1999) and WebWork (Miller, Palaniswami *et al.* 1998) enactment services have been used in prototyping and deploying applications to various domains, such as bio-informatics (Hall, Miller *et al.* 2000), healthcare (Anyanwu, Sheth *et al.* 1999), telecommunications (Luo 2000), the military (Kang, Froscher *et al.* 1999), and university administration (CAPA 1997).

Our experience with real-world applications has made us aware that existing workflow systems, both products and research prototypes, provide a set of indispensable functionalities that manage and streamline business processes. Yet, organizations operating in e-commerce and in global economies that include competitive and constantly changing markets have a new set of requirements that have not been answered by current workflow technologies. One important missing requirement is the management of Quality of Service (QoS). Organizations operating in modern markets, such as e-commerce activities and distributed Web services interactions, require quality of service management. Products and services with well-defined specifications must be available to customers. An appropriate control of quality leads to the creation of quality products and services; these, in turn, fulfill customer expectations and achieve customer satisfaction.

While QoS has been a major concern in the areas of networking (Cruz 1995; Georgiadis, Guerin *et al.* 1996), real-time applications (Clark, Shenker *et al.* 1992) and middleware (Zinky, Bakken *et al.* 1997; Frolund and Koistinen 1998; Hiltunen, Schlichting *et al.* 2000), few research groups have concentrated their efforts on enhancing workflow systems to support workflow Quality of Service management.

For organizations, being able to characterize workflows based on QoS has four distinct advantages. First, it allows organizations to translate their vision into their business processes more efficiently, since workflow can be designed according to QoS metrics. For e-commerce processes it is important to know the QoS an application will exhibit before making the service available to its customers. Second, it allows for the selection and execution of workflows based on their QoS, to better fulfill customer expectations. As workflow systems carry out more complex and mission-critical applications, QoS analysis serves to ensure that each application meets user requirements. For e-commerce processes, it is important to know the QoS an application will exhibit before making the service available to customers. Third, it makes possible the monitoring of workflows based on QoS. Workflows must be rigorously and constantly monitored throughout their life cycles to assure compliance both with initial QoS requirements and targeted objectives. QoS monitoring allows adaptation strategies to be triggered when undesired metrics are identified or when threshold values are reached. Fourth, it allows for the evaluation of alternative strategies when adaptation becomes necessary. The unpredictable nature of the surrounding environment has an important impact on the strategies, methodologies, and structure of business processes. Thus, in order to complete a workflow according to initial QoS requirements, it is necessary to expect to adapt, replan, and reschedule a workflow in response to unexpected progress, delays, or technical conditions. When adaptation is necessary, a set of potential alternatives is generated, with the objective of changing a workflow as its QoS continues to meet initial requirements. For each alternative, prior to actually carrying out the adaptation in a running workflow, it is necessary to estimate its impact on the workflow QoS. For example, when a workflow becomes unavailable due to the malfunction of its components, it is indispensable to evaluate the adaptive strategies that can be applied to correct the process. It is essential that the services rendered follow customer specifications to meet their expectations and ensure satisfaction. Customer expectations

and satisfaction can be translated into the quality of service rendered. Organizations have realized that quality of service management is an important factor in their operations. Quality models, such as ISO9000 (ISO9000 2002), have been created to help organizations and their individual performers meet customer needs.

Workflow QoS is composed of different dimensions that are used to characterize workflow schema and instances. The effort of developing a comprehensive QoS model specification and its computation, covering various quality dimensions, is innovative. Most of the research carried out in order to extend workflow systems' capabilities to include project management features has mainly been done for the time dimension (Kao and GarciaMolina 1993; Bussler 1998; Eder, Panagos *et al.* 1999; Marjanovic and Orlowska 1999; Dadam, Reichert *et al.* 2000; Sadiq, Marjanovic *et al.* 2000; Son, Kim *et al.* 2001); this is only one of the dimensions under the workflow QoS umbrella. Even though some WfMSs currently offer time management support, the technology available is rudimentary (Eder, Panagos *et al.* 1999). Research on workflow reliability issues has also been conducted, but the work was mostly on system implementation (Kamath, Alonso *et al.* 1996; Tang and Veijalainen 1999; Wheater and Shrivastava 2000). The Crossflow project (Klingemann, Wäsch *et al.* 1999; Damen, Derks *et al.* 2000; Grefen, Aberer *et al.* 2000) is the one that most closely relates to our work. Not only is time considered, but also the cost associated with workflow executions is taken into account. In Crossflow, the information about past workflow execution is collected in a log. From this information, a continuous-time Markov chain (CTMC) is derived. Since Markov chains do not directly support the concept of parallel executions introduced by the *and*-split/*and*-join structure, the power set of the parallel activities of the tasks inside an *and*-split/*and*-join structure needs to be constructed. While for small workflows the computation of a power set is affordable, this may not be the case for large workflows with a parallel nature, for which the power set can reach millions of states. Our approach

uses a different concept to compute quality of service dimensions, one which does not suffer from exponential complexity.

Our goal is to develop both a model for the specification of workflow QoS and methods to analyze and monitor QoS. We start by investigating the relevant quality of service dimensions which are necessary to correctly characterize workflows. We not only target the time dimension, but also investigate other dimensions required to develop a real and usable workflow QoS model. Once the QoS and associated dimensions are selected, it is necessary to develop algorithms and to select methods to compute QoS. In workflows, quality metrics are associated with tasks, and tasks compose workflows. The computation of workflow QoS is done based on the QoS of the tasks that compose a workflow. We present an algorithm and also show how a workflow system can be coupled with a simulation system in order to predict QoS. Through this paper, the word 'task' or 'workflow task' corresponds to a traditional workflow task or a Web service. It will later become evident that in order for our model to be applied to workflows, tasks or Web service only have to adhere to the QoS model.

This paper is structured as follows. Section 2.3 describes a workflow process that illustrates a real world scenario which will be used to exemplify QoS through the rest of the paper. Based on our scenario, a set of new requirements is derived and the current limitations of WfMSs technology are stated. In section 2.4, we introduce our workflow QoS model and describe each of its dimensions. Section 2.5 describes how the quality of service of workflow tasks is calculated. In Section 2.6, we present an algorithm to compute and estimate workflow QoS, and we also describe how simulation techniques can be used for QoS estimation. Section 2.7 presents an example of how to compute the QoS for the workflow introduced in our initial scenario. Section 2.8 discusses the related work in the QoS area; section 2.9 presents future work on workflow QoS. Finally, section 2.10 presents our conclusions.

2.3 SCENARIO

The Fungal Genome Resource laboratory (FGR 2002) at the University of Georgia has realized that to be competitive and efficient it must adopt a new and modern information system infrastructure. Therefore, a first step was taken in that direction with the adoption of a workflow management system (METEOR (Kochut, Sheth *et al.* 1999)) to support its laboratory processes (Hall, Miller *et al.* 2000). Since the laboratory supplies several genome services to its customers, the adoption of a WfMS has enabled the logic of laboratory processes to be captured in a workflow schema. As a result, all the services available to customers are stored and executed under the supervision of the workflow system.

2.3.1 WORKFLOW STRUCTURE

Before discussing this scenario in detail, we review the basis elements of the METEOR workflow model.

A workflow is composed of tasks and transitions. Tasks are represented using a circle, networks (sub-workflows) using rounded rectangles, and transitions are represented using an arrow. Transitions express dependencies between tasks and are associated with an enabling probability (p_1, p_2,.., p_n). When a task has only one outgoing transition, the enabling probability is 1. In such a case, the probability can be omitted from the graph. A task with more than one outgoing transition can be classified as an *and-split* or *xor-split*. *And-split* tasks enable all their outgoing transitions after completing their execution. *Xor-split* tasks enable only one outgoing transition after completing their execution. *And-split* tasks are represented with a '*' and *xor-split* tasks are represented with a '+'. A task with more than one incoming transition can be classified as an *and-join* or *xor-join*. *And-join* tasks start their execution when all their incoming transitions are enabled. *Xor-join* tasks are executed as soon as one of the incoming transitions is enabled. As with *and-split* and *xor-split* tasks, *and-join* tasks and *xor-join* tasks are

represented with the symbol '*' and '+', respectively. When no symbol is present to indicate the input or output logic of a task, then it is assumed to be an *xor*.

2.3.2 WORKFLOW DESCRIPTION

Genomic projects involve highly specialized personnel and researchers, sophisticated equipment, and specialized computations involving large amounts of data. The characteristics of the human and technological resources involved, often geographically distributed, require a sophisticated coordination infrastructure to manage not only laboratory personnel and equipment, but also the flow of data generated.

One of the services supplied by the research laboratory is the DNA Sequencing workflow. A simplified version of the DNA Sequencing workflow is depicted in Figure 2-1. The complete description of the workflow can be found in (Cardoso 2002).

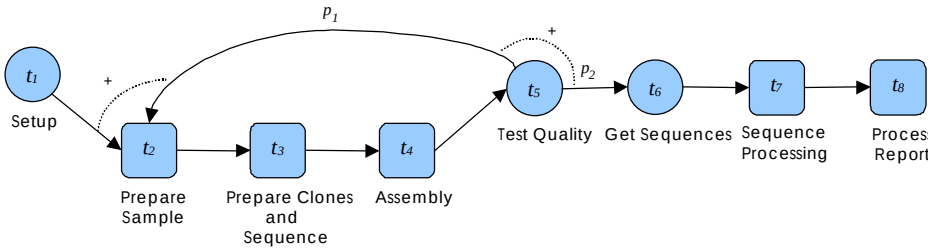

Figure 2-1 – DNA Sequencing workflow

The workflow is composed of eight main tasks: *Setup*, *Prepare Sample*, *Prepare Clone and Sequence*, *Assembly*, *Get Sequences*, *Sequence Processing*, and *Process Report*. Each individual task carries out a particular function; if necessary, the workflow can be spread across multiple research centers.

The *Setup* task is responsible for initializing internal variables of the workflow process.

The second task, *Prepare Sample*, consists of isolating DNA from a biological sample. The samples can be prepared using a variety of protocols. These protocols need

to be followed rigorously in order to obtain DNA that is not degraded in any form. A correctly prepared sample will originate a better DNA sequencing, since the quality of the DNA template is one of the most critical factors in DNA sequencing.

The task *Prepare Clones and Sequence* clones specific regions of the genome from DNA isolated in the previous step. This step can be fully automated by computer control (using, for example, a robotic system). This task also executes the sequencing, which uses DNA sequencing machines to read each biochemical "letter" (A, G, C or T) of a cloned DNA fragment. The output is composed of short decoded segments (a sequence such as AGGCATTCCAG…). The use of automated sequencers has revolutionized the field of bioinformatics by enabling scientists to catalogue sequence information hundreds of times faster than was possible with pre-existing scanning techniques. This new approach allows for automatic recognition, without major human intervention.

The *Assembly* task analyzes the DNA segments generated in the sequencing task. This step includes the assembly of larger contiguous blocks of sequences of DNA from small overlapping fragments. This is complicated by the fact that similar sequences occur many times in many places of the genome.

The *Test Quality* task screens for the *Escherichia coli* (*E. coli*) contaminant in DNA contigs. The clones grown in bacterial hosts are likely to be contaminated. A quick and effective way to screen for the *E. coli* contaminant is to compare a given DNA sequence to the *E. coli* genome. For *E. coli,* this task is made easier by the availability of its full genome.

Get Sequences is a simple task that downloads the sequences created in the assembly step, using the FTP protocol.

The *Sequence Processing* task analyzes the DNA segments generated in the assembly step. The goal of this task is to find DNA sequences in order to identify macromolecules with related structures and functions. The new DNA sequence is

compared to a repository of known sequences (*e.g.*, Swiss-Prot or GenBank), using one of a number of computational biology applications for comparison.

After obtaining the desired data from the *Sequence Processing* task, the results are stored, e-mailed, and a report is created. The *Process Report* task stores the data generated in the previous task in a database and creates a final report. It is responsible for electronically mailing the sequencing results to the persons involved in this process, such as researchers and lab technicians.

2.3.3 WORKFLOW APPLICATION REQUIREMENTS

In its normal operation, the Fungal Genome Resource laboratory executes the DNA Sequencing workflow in a regular manner. Workflow instances are started in order to render the sequencing services. In this scenario, and with current workflow technology, the execution of the workflow instances is carried out without any quality of service management on important parameters such as delivery deadlines, fidelity, quality, reliability, and cost of service. The laboratory wishes to be able to state a detailed list of requirements for the service to be rendered to its customers. Its requirements include the following:

- The final report has to be delivered in 31 weeks or less, as specified by the customer (*e.g.*, NIH).
- The profit margin has to be 10%. For example, if a customer pays $1,100 for a sequencing, then the execution of the DNA Sequencing workflow must have a cost for the laboratory that is less than $1,000.
- The error rate of the task *Prepare Clones and Sequence* has to be at most e, and the data quality of the task *Sequence Processing* has to be at least a.

- In some situations, the client may require an urgent execution of DNA sequencing. Therefore, the workflow has to exhibit high levels of reliability, since workflow failures would delay the sequencing process.

The requirements for the genetic workflow application presented underline four non-functional requirements: time, cost, fidelity, and reliability. While the specification of such quality requirements is important, current WfMSs do not include the functions to delineate their specification or management.

2.3.4 CURRENT WFMSS LIMITATIONS

The lack of a mechanism to specify workflow QoS is a current limitation of WfMSs. However, this is not the only missing element; once a workflow QoS model is defined, three additional components need to be developed: *estimation algorithms and methods*, *monitoring tools*, and *mechanisms to control* the quality of service. Only the development of integrated solutions composed of those four modules (specification, estimation, monitoring, and control) can result in a sophisticated quality management framework. The objectives and functionalities of each module include the following:

- A quality of service model must be developed to allow for the *specification* of workflow Quality of Service (QoS) metrics. This model allows suppliers to specify the duration, quality, cost, fidelity, *etc.*, of the services and products to be delivered. Specifications can be set at design-time, when designers build workflow applications, or they can be adjusted at run-time.
- Algorithms and methods must be developed to *estimate* the quality of service of a workflow both before instances are started and during instance execution. The estimation of QoS before instantiation allows suppliers to ensure that the workflow processes to be executed will indeed exhibit the quality of service

requested by customers. The analysis of workflow QoS during instance execution allows workflow systems to constantly compute QoS metrics and register any deviations from the initial requirements.

- Tools must be available to *monitor* the quality of service of running workflow instances. Workflow users and managers need to receive information about the QoS status and possible deviations from the desired metrics that might occur. In our scenario, let us assume that for some unknown reason the *matching factor* of the DNA Sequencing data drops below a threshold expressed by the customer. The *matching factor* reflects the degree of similarity between the query sequence ("probe") and the compared ("subject") sequence stored in a sequence database. The use of workflow QoS monitoring tools can automatically detect this variation in fidelity and automatically notify interested users.
- Mechanisms must be available which *control* the quality of service of workflow instances. Control is necessary when instances do not behave according to initial requirements. Let us consider the following example: workflow instances are running correctly and the quality of service specifications are being followed when a task fails. The task *Prepare Clone and Sequence* stops its processing because one of the associated machines has a mechanical problem. As a consequence, workflow QoS specifications of time are no longer satisfied, and the WfMS raises a warning, an alert, or an exception. The faulty task needs to be replaced by an equivalent task to restore the soundness of the system. This replacement can be accomplished by applying dynamic changes to the workflow instances, either manually or automatically (Cardoso, Luo *et al.* 2001).

While these four areas of research are important and indispensable for adequate quality of service management, in this paper we focus on the specification, estimation, and monitoring of workflow QoS.

2.4 WORKFLOW QUALITY OF SERVICE

As stated earlier, the quality of service is an important issue for workflow systems. The international quality standard ISO 8402 (part of the ISO 9000 (ISO9000 2002)) describes quality as *"the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs."* This definition implies a relation between the characteristics of products or services rendered and the initial requirements or implied needs. In our opinion, this definition of quality, which includes an important relationship between requirements and characteristics, is relevant and applicable to the domain of WfMSs. For us, workflow QoS represents *the quantitative and qualitative characteristics of a workflow application necessary to achieve a set of initial requirements.*

Workflow QoS addresses the non-functional issues of workflows rather than workflow process operations. Quantitative characteristics can be evaluated in terms of concrete measures such as workflow execution time, cost, *etc*. Kobielus (1997) suggests that dimensions such as time, cost, and quality should constitute the criteria that workflow systems should include and might benefit from. Qualitative characteristics specify the expected services offered by the system, such as security and fault-tolerance mechanisms. QoS should be seen as an integral aspect of workflows; therefore, it should be integrated with workflow specifications. The first step is to define a workflow QoS model.

2.4.1 WORKFLOW QOS MODEL

Quality of service can be characterized according to various dimensions. We have investigated related work to decide which dimensions would be relevant to compose our QoS model. Our research targeted two distinct areas: operations management for organizations and quality of service for software systems. The study of those two areas is important, since workflow systems are widely used to model organizational business processes, and workflow systems are themselves software systems.

On the organizational side, Stalk and Hout (1990) and Rommel *et al*. (1995) investigated the features with which successful companies assert themselves in competitive world markets. Their results indicated that success is related to the capability to compete with other organizations, and it is based upon three essential pillars: *time*, *cost*, and *quality*. These three dimensions have been a major concern for organizations. Garvin (1988) associates eight dimensions with quality, including performance and reliability. Software systems' quality of service has also been extensively studied. Major contributions can be found in the areas of networking (Cruz 1995; Georgiadis, Guerin *et al.* 1996), real-time applications (Clark, Shenker *et al.* 1992) and middleware (Zinky, Bakken *et al.* 1997; Hiltunen, Schlichting *et al.* 2000). For middleware systems, Frlund and Koistinen (1998) present a set of practical dimensions for distributed object systems' reliability and performance, which include TTR (time to repair), TTF (time to failure), availability, failure masking, and server failure. For data networks, the QoS generally focus on domain-specific dimensions such as bandwidth, latency, jitter, and loss (Nahrstedt and Smith 1996).

Our past work on deploying workflow applications has made us aware of the need for workflow process QoS management. Additionally, we have realized that workflow processes have a particular set of requirements which are domain dependent and that need to be accounted for when creating a QoS model. Based on previous studies and our experience in the workflow domain, we have constructed a QoS model composed of the following dimensions: *time*, *cost*, *reliability*, and *fidelity*. According to Weikum (1999), information services QoS can be divided into three categories: system centric, process centric, and information centric. Our model specifies quality dimensions that include the system and process categories. QoS specifications are set for task definitions. Based on this information, QoS metrics are computed for workflows (see section 2.6).

Other researchers have also identified the need for a QoS process model. A good example is the DAML-S specification (Ankolekar, Burstein *et al.* 2001; DAML-S 2001),

which semantically describes business processes (as in the composition of Web services). The use of semantic information facilitates process interoperability between trading partners involved in e-commerce activities. This specification includes constructs which specify quality of service parameters, such as quality guarantees, quality rating, and degree of quality. While DAML-S has identified the importance of Web services and business processes specifications, the QoS model adopted should be significantly improved in order to supply a more functional solution for its users. One current limitation of DAML-S' QoS model is that it does not provide a detailed set of classes and properties to represent quality of service metrics. The QoS model needs to be extended to allow for a precise characterization of each dimension. The addition of semantic concepts, such as minimum, average, maximum, and the distribution function associated with a dimension, will allow the implementation of algorithms for the automatic computation of QoS metrics for processes based on atomic tasks and sub-processes' QoS metrics.

2.4.2 TASK TIME

Time is a common and universal measure of performance. For workflow systems, it can be defined as the total time needed by an instance to transform a set of inputs into outputs. The philosophy behind a time-based strategy usually demands that businesses deliver the most value as rapidly as possible. Shorter workflow execution time allows for a faster production of new products, thus providing a competitive advantage, since the products are more rapidly introduced into the market. Additionally, reducing the time taken to execute a set of tasks in a workflow process makes it possible for an organization to be more responsive to customers' needs. Therefore, it is important to enhance WfMS to include time-based process execution.

The first measure of time is **task response time** (T). Task response time corresponds to the time an instance takes to be processed by a task. The task response time can be

broken down into two major components: delay time and process time. **Delay time** (DT) refers to the non-value-added time needed in order for an instance to be processed by a task. This includes, for example, the instance queuing delay and the setup time of the task. While, those two metrics are part of the task operation, they do not add any value to it. **Process time** (PT) is the time a workflow instance takes at a task while being processed; in other words, it corresponds to the time a task needs to process an instance. Therefore, *task response time* for a task *t* can be computed as follows:

$$T(t) = DT(t) + PT(t)$$

The delay time can be further broken down into queuing delay and setup delay. *Queuing delay* is the time instances spend waiting in a tasklist, before the instance is selected for processing. *Setup delay* is the time an instance spends waiting for the task to be set up. Setup activities may correspond to the warming process carried out by a machine before executing any operation, or to the execution of self-checking procedures. Another time metric that may be considered to integrate with the delay time is the *synchronization delay*, which corresponds to the time a workflow instance waits for mates in an *and-join* task (synchronization). In our QoS model, this metric is not part of the task response time. This is because the algorithm we use to estimate workflow QoS can derive this metric directly from the workflow structure and from the task response time. This will become clearer when we describe workflow QoS computation.

2.4.3 TASK COST

Task cost represents the cost associated with the execution of workflow tasks. Cost is an important factor, since organizations need to operate according to their financial plan. It is fundamental for organizations that wish to reduce their expenditures on internal processes and wish to control product and service cost. During workflow design, both prior to workflow instantiation and during workflow execution, it is necessary to estimate

the cost of the execution in order to guarantee that financial plans are followed. The cost of executing a single task includes the cost of using equipment, the cost of human involvement, and any supplies and commodities needed to complete the task. The following cost functions are used to compute the cost associated with the execution of a task.

Task cost (C) is the cost incurred when a task *t* is executed; it can be broken down into two major components: enactment cost and realization cost.

$$C(t) = EC(t) + RC(t)$$

The **enactment cost** (EC) is the cost associated with the management of the workflow system and with workflow instances monitoring. The **realization cost** (RC) is the cost associated with the runtime execution of the task. It can be broken down into: *direct labor cost, machine cost, direct material cost,* and *setup cost*. *Direct labor cost* is the cost associated with the person carrying out the execution of a workflow human task (Kochut, Sheth *et al.* 1999), or the cost associated with the execution of an automatic task with partial human involvement. *Machine cost* is the cost associated with the execution of an automatic task. This can correspond to the cost of running a particular piece of software or the cost of operating a machine. *Direct material cost* is the cost of the materials, resources, and inventory used during the execution of a workflow task. *Setup cost* is the cost to set up any resource used prior to the execution of a workflow task.

2.4.4 TASK RELIABILITY

In an early work on workflow modeling, Krishnakumar and Sheth (1995) represented the execution behavior of each task, using task structures. Each workflow task structure has an initial state, an execution state, and two distinct terminating states. One of the states indicates that a task has failed (for non-transactional tasks) or was aborted (for transactional and open 2PC tasks), while the other state indicates that a task is done or

committed (Figure 2-2). The model used to represent each task indicates that only one starting point exists when performing a task, but two different states can be reached upon its execution. Based on this task model structure, we introduce the *reliability* dimension. This QoS dimension provides information concerning the relationship between the number of times the state done/committed is reached and the number of times the failed/aborted state is reached after the execution of a task.

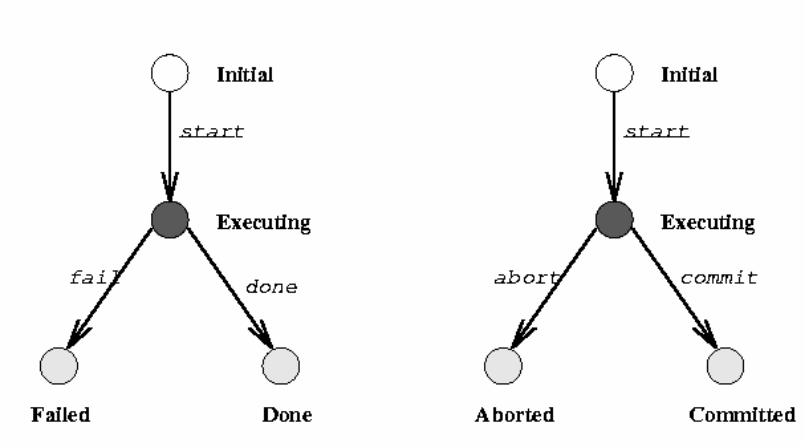

Figure 2-2 - Two task structures (Krishnakumar and Sheth 1995)

Task Reliability (R) corresponds to the likelihood that the components will perform according to; it is a function of the failure rate. To describe task reliability we follow a discrete-time modeling approach. We have selected this solution since workflow task behavior is most of the time characterized in respect to the number of executions. Discrete-time models are adequate for systems that respond to occasional demands, such as database systems (*i.e.*, discrete-time domain). This dimension (Table 2-1) follows from one of the popular discrete-time stable reliability models proposed in (Nelson 1973), where failure rate is given as the ratio of *successful executions/scheduled executions*.

Table 2-1 – Task reliability

R(*t*) = 1 – failure rate

For each task, the WfMS keeps track of the number of times the task has been scheduled for execution and how many times the task has been successfully executed. R(*t*) is a stable model, since when software failure occurs no fault removal is performed.

Alternatively, continuous-time reliability models can be used when the failures of the malfunctioning equipment or software can be expressed in terms of times between failures, or in terms of the number of failures that occurred in a given time interval. Such reliability models are more suitable when workflows include tasks that control equipment or machines that have failure specifications determined by the manufacturer. Goel (1985) classified reliability models into four kinds: input domain-based models, times-between-failures models, failure-count models, and fault seeding models. Ireson, Jr *et al.* (1996) presents several software reliability models which can be used to model this QoS dimension. The ideal situation would be to associate with each workflow task a reliability model representing its working behavior. While this is possible, we believe that the common workflow system users do not have enough knowledge and expertise to apply such models.

2.4.5 TASK FIDELITY

We view fidelity as a function of effective design; it refers to an intrinsic property(ies) or characteristic(s) of a good produced or service rendered. Fidelity reflects how well a product is being produced and how well a service is being rendered. Fidelity is often difficult to define and measure because it is subject to judgments and perceptions. Nevertheless, the fidelity of workflows should be predicted whenever feasible and carefully controlled when needed (Kolarik 1995; Franceschini 2002).

Workflow tasks have a fidelity (F) vector dimension composed of a set of fidelity attributes ($F(t).a_r$), that reflect and quantify task operations. Each fidelity attribute refers to a property or characteristic of the product being created, transformed, or analyzed. Fidelity attributes are used by the workflow system to compute how well workflows, instances, and tasks are meeting user specifications. For example, the *Test Quality* task check the fidelity of the attribute $F(t).a_{E.\ coli\ matching}$. This attribute reflects the probability that the sample being sequenced is contaminated. Each task is associated with a fidelity function $F(t)$, which represents the local normalized fidelity:

$$F(t) = |f_1(F(t).a_i)|\, wi_1 + |\, f_2(F(t).a_j)|\, wi_2 + |\, f_3(F(t).a_k)|\, wi_3 + \ldots + |\, f_n(F(t).a_l)|\, wi_n$$

The formula weights the fidelity attributes, which can be transformed to more appropriate values using a function f_n, and are normalized to the scale [0..1]. The sum of the weights wi_k is equal to 1. In view of the fact humans often feel awkward in handling and interpreting such quantitative values (Tversky and Kahneman 1974), we allow the designer with the help of a domain expert to map the value resulting from applying the fidelity function to a qualitative scale (Miles and Huberman 1994). This qualitative indicator is used to detect areas of a workflow with anomalies and undesired behavior. An example of a mapping scale for quantitative and qualitative values is shown in Table 2-2. The workflow designer is responsible for the creation of the mapping table. The table is created by first selecting a set of qualitative terms that characterize the fidelity. The use of qualitative terms may facilitate the human understanding of the fidelity concept exhibited by workflows in some cases.

Table 2-2 – Example of a fidelity-mapping table

Qualitative Fidelity	Quantitative Fidelity
Unacceptable	[0.00.. 0.20]
Poor	[0.21.. 0.40]
Satisfactory	[0.41.. 0.60]
Good	[0.61.. 0.80]
Perfect	[0.81.. 1.00]

Depending on the task type, a task uses different strategies to set fidelity attributes. Three scenarios can be drawn: automatic tasks controlling hardware, automatic tasks controlling software, and human tasks. For an automated task controlling a hardware device, the fidelity attribute can be set after reading the output status line of the device. For example, the task *Sequencing* controls DNA sequencing, which is carried out automatically by a sequencer. When the sequencing finishes, the machine generates several output files to describe how the process was executed. These values can be passed on to the task, which automatically updates its fidelity attributes. For automated tasks controlling a software application, the same procedure can be applied. For example, the task *Sequence Processing* executes various algorithms on the sequences received. One of the algorithms used is BLAST (Altschul, Gish *et al.* 1990). This algorithm searches DNA sequences in a database to identify macromolecules with related structures and functions. Once the search is concluded, the algorithm returns a value indicating the confidence of the matching. For this task, the returned value from the execution of the algorithm will be used to describe the fidelity of the task's execution. For human tasks, the procedure has to be manual. Therefore, it is the responsibility of the user to manually input information

relative to the fidelity of the task executed. In the case of the task *Prepare Sample,* the lab technician sets the fidelity attribute quality of clones manually, after a visual identification. For quality assurance reasons the attributes should be set or checked by a person other than the one who that carried out the task execution. If evaluating the fidelity of a task cannot be accurately done by a human, an option is to place – when possible – an automatic task after the human task to automatically check the fidelity.

The fidelity information can be used to effectively monitor workflow executions. Typically, during the lifetime of an instance, qualitative information describing task fidelity is displayed on graphical monitors as the tasks are executed. Managers can easily identify tasks which exhibit unsatisfactory fidelity metrics.

2.4.6 QoS Model Discussion

One of the most popular workflow classifications distinguishes between *ad hoc* workflows, administrative workflows, and production workflows. This classification was first mentioned by (McCready 1992). The main differences between these types include structure, repetitiveness, predictability, complexity, and degree of automation.

We recognize that the QoS model presented here is better suited for production workflows (McCready 1992) since they are more structured, predictable, and repetitive. Production workflows involve complex and highly-structured processes, whose execution requires a high number of transaction accessing different information systems. These characteristics allow the construction of adequate QoS models for workflow tasks. In the case of *ad hoc* workflows, the information, the behavior, and the timing of tasks are largely unstructured, which makes the procedure of constructing a good QoS model more difficult and complex.

2.5 CREATION OF QOS ESTIMATES

In order to facilitate the analysis of workflow QoS, it is necessary to initialize task QoS metrics and also initialize stochastic information which indicates the probability of transitions being fired at runtime. Once tasks and transitions have their estimates set, algorithms and mechanisms, such as simulation, can be applied to compute overall workflow QoS.

2.5.1 QOS ESTIMATES FOR TASKS

Having previously defined the QoS dimensions for tasks, we now target the estimation of QoS metrics of tasks. The specification of QoS metrics for tasks is made at design time and re-computed at runtime, when tasks are executed. During the graphical construction of a workflow process, the designer sets QoS estimates for each task. The estimates characterize the quality of service that the tasks will exhibit at runtime.

Setting initial QoS metrics for some workflow tasks may be relatively simple. For example, setting the QoS for a task controlling a DNA sequencer can be done based on the time, cost, and reliability specifications given by the manufacturer of the DNA sequencer. In other cases, setting initial QoS metrics may prove to be difficult. This is the case for tasks that heavily depend on user input and system environment. For such tasks, it is convenient to study the workflow task based on real operations. The estimates are based on data collected while testing the task. The idea is to test the task based on specific inputs. This can be achieved by the elaboration of an operational profile (Musa 1993). In an operational profile, the input space is partitioned into domains, and each input is associated with a probability of being selected during operational use. The probability is employed in the input domain to guide input generation. The density function built from the probabilities is called the operational profile of the task. At runtime, tasks have a probability associated with each input. Musa (1999) described a detailed procedure for developing a practical operational profile for testing purposes.

The task runtime behavior specification is composed of two classes of information (Table 2-3): basic and distributional. The basic class associates with each task's QoS dimension the minimum value, average value, and maximum value the dimension can take. For example, the cost dimension corresponds to the minimum, average, and maximum cost associated with the execution of a task. The second class, the distributional class, corresponds to the specification of a constant or of a distribution function (such as Exponential, Lognormal, Normal, Rayleigh, Time-Independent, Weibull, and Uniform) which statistically describes task behavior at runtime. For example, Table 2-3 and Table 2-4 show the QoS dimensions for an automatic task (the *SP FASTA* task) and for a manual task (the *Prepare Sample* task; see section 2.3.2 for tasks descriptions).

Table 2-3 – Task QoS for an automatic task

	Basic class			**Distributional class**
	Min value	Avg value	Max value	Dist. Function
Time	0.291	0.674	0.895	Normal(0.674, 0.143)
Cost	0	0	0	0.0
Reliability	-	100%	-	1.0
Fidelity.a_i	0.63	0.81	0.92	Trapezoidal(0.7,1,1,4)

Table 2-4 – Task QoS for a manual task

	Basic class			Distributional class
	Min value	Avg value	Max value	Dist. Function
Time	192	196	199	Normal(196, 1)
Cost	576	576	576	576.0
Reliability	-	100%	-	1.0
Fidelity.a_i	-	-	-	-

The values specified in the basic class are typically employed by mathematical methods in order to compute workflow QoS metrics, while the distributional class information is used by simulation systems to compute workflow QoS. To devise values for the two classes, the designer typically applies the functions presented in the previous section to derive the task's QoS metrics. We recognize that the specification of time, cost, fidelity, and reliability is a complex operation, which when not carried out properly can lead to the specification of incorrect values. Additionally, the initial specification may not remain valid over time. To overcome this difficulty, a task's QoS values can be periodically re-computed for the basic class, based on previous executions. The distributional class may also need to have its distribution re-computed. At runtime, the workflow system keeps track of actual values for the QoS dimensions monitored. QoS runtime metrics are saved and used to re-compute the QoS values for the basic class which were specified at design time. The workflow system re-computes the QoS values for each dimension; this allows the system to make more accurate estimations based on recent instance executions.

The re-computation of QoS task metrics is based on data coming from designer specifications and from the workflow system log. Four scenarios can occur: a) For a

specific task *t* and a particular dimension *Dim*, the average is calculated based only on information introduced by the designer (designer average); b) the average of a task *t* dimension is calculated based on all its executions independently of the workflow that executed it (multi-workflow average); c) the average of the dimension *Dim* is calculated based on all the times task *t* was executed in any instance from workflow *w* (workflow average); and d) the average of the dimension of all the times task *t* was executed in instance *i* of workflow *w* (instance average). Scenario d) can only occur when loops exist in a workflow.

The averages described in Table 2-5 are computed at runtime and made available to the workflow system. While Table 2-5 shows only how to compute average metrics, similar formulae can be used to compute minimum and maximum values.

Table 2-5 – Designer, multi-workflow, workflow and instance average

Designer $Average_{Dim}(t)$	Average specified by the designer in the basic class for dimension Dim
Multi-Workflow $Average_{Dim}(t)$	Average of the dimension Dim for task *t* executed in the context of any workflow
Workflow $Average_{Dim}(t, w)$	Average of the dimension Dim for task *t* executed in the context of any instance of workflow *w*
Instance $Average_{Dim}(t, w, i)$	Average of the dimension Dim for task *t* executed in the context of instance *i* of workflow *w*

The task QoS for a particular dimension can be determined at different levels; it is computed following the equations described in Table 2-6.

Table 2-6 – QoS dimensions computed at runtime

a)	$QoS_{Dim}(t)$	Designer $Average_{Dim}(t)$
b)	$QoS_{Dim}(t)$	wi_1* Designer $Average_{Dim}(t)$ + wi_2* Multi-Workflow $Average_{Dim}(t)$
c)	$QoS_{Dim}(t, w)$	wi_1* Designer $Average_{Dim}(t)$ + wi_2* Multi-Workflow $Average_{Dim}(t)$ + wi_3*Workflow $Average_{Dim}(t, w)$
d)	$QoS_{Dim}(t, w, i)$	wi_1* Designer $Average_{Dim}(t)$ + wi_2* Multi-Workflow $Average_{Dim}(t)$ + wi_3* Workflow $Average_{Dim}(t, w)$ + wi_4* Instance Workflow $Average_{Dim}(t,w, i)$

The workflow system uses the formulae from Table 2-6 to predict the QoS of tasks. The weights wi_j are set manually. They reflect the degree of correlation between the workflow under analysis and other workflows for which a set of common tasks is shared. Since the values entered by the designer may contain extraneous data and therefore be imprecise, a Bayesian approach (Bernardo and Smith 1994) might be considered to make use of prior knowledge in order to improve the accuracy of the weights wi_j.

Let us assume that we have an instance i of workflow w running and that we desire to predict the QoS of task $t \in w$. The following rules are used to choose which formula to apply when predicting QoS. If task t has never been executed before, then formula a) is chosen to predict task QoS, since there is no other data available. If task t has been executed previously, but in the context of workflow w_n, and w != w_n, then formula b) is chosen. In this case we can assume that the execution of t in workflow w_n will give a

good indication of its behavior in workflow *w*. If task *t* has been previously executed in the context of workflow *w*, but not from instance *i*, then formula c) is chosen. Finally, if task *t* has been previously executed in the context of workflow *w*, and instance *i*, meaning that a loop has been executed, then formula d) is used.

2.5.2 PROBABILITIES ESTIMATES FOR TRANSITIONS

In the same way we seed tasks' QoS, we also need to seed workflow transitions. Initially, the designer sets the transition probabilities at design time. At runtime, the transitions' probabilities are re-computed. The method used to re-compute the transitions' probabilities follows the same lines of the method used to re-compute tasks' QoS. When a workflow has never been executed, the values for the transitions are obviously taken from initial designer specifications. When instances of a workflow *w* have already been executed, then the data used to re-compute the probabilities come from initial designer specifications for workflow *w*, from other executed instances of workflow *w*, and if available, from the instance of workflow *w* for which we wish to predict the QoS. This corresponds to the use of functions similar to the ones previously defined for tasks' QoS (see Table 2-6).

2.6 QOS COMPUTATION

Once QoS estimates for tasks and for transitions are determined, we can compute overall workflow QoS. We describe two modeling techniques that can be used to compute QoS metrics for a given workflow process: mathematical modeling and simulation modeling. The selection of the method is based on a tradeoff between time and the accuracy of results. The mathematical method is computationally faster, but it yields results which may not be as accurate as the results obtained by simulation. (Note that our mathematical models could be extended to queuing network models (Lazowska, Zhorjan *et al.* 1984), but this requires making some simplifying assumptions).

2.6.1 MATHEMATICAL MODELING

The stochastic workflow reduction method consists of applying a set of reduction rules to a workflow until only one atomic task (Kochut, Sheth *et al.* 1999) exists. Each time a reduction rule is applied, the workflow structure changes. After several iterations only one task will remain. When this state is reached, the remaining task contains the QoS metrics corresponding to the workflow under analysis.

The set of reduction rules that can be applied to a given workflow corresponds to the set of inverse operations that can be used to construct a workflow. We have decided to only allow the construction of workflows which are based on a set of predefined construction systems; this protects users from designing invalid workflows. Invalid workflows contain design errors, such as non-termination, deadlocks, and split of instances (Aalst 1999). While in this paper we do not prove that a workflow graph can be reduced by using the proposed set of reduction systems, this can be accomplished, proving that all the reduction systems form a "finite Church-Rosser" transformation. Work on graph reduction can be found in Allen (1970) and Knuth (1971).

To compute QoS metrics, we have developed the *SWR*(*w*) algorithm (Cardoso 2002), which uses a set of six distinct reduction rules: (1) sequential, (2) parallel, (3) conditional, (4) fault-tolerant, (5) loop, and (6) network.

Additional reduction rules can be developed. We have decided to present the reduction concept with only six reduction rules, for two reasons. The first reason is because a vast majority of workflow systems support the implementation of the reduction rules presented. Based on a study on fifteen major workflow systems and the workflow patterns that they support (Aalst, Barros *et al.* 2002), fifteen of the workflow systems studied supported the reduction rules (1)(2)(3), ten supported the reduction rule (5), and eight supported the reduction rules (4). The study does not discuss network patterns. The network pattern is intended to provide a structural and hierarchical division of a given

workflow design into levels, in order to facilitate its understanding by the grouping of related tasks into functional units. The second reason is that the reduction rules are simple, making it easy to understand the idea behind the workflow reduction process.

2.6.1.1 REDUCTION SYSTEMS

Reduction of a Sequential System. Figure 2-3 illustrates how two sequential workflow tasks t_i and t_j can be reduced to a single task t_{ij}. In this reduction, the incoming transitions of t_i and outgoing transition of tasks t_j are transferred to task t_{ij}.

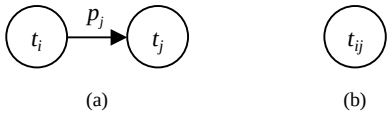

Figure 2-3 - Sequential system reduction

In a sequential system, $p_j = 1$. This reduction can only be applied if the following two conditions are satisfied: a) t_i is not a *xor/and* split and b) t_j is not a *xor*/*and* join. These conditions prevent this reduction from being applied to parallel, conditional, and loop systems. To compute the QoS of the reduction, the following formulae are applied:

$$T(t_{ij}) = T(t_i) + T(t_j)$$

$$C(t_{ij}) = C(t_i) + C(t_j)$$

$$R(t_{ij}) = R(t_i) * R(t_j)$$

$$F(t_{ij}).a_r = f(F(t_i), F(t_j))$$

Reduction of a Parallel System. Figure 2-4 illustrates how a system of parallel tasks t_1, t_2, …, t_n, an *and* split task t_a, and an *and* join task t_b can be reduced to a sequence of three

tasks t_a, t_{1n}, and t_b. In this reduction, the incoming transitions of t_a and the outgoing transition of tasks t_b remain the same. The only outgoing transitions from task t_a and the only incoming transitions from task t_b are the ones shown in the figure below. The probabilities of p_{a1}, p_{a2},…, p_{1n} and p_{1b}, p_{2b},…, p_{nb} are equal to 1.

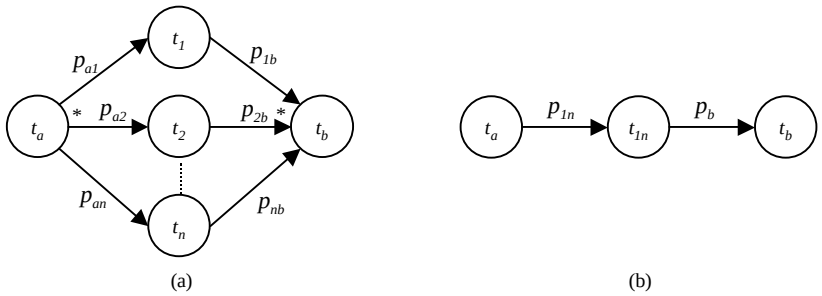

Figure 2-4 - Parallel system reduction

The QoS of tasks t_a and t_b remain unchanged, and $p_{1n} = p_b = 1$. To compute the QoS of the reduction the following formulae are applied:

$$T(t_{1n}) = \text{Max}_{i \in \{1..n\}} \{T(t_i)\}$$

$$C(t_{1n}) = \sum_{1 \le i \le n} C(t_i)$$

$$R(t_{1n}) = \prod_{1 \le i \le n} R(t_i)$$

$$F(t_{1n}).a_r = f(F(t_1), F(t_2), \ldots, F(t_n))$$

Reduction of a Conditional System. Figure 2-5 illustrates how a system of conditional tasks t_1, t_2, …, t_n, a *xor* split (task t_a), and a *xor* join (task t_b) can be reduced to a sequence of three tasks t_a, t_{1n}, and t_b. Task t_a and task t_b do not have any other outgoing transitions and incoming transitions, respectively, other than the ones shown in the figure. In this

reduction the incoming transitions of t_a and outgoing transition of tasks t_b remain the same, and $\sum_{i=1}^{n} p_{ai} = 1$.

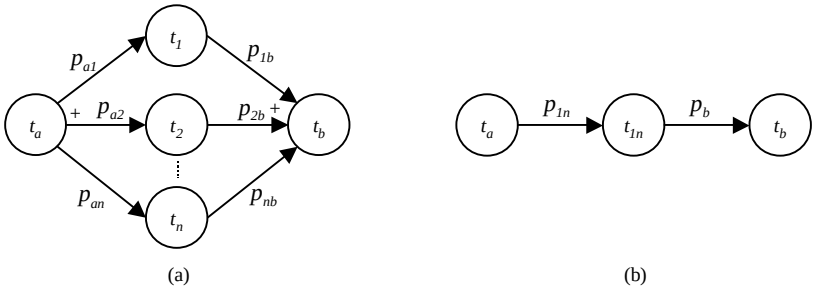

Figure 2-5 - Conditional system reduction

The QoS of tasks t_a and t_b remain unchanged, and $p_{1n} = p_b = 1$. To compute the QoS of the reduction the following formulae are applied:

$$T(t_{1n}) = \sum_{1 \le i \le n} p_{ai} * T(t_i)$$

$$C(t_{1n}) = \sum_{1 \le i \le n} p_{ai} * C(t_i)$$

$$R(t_{1n}) = \sum_{1 \le i \le n} p_{ai} * R(t_i)$$

$$F(t_{1n}).a_r = f(p_{a1}, F(t_1), p_{a2}, F(t_2), \ldots, p_{an}, F(t_n))$$

Reduction of a Loop System. Loop systems can be characterized by simple and dual loop systems. Figure 2-6 illustrates how a simple loop system can be reduced. A simple loop system in task t_i can be reduced to a task t_{li}. In this reduction, $p_i + \sum_{i=1}^{n} p_{oi} = 1$.

Once the reduction is applied, the probabilities of the outgoing transitions of task t_{li} are changed to $p_{lk} = \frac{p_{ok}}{1 - p_i}$, and $\sum_{k=1}^{n} p_{lk} = 1$.

Figure 2-6 – Simple loop system reduction

To compute the QoS of the reduction the following formulae are applied:

$$T(t_{li}) = \frac{T(t_i)}{1 - p_i}$$

$$C(t_{li}) = \frac{C(t_i)}{1 - p_i}$$

$$R(t_{li}) = \frac{(1 - p_i) * R(t_i)}{1 - p_i R(t_i)}$$

$$F(t_{li}).a_r = f(p_i, F(t_i))$$

Figure 2-7 illustrates how a dual loop system can be reduced. A dual loop system composed of two tasks t_i and t_j can be reduced to a single task t_{ij}. In this reduction, $p_i + \sum_{i=1}^{n} p_{oi} = 1$. Once the reduction is applied, the probabilities of the outgoing transitions of task t_{ij} are changed to $p_{lk} = \frac{p_{ok}}{1 - p_i}$, and $\sum_{k=1}^{n} p_{lk} = 1$.

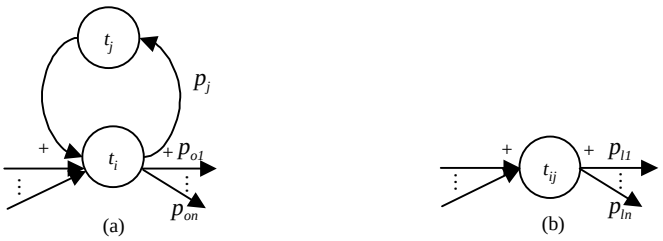

Figure 2-7 – Dual loop system reduction

To compute the QoS of the reduction the following formulae are applied:

$$T(t_{ij}) = \frac{T(t_i) + T(t_j) - (1 - p_j)T(t_j)}{(1 - p_j)}$$

$$C(t_{ij}) = \frac{C(t_i) + C(t_j) - (1 - p_j)C(t_j)}{(1 - p_j)}$$

$$R(t_{ij}) = \frac{(1 - p_j) * R(t_i)}{1 - p_j R(t_i) R(t_j)}$$

$$F(t_{ij}).a_r = f(F(t_i), p_j, F(t_j))$$

Reduction of a Fault-Tolerant System. Figure 2-8 illustrates how a fault-tolerant system with tasks t_1, t_2, …, t_n, an *and* split (task t_a), and a *xor* join (task t_b) can be reduced to a sequence of three tasks t_a, t_{1n}, and t_b. The execution of a fault-tolerant system starts with the execution of task t_a and ends with the completion of task t_b. Task t_b will be executed only if k tasks from the set $\{t_1, t_2, \ldots, t_n\}$ are executed successfully. In this reduction, the incoming transitions of t_a and the outgoing transition of tasks t_b remain the same, and $\forall i \in \{1..n\}, p_{ai} = 1, p_{ib} = 1$.

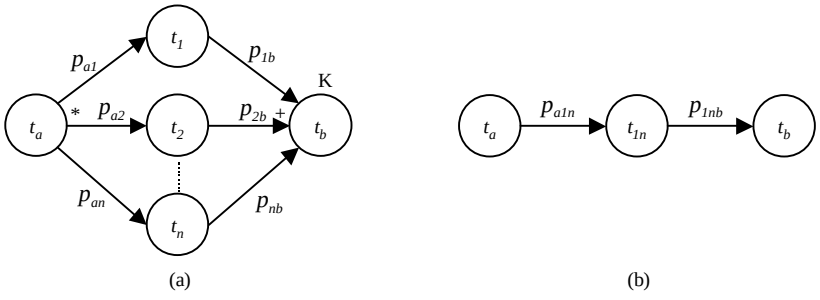

Figure 2-8 – Fault-Tolerant system reduction

The QoS of tasks t_a and t_b remain unchanged, and $p_{a1n} = p_{1nb} = 1$. To compute the QoS of the reduction the following formulae are applied:

$$T(t_{1n}) = \underset{k}{Min}(\{T(t_1),...,T(t_n)\})$$

$$C(t_{1n}) = \sum_{1 \le i \le n} C(t_I)$$

$$R(t_{1n}) = \sum_{i_1=0}^{1} \ldots \sum_{i_n=0}^{1} f(\sum_{j=1}^{n} i_j - k) * ((1-i_1) + (2i_1 - 1)R(t_1)) * ... * ((1-i_n) + (2i_n - 1)R(t_n))$$

$$F(t_{1n}).a_r = f(p_{a1}, F(t_1), p_{a2}, F(t_2), \ldots, p_{an}, F(t_n), k)$$

The function $\underset{k}{Min}(s)$ selects the k minimum value from set s, and function $f(x)$ is defined as followed:

$$f(x) = \begin{cases} 0, x < 0 \\ 1, x \ge 0 \end{cases}$$

The formula $R(t_{1n})$ is utilized to compute reliability and corresponds to the sum of all the probabilistic states for which more than k tasks execute successfully. The summation over i_1, …, i_n corresponds to the generation of a binary sequence for which 0 represents the failing of a task, and 1 represents its success. For example, in a fault-tolerant system with three parallel tasks (n=3), the values of the indexes i_1=1, i_2=0, and i_3=1 represent the probabilistic state for which tasks t_1 and t_3 succeed and task t_2 fails. The term $f(\sum_{j=1}^{n} i_j - k)$ is used to indicate if a probabilistic state should be considered in the reliability computation. A probabilistic state is considered only if the number of tasks succeeding is greater or equal to k, *i.e.*, $\sum_{j=1}^{n} i_j \ge k$ (or equivalently $\sum_{j=1}^{n} i_j - k \ge 0$). In our previous example, since i_1=1, i_2=0, i_3=1 and $\sum_{j=1}^{n} i_j = 2$, the probabilistic state (i_1=1, i_2=0, i_3=1) will be only considered if $k \le 2$.

Reduction of a Network System. A network task represents a sub-workflow (Figure 2-9). It can be viewed as a black box encapsulating an unknown workflow realization with a certain QoS. A network task n_s, having only one task t_i, can be replaced by an atomic task t_j. This reduction can be applied only when the QoS of task t_i is known. In this replacement, the QoS of the atomic task t_j is set to the workflow QoS of the task t_i, *i.e.*, $X(t_j) = X(t_i)$, $X \in \{T, C, R, F\}$.

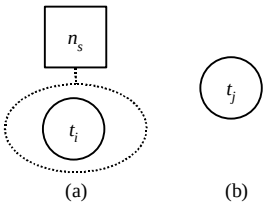

Figure 2-9 - Network system reduction

The input and output transitions of the network task n_s are transferred to the atomic task t_j.

2.6.1.2 TIME, COST, RELIABILITY, AND FIDELITY COMPUTATIONS

Time and Cost. The operations used to compute the time and cost dimensions are fairly intuitive.

Reliability. For the reliability dimension we have used concepts from system and software reliability theory (Hoyland and Rausand 1994; Ireson, Jr. *et al.* 1996; Musa 1999). The reliability functions used when applying workflow reduction systems assume that tasks behave independently. While this assumption is widely employed when modeling hardware systems, it is considered by some to be inappropriate for software systems since they tend to violate the independence supposition of the individual software systems.

Mason and Woit (1998) show that an application's structure has an influence on the dependability derived from the reliability of its components. Their work presents a theory

based on a set of rules which when applied to the construction of an application can result in systems which do not violate the underlying assumptions of the typical reliability models, *i.e.*, system independence. In order to understand the dependence of software components it is necessary to understand the difference between the terms “uses” and “invokes” (Parnas 1974; Parnas 2001). The utilization of “use” methodology creates a dependency between modules or procedures. This is because if a module A calls a module B, then the state of A depends on the results of B. Using the “invokes” methodology this problem does not arise, since when module A calls module B, module A does not wait or depend on B’s execution results. Based on this observation, Mason and Woit (1998) state that to reduce the dependence of modules in a system or application a, “uses” methodology should not be present to interconnect the components; instead, a “invokes” methodology should be present. Additionally, the module’s implementation details cannot affect the correctness of other modules in the system (state independence).

The architecture of workflow systems directly follows the two points that allow for a reduction of task dependencies. Workflow systems such as ORBWork (Kochut, Sheth *et al.* 1999) use a message-passing architecture and thus exhibit “invokes” characteristics. Additionally, tasks are independent from the implementation point of view, and therefore they are state independent. Due to the architecture of typical WfMSs, workflow applications have a reduced dependency factor among tasks; we make the assumption that the dependencies can be ignored in most of cases. Nevertheless, if tasks exhibit strong dependencies due to the data transferred, a profiling approach may need to be considered. Hamlet *et al.* (2001) proposed the use of operational profiles that are passed between connected components to more effectively compute the reliability of the global system.

Fidelity. While time, cost, and reliability are common and universal measurements, fidelity is a function of effective design which refers to an intrinsic property(ies) or characteristic(s) of a good produced by a task realization.

Since fidelity fully depends on the intrinsic properties and characteristics of the goods produced, it is not a universal measurement. This means that for each reduction rule presented previously, it is not possible to specify a general and universal formula to compute fidelity. Thus, for each reduction system (except for network systems) and for each fidelity attribute, a specific formula needs to be specified. For example, the Swiss watchmaker TAG Heuer conducts a series of sixty tests to their watches during the manufacturing process. Specific tasks carry out the tests, which are placed at strategic locations in the process. Each testing task can have a fidelity attribute associated with it that represents the number of tests that have been passed when the task was executed. In this case, the following fidelity function can be specified for the sequential reduction rule:

$$F(t_{ij}).a_{\text{number of tests passed}} = f(F(t_i), F(t_j)) \text{ and}$$

$$f(vx, vy) = vx.a_{\text{number of tests passed}} + vy.a_{\text{number of tests passed}}$$

In this example, the function *f* is additive and simply adds the number of tests passed by each task. In other cases, the function *f* can be multiplicative, and therefore can be similar to the functions employed to compute metrics for the reliability dimension.

It is the responsibility of the designer to set for each fidelity attribute involved in a workflow the fidelity functions (*f*) to be used when computing workflow QoS. The designer can select a function from available sets of fidelity functions specifically constructed to match particular domain requirements. Alternatively, if the functions needed cannot be found due to their specificity, the designer can manually define new functions to meet his/her requirements.

2.6.2 SIMULATION MODELING

In order to follow organizational strategies and meet organizational goals, workflow systems need to be able to analyze workflows according to their QoS. While mathematical methods can be effectively used (see previous section), another alternative is to utilize simulation analysis (Miller, Cardoso *et al.* 2002). Simulation can play an important role in tuning the quality of service metrics of workflows by exploring "what-if" questions. When the need to adapt or to change a workflow is detected, deciding what changes to carry out can be very difficult. Before a change is actually made, its possible effects can be explored with simulation. To facilitate rapid feedback, the workflow system and the simulation system need to interoperate. In particular, workflow specification documents need to be translated into simulation model specification documents so that the new model can be executed/animated on-the-fly.

In our project, these capabilities involve a loosely-coupled integration between the METEOR WfMS and the JSIM simulation system (Nair, Miller *et al.* 1996; Miller, Nair *et al.* 1997; Miller, Seila *et al.* 2000). Workflow is concerned with scheduling and transformations that take place in tasks, while simulation is mainly concerned with system performance. For modeling purposes, a workflow can be abstractly represented by using directed graphs (*e.g.*, one for control flow and one for data flow, or one for both). Since both models are represented as directed graphs, interoperation is facilitated. In order to carry out a simulation, the appropriate workflow model is retrieved from the repository and translated into a JSIM simulation model specification. The simulation model is displayed graphically and then executed/animated. Statistical results which indicate workflows QoS are collected and displayed.

In order to simulate METEOR workflows, we are enhancing the JSIM Web-Based Simulation System. In JSIM, simulation entities flow through a digraph consisting of the following types of nodes.

Table 2-7 – Nodes in JSIM

Source	Produces entities with random times
Server	Provides service to entities
Facility	Inherits from server, adds a waiting queue
Signal	Alters number of service units in a server(s)
Sink	Sink consumes entities and records statistics

These nodes are connected together with transports, which move entities from one node to the next. These edges provide a smooth motion of entities when a simulation model is animated. These edges are labeled with branching probabilities.

The mapping of a workflow digraph to a simulation digraph is straightforward. A METEOR *start*, *stop* task will be mapped to a JSIM Source and Sink node, respectively. A METEOR human task will be mapped to a JSIM Facility, with the number of service units equal to the number of human participants carrying out the task and feeding of the same worklist. A METEOR transactional/non-transactional task will be mapped to a JSIM Facility, with the number of service units equal to the number of processors available to execute the task. These default mappings can be customized (*e.g.*, a non-transactional task that does not allow requests to be queued should be mapped to a JSIM Server). Each edge in the METEOR digraph will be mapped to a corresponding edge in the JSIM digraph. In METEOR, edges are labeled with the data type of objects flowing along the edge. In the case of *xor* nodes, they are also labeled with Boolean expressions. (The first one that evaluates to true will be the edge selected.) In the current version of JSIM, data flow must be handled by custom coding. A Boolean expression is mapped to the probability that the condition will evaluate to true and that none of the preceding

conditions will evaluate to true. For more details on mapping workflow specifications into simulation models specifications, see Chandvasekavan *et al.* (2002).

2.6.3 WORKFLOW QOS METRICS OF INTEREST

In this section, we list the workflow QoS metrics which are of interest to compute. The computation can be done at either design time or runtime. At design time, QoS computations help the designer to compose workflows that will exhibit QoS metrics which accord with initial requirements. At runtime, the computation of QoS allows the manager and administrator to identify workflow instances that have ceased to meet initial QoS requirements. This situation may occur when tasks fail, break down, or when necessary services are unavailable. The metrics presented can be automatically computed using the SWR algorithm.

2.6.3.1 WORKFLOW TIME

The workflow monitor records the total time workflow instances spend within a process. When a workflow process is executed, instances enter the process, then proceed through various tasks, and finally exit the workflow process. For example, in our scenario, the DNA Sequencing had a time constraint; it had to be completed in less than 31 weeks. The WfMS needs to constantly monitor and estimate the time remaining for instance termination. In Table Table 2-8 and Table 2-9, we show four important measurements for workflow time-based executions: *workflow response time*, *workflow delay time*, *minimum workflow response time*, and *workflow response time efficiency*.

Table 2-8 – Workflow QoS metrics for the time dimension (Part A)

Workflow Response Time (T)	T(w) = T(*SWR*(w))
The workflow response time is the total amount of time that a workflow instance spends within a workflow process before it finishes. The response time in a workflow is equal to the sum of the response times at the individual tasks, less any time that two or more tasks are superimposed on one another. Two or more tasks superimpose their response time when they are executed in parallel.	
Workflow Delay Time (DT)	DT(*w*) = DT(*SWR*(*w*))
The workflow delay time, sometimes called "waiting time," is the total amount of time that a workflow instance spends in a workflow, while not being processed by a task. The average delay time in a workflow is equal to the sum of the delay times at the individual tasks, less any time that two or more tasks are superimposed.	

Table 2-9 – Workflow QoS metrics for the time dimension (Part B)

Minimum Workflow Response Time (min T)	$\min T(w) = \min T(SWR(w))$
The minimum workflow response time, sometimes called the "service time" of a workflow, is the time required for a workflow instance to be processed, not accounting for any task delay time. Thus, it includes only the task response time, ignoring completely the impact of the task delay time. The minimum workflow response time is equal to the sum of the process time at the individual tasks, less any time that two or more tasks superimpose.	
Workflow Response Time Efficiency (E)	$E(w) = \frac{\min T(w)}{T(w)}$
The workflow response time efficiency is the ratio of the minimum workflow response time and the workflow response time. It is instructive to compare these two measures, since instance efficiency measurement provides an indication of the time an instance is delayed during its execution and also indicates the degree a workflow process can be improved by reducing its response time.	

2.6.3.2 WORKFLOW COST, RELIABILITY, AND FIDELITY

In Table 2-10, we show three other QoS measurements for workflows.

Table 2-10 – Workflow QoS metrics for the cost, reliability, and fidelity dimension

Workflow Cost (C)	$C(w) = C(SWR(w))$
Workflow cost (C) analysis measures the cost incurred during the execution of a workflow. When a workflow process is executed, various tasks, with their associated costs, are also executed. Cost-based workflows need to have their associated cost calculated so that managers can make sure that operations are within initial budgets.	
Workflow Reliability (R)	$R(w) = R(SWR(w))$
Workflow reliability (R) corresponds to the likelihood that a workflow will perform for its users on demand.	
Workflow Fidelity (F)	$F(w)._{attribute} = F(attribute, SWR(w))$
Workflow fidelity (F) is a function of effective design; it refers to the intrinsic properties or characteristics of a good produced or a service rendered.	

2.7 WORKFLOW QOS COMPUTATION EXAMPLE

The Fungal Genome Resource (FGR) laboratory is in the process of reengineering their workflows. The laboratory technicians, domain experts, and managers have agreed that an alteration to the *Prepare and Sequence* (Figure 2-10) and *Sequence Processing* (Figure 2-11) workflows would potentially be beneficial when sequencing DNA.

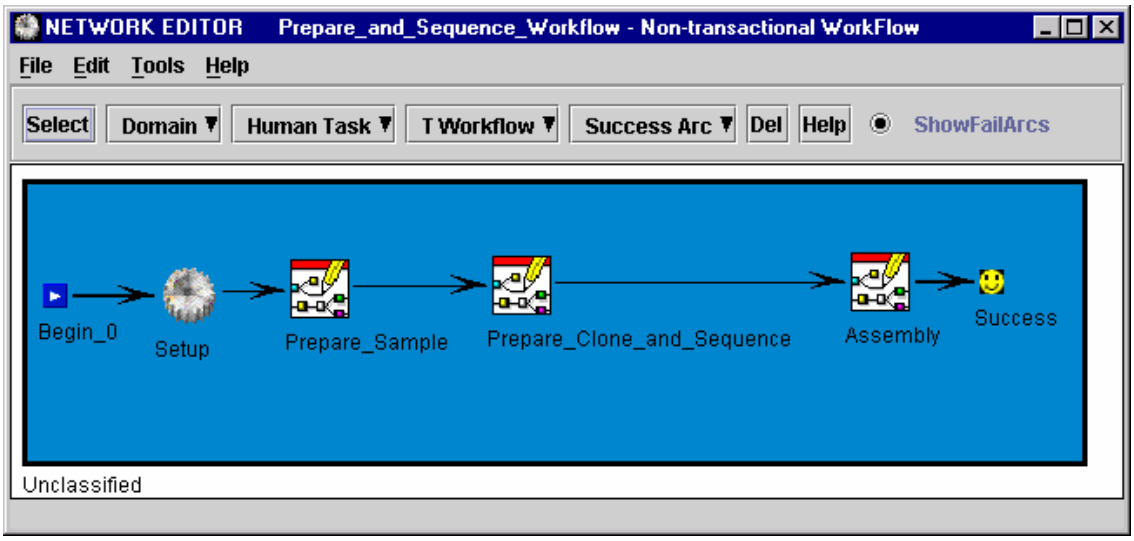

Figure 2-10 – Prepare and Sequence Workflow

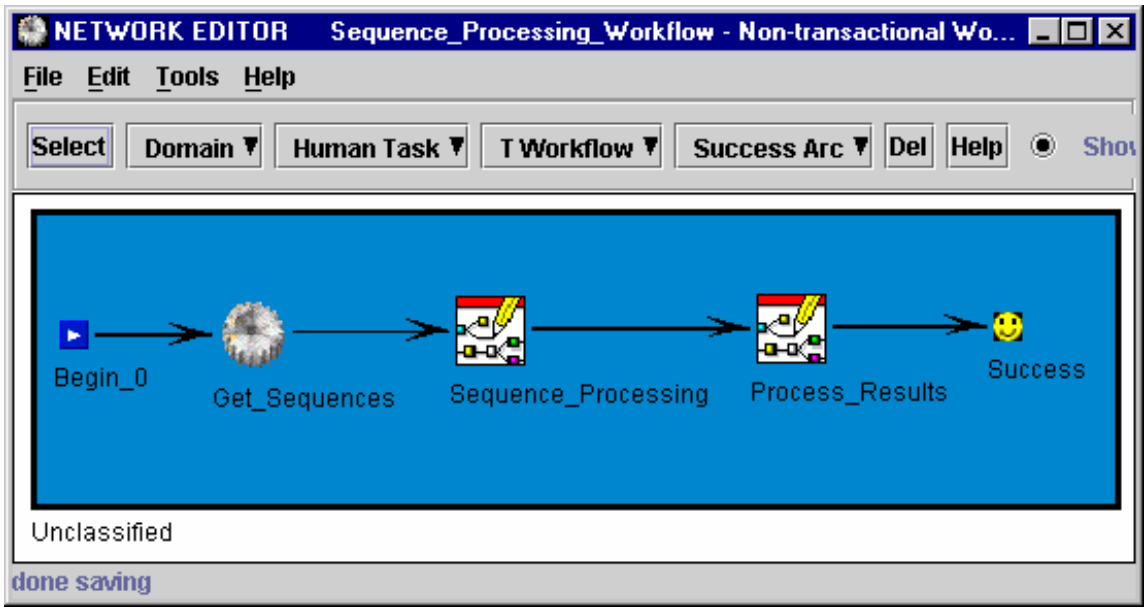

Figure 2-11 – Sequence Processing Workflow

To improve the efficiency of the processes being managed by the workflow system, the bioinformatics researchers decided to merge the two processes. The researchers noticed that the quality of the DNA sequencing obtained was in some cases useless due to *E. coli* contamination. Additionally, it was felt that it would be advantageous to use other

algorithms in the sequence processing phase. Therefore, to improve the quality of the process, the *Test Quality* task and the *SP FASTA* task were added.

Clones grown in bacterial hosts are likely to become contaminated. A quick and effective way to screen for the *Escherichia coli* (*E. coli*) contaminants is to compare the clones against the *E. coli* genome. For *E. coli*, this task is made easier with the availability of its full genome.

The task *SP FASTA* has of the same objective of the task *SP BLAST* (a task of the sequence processing sub-workflow). Both tasks compare new DNA sequences to a repository of known sequences (*e.g.*, Swiss-Prot or GenBank.) The objective is to find sequences with homologous relationships to assign potential biological functions and classifying sequences into functional families. All sequence comparison methods, however, suffer from certain limitations. Consequently, it is advantageous to try more than one comparison algorithm during the sequence processing phase. For this reason, it was decided to employ the BLAST (Altschul, Gish *et al.* 1990) and FASTA (Pearson and Lipman 1988) programs to compare sequences.

The following actions were taken to reengineer the existing workflows:

1) Merge the *Prepare and Sequence* workflow from Figure 2-10 and the *Sequence Processing* workflow from Figure 2-11,
2) Add the task *Test Quality* to test the existence of *E. coli* in sequences, and
3) Execute the search for sequences in genome databases using an additional search algorithm (FASTA).

At this point, the alterations to introduce into the processes have been identified. From the functional perspective, the lab personnel, domain experts, and workflow designer all agreed that the new workflow will accomplish the intended objective. The new re-engineered workflow is named *DNA Sequencing.* It is illustrated in Figure 2-12.

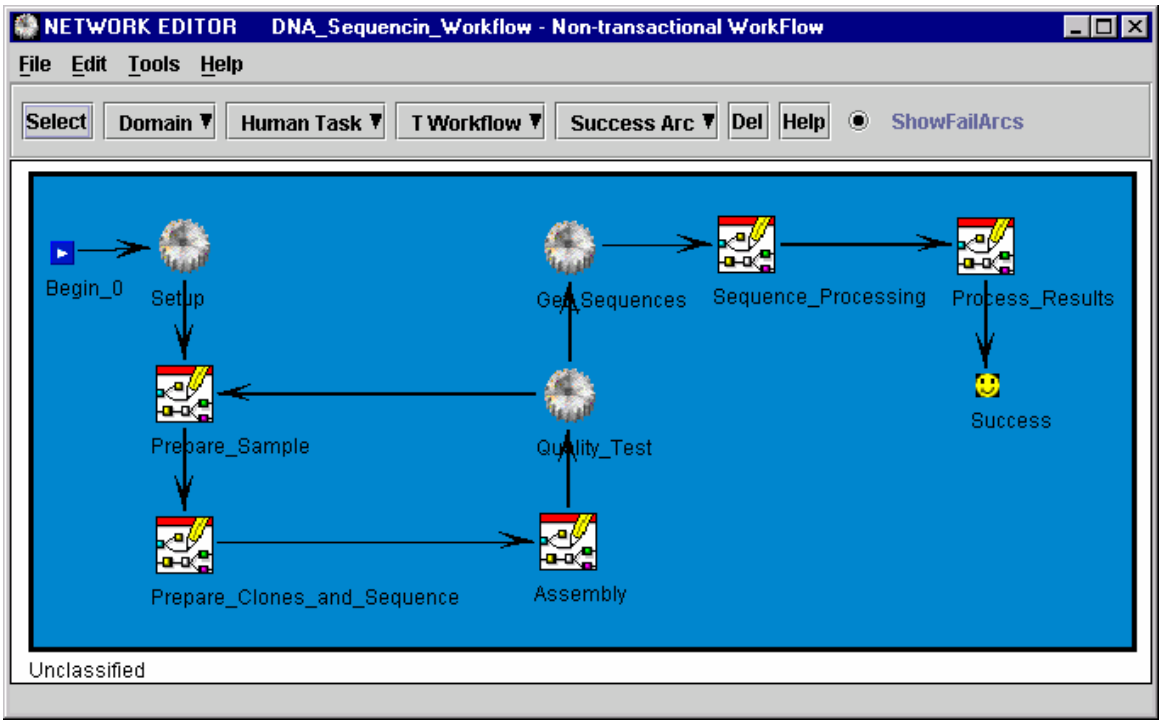

Figure 2-12 – DNA Sequencing Workflow

2.7.1 SETTING QOS METRICS

While the workflow design meets the functional objectives, non-functional requirements also need to be met. Prior to the execution of the new workflow, an analysis is necessary to guarantee that the changes to be introduced will actually produce a workflow that meets desired QoS requirements, *i.e.*, that the workflow time, cost, reliability, and fidelity remain within acceptable thresholds. To accomplish this, it is necessary to analyze the QoS metrics and use the *SWR* algorithm (Cardoso 2002) to compute workflow quality of service metrics.

The first step is to gather QoS estimates for the tasks involved in the *Prepare and Sequence* and *Sequence Processing* workflows. These workflows have been executed several times in the past, and the workflow system has recorded their QoS metrics. The designer QoS estimates have been set using the following methods. (We have omitted the designer QoS specification for the distributional class since this experiment does not

involve the use of a simulation system to compute and predict QoS metrics.) For human tasks, the laboratory technicians and researchers have provided estimates for the QoS dimensions. For automated tasks, we have used training sets. For example, for the *SP BLAST* task we have constructed a training set of sequences of different lengths. The sequences have been processed with BLAST, and their QoS has been recorded. For the time dimension, we have used linear regression to predict future metrics (since the BLAST algorithm has a linear running time (Altschul, Gish *et al.* 1990).) Equation 1 was used to estimate the BLAST running time to process a sequence:

$$y = a + bx, \quad a = \overline{Y} - b\overline{X} \quad \text{and} \quad b = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2} \tag{1}$$

where x is the independent data (input size) and y is the dependent data (running time). The estimated function is defined as:

$$y = a + bx, \text{ with } a = 78.37, b = 0.0071 \tag{2}$$

The only task with a fidelity function is the *SP BLAST* task. The fidelity attribute *HITS* indicates the percentage of sequences processed with an E value lower than e^{-15}. The E value is an indication of the probability that the match between a query sequence and a sequence stored in a database occurred by chance. For close matches, this number is typically very small.

$$F(t_{BSP\ BLAST})._{HITS} = \text{percentage of sequences with } E < e^{-15}$$

For the new tasks introduced (*Test Quality* and *SP FASTA*), no QoS runtime information is available. The only QoS information that can be used to compute the

workflow QoS is the one the designer specified at design time. The initial QoS estimates are shown in Table 2-11.

Table 2-11 – Test Quality and FASTA initial QoS estimates

	Designer Specifications			
Tasks	**T(t)**	**C(t)**	**R(t)**	**F(t)**
Quality Test	0.01	$0.0	100%	n/a
SP FASTA	9.59	$0.0	100%	0.65

Since the *SP FASTA* task is an automated task, we have used a training set of sequences to derive and set designer QoS estimates. For the time dimension, we have used the linear regression from Equation 1 and defined the function represented in Equation 3 to estimate its duration (FASTA has a linear running time (Pearson and Lipman 1988).)

$$y = a + bx, \text{ with } a = 1061.9, b = 4.11 \quad (3)$$

As for the *SP BLAST* task, the following fidelity function has been utilized to characterize the quality of the results obtained by the task *SP FASTA*:

$$F(t_{SP\ FASTA}).{}_{HITS} = \text{percentage of sequences with } E < 0.01$$

Generally, a value of 0.01 or below is statistically very significant, and a value between 0.01 and 0.05 is the borderline.

To make the workflow QoS computation possible for the fidelity dimension, formulae have been defined for the reduction systems. As an example, for parallel systems and for the *HITS* fidelity attribute, the following function has been defined:

$$F(t_{1n}).\text{HITS} = f(F(t_1), F(t_2), \ldots, F(t_n)) = \frac{\sum_{1 \le i \le n} F(t_i).\text{HITS}}{\#\text{ of tasks with the fidelity attribute HITS}}$$

Using the above formula in the *DNA Sequencing* workflow will result in the application of the following function:

$$F(t_{\text{SP BLAST FASTA}}).\text{HITS} = (F(t_{\text{SP BLAST}}).\text{HITS} + F(t_{\text{SP FASTA}}).\text{HITS})/2$$

This function represents only a possible computation for the *HITS* fidelity attribute. It is shown here with the solely objective of illustrating how fidelity attributes are computed. Additional studies of the FASTA and BLAST applications would give more information on the processing of sequences that could be used to a more precise definition of this function.

2.7.2 Computing QoS Metrics

The domain experts believe that there is a strong agreement between the tasks QoS exhibited during the execution of the *Prepare and Sequence* and the *Sequence Processing* workflows, and the expected QoS of the tasks to be scheduled by the *DNA Sequencing* workflow. This belief is based on the fact that the tasks executed in the two initial workflows will be executed without any change by the newly constructed workflow. The following functions (see also Table 2-5) have been utilized to re-compute QoS metrics based on designer and runtime information:

Table 2-12 – Re-computation of the QoS dimensions for the DNA Sequencing workflow

b)	$QoS_{Dim}(t)$	0.2*Designer $Average_{Dim}(t)$ + 0.8*Multi-Workflow $Average_{Dim}(t)$
c)	$QoS_{Dim}(t, w)$	0.2*Designer $Average_{Dim}(t)$ + 0.2*Multi-Workflow $Average_{Dim}(t)$ + 0.6*Workflow $Average_{Dim}(t, w)$

To represent the QoS agreement among tasks from different workflows, the domain experts have decided to set the weights according to the following beliefs. For formula b), the domain experts believe that the recorded QoS of tasks previously executed will give good estimates for the execution of tasks scheduled by the new workflow. Thus, the experts set the weights wi_1 and wi_2 of formula b) to 0.2 and 0.8, respectively. The domain experts also believe that as soon as tasks are scheduled by the new workflow, the QoS estimates should rely on the latest QoS data recorded from the *DNA Sequencing* workflow. Also, they consider that when QoS data is available from the *DNA Sequencing* workflow, the importance given to the designer estimates should have the same influence as the QoS estimates recorded for the execution of tasks scheduled by other workflows than the *DNA Sequencing*. Therefore, for formula c), the experts set the weights wi_1, wi_2, and wi_3 to 0.2, 0.2, and 0.6, respectively. In our experiments, we only predict workflow QoS metrics before the execution of workflow, not during workflow execution; thus, we did not to set the weights for formula c) from Table 2-6.

Since the new workflow has a loop that did not exist in any of the previously executed workflows, it is necessary to estimate the probability of the transition (*Test Quality, Prepare Sample*) to be enabled at runtime. Based on prior knowledge of sequencing experiments, the researchers calculate that approximately 10% of the DNA sequence will contain *E. coli* bacteria and that thus there is a 10% probability of the loop back transition being enabled.

2.7.3 RESULTS

We have run a set of ten experiments. Each experiment involved the execution of the SWR algorithm to predict QoS metrics of the *DNA Sequencing* workflow and the actual execution of the workflow. The results are shown for the four QoS dimensions in Figure 2-13, Figure 2-14, Figure 2-15, and Figure 2-16. The diamonds indicate the QoS estimates given by the SWR algorithm and the squares indicate the runtime metrics.

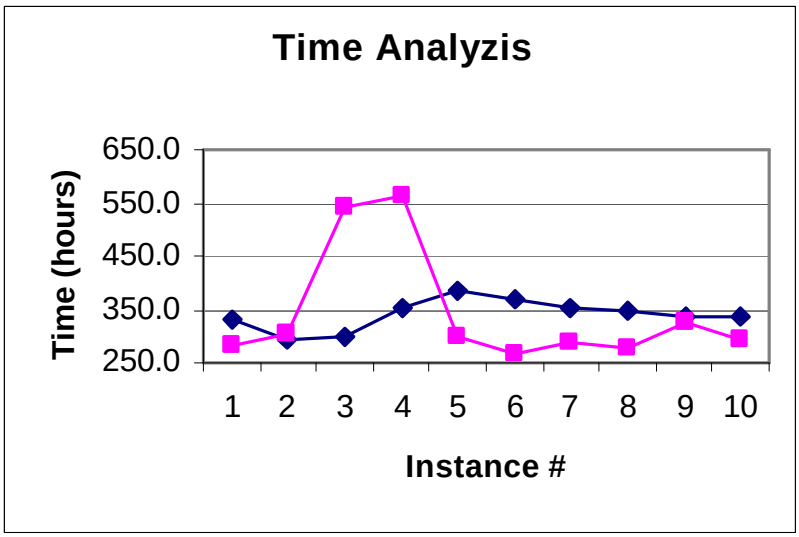

Figure 2-13 – Experiment results (Time Analysis)

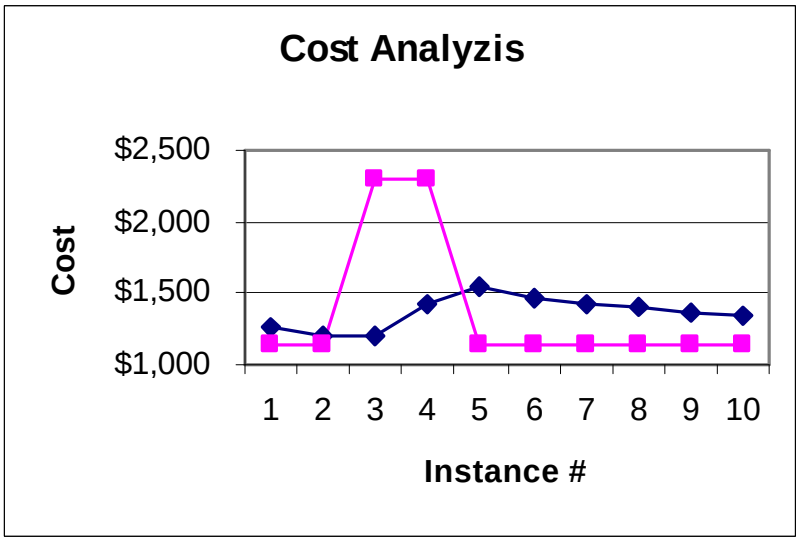

Figure 2-14 – Experiment results (Cost Analysis)

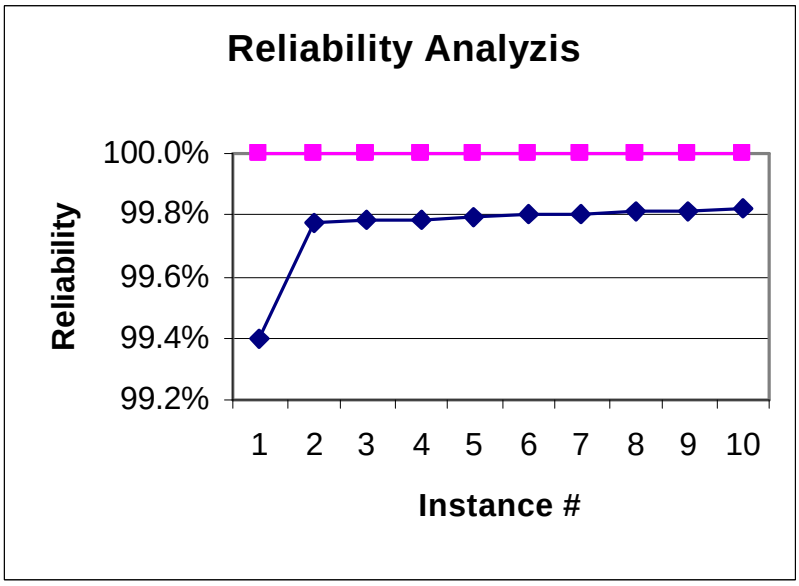

Figure 2-15 – Experiment results (Reliability Analysis)

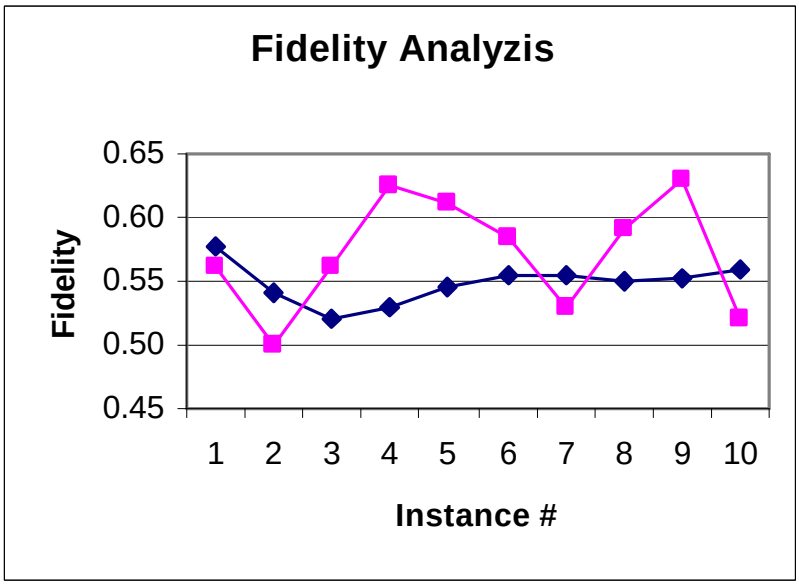

Figure 2-16 – Experiment results (Fidelity Analysis)

For the time analysis, the most relevant information that can be interpreted from the chart is the observation that the instances 3 and 4 have registered actual running times that are considerably different from the values estimated. This is due to the topology of the workflow. During the process, it is expected that some DNA sequences will contain *E. coli* contamination. When this happens, re-work is needed, and the first part of the workflow, involving the tasks *Prepare Sample*, *Prepare Clone and Sequence*, and *Assembly*, has to be re-executed. The first part of the workflow takes approximately 99% of the overall workflow execution time. Thus, when *E. coli* contamination is present in a sequence, the time needed to execute the workflow almost doubles. Since it is impossible to know if a DNA sequence will contain *E. coli* or not, the SWR algorithm gives an estimate for instance 3 which is significantly different from the registered values. When instance 4 is executed, the QoS metrics from the previous instance are considered for the QoS estimation. As a result, it can be seen in the chart that the SWR estimation converges to the mean of the recent time metrics recorded. If more instances detect the presence of

E. coli contamination, the results of the SWR algorithm for the time dimension will gradually converge to the 550 hours level. When instances number 5 through 10 are executed, they do not detect the presence of contamination in the sequences processed. As a result, the SWR estimates are more accurate, and the estimates start to slowly converge at lower time values.

The costs associated with each task have been provided from technical datasheets describing the DNA Sequencing process. For the cost analysis, the results observed are strongly linked to the results obtained from the time analysis. Again, instances 3 and 4 have recorded actual costs that are considerably different from the values estimated. This is due to the existence of *E. coli* contamination in the sequences processed. When contamination is detected, the re-work necessary to carry out the sequencing double the cost of the instance. This is because the cost of an instance is totally determined by the tasks *Prepare Sample*, *Prepare Clone and Sequence*, and *Assembly*, which are involved in any necessary re-work. All the other tasks, which are mainly automated software tasks, are considered to have a zero cost. As with the time analysis, the convergence of the SWR algorithm towards recent registered metrics can be seen. One particularity of the DNA Sequencing workflow is the discrete linearity of its cost. When no re-work is necessary because no contamination is detected, the cost of the instance is c. If contamination is found, then re-work is needed, and the cost of the instance is $2c$. If contamination is found n times during the sequencing process, the cost of the instance will be nc. This property for the cost dimension can be observed from the chart, where instances with no re-work always have the same cost ($1,152), and instances that need re-working one time have a cost of $2,304.

The fidelity analysis shows the creation of very good estimates. It can be seen that the SWR algorithm constantly changes its convergence as a response to recently recorded QoS metrics. The runtime fidelity metrics are within a small range, as predicted from the estimates.

The reliability analysis is relatively easy to interpret. For the first instance executed, the SWR algorithm has used information specified by the designer and derived from task executions from the *Prepare and Sequence* and *Sequence Processing* workflows. The information suggests that the reliability of the new workflow design will be 99.4%. But during our experiments, the ten instances executed never failed. Thus, a 100% reliability value has been registered for each workflow instance. During the instance executions, the reliability estimates given by the SWR algorithm slowly converge to 100%. Nevertheless, it is expected that as the workflow system executes more instances, the reliability of the DNA Sequencing workflow will decrease.

For all the QoS dimensions, the degree of convergence of the SWR algorithm is directly dependent on the weights that have been set for the re-computation of the QoS dimensions (see Table 2-1 for the weights used in the DNA Sequencing workflow). A higher weight associated with the multi-workflow function implies a faster convergence when the SWR algorithm is applied. The same principal applies to the instance workflow function.

2.8 Related Work

The work found in the literature on quality of service for WfMS is limited. The Crossflow project (Klingemann, Wäsch *et al.* 1999; Damen, Derks *et al.* 2000; Grefen, Aberer *et al.* 2000) has made the major contribution. In their approach, a continuous-time Markov chain (CTMC) is used to subsequently calculate the time and the cost associated with workflow executions. While the research on quality of service for WfMS is limited, the research on time management, which is under the umbrella of workflow QoS, has been more active and productive. Eder *et al*. (1999) and Pozewaunig *et al*. (1997) present an extension of CMP and PERT by annotating workflow graphs with time, in order to check the validity of time constraints at process build-time and instantiation-time, and to take pre-emptive actions at run-time. The major limitation of their approach is that only

directed acyclic graphs (DAG) can be modeled. This is a significant limitation since the many workflows have cyclic graphs. Cycles are, in general, used to represent re-work actions or repetitive activities within a workflow. Our approach deals with acyclic workflows as well as with cyclic workflows. Our experience on modeling real-world applications has shown that a significant number of workflows have cyclic graphs. Dadam *et al.* (Reichert and Dadam 1998; 2000) also recognize that time is an important aspect of workflow execution. With each workflow task, minimal and maximal durations may be specified. The system supports the specification and monitoring of deadlines. The monitoring system notifies users when deadlines are going to be missed. It also checks if minimal and maximal time distances between tasks are followed according to initial specifications. Marjanovic and Orlowska (1999) describe a workflow model enriched with modeling constructs and algorithms for checking the consistency of workflow temporal constraints. Their work mainly focuses on how to manage workflow changes, while accounting for temporal constraints. Son *et al.* (2001) present a solution for the deadline allocation problem based on queuing networks. Their work also uses graph reduction techniques, but these are applied to queuing theory. Studies on workflow reliability can also be found in the literature. The research is mainly concentrated on system implementation issues. In (Kamath, Alonso *et al.* 1996) the authors propose an architecture to enhance workflow systems' reliability via replication. Different reliability levels for different categories of process instances are used. Tang and Veijalainen (1999) propose the use of a fragmentation technique to provide higher reliability, without using a replication-based solution. Wheater and Shrivastava (1998) describe a workflow system that relies on a middleware infrastructure to provide a fault-tolerant execution environment, enhancing system and applications reliability.

Although the work on quality of service for workflows is lacking, a significant amount of research has been done in the areas of networking (Cruz 1995; Georgiadis, Guerin *et al.* 1996), real-time applications (Clark, Shenker *et al.* 1992) and middleware

(Zinky, Bakken *et al.* 1997; Frolund and Koistinen 1998; Hiltunen, Schlichting *et al.* 2000).

Recently, in the area of Web services, researchers have also manifested an interest in QoS. The DAML-S (Ankolekar, Burstein *et al.* 2001; DAML-S 2001) specification allows the semantic description of business processes. The specification includes constructs which specify quality of service parameters, such as quality guarantees, quality rating, and degree of quality. One current limitation of DAML-S' QoS model is that every process needs to have QoS metrics specified by the user.

2.9 FUTURE WORK

The workflow QoS model presented in this paper can be extended in two additional dimensions which are useful for workflow systems with stronger requirements. The first dimension is *maintainability*. Maintainability corresponds to the mean time necessary to repair workflow failures; it is the average time spent to maintain workflows in a condition where they can perform their intended function. Maintenance actions mainly involve the correction of failures during workflow execution. Workflow systems record the period of time necessary for a faulty task to be repaired. The time spent to repair a workflow component depends on the type of error that has occurred. Reparative actions can be as simple as restarting a workflow scheduler that has crashed (Kochut, Sheth *et al.* 1999), or they can be more complex, involving the installation of an ORB infrastructure in a new machine to transfer workflow schedulers, for example. To increase maintainability, advanced mechanisms have been developed to allow workflow systems to automatically recover from errors. Luo *et al.* (2000) describe the architecture and implementation of an exception-handling mechanism. The system detects and propagates exceptions which occur during instances execution to an exception-handling module. The system, based on case-based reasoning theory, derives exception handlers to repair damaged workflows

(Luo, Sheth *et al.* 1998). The system has the ability to adapt itself over time. The knowledge acquired in past experiences is used in the resolution of new problems.

The second dimension that can be included is the *trust* dimension. The use of workflow systems to coordinate and manage Web services compels the development of techniques to appraise the global security level of workflows specifications. Workflow systems and applications face several security problems, and dedicated mechanisms are needed to increase the level of security. Major problems include the distributed nature of WfMSs, the use of non-secure networks (*i.e.,* the Internet), the use of Web servers to access workflow systems data, and the potential multi-organizational span of workflows. Systems security level is assessed through the existence of security mechanisms (such as authentication, access control, labels, audits, system integrity, security policy, *etc*.) and through the use of development techniques (such as formal specifications, formal proofs, tests, *etc*.). The importance of developing secure workflow systems has been recognized, and prototypes combining workflow and security technology have already been developed. We have extended workflow technology with the implementation of two security modules. The first one (Miller, Fan *et al.* 1999) and (Fan 1999) describes a workflow security architecture which targets the five security services (authentication, access control, data confidentiality, data integrity, and non-repudiation) recommended by the International Standards Organization for network-based information systems. The second one (Kang, Froscher *et al.* 1999) describes a multilevel secure (MLS) workflow system to enable distributed users and workflow applications to cooperate across classification levels. MLS workflow systems allow users to program multilevel mission logic, to securely coordinate distributed tasks, and to monitor the progress of the workflow across classification levels.

The functions used to compute the QoS dimensions at runtime (Table 2-6) have their terms weighted. The user is responsible for setting the weights (wi_1, wi_2, wi_3, and wi_4). These weights remain constant as the workflow system registers new workflow

executions. Additional research would be useful to analyze the effect of substituting the constant weights with variable weights. The idea would be to allow the workflow system to automatically change the weights based on the number of workflow executions. As more instances are registered for a workflow *w,* the weights specified for the Designer and Multi-Workflow functions can be decreased and the weight associated with the Workflow function increased. This corresponds with the belief that over time the QoS metrics of the instances of the workflow *w* will give more accurate and fresh data to be used with the SWR algorithm. The use of Bayesian estimates (Bernardo and Smith 1994) are one of the solutions that can be investigated to enable the automatic adjustments of the weights.

2.10 CONCLUSIONS

New trading models, such as e-commerce, bring a new set of challenges and requirements that need to be explored and answered. Many E-commerce applications composed of Web services forming workflows, which in turns represent an abstraction of cross-organizational business processes. The use of workflows and workflow systems to conduct and coordinate businesses in a heterogeneous and distributed environment has an immediate operational requirement: the management of workflow quality of service. The composition of Web services, and therefore workflows, cannot be undertaken while ignoring the importance of quality of service measurements. Trading agreements between suppliers and customers include the specification of QoS items such as products or services to be delivered, deadlines, quality of products, and cost of service. The correct management of such Quality of Service (QoS) specifications directly impacts the success of organizations participating in e-commerce and also directly impacts the success and evolution of e-commerce itself.

In this paper, as a starting point, we show the importance of quality of service management for workflow and workflow systems. Based on our experience with the

development of workflow applications for various domains and with emergent workflow requirements, we present a QoS model. This model allows for the description of workflow components from a quality of service perspective; it includes four dimensions: time, cost, reliability, and fidelity. The use of QoS increases the added value of workflow systems to organizations, since non-functional aspects of workflows can be described. The model is predictive. Based on the QoS of workflow components (tasks), the QoS of workflows (networks) can be automatically computed. This feature is important, especially for large processes which in some cases may contain hundreds of tasks. We present a mathematical model that formally describes the formulae to compute QoS metrics among workflow tasks. Based on these formulae, we have developed an algorithm (SWR algorithm) to automatically compute the overall QoS of a workflow. The algorithm applies a set of reduction rules to a workflow, until only one task remains which represents the QoS for the entire workflow. We also describe how a simulation system can be used with a workflow system to carry out efficient workflow QoS simulations.

To test the validity of our QoS model and of our mathematical model we have deployed a set of production workflows in the area of genetics (Fungal Genome Resource laboratory). We have executed several workflow instances and the generated QoS data have been collected and analyzed. The analysis of the data corroborates our initial hypothesis which states that our QoS model and mathematical model give a suitable framework to predict and analyze the QoS of production workflows.

2.11 REFERENCES

Aalst, W. M. P. v. d. (1999). Generic Workflow Models: How to Handle Dynamic Change and Capture Management Information. Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems (CoopIS'99), Edinburgh, Scotland, IEEE Computer Society Press. pp. 115-126.

Aalst, W. M. P. v. d., A. P. Barros, A. H. M. t. Hofstede and B. Kiepuszeski (2002). Workflow patterns homepage. http://tmitwww.tm.tue.nl/research/patterns.

Allen, F. E. (1970). "Control Flow Analysis." SIGPAN Notices **5**(7): 1-19.

Altschul, S. F., W. Gish, W. Miller, E. W. Myers and D. J. Lipman (1990). "Basic local alignment search tool." Journal of Molecular Biology **215**: 403-410.

Ankolekar, A., M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara and H. Zeng (2001). DAML-S: Semantic Markup for Web Services. Proceedings of the International Semantic Web Working Symposium (SWWS). pp. 39-54.

Anyanwu, K., A. P. Sheth, J. A. Miller, K. J. Kochut and K. Bhukhanwala (1999). "Healthcare Enterprise Process Development and Integration.," LSDIS Lab, Department of Computer Science, University of Georgia, Athens, GA, Technical Report.

Bernardo, J. M. and A. F. M. Smith (1994). Bayesian Theory, Wiley.

Bussler, C. (1998). Workflow Instance Scheduling with Project Management Tools. 9th Workshop on Database and Expert Systems Applications DEXA'98, Vienna, Austria, IEEE Computer Society Press. pp. 753-758.

CAPA (1997). "Course Approval Process Automation (CAPA)," LSDIS Lab, Department of Computer Science, University of Georgia, Athens, GA. July 1, 1996 - June 30, 1997.

Cardoso, J. (2002). Stochastic Workflow Reduction Algorithm. LSDIS Lab, Department of Computer Science, University of Georgia, http://lsdis.cs.uga.edu/proj/meteor/QoS/SWR_Algorithm.htm.

Cardoso, J. (2002). Workflow Quality of Service and Semantic Workflow Composition. Ph.D. Dissertation. Department of Computer Science, University of Georgia, Athens, GA.

Cardoso, J., Z. Luo, J. Miller, A. Sheth and K. Kochut (2001). Survivability Architecture for Workflow Management Systems. Proceedings of the 39th Annual ACM Southeast Conference, Athens, GA. pp. 207-216.

Chandvasekavan, S., G. Silver, J. A. Miller, J. S. Cardoso and A. P. Sheth (2002). Composite Web Service: Performance Evaluation and Simulation. Proceedings of the 2002 Winter Simulation Conference, San Diego, California (in progress).

Chen, Q., U. Dayal, M. Hsu and M. L. Griss (2000). Dynamic-Agents, Workflow and XML for E-Commerce Automation. EC-Web. pp. 314-323.

Clark, D., S. Shenker and L. Zhang (1992). Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. Proceedings of ACM SIGCOMM. pp. 14-26.

Cruz, R. L. (1995). "Quality of service guarantees in virtual circuit switched networks." IEEE J. Select. Areas Commun. **13**(6): 1048-1056.

Dadam, P., M. Reichert and K. Kuhn (2000). Clinical Workflows: the Killer Application for Process Oriented Information Systems. 4th International Conference on Business Information Systems (BIS 2000), Poznan, Poland. pp. 36-59.

Damen, Z., W. Derks, M. Duitshof and H. Ensing (2000). Business-to-business E-Commerce in a Logistics Domain. The CAiSE*00 Workshop on Infrastructures for Dynamic Business-to-Business Service Outsourcing, Stockholm, Sweden.

DAML-S (2001). "Technical Overview - a white paper describing the key elements of DAML-S."

Eder, J., E. Panagos, H. Pozewaunig and M. Rabinovich (1999). Time Management in Workflow Systems. BIS'99 3rd International Conference on Business Information Systems, Poznan, Poland, Springer Verlag. pp. 265-280.

Fan, M. (1999). Security for the METEOR Workflow Management System. M.Sc. Thesis. Department of Computer Science, University of Georgia, Athens, GA.

Fensel, D. and C. Bussler (2002). The Web Service Modeling Framework. Vrije Universiteit Amsterdam (VU) and Oracle Corporation, http://www.cs.vu.nl/~dieter/ftp/paper/wsmf.pdf.

FGR (2002). Fungal Genome Resource laboratory, http://gene.genetics.uga.edu/.

Franceschini, F. (2002). Advanced quality function deployment. Boca Raton, FL, St. Lucie Press.

Frlund, S. and J. Koistinen (1998). "Quality-of-Service Specification in Distributed Object Systems." Distributed Systems Engineering Journal **5**(4).

Frolund, S. and J. Koistinen (1998). "Quality-of-Service Specification in Distributed Object Systems." Distributed Systems Engineering Journal **5**(4): 179-202.

Garvin, D. (1988). Managing Quality: The Strategic and Competitive Edge. New York, Free Press.

Georgiadis, L., R. Guerin, V. Peris and K. Sivarajan (1996). "Efficient Network QoS Provisioning Based on Per Node Traffic Shaping." IEEE ACM Transactions on Networking **4**(4): 482-501.

German Shegalov, Michael Gillmann and G. Weikum (2001). "XML-enabled workflow management for e-services across heterogeneous platforms." The VLDB Joumal **10**: 91-103.

Goel, A. L. (1985). "Software reliability models: assumptions, limitations, and applicability." IEEE Transactions on Software Engineering **11**(12): 1411-1423.

Grefen, P., K. Aberer, Y. Hoffner and H. Ludwig (2000). "CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises." International Journal of Computer Systems Science & Engineering **15**(5): 227-290.

Hall, D., J. A. Miller, J. Arnold, K. J. Kochut, A. P. Sheth and M. J. Weise (2000). "Using Workflow to Build an Information Management System for a Geographically Distributed Genome Sequence Initiative," LSDIS Lab, Department of Computer Science, University of Georgia, Athens, GA, Technical Report.

Hamlet, D., D. Mason and D. Woit (2001). Theory of Software Component Reliability. 23rd International Conference on Software Engineering ICSE'2001. pp. 361-370.

Hiltunen, M. A., R. D. Schlichting, C. A. Ugarte and G. T. Wong. (2000). Survivability through Customization and Adaptability: The Cactus Approach. DARPA Information Survivability Conference and Exposition (DISCEX 2000). pp. 294-307.

Hoyland, A. and M. Rausand (1994). System Reliability Theory: Models and Statistical Methods, Wiley, John & Sons, Incorporated.

Ireson, W. G., C. F. C. Jr. and R. Y. Moss (1996). Handbook of reliability engineering and management. New York, McGraw Hill.

ISO9000 (2002). ISO9000. International Organization for Standardization, http://www.iso.ch/iso/en/iso9000-14000/iso9000/iso9000index.html.

Kamath, M., G. Alonso, R. Guenthor and C. Mohan (1996). Providing High Availability in Very Large Workflow Management Systems. Proceedings of the 5th International Conference on Extending Database Technology, Avignon. pp. 427-442.

Kang, M. H., J. N. Froscher, A. P. Sheth, K. J. Kochut and J. A. Miller (1999). A Multilevel Secure Workflow Management System. Proceedings of the 11th Conference on Advanced Information Systems Engineering, Heidelberg, Germany, Springer. pp. 271-285.

Kao, B. and H. GarciaMolina (1993). Deadline assignment in a distributed soft realtime system. Proceedings of the 13th International Conference on Distributed Computing Systems. pp. 428-437.

Klingemann, J., J. Wäsch and K. Aberer (1999). Deriving Service Models in Cross-Organizational Workflows. Proceedings of RIDE - Information Technology for Virtual Enterprises (RIDE-VE '99), Sydney, Australia. pp. 100-107.

Knuth, D. E. (1971). "An Empirical Study of FORTRAN Programs." Software Practices and Experience **1**(12): 105-134.

Kobielus, J. G. (1997). Workflow Strategies, IDG Books Worldwide.

Kochut, K. J., A. P. Sheth and J. A. Miller (1999). "ORBWork: A CORBA-Based Fully Distributed, Scalable and Dynamic Workflow Enactment Service for METEOR,"

Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia, Athens, GA.

Kolarik, W. J. (1995). Creating Quality: Concepts, Systems, Strategies, and Tools. New York, McGraw-Hill.

Krishnakumar, N. and A. Sheth (1995). "Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operations." Distributed and Parallel Databases Journal **3**(2): 155-186.

Lazowska, E. D., J. Zhorjan, S. G. Graham and K. C. Sevcik (1984). Quantitative System Performance: Computer System Analysis Using Queueing Network Models, Prentice Hall.

Leymann, F. (2001). Web Services Flow Language (WSFL 1.0). IBM Corporation, http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

Luo, Z. (2000). Knowledge Sharing, Coordinated Exception Handling, and Intelligent Problem Solving to Support Cross-Organizational Business Processes. Ph.D. Dissertation. Department of Computer Science, University of Georgia, Athens, GA.

Luo, Z., A. P. Sheth, J. A. Miller and K. J. Kochut (1998). Defeasible Workflow, its Computation, and Exception Handling. Proceedings of 1998 Computer-Supported Cooperative Work (CSCW 1998), Towards Adaptive Workflow Systems Workshop, Seattle, WA.

Marjanovic, O. and M. Orlowska (1999). "On modeling and verification of temporal constraints in production workflows." Knowledge and Information Systems **1**(2): 157-192.

Mason, D. and D. Woit (1998). Software system reliability from component reliability. Proceedings of 1998 Workshop on Software Reliability Engineering (SRE'98), Ottawa, Ontario.

McCready, S. (1992). There is more than one kind of workflow software. Computerworld. **November 2:** 86-90.

Miles, M. B. and A. M. Huberman (1994). Qualitative data analysis: an expanded sourcebook. Thousand Oaks, California, Sage Publications.

Miller, J. A., J. S. Cardoso and G. Silver (2002). Using Simulation to Facilitate Effective Workflow Adaptation. Proceedings of the 35th Annual Simulation Symposium (ANSS'02), San Diego, California. pp. 177-181.

Miller, J. A., M. Fan, S. Wu, I. B. Arpinar, A. P. Sheth and K. J. Kochut (1999). "Security for the METEOR Workflow Management System," Department of Computer Science, University of Georgia, Athens, GA, Technical Report, pp. 33.

Miller, J. A., R. Nair, Z. Zhang and H. Zhao (1997). JSIM: A Java-Based Simulation and Animation Environment. Proceedings of the 30th Annual Simulation Symposium, Atlanta, GA. pp. 786-793.

Miller, J. A., D. Palaniswami, A. P. Sheth, K. J. Kochut and H. Singh (1998). "WebWork: METEOR2's Web-based Workflow Management System." Journal of Intelligence Information Management Systems: Integrating Artificial Intelligence and Database Technologies (JIIS) **10**(2): 185-215.

Miller, J. A., A. F. Seila and X. Xiang (2000). "The JSIM Web-Based Simulation Environment." Future Generation Computer Systems: Special Issue on Web-Based Modeling and Simulation **17**(2): 119-133.

Musa, J. D. (1993). "Operational Profiles in Software-Reliability Engineering." IEEE Software **10**(2): 14-32.

Musa, J. D. (1999). Software reliability engineering: more reliable software, faster development and testing. New York, McGraw-Hill.

Nahrstedt, K. and J. M. Smith (1996). "Design, Implementation and Experiences of the OMEGA End-point Architecture." IEEE JSAC **14**(7): 1263-1279.

Nair, R., J. A. Miller and Z. Zhang (1996). A Java-Based Query Driven Simulation Environment. Proceedings of the 1996 Winter Simulation Conference, Colorado, CA. pp. 786-793.

Nelson, E. C. (1973). "A Statistical Basis for Software Reliability," TRW Software Series March.

Parnas, D. L. (1974). On a 'Buzzword': Hierarchical Structure. Proceedings of the IFIP Congress 1974, North Holland. pp. 336-339.

Parnas, D. L. (2001). Software fundamentals: collected papers by David L. Parnas. Boston, Addison-Wesley.

Pearson, W. R. and D. J. Lipman (1988). Improved tools for biological sequence comparison. Proceedings of the National Academy of Science of the USA. pp. 2444-2448.

Pozewaunig, H., J. Eder and W. Liebhart (1997). ePERT: Extending PERT for workflow management systems. First European Symposium in Advances in Databases and Information Systems (ADBIS), St. Petersburg, Russia. pp. 217-224.

Reichert, M. and P. Dadam (1998). "ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control." Journal of Intelligent Information Systems - Special Issue on Workflow Managament **10**(2): 93-129.

Rommel, G. (1995). Simplicity wins: how Germany's mid-sized industrial companies succeed. Boston, Mass, Harvard Business School Press.

Sadiq, S., O. Marjanovic and M. E. Orlowska (2000). "Managing Change and Time in Dynamic Workflow Processes." The International Journal of Cooperative Information Systems **9**(1, 2): 93-116.

Shrivastava, S. K. and S. M. Wheater (1998). Architectural Support for Dynamic Reconfiguration of Distributed Workflow Applications. IEEE Proceedings Software Engineering. pp. 155-162.

Son, J. H., J. H. Kim and M. H. Kim (2001). "Deadline Allocation in a Time-Constrained Workflow." International Journal of Cooperative Information Systems (IJCIS) **10**(4): 509-530.

Stalk, G. and T. M. Hout (1990). Competing against time: how timebased competition is reshaping global markets. New York, Free Press.

Tang, J. and J. Veijalainen (1999). "Using Fragmentation To Increase Reliability For Workflow Systems." Society for Design and Process Science **3**(2): 33-48.

Tversky, A. and D. Kahneman (1974). "Judgement under uncertainty: Heuristics and biases." Science **185**: 1124-1131.

Weikum, G. (1999). Towards Guaranteed Quality and Dependability of Information Service. Proceedings of the Conference Datenbanksysteme in Buro, Technik und Wissenschaft, Freiburg, Germany, Springer Verlag. pp. 379-409.

Wheater, S. M. and S. K. Shrivastava (2000). "Reliability Mechanisms in the OPENflow Distributed Workflow System," Department of Computing Science, University of Newcastle upon Tyne Technical Report 31, Esprit LTR Project No. 24962 - C3DS First year Report, pp. 269-288.

Zinky, J., D. Bakken and R. Schantz (1997). "Architectural Support for Quality of Service for CORBA Objects." Theory and Practice of Object Systems **3**(1): 1-20.

CHAPTER 3

IMPLEMENTING QUALITY OF SERVICE FOR WORKFLOW MANAGEMENT SYSTEMS[2]

[2] Cardoso, J.S., A. Sheth, and K. Kochut. Submitted to the International Journal of Cooperative Information Systems (07/12/2002).

3.1 ABSTRACT

Workflow management systems (WfMSs) have been used to support various types of business processes. As organizations adopt new working models, such as e-commerce, new challenges arise for workflow systems. One such challenge is that of quality of service (QoS) management. QoS management includes mechanisms that specify, compute, monitor, and control the quality of service of the products or services to be delivered. A good management of QoS directly impacts the success of organizations participating in e-commerce activities by better fulfilling customer expectations and achieving customer satisfaction. In this paper we present an implementation of a comprehensive QoS model for workflows we have specified earlier. While the implementation is being carried out for the METEOR workflow system, the ideas presented here can also be applied to other workflow systems. In this work we describe the components that have been changed, or added, and discuss how they interact to enable the specification, computation, and monitoring of QoS.

3.2 INTRODUCTION

Organizations are constantly seeking new and innovative information systems to better fulfill their missions and strategic goals. The use of workflow Management Systems (WfMSs) allows organizations to streamline and automate business processes and reengineer their structure, as well as increase efficiency and reduce costs. Workflow systems are also a valuable asset for managing e-commerce applications that span multiple organizations (Sheth, Aalst *et al*. 1999). As the number of online services increases, workflow systems are needed to coordinate and manage the interaction among Web services (Berners-Lee 2001; Fensel and Bussler 2002).

Organizations operating in modern markets, such as e-commerce, require systematic design, planning, control, and management of business processes. One particular

important aspect is the quality of service (QoS) management. Products and services with well-defined specifications must be available to customers. This is especially important since when using the Internet to trade goods, customers do not have a tangible access to the products to be delivered. A good management of quality leads to the creation of quality products and services, which in turn fulfill customer expectations and achieve customer satisfaction. The customer's expectations and satisfaction can be translated into the quality of service rendered. Equally importantly, QoS is needed as a basis for contracts that govern e-commerce activities between trading partners.

Workflow systems should be viewed as more than just automating or mechanizing tools. They can also be used to analyze, reshape, and reengineer the way business is done. One way to achieve these objectives is through QoS analysis involving such QoS metrics as, time, cost, reliability, and fidelity. At runtime, if the monitoring of a workflow indicates the presence of unsatisfactory QoS metrics, strategies can be employed to redesign, reengineer, or dynamically adapt the workflow.

For organizations, being able to characterize workflows based on their QoS has three direct advantages. First, it allows organizations to translate their vision into their business processes more efficiently, since workflow can be designed according to QoS metrics. Second, it allows for the selection and execution of workflows based on their QoS in order to better fulfill customers' expectations. Third, it also makes possible the monitoring and control of workflows based on QoS, setting up compensation strategies when undesired metrics are identified, or use it as a tool to manage contract commitments.

The requirement of process QoS management is a new challenge for workflow systems. While QoS has been a major concern for networking, real-time applications, and middleware, few research groups have concentrated their efforts on enhancing workflow systems to support workflow quality of service (QoS) capabilities or a subset of them. Most of the research carried out to extend the functionality of workflow systems QoS has

only been done in the time dimension, which is only one of the dimensions under the QoS umbrella. Furthermore, the solutions and technologies presented are still preliminary and limited (Eder, Panagos *et al*. 1999).

Our work in this area started with the definition of a QoS model for workflows (Cardoso, Miller *et al*. 2002). The model includes four dimensions: time, cost, reliability, and fidelity. These dimensions allow for the specification of non-functional QoS metrics and for the computation of overall workflow QoS based on individual task QoS.

This paper enumerates and describes the enhancements that need to be made to workflow systems to support processes constrained by QoS requirements, such as e-commerce workflows. The enhancements include the development and support of a comprehensive QoS model and the implementation of methodologies (a mathematical model and simulation) to compute and predict workflow QoS. We have developed a stochastic workflow reduction algorithm (SWR) for the step-by-step computation of QoS metrics. Our work has been carried out for the METEOR system to allow the specification, computation, and management of QoS. The support of QoS requires the modification and extension of several workflow system components, and the development of additional modules. While the implementation was made for the METEOR system, and the development is based on a specific conceptual model, the main ideas presented in this study can be applied to the vast majority of workflow systems available (Aalst, Barros *et al*. 2002).

This paper is structured as follows. In section 3.3, we present the related work that has been done in the context of QoS management. In section 3.4, we briefly describe our QoS model and each of its dimensions. These descriptions will allow for a better understanding of QoS implementation. Section 3.5 is extensive and describes the modification of existing workflow system components and the creation of new modules that have been developed to support the workflow QoS concept. Each of workflow components and new modules are analyzed individually. Section 3.6 explains how QoS

can be computed, as based on QoS tasks. We briefly present the idea behind one algorithm that we have developed, and we also describe how simulation techniques can be used to compute workflow QoS. Finally, section 3.7 presents our conclusions.

3.3 Related Work

While QoS has been a major concern for networking (Cruz 1995; Georgiadis, Guerin *et al*. 1996), real-time applications (Clark, Shenker *et al*. 1992) and middleware (Zinky, Bakken *et al*. 1997; Frlund and Koistinen 1998; Hiltunen, Schlichting *et al*. 2000), few research groups have concentrated their efforts on enhancing workflow systems to support workflow quality of service (QoS) specifications and management.

The work found in the literature on quality of service for WfMS is limited. The Crossflow project (Klingemann, Wäsch *et al*. 1999; Damen, Derks *et al*. 2000; Grefen, Aberer *et al*. 2000) has made an early contribution by considering time and cost. In their approach, a continuous-time Markov chain (CTMC) is used to calculate the time and cost associated with workflow executions. While the research on QoS for WfMS is limited, the research on time management, which is one component of workflow QoS, has been more active and productive. Eder (1999) and Pozewaunig (1997) extend CMP and PERT by annotating workflow graphs with time. At process build-time, instantiation-time, and runtime the annotations are used to check the validity of time constraints. A significant limitation of their approach is that only direct acyclic graphs (DAG) can be modeled, especially because many real-world workflows have cyclic graphs. Cycles are in general used to represent re-work actions or repetitive activities within a workflow. Reichert (1998) and Dadam (2000) also recognize time as an important aspect of workflow execution. In their approach, it is possible to specify a deadline involving minimal and maximal durations for execution of each task. At runtime, the workflow system monitors the specified deadlines and notifies users when deadlines are missed. The system also checks if minimal and maximal time distances between tasks are followed according to

initial specifications. Marjanovic and Orlowska (1999) describe a workflow model enriched by modeling constructs and algorithms that check the consistency of workflow temporal constraints. Their work mainly focuses on how to manage workflow changes, while at the same time accounting for temporal constraints. Son and Kim (2001) present a solution for the deadline allocation problem based on queuing networks. Their work also uses graph reduction techniques, but applied to queuing networks.

Recently, researchers have been interested in QoS in the area of Web services. In the DAML-S (DAML-S 2001) specification, use of an ontology allows and facilitates process interoperability between trading partners involved in e-commerce activities. The specification includes tags to specify the quality of service parameters, such as quality guarantees, quality rating, and degree of quality. While DAML-S has identified specifications for Web service and business processes as a key specification component, the QoS model which should be adopted needs to be significantly improved to supply a realistic solution to its users. One current limitation of the DAML-S' QoS model is that it does not provide a detailed set of classes and properties that represent QoS metrics. The QoS model needs to be extended to allow for a precise characterization of each dimension. Furthermore, a model to compute overall QoS of process specified as composition of Web Services is not provided. The addition of concepts that represent the minimum, average, maximum, and distribution functions associated with dimension, such as cost and duration, will allow for the implementation of algorithms for the automatic computation of QoS metrics of processes, as based on sub-processes' QoS metrics.

3.4 WORKFLOW QUALITY OF SERVICE

In the work presented here, workflow QoS represents the quantitative and qualitative characteristics of a workflow application which is necessary to achieve a set of initial requirements. Workflow QoS addresses the non-functional issues of workflows, rather than workflow process operations. Quantitative characteristics can be evaluated in terms

of concrete measures such as workflow execution time, cost, *etc*. Kobielus (1997) suggests that dimensions such as time, cost and quality should be a criteria that workflow systems should include and might benefit from. Qualitative characteristics specify the expected services offered by the system, such as security and fault-tolerance mechanisms. QoS should be seen as an integral aspect of workflows, and therefore it should be embedded in workflow specifications and WfMSs.

Quality of service can be characterized along various dimensions. We have investigated related work to decide which dimensions would be relevant in composing our QoS model. Our research targeted two distinct areas: operations management in organizations (Garvin 1988; Stalk and Hout 1990; Rommel 1995) and quality of service for software systems, which include networking (Cruz 1995; Georgiadis, Guerin *et al*. 1996; Nahrstedt and Smith 1996), middleware areas (Zinky, Bakken *et al*. 1997; Frlund and Koistinen 1998; Hiltunen, Schlichting *et al*. 2000), and real-time applications (Clark, Shenker *et al*. 1992). The study of those two areas is important, since workflow systems are widely used to model organizational business processes, and since workflow systems are themselves software systems.

3.4.1 QoS Model

Weikum (1999) divided information services QoS into three categories: system centric, process centric, and information centric. Based on previous studies and on our experience in the workflow domain, we have constructed a QoS model that includes system and process categories. Our model is composed of four dimensions: *time*, *cost*, *fidelity*, and *reliability*.

Time (T) is a common and universal measure of performance. For workflow systems, it can be defined as the total time needed by an instance in order to transform a set of inputs into outputs. Task response time (T) corresponds to the time an instance takes to be

processed by a task. The *task response time* can be broken down into major components which include: process time, queuing delay, setup delay, and synchronization delay.

Cost (C) represents the cost associated with the execution of workflow tasks. During workflow design, prior to workflow instantiation, and during workflow execution it is necessary to estimate the cost of the execution to guarantee that financial plans are followed. Task cost is the cost incurred when a task *t* is executed; it can be broken down into major components, which include realization cost and enactment cost.

We view **Fidelity** (F) as a function of effective design; it refers to an intrinsic property or characteristic of a good produced or of a service rendered. Fidelity reflects how well a product is being produced and how well a service is being rendered. Fidelity is often difficult to define and measure because it can be subjective. Nevertheless, the fidelity of workflows should be predicted when possible and carefully controlled when needed. Workflow tasks have a fidelity vector dimension composed by a set of fidelity attributes ($F(t).a_i$) to reflect, qualify, and quantify task operations. Each fidelity attribute refers to a property or characteristic of the product being created, transformed, or analyzed. Fidelity attributes are used by the workflow system to compute how well workflows, instances, and tasks are meeting user specifications. For automatic tasks (Kochut, Sheth et al. 1999) the fidelity can be set automatically. For a human task, we must really on the person in charge of the task realization to set the fidelity attributes.

Task **Reliability** (R) corresponds to the likelihood that the components will perform when the user demands them; it is a function of the failure rate. Depending on the workflow system and task conceptual model, tasks instances can be placed into different states, typically described by a state transition diagram (task structure) during their execution. Two final states exist. One represents the success of a task realization, and the

other represents the failure of a task realization. The reliability dimension is a function of the number of times the success state is reached and the number of times the failure state is reached.

3.5 WORKFLOW QOS IMPLEMENTATION

The QoS model that we have developed is being implemented for the METEOR workflow management system. The METEOR project is represented by both a research system (METEOR 2002), and a suite of commercial systems that provide an open system based, high-end workflow management solution, as well as an enterprise application integration infrastructure. The work discussed in this paper is part of the research system and is not part of any commercial product yet.

METEOR's architecture includes four services: Enactment, Manager, Builder, and Repository. The enactment service includes two systems: ORBWork (Kochut, Sheth *et al*. 1999) and WebWork (Miller, Palaniswami *et al*. 1998). The task of the enactment service is to provide an execution environment for processing workflow instances. Both ORBWork and WebWork use fully distributed implementations. WebWork, an entirely Web-based enactment service, is a comparatively light-weight implementation that is well-suited for less complex applications that involve limited data exchange and do not need to be dynamically changed. ORBWork is targeted for more demanding, mission-critical enterprise applications requiring high salability, robustness and dynamic adaptation. The current version of ORBWork has been designed to address a variety of shortcomings found in today's workflow systems. It supports interoperability standards such as JFLOW (OMG 1998) and SWAP (Swenson 1998). Although we started with the use of open standards such as Java and CORBA to make it a good candidate for interoperating with existing systems from a variety of distributed and heterogeneous computing environments, recently a Java-only version (replacing CORBA with RMI) has also been completed. With recently added modules, it also includes a repository for reuse

(Song 2001), dynamic changes (Chen 2000) at the instance level and an exception-handling mechanism (Luo 2000). ORBWork has been used in prototyping and deploying workflow applications in various domains, such as bio-informatics (Hall, Miller *et al*. 2000), healthcare (Anyanwu, Sheth *et al*. 1999), telecommunications (Luo 2000), defense (Kang, Froscher *et al*. 1999), and university administration (CAPA 1997).

In this section we describe the components that make up the METEOR system and the components that have been modified, extended, and created to enable QoS management. Changes have been made to four services: the Enactment, the Manager, the Builder, and the Repository. These components and their relationship to the overall workflow system are illustrated in Figure 3-1.

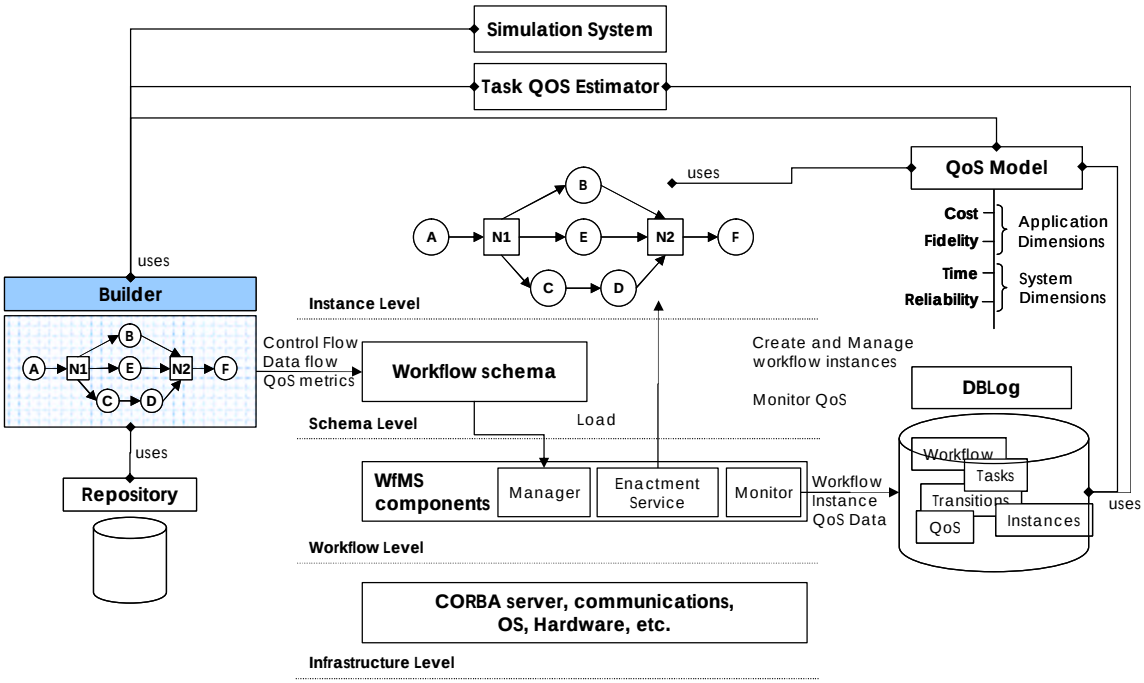

Figure 3-1 – QoS Management Architecture

3.5.1 ENACTMENT SERVICE

In this section we describe the modifications that have been made to the ORBWork enactment system. The components analyzed include task schedulers, task managers, and monitors.

In ORBWork enactment service, task schedulers, task managers, and tasks are responsible for managing runtime QoS metrics. From the implementation point of view, we divide the management of the QoS dimensions into two classes: the system and the application dimensions. The system dimensions (time and reliability) are the responsibility of task schedulers, while the application dimensions (cost and fidelity) are the responsibility of task managers and tasks. Since task schedulers decide the starting time of task execution and are notified when tasks are complete, they set the time dimension of the QoS. Additionally, the supervision of tasks completion puts them in charge of managing the reliability dimension. These two dimensions are called system dimensions because it is a system component (the enactment system) that is responsible for registering the time and reliability metrics at runtime. For the cost and fidelity dimensions, task managers are the candidate components since they include the necessary functions to initialize tasks with estimated QoS metrics. The cost and fidelity dimensions are called application dimensions since they are manipulated and modified by a task realization.

3.5.1.1 TASK SCHEDULERS

ORBWork follows a fully distributed scheduling strategy. The scheduling responsibilities are shared among a number of participating task schedulers, according to workflow definitions. The distributed schedulers maintain a workflow data specification that has been received during workflow installation. Each task scheduler provides a well-constrained subset of the HTTP protocol and thus implements a lightweight, local Web server. The scheduler accesses workflow specifications through the HTTP protocol,

directly from specification files or from the repository. Each set of task specifications includes input dependency (input transitions), output transitions with their associated conditions, and date objects that are sent into and out of the task. As discussed previously, task schedulers are responsible for managing the time and reliability dimensions. We discuss each one of these separately in the following sections.

Managing Time

In section 3.4 we have classified task response time (T) as the time an instance takes to be processed by a task. Task response time is composed of two major components: *delay time* (DT) and *process time* (PT). Delay time is further broken down into *queuing delay* (QD) and *setup delay* (SD). This makes the response time of a task *t* represented as followed:

$$T(t) = DT(t) + PT(t) = QD(t) + SD(t) + PT(t)$$

Another important time metric is the *synchronization delay* (SyncD). This measure corresponds to the time *and-join* tasks spend waiting for all the incoming transitions to be enabled. The SyncD(*t*) of a task *t* is the difference of the time t_b registered when all the incoming transitions of task *t* are enabled and the time t_a registered when the first incoming transition was enabled, *i.e.* $t_b - t_a$. This measure gives valuable information that can be used to re-engineer business processes to increase their time efficiency.

To efficiently manage the time dimension, workflow systems must register values for each of the functions involved in the calculation of task response time (T). The time dimension has its values set according to the task structure illustrated in Figure 3-2. Each state has been mapped to one of the functions that compose the time dimension. ORBWork system follows this task structure to represent workflow task execution behavior (Krishnakumar and Sheth 1995). To more effectively support QoS management,

the original structure has been extended, with the inclusion of the *Pre-Init*, as shown in Figure 3-2.

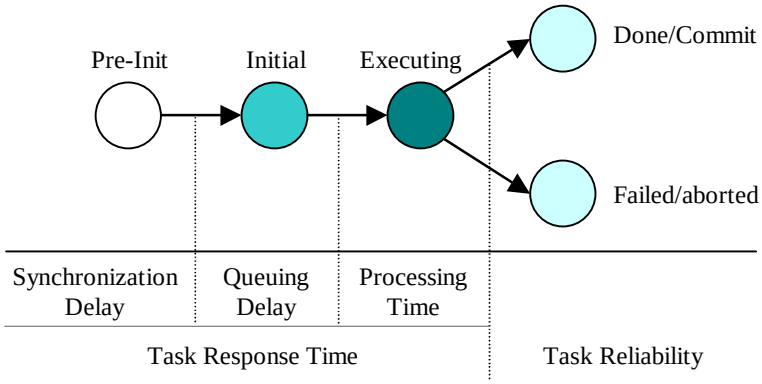

Figure 3-2 – Revised task structure (extended from (Krishnakumar and Sheth 1995))

The synchronization delay time is calculated based on the difference between the time registered when a task leaves the *pre-init* state and the time registered when it enters the state. A task *t* remains in the *pre-init* state as long as its task scheduler is waiting for another transition to be enabled in order to place the task into an *initial* state. This only happens with synchronization tasks, *i.e.*, *and-join* tasks (Kochut 1999), since they need to wait until all their incoming transitions are enabled before continuing to the next state. For all other types of input and output logic (*xor-split*, *xor-join*, *and-split*) the synchronization delay time is set to zero.

As for the synchronization delay time, the queuing time is the difference between the time a task leaves and enters the *initial* state. A task in the *initial* state indicates that the task is in a queue waiting to be scheduled (by its task scheduler). ORBWork task schedulers treat their queues with a FIFO policy. One interesting queuing policy variation is associated with the scheduling of human-tasks. For a human-task instance, being in the *initial* state means that the task has been placed in a worklist for human processing. A user can select any human-task in a worklist, as long as the user role matches the task role. In this case, the queuing policy is SIRO (Serve In Random Order). Depending on

the workflow system, other useful queuing policies can be used, such as priority queues. When a task instance enters a queue a time-stamp is attached to it. When the task is removed from the queue for scheduling, another time-stamp is attached to it so that the total queuing time can be calculated later.

When a task is ready to be executed it transits to the *executing* state. As with the previous calculations, the time a task remains in this state corresponds to the processing time.

Managing Reliability

During a task realization, a number of undesirable events may occur. Depending on the successful or unsuccessful execution of a task, it can be placed in the done or fail state (for non-transactional tasks) and commit or abort (for transactional tasks). The former state indicates that the task execution was unsuccessful, while the latter state indicates that a task is executed successfully (Krishnakumar and Sheth 1995).

When an undesirable event occurs, an exception is generated. An exception is viewed as an occurrence of some abnormal event that the underlying workflow management system can detect and react to. If an exception occurs during the invocation of a task realization, its task enters the fail/abort state. In our implementation, it is the responsibility of task schedulers to identify the final state of a task execution in order to subsequently set the reliability dimension. Later this information is used to compute the failure rate, which is the ratio between the number of times the failed/aborted state is reached and the number of times state done/committed is reached. To describe task reliability we follow a discrete-time modeling approach. Discrete-time models are adequate for systems that respond to occasional demands such as database systems. We use the stable reliability model proposed by Nelson (1973), for which the reliability of a task t is $R(t) = 1$ - failure rate.

3.5.1.2 TASK MANAGERS AND TASKS

When a task is ready to execute, a task scheduler activates an associated task manager. The task manager oversees the execution of the task itself. Task managers are implemented as an object and are classified as transactional or non-transactional, depending on the task managed. Human tasks do not have an associated task manager. Once activated, the task manager stays active until the task itself completes. Once the task has completed or terminated prematurely with a fault, the task manager notifies its task scheduler. The task manager is responsible for creating and initializing a QoS data structure from QoS specifications (for the cost and fidelity dimensions) for the task overseen. When the supervised task starts its execution, the data structure is transferred to it. If the task is a non-transactional one (typically performed by a computer program), a set of methods is available to programmatically change the initial QoS estimates. No methods are supplied to change the time and reliability dimensions since the task schedulers are responsible for controlling these dimensions. For transactional tasks (*i.e.*, a database operation), only the time and reliability dimensions are dynamically set at runtime. The cost and fidelity dimensions, once initialized from the QoS specifications, cannot be changed. This is because database systems do not make available information evaluating the cost and the fidelity of the operations executed. Once the task completes its execution, the QoS data structure is transferred back to the task manager, and later from the task manager to the task scheduler. The only responsibility of the task scheduler will be to incorporate the metrics registered for the time and reliability dimensions (see section 3.5.1.1) into the QoS data structure and send it to the monitor to be processed (see next section).

In the case of human tasks (performed directly by end-users), the QoS specifications for the cost and fidelity dimensions are included in interface page(s) (as HTML templates) presented to the end-user. When executing a human task, the user can directly

set the cost and fidelity dimensions to values reflecting how the task was carried out. As mentioned previously, human-tasks do not have a task manager associated with them, and therefore a specific task scheduler is responsible for the task supervision. When the task completes its realization, the task scheduler parses the interface page(s) and retrieves the new QoS metrics that the user may have modified.

3.5.1.3 MONITOR

When workflows are installed and instances are executed, the enactment system generates information messages (events) describing the activities being carried out. The monitor is an independent component represented by an object that records all of the events for all of the workflows being processed by the enactment service. Depending on the system setup parameters, the ORBWork monitor can display the events it receives to the console or store them in a readable log file. To extend the functionality and usability of the monitor two distinct APIs have been developed: the HTTPlog and the DBlog.

The first one uses the HTTP protocol to send status information from the ORBWork monitor to remote clients. The information can be viewed remotely, using a monitor client. This is particularly suitable for administrators that need to periodically check the status of running instances. The second API, the DBlog, has been developed to store the status and QoS events generated in a relational database. When a workflow is installed and executed, task QoS estimates, runtime QoS metrics, and transition frequencies are stored in the database. The stored information will be later utilized to create a QoS profile for the tasks and to enable the computation of the workflow QoS.

3.5.1.4 DBLOG

The DBlog is a suitable interface that the monitor uses to store workflow runtime data in a database. The runtime data generated from workflow installations and instances execution is propagated to the DBlog that will be in charge of storing the information into

a specified database. Figure 3-3 shows the database schema used to store workflow-related data and tasks QoS metrics (designer and runtime metrics).

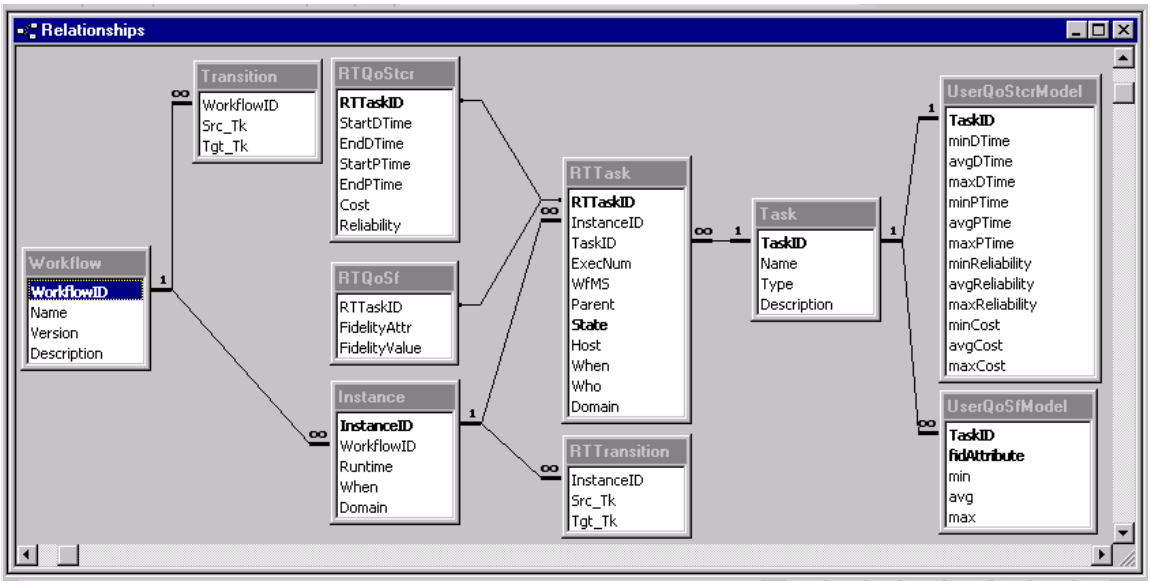

Figure 3-3 - Database Schema with QoS support

The data model includes metadata describing workflows and workflow versions, tasks, instances, transitions, and runtime QoS metrics. In addition to storing runtime QoS, we also store designer-defined QoS estimates. The data model captures the information necessary to subsequently run suitable tools to analyze workflow QoS. One of the primary goals of using a database system loosely coupled with the workflow system is to enable different tools to be used to analyze QoS, such as project management and statistical tools.

DBlog is populated when workflows are installed and instances executed. The DBlog schema was designed to store three distinct categories of information, reflecting workflow systems operations with QoS management. The first category corresponds to data events generated when workflows are installed. During installation, information describing workflow structure (which includes tasks and transitions) is stored. The second category of information to be stored corresponds to the QoS estimates for tasks

and transitions that are specified at the workflow design phase. The third category corresponds to the information which describes how instances are behaving at runtime. This includes data indicating the tasks' processing time, cost, and the enabling of transitions. The monitoring of transitions is important to build functions which probabilistically describe their enabled rate. The computation of workflow QoS metrics is based on this stochastic structure.

Since the database stores real-time runtime information of tasks QoS metrics, we are also investigating the implementation of mechanisms to automatically notify or alert operators and supervisors when QoS metrics reach threshold values, so that corrective actions can be taken immediately.

3.5.2 MANAGER

The manager is used to install and administer workflow definitions (schema), and to start workflow instances. When a workflow is installed, the manager activates all of the necessary task schedulers to carry out the execution of instances. The manager is implemented as an object and has an interface that allows clients to interact with it. The manager does not participate in any task scheduling activities. It is only necessary at the time a new workflow is installed or modified. When a workflow is installed, trace messages are sent to the monitor indicating the workflow installed and its associated tasks. The information send to the monitor also includes the initial QoS estimates that the user has set during the workflow design. When the monitor receives this information (workflow topology, tasks, and QoS estimates), it uses the DBlog interface to store it in a database for later QoS processing.

3.5.3 WORKFLOW BUILDER

The workflow builder tool is used to graphically design and specify a workflow. In most cases, after a workflow design no extra work is necessary and it can be converted

automatically to an application by a code generator. The builder is used to specify workflow topology, tasks, transitions (control flow and data flow), data objects, task invocation, roles, and security domains (Kang, Park *et al*. 2001). During the design phase, the designer is shielded from the underlying details of the runtime environment and infrastructure, separating the workflow definition from the enactment service on which it will be installed and executed. To support workflow QoS management the designer must be able to set estimates for transition probabilities and QoS estimates for tasks. This information is later combined with historical data, which plays a larger role as more instances are executed, to create a runtime QoS model for tasks and a probability model for transitions.

The workflow model and the task model have been extended to support the specification of QoS metrics. To support these extensions, the builder has been enhanced to allow designers to associate probabilities with transitions and to make possible the specification of initial QoS metrics for tasks (see section 3.5.3.1). Previously, the workflow model only included data flow mappings associated with transitions. The association of probabilities with transitions transforms a workflow into a stochastic workflow. The stochastic information indicates the probability of a transition being fired at runtime. The QoS model specified for each task and transitions probabilities are embedded into the workflow definition and stored in XML format.

3.5.3.1 SETTING INITIAL TASK QOS ESTIMATES

At design time, each task receives information which includes its type, input and output parameters, input and output logic, realization, exceptions generated, *etc*. All this information makes up the task model. The task model has been extended to accommodate the QoS model. Task QoS is initialized at design time and re-computed at runtime when tasks are executed. During the graphical construction of a workflow process, each task

receives information estimating its quality of service behavior at runtime. This includes information about its cost, time (duration), reliability, and fidelity.

The task QoS estimates are composed of two classes of information: basic and distributional. The basic class associates with each task QoS dimension the estimates of the minimum, average, and maximum values that the dimension can take. For example, for the cost dimension, it corresponds to the minimum, average, and maximum costs associated with the execution of a task. The second class, the distributional class, corresponds to the specification of a distribution function (such as Exponential, Normal, Gamma, Weibull, and Uniform) which statistically describes tasks behavior at runtime. For example, the time QoS dimension of a task can be describe by using the normal or uniform distribution function. Figure 3-4 illustrates the graphical interface that is used to specify the basic and distributional information to setup initial QoS metrics.

The values specified in the basic class are typically used by mathematical methods to compute and predict workflow QoS metrics (see SWR algorithm in Appendix), while the distributional class information is used by simulation systems to compute workflow QoS (see section 3.6.2). To devise values for the two classes, the user typically applies QoS models presented in Cardoso, Miller *et al*. (2002). We recognize that the specification of cost, time, fidelity, and reliability is a complex operation, and when not carried out properly can lead to the specification of incorrect values.

Once the design of a workflow is completed, it is compiled. The compilation generates a set of specification files and realization files for each task. The specification files (Spec files) include information describing the control and data flow of each task.

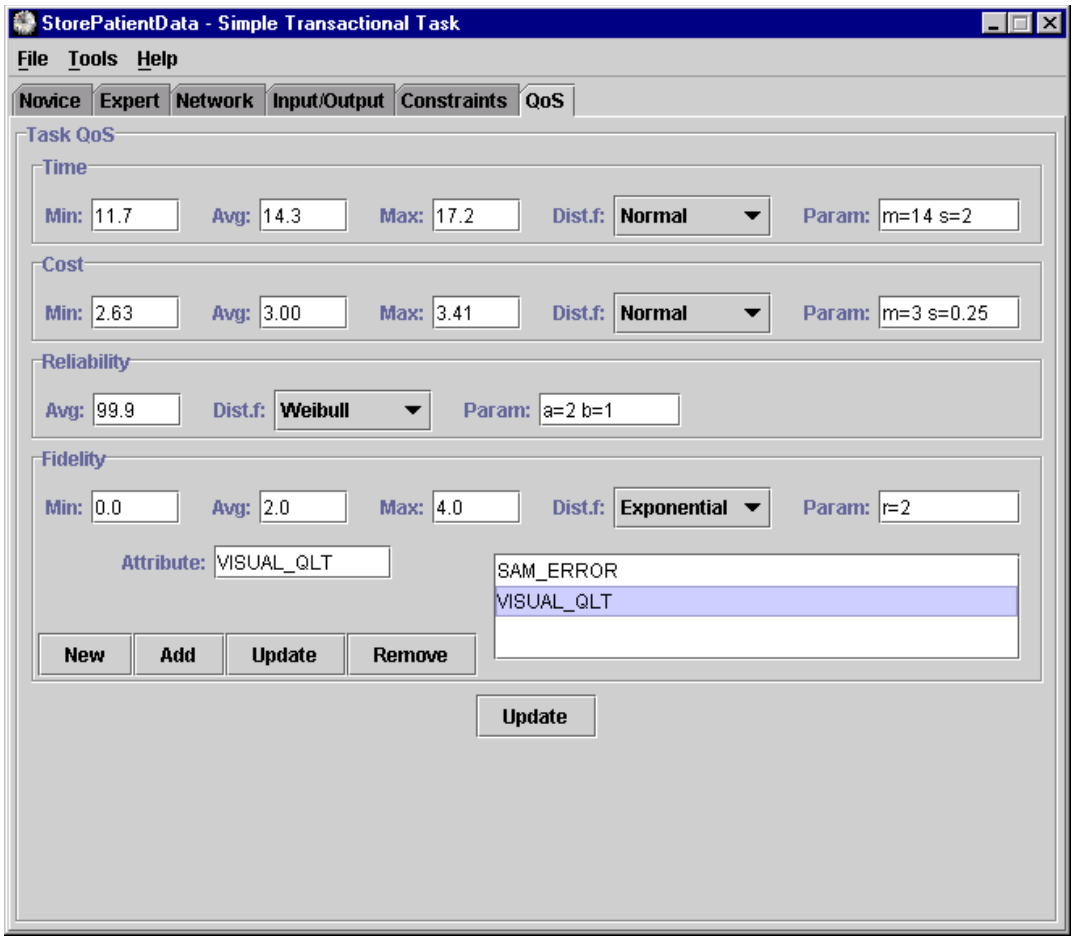

Figure 3-4 – Task QoS basic and distributional class

The realization files include the operations or instructions for a task to be executed at runtime. For human tasks, HTML files are generated, since they are carried out using a web browser. For non-transactional tasks, java code files are generated and compiled. At runtime, the executables are executed automatically by the enactment system. Finally, for non-transactional tasks a file containing the necessary data to connect to databases is generated. To enable the enactment service to acquire and manipulate QoS information, the builder has been extended to generate QoS specification files for each task. For human tasks we have decided to embed the QoS metrics directly into the HTML forms that are generated.

3.5.3.2 RE-COMPUTING QOS ESTIMATES

The initial QoS specifications may not be valid over time. To overcome this difficulty we re-compute task QoS values for the basic class, based on previous executions. The same applies for transitions. The distributional class also needs to have its distribution re-computed. This involves the analysis of runtime QoS metrics to make sure that the QoS distribution functions associated with a task remain valid or need to be modified

The re-computation of QoS estimates for tasks and for transition probabilities is done based on runtime data generated from past workflow executions that have been stored in the database log (section 3.5.1.4). We have developed a QoS Estimator module that lies between the builder and the database log. The QoS Estimator creates a QoS model for tasks based on the information stored in the DBlog. It also calculates transition probability functions based on the transitions enabled at runtime. Figure 3-5 illustrates the architecture of the QoS Estimator module. When a workflow is being designed, if the tasks selected to compose the workflow have been previously executed, then their QoS metrics are re-computed automatically using the QoS Estimator module.

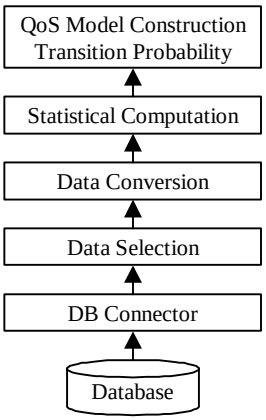

Figure 3-5 – QoS Estimator Module

DB connector

The DB Connector is responsible for the establishment of a connection to the database. Currently, we support relational databases that implement the JDBC protocol.

Data Selection

The data selection component allows for the selection of task QoS metrics, as defined by the designer and tasks previously executed. Four distinct selection modes exist, and for each one a specific selection function has been constructed. The functions are shown in Table 3-1. The component can select tasks QoS metrics from information introduced by the user at design time, from tasks executed in the context of any workflow, from tasks executed in the context of a specific workflow *w*, and from tasks executed from a particular instance *i* of workflow *w*.

Table 3-1 – Select functions of the Data Selection Component

Selection function	Description
UD_Select(*t*)	Selects the designer defined QoS metrics of task *t* specified by the designer in the basic class.
RT_Select(*t*)	Selects the runtime QoS metrics of all the executions of task *t*.
RT_Select(*t, w*)	Selects the runtime QoS metrics of all the executions of task *t* in any instance of workflow *w*.
RT_Select(*t, w, i*)	Selects the runtime QoS metrics of all the executions of task *t* in instance *i* of workflow *w*.

Data Conversion

Once a subset of the tasks present in the database log is selected, the data describing their QoS may need to be converted to a suitable format in order to be processed by the Statistical Computation component. The data conversion component is responsible for this conversion. For example, if the processing time of a task is stored using its *start execution date* and *end execution date*, the data conversion component applies the function $f(t) = end\_execution\_date(t) - start\_execution\_date(t)$ to compute the processing time (PT). As another example, let us assume that the reliability of a task is stored in the database using the keywords *done*, *fail*, *commit*, and *abort* (as in ORBWork). In this case, the data conversion component converts the keywords *done* and *commit* to the value 1, indicating the success of the task, and converts the keywords *fail* and *abort* to the value 0, indicating the failure of the task. This abstraction allows the statistical component to be independent from any particular choice of storing runtime information.

Statistical Computation

Once an appropriate set of tasks has been retrieved from the database and their QoS data has been converted to a suitable format, it is transferred to the statistical computation component to estimate QoS metrics. Currently, the module only computes the minimum, average, and maximum for QoS dimensions, but additional statistical functions can be easily included, such as standard deviations, average deviation, and variance.

Four distinct functions have been developed to compute estimates for the tasks selected in the previous step; these are shown in Table 3-2. Each function is to be used when computing QoS dimensions and corresponds to four scenarios that can occur. The first function is utilized to retrieve, for a specific task *t* and a particular dimension *Dim*, the average specified by the designer. This function is used when QoS estimates are needed and no runtime QoS information is available. The second function calculates the average of dimension *Dim* metrics for task *t*, based on all task *t* executions, independently

of the workflow that has executed it. The third function calculates the average of a task *t* dimension *Dim*, based on all the times task *t* was executed in any instance from workflow *w*. Finally, the last function (d) calculates the average of the dimension *Dim* of all the task *t* executions, from instance *i* of workflow *w*. This scenario can only occur when loops exist in a workflow, and they often do.

Table 3-2 – Designer, multi-workflow, workflow and instance average

Function	Description
a) Designer $Average_{Dim}(t)$	Average specified by the designer in the basic class for dimension Dim.
b) Multi-Workflow $Average_{Dim}(t)$	Computes the average of the dimension *Dim* of all the executions of task *t*.
c) Workflow $Average_{Dim}(t, w)$	Computes the average of the dimension *Dim* of all the executions of task *t* in any instance of workflow *w*.
d) Instance $Average_{Dim}(t, w, i)$	Computes the average of the dimension *Dim* of all the executions of task *t* in instances *i* of workflow *w*.

Similar to the functions used to compute averages as shown in Table 3-2 we also support functions to compute the minimum and maximum for QoS dimensions.

QoS Model Construction

The QoS Model Construction component uses the information computed in the statistical computation component and applies the functions presented in Table 3-3 in order to re-

compute a QoS model for tasks. The weights wi_j are set manually, and they reflect the degree of correlation between the workflow under analysis and other workflows for which a set of common tasks is shared.

Table 3-3 – QoS dimensions re-computed at runtime

a)	$QoS_{Dim}(t)$	Designer $Average_{Dim}(t)$
b)	$QoS_{Dim}(t)$	wi_1 * Designer $Average_{Dim}(t)$ + wi_2 * Multi-Workflow $Average_{Dim}(t)$
c)	$QoS_{Dim}(t, w)$	wi_1 * Designer $Average_{Dim}(t)$ + wi_2 * Multi-Workflow $Average_{Dim}(t)$ + wi_3 * Workflow $Average_{Dim}(t, w)$
d)	$QoS_{Dim}(t, w, i)$	wi_1 * Designer $Average_{Dim}(t)$ + wi_2 * Multi-Workflow $Average_{Dim}(t)$ + wi_3 * Workflow $Average_{Dim}(t, w)$ + wi_4 * Instance Workflow $Average_{Dim}(t, w, i)$

Let us assume that we have an instance i of workflow w running, and we desire to predict the QoS of task $t \in w$. The following rules are used to choose which formula to apply when predicting QoS. If task t has never been executed before, then formula a) is chosen to predict the task QoS, since there is no other data available. If task t has been executed previously, but in the context of workflow w_n, and w != w_n, then formula b) is chosen. In this case we assume that the execution of t in workflow w_n will give a good indication of its behavior in workflow w. If task t has been previously executed in the context of workflow w, but not from instance i, then formula c) is chosen. Finally, if task t has been previously executed in the context of workflow w, and instance i, meaning that a loop has been executed, then formula d) is used.

The method used to re-compute transitions' probability follows the same lines as for the method used to re-compute tasks' QoS. When a workflow has never been executed, the values for the transitions are obviously taken from initial designer specifications, the only information available. When instances of a workflow *w* have already been executed, then the data used to re-compute the probabilities come from initial designer specifications for workflow *w* and from the executed instances.

Figure 3-6 shows the graphical user interface available to set the QoS functions and their associated weights, and to visualize the QoS estimates automatically computed for workflows, instances, tasks, and transitions. The QoS computation is carried out using the SWR algorithm (see section 3.8).

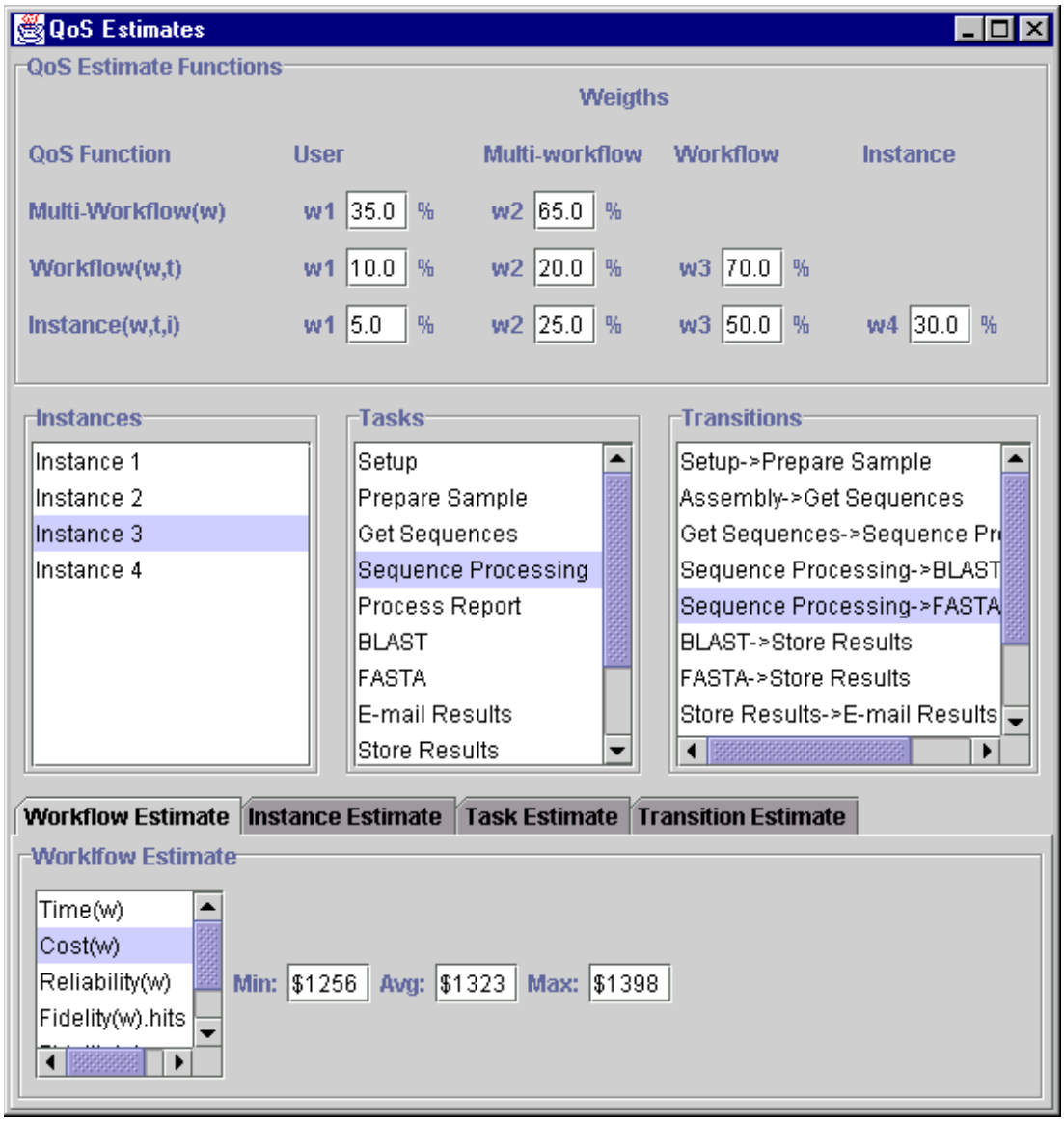

Figure 3-6 – The GUI to calculate QoS estimates

3.5.4 WORKFLOW REPOSITORY SERVICE

Our workflow builder is coupled with a repository. The repository is responsible for maintaining information about workflow definitions and associated workflow applications. The repository tool allows users to retrieve, update, and store workflow definitions (Song 2001). A user can browse the contents of the repository and find already existing workflow definitions fragments (either sub-workflows or individual tasks) to be incorporated into a workflow being created. The repository service is also available to the enactment service; it provides the necessary information about a workflow application to be started. The repository supplies a practical and efficient access to workflow definitions, based on queries. In order to query and search the repository based on QoS requirements the repository needs to be extended. This functionality is useful since it allows users to find tasks with specific QoS metrics when composing workflows with initial QoS requirements, such as low cost or high availability. While we have not implemented this feature yet, we consider it indispensable for QoS based workflow composition; and will support it in a future version of this system.

3.6 WORKFLOW QOS ANALYSIS AND SIMULATION

Having made a graphical (abstract) representation of an organizational process model, a workflow contains information which can be used as a basis for analysis. The analysis focuses on workflow topology (tasks and transitions) and on the QoS metrics. Analyzing workflows allows us to gather information about workflow QoS metrics, which include processing time, delay time, cost, fidelity, and reliability. The QoS information makes workflow structures more transparent and quantifiable, allowing inefficiencies and performance problems such as bottlenecks, to be found.

We describe two methods that the builder can use to compute QoS metrics for a given workflow process: mathematical modeling and simulation modeling. The selection

of the method is based on a tradeoff between time and the accuracy of results. The mathematical method is computationally faster, but yields results which may not be as accurate as the ones obtained with simulation. Workflow modeling is a continuous activity, where processes are continuously improved to increase efficiency and meet organizational goals and strategies.

3.6.1 MATHEMATICAL MODELING

Comprehensive solutions to the challenges encountered in synthesizing QoS for composite services have been discussed in detail (Cardoso, Miller *et al*. 2002). We have developed a stochastic workflow reduction algorithm (SWR) for step-by-step computation of aggregate QoS properties. The code, examples, and documentation for the algorithm can be found in Cardoso (2002). At each step a reduction rule is applied to shrink the workflow. Also at each step, the response time (T), cost (C), fidelity (F) and reliability (R) of the tasks involved is computed. Additional task metrics can also be individually computed, such as task queuing time and setup time. The reduction process is continued until only one atomic task (Kochut, Sheth *et al*. 1999) is left in a workflow. When this state is reached, the remaining task contains the QoS metrics corresponding to the workflow under analysis. The set of reduction rules that can be applied to a composite service (*i.e.*, workflow) corresponds to the set of inverse operations that can be used to construct a workflow of services. We have decided to allow only the construction of workflows based on a set of predefined construction rules to protect users from designing invalid workflows. Invalid workflows contain design errors, such as non-termination, deadlocks, and the split of instances (Aalst 1999). To compute QoS metrics, we use a set of six distinct reduction rules: (1) sequential, (2) parallel, (3) conditional, (4) fault-tolerant, (5) loop, and (6) network. As an illustration, we will show how reduction works for a parallel system of tasks.

Reduction of a Parallel System. Figure 3-7 illustrates how a system of parallel tasks t_1, t_2, …, t_n, an *and-split* task t_a, and an *and-join* task t_b can be reduced to a sequence of three tasks t_a, t_{1n}, and t_b. In this reduction the incoming transitions of t_a and the outgoing transitions of tasks t_b remain the same. The only outgoing transitions from task t_a and the only incoming transitions from task t_b are the ones shown in the figure. In a parallel system, the probabilities of p_{a1}, p_{a2},…, p_{1n} and p_{1b}, p_{2b},…, p_{nb} are equal to 1.

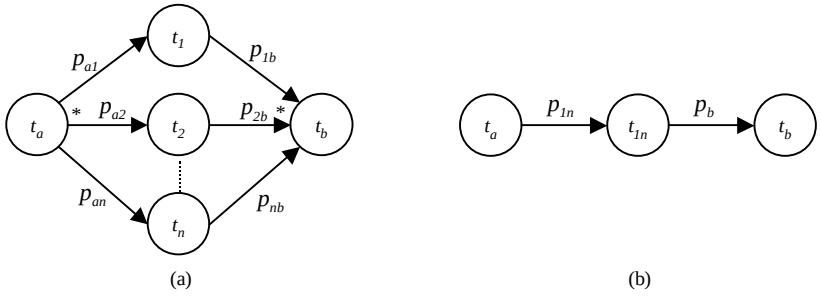

Figure 3-7 - Parallel system reduction

After applying the reduction, the QoS of tasks t_a and t_b remain unchanged, and $p_{1n} = p_b = 1$. To compute the QoS for this reduction the following formulae are applied:

$$T(t_{1n}) = \text{Max}_{i \in 1 \le i \le n} \{T(t_i)\}$$

$$C(t_{1n}) = \sum_{1 \le i \le n} C(t_i)$$

$$R(t_{1n}) = \prod_{1 \le i \le n} R(t_i)$$

$$F(t_{1n}).a_r = f(F(t_1), F(t_2), \ldots, F(t_n))$$

When a workflow needs to be analyzed, the builder converts the workflow data structure supported by the builder to one supported by the SWR algorithm. Once a workflow is in a suitable data format and each task has their QoS metrics and transition

probabilities computed, it is transferred to the SWR algorithm. The algorithm outputs a single atomic task which contains the QoS metrics corresponding to the input workflow.

3.6.2 SIMULATION MODELS

While mathematical methods can be effectively used, another alternative is to utilize simulation analysis (Miller, Cardoso *et al*. 2002). Simulation can play an important role in fine-tuning tuning the quality of service metrics of workflows, by exploring “what-if" questions. When the need to adapt or to change a workflow is detected, deciding what changes to carry out can be very difficult. Before a change is actually made, its possible effects can be explored with simulation. To facilitate rapid feedback, the workflow system and the simulation system need to interoperate. In particular, workflow specification documents need to be translated into simulation model specification documents so that the new model can be executed/animated on-the-fly.

In our project, these capabilities involve a loosely-coupled integration of the METEOR WfMS and the JSIM simulation system (Nair, Miller *et al*. 1996; Miller, Nair *et al*. 1997; Miller, Seila *et al*. 2000). Workflow is concerned with scheduling and transformations that take place in tasks, while simulation is mainly concerned with system performance. For modeling purposes, a workflow can be abstractly represented by using directed graphs (*e.g.*, one for control flow and one for data flow, or one for both). Since both models are represented as directed graphs, interoperation is facilitated. In order to carry out a simulation, the appropriate workflow model is retrieved from the repository, and the distribution functions defined in the QoS distributional class (see section 3.5.3.1) are used to create a JSIM simulation model specification. The simulation model is displayed graphically and then executed/animated. Statistical results are collected and displayed, indicating workflows QoS.

3.7 CONCLUSIONS

Organizations operating in global and competitive markets require a high level of quality of service management. The use of workflow systems to automate, support, coordinate, and manage business processes enables organizations to reduce costs and increase efficiency. Workflow systems should be viewed as more than just automating or mechanizing driving forces. They should be used to reshape and re-engineer the way business is done. One way to achieve continuous process improvement is to view and analyze processes from a QoS perspective. This allows workflows to be designed and adapted according to quality of service constraints drawn from organizational goals and strategies. A good management of QoS leads to the creation of quality products and services, which in turn fulfills customer expectations and achieves customer satisfaction. This becomes increasingly important when workflow systems are used in new organizational and trading models, such as in virtual organizations and e-commerce activities that span organizational boundaries.

While QoS management is of a high importance to organizations, current WfMSs and workflow applications do not provide full solutions to support QoS. Two research areas need to be explored. On one hand, a good theoretical QoS model is necessary to formally specify, represent, and calculate QoS metrics. On the other hand, experimental workflow systems need to be developed to identify the challenges and difficulties that the implementation of QoS management faces. We have already developed a QoS theoretical model, and in this paper we explain how the model was implemented in the METEOR system.

The support of QoS management requires the modification and extension of most of workflow system components. This includes the enactment system, the workflow builder (or designer), the monitor, the code generator, the repository, the workflow model, and the task model. Additionally, new components need to be implemented, such as a QoS

estimator module to create QoS estimates for tasks and probabilities for transitions. The monitor needs an additional interface so that runtime tasks QoS metrics are propagated and logged into a database for data processing purposes.

Algorithms and methods are necessary to predict overall workflow QoS metrics. For this purpose, we present a mathematical model and explain how simulation can be used to calculate and predict workflow QoS. Both approaches enable a predictive computation of workflows QoS based on tasks QoS estimates. The mathematical method is computationally faster, but yields results which may not be as precise as the ones obtained with simulation. The choice of the method is based on a tradeoff between time and the accuracy of results.

3.8 APPENDIX

The SWR (Stochastic Workflow Reduction) algorithm uses the set of reduction rules presented in (Cardoso, Miller *et al*. 2002) to compute workflow QoS metrics. The algorithm iteratively applies the reduction rules to a workflow until only one atomic task remains. At each iteration, the response time (T), cost (C), reliability (R), and fidelity (F) of the tasks involved is computed. Additional task metrics can also be computed, such as task queue time and setup time. If at any point no more reduction rules can be applied and the size of the workflow is greater than 1, then the initial workflow design was incorrect. An outline of the algorithm is presented in Listing 3-1.

```
QoS SWR (workflow wf) begin
    boolean changes = true;

    while changes begin
        changes = false;

        forall task in wf and  no changes begin

            changes = applySequentialRule(wf, task);
            if changes continue;

            changes = applyParallelRule(wf, task);
            if changes continue;

            changes = applyConditionalRule(wf, task);
            if changes continue;

            change = applyBasicLoopRule(wf, task);
            if changes continue;

            change = applyDualLoopRule(wf, task);
            if changes continue;

            change = applyNetworkRule(wf, task);
            if changes continue;

        end forall
    end while

    if workflow_size(wf) > 1 then error("invalid workflow schema")
    else begin
        atomic _task = getAtomicTask(wf);
        return atomic_task.QoS;
    end

end function
```

Listing 3-1 – The SWR algorithm

To check if a reduction rule can be applied a set of conditions are tested. In Listing 3-2 we illustrate the *applyConditionalRule* function. From line 3 to line 22, several conditions are tested to ensure that the conditional rule can be applied.

Once this is done, the QoS of the system being reduced is calculated (line 23 and 24) and the workflow is transformed (line 25 and 26). The transformation involves substituting the system being reduced (sequential, parallel, conditional, basic loop, dual loop, or network system) with a new task that has the QoS corresponding to the reduction.

```
1)  boolean applyConditionalRule(workflow wf, task tk) begin

2)  // check if the task tk is a "xor split" and if it is not a network task
3)  if isaXORsplit(tk) and not isaNetwork(tk) begin

4)      // get the tasks involved in the xor-split and xor-join system
5)      task[] next_tasks = wf.getNextTasks(tk);

6)      // check if all the tasks involved in the xor-split and xor-join system only have
7)      // one input and one output
8)      if not hasOneInputOneOutput(next_tks) return false;

9)      // get a task between the xor-split and xor-join task
10)     task a_next_tk = next_tks.getTask();

11)     // get the xor-join task
12)     task xor_join = wf.getNextTask(a_next_tk);

13)     // check if the xor_join task is indeed a "xor join", if the xor_join is not a
          network
14)     // task, and if the tasks involved in the xor-split and xor-join system are not
15)     // network tasks
16)     if not isaXORjoin(xor_join) or isaNetwork(xor_join) or isaNetwork(next_tks)
17)         return false;

18)     // check if the tasks following the xor-split are connected to the samexor-join
19)     if not sameDstTask(next_tks, xor_join) return false;

20)     // check if the xor-split degree is equal to the xor-join degree
21)     if wf.getNextTasks(tk).size != wf.getPrevTasks(xor_join).size
22)         return false;

23)     // compute the QoS for the conditional system
24)     QoS qos =  computeQoSConditionalSystem(wf, tk);

25)     // change the workflow structure and set the QoS for the new task created
26)     ….
27)     return true;
28) end if

29) return false;
30) end function
```

Listing 3-2 – The applyConditionalRule function

3.9 REFERENCES

Aalst, W. M. P. v. d. (1999). Generic Workflow Models: How to Handle Dynamic Change and Capture Management Information. Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems (CoopIS'99), Edinburgh, Scotland, IEEE Computer Society Press. pp. 115-126.

Aalst, W. M. P. v. d., A. P. Barros, A. H. M. t. Hofstede and B. Kiepuszeski (2002). Workflow patterns homepage. http://tmitwww.tm.tue.nl/research/patterns.

Anyanwu, K., A. P. Sheth, J. A. Miller, K. J. Kochut and K. Bhukhanwala (1999). "Healthcare Enterprise Process Development and Integration.," LSDIS Lab, Department of Computer Science, University of Georgia, Athens, GA, Technical Report.

Berners-Lee, T. (2001). Keynote presentation on web services and the future of the web. Software Development Expo 2001 Visionary Keynote, http://www.technetcast.com/tnc_play_stream.html?stream_id=616.

CAPA (1997). "Course Approval Process Automation (CAPA)," LSDIS Lab, Department of Computer Science, University of Georgia, Athens, GA. July 1, 1996 - June 30, 1997.

Cardoso, J. (2002). Stochastic Workflow Reduction Algorithm. LSDIS Lab, Department of Computer Science, University of Georgia, http://lsdis.cs.uga.edu/proj/meteor/QoS/SWR_Algorithm.htm.

Cardoso, J., A. Sheth and J. Miller (2002). Workflow Quality of Service. International Conference on Enterprise Integration and Modeling Technology and International

Enterprise Modeling Conference (ICEIMT/IEMC'02), Valencia, Spain, Kluwer Publishers.

Chen, Y. (2000). Design and Implementation of Dynamic Process Definition Modifications in OrbWork Enactment System. M.Sc. Thesis. Department of Computer Science, University of Georgia, Athens, GA.

Clark, D., S. Shenker and L. Zhang (1992). Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. Proceedings of ACM SIGCOMM. pp. 14-26.

Cruz, R. L. (1995). "Quality of service guarantees in virtual circuit switched networks." IEEE J. Select. Areas Commun. **13**(6): 1048-1056.

Dadam, P., M. Reichert and K. Kuhn (2000). Clinical Workflows: the Killer Application for Process Oriented Information Systems. 4th International Conference on Business Information Systems (BIS 2000), Poznan, Poland. pp. 36-59.

Damen, Z., W. Derks, M. Duitshof and H. Ensing (2000). Business-to-business E-Commerce in a Logistics Domain. The CAiSE*00 Workshop on Infrastructures for Dynamic Business-to-Business Service Outsourcing, Stockholm, Sweden.

DAML-S (2001). "Technical Overview - a white paper describing the key elements of DAML-S."

Eder, J., E. Panagos, H. Pozewaunig and M. Rabinovich (1999). Time Management in Workflow Systems. BIS'99 3rd International Conference on Business Information Systems, Poznan, Poland, Springer Verlag. pp. 265-280.

Fensel, D. and C. Bussler (2002). The Web Service Modeling Framework. Vrije Universiteit Amsterdam (VU) and Oracle Corporation, http://www.cs.vu.nl/~dieter/ftp/paper/wsmf.pdf.

Frlund, S. and J. Koistinen (1998). "Quality-of-Service Specification in Distributed Object Systems." Distributed Systems Engineering Journal **5**(4).

Garvin, D. (1988). Managing Quality: The Strategic and Competitive Edge. New York, Free Press.

Georgiadis, L., R. Guerin, V. Peris and K. Sivarajan (1996). "Efficient Network QoS Provisioning Based on Per Node Traffic Shaping." IEEE ACM Transactions on Networking **4**(4): 482-501.

Grefen, P., K. Aberer, Y. Hoffner and H. Ludwig (2000). "CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises." International Journal of Computer Systems Science & Engineering **15**(5): 227-290.

Hall, D., J. A. Miller, J. Arnold, K. J. Kochut, A. P. Sheth and M. J. Weise (2000). "Using Workflow to Build an Information Management System for a Geographically Distributed Genome Sequence Initiative," LSDIS Lab, Department of Computer Science, University of Georgia, Athens, GA, Technical Report.

Hiltunen, M. A., R. D. Schlichting, C. A. Ugarte and G. T. Wong. (2000). Survivability through Customization and Adaptability: The Cactus Approach. DARPA Information Survivability Conference and Exposition (DISCEX 2000). pp. 294-307.

Kang, M. H., J. N. Froscher, A. P. Sheth, K. J. Kochut and J. A. Miller (1999). A Multilevel Secure Workflow Management System. Proceedings of the 11th Conference on Advanced Information Systems Engineering, Heidelberg, Germany, Springer. pp. 271-285.

Kang, M. H., J. S. Park and J. N. Froscher (2001). Access Control Mechanisms for Inter-organizational Workflows. Proceedings of 6th ACM Symposium on Access Control Models and Technologies, Chantilly, VA.

Klingemann, J., J. Wäsch and K. Aberer (1999). Deriving Service Models in Cross-Organizational Workflows. Proceedings of RIDE - Information Technology for Virtual Enterprises (RIDE-VE '99), Sydney, Australia. pp. 100-107.

Kobielus, J. G. (1997). Workflow Strategies, IDG Books Worldwide.

Kochut, K. J. (1999). "METEOR Model version 3," Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia, Athens, GA.

Kochut, K. J., A. P. Sheth and J. A. Miller (1999). "ORBWork: A CORBA-Based Fully Distributed, Scalable and Dynamic Workflow Enactment Service for METEOR," Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia, Athens, GA.

Krishnakumar, N. and A. Sheth (1995). "Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operations." Distributed and Parallel Databases Journal **3**(2): 155-186.

Luo, Z. (2000). Knowledge Sharing, Coordinated Exception Handling, and Intelligent Problem Solving to Support Cross-Organizational Business Processes. Ph.D. Dissertation. Department of Computer Science, University of Georgia, Athens, GA.

Marjanovic, O. and M. Orlowska (1999). "On modeling and verification of temporal constraints in production workflows." Knowledge and Information Systems **1**(2): 157-192.

METEOR (2002). METEOR (Managing End-To-End OpeRations) Project Home Page. LSDIS Lab, http://lsdis.cs.uga.edu/proj/meteor/meteor.html.

Miller, J. A., J. S. Cardoso and G. Silver (2002). Using Simulation to Facilitate Effective Workflow Adaptation. Proceedings of the 35th Annual Simulation Symposium (ANSS'02), San Diego, California. pp. 177-181.

Miller, J. A., R. Nair, Z. Zhang and H. Zhao (1997). JSIM: A Java-Based Simulation and Animation Environment. Proceedings of the 30th Annual Simulation Symposium, Atlanta, GA. pp. 786-793.

Miller, J. A., D. Palaniswami, A. P. Sheth, K. J. Kochut and H. Singh (1998). "WebWork: METEOR2's Web-based Workflow Management System." Journal of Intelligence Information Management Systems: Integrating Artificial Intelligence and Database Technologies (JIIS) **10**(2): 185-215.

Miller, J. A., A. F. Seila and X. Xiang (2000). "The JSIM Web-Based Simulation Environment." Future Generation Computer Systems: Special Issue on Web-Based Modeling and Simulation **17**(2): 119-133.

Nahrstedt, K. and J. M. Smith (1996). "Design, Implementation and Experiences of the OMEGA End-point Architecture." IEEE JSAC **14**(7): 1263-1279.

Nair, R., J. A. Miller and Z. Zhang (1996). A Java-Based Query Driven Simulation Environment. Proceedings of the 1996 Winter Simulation Conference, Colorado, CA. pp. 786-793.

Nelson, E. C. (1973). "A Statistical Basis for Software Reliability," TRW Software Series March.

OMG (1998). BODTF RFP #2 Submission, Workflow Management Facility, Revised Submission, ftp://ftp.omg.org/pub/docs/bom/98-06-07.pdf.

Pozewaunig, H., J. Eder and W. Liebhart (1997). ePERT: Extending PERT for workflow management systems. First European Symposium in Advances in Databases and Information Systems (ADBIS), St. Petersburg, Russia. pp. 217-224.

Reichert, M. and P. Dadam (1998). "ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control." Journal of Intelligent Information Systems - Special Issue on Workflow Managament **10**(2): 93-129.

Rommel, G. (1995). Simplicity wins: how Germany's mid-sized industrial companies succeed. Boston, Mass, Harvard Business School Press.

Sheth, A. P., W. v. d. Aalst and I. B. Arpinar (1999). "Processes Driving the Networked Economy." IEEE Concurrency **7**(3): 18-31.

Son, J. H., J. H. Kim and M. H. Kim (2001). "Deadline Allocation in a Time-Constrained Workflow." International Journal of Cooperative Information Systems (IJCIS) **10**(4): 509-530.

Song, M. (2001). RepoX: A Repository for Workflow Designs and Specifications. M.Sc. Department of Computer Science, University of Georgia, Athens.

Stalk, G. and T. M. Hout (1990). Competing against time: how timebased competition is reshaping global markets. New York, Free Press.

Swenson, K. (1998). SWAP - Simple Workflow Access Protocol.

Weikum, G. (1999). Towards Guaranteed Quality and Dependability of Information Service. Proceedings of the Conference Datenbanksysteme in Buro, Technik und Wissenschaft, Freiburg, Germany, Springer Verlag. pp. 379-409.

Zinky, J., D. Bakken and R. Schantz (1997). "Architectural Support for Quality of Service for CORBA Objects." Theory and Practice of Object Systems **3**(1): 1-20.

CHAPTER 4

SEMANTIC E-WORKFLOW COMPOSITION[3]

[3] Cardoso, J.S. and A. Sheth. Submitted to the Journal of Intelligent Information Systems (07/12/2002).

4.1 ABSTRACT

Systems and infrastructures are currently being developed to support Web services. The main idea is to encapsulate an organization's functionality within an appropriate interface and advertise it as Web services. While in some cases Web services may be utilized in an isolated form, it is normal to expect Web services to be integrated as part of workflow processes. The composition of workflow processes that model e-service applications differs from the design of traditional workflows, in terms of the number of tasks (Web services) available to the composition process, in their heterogeneity, and in their autonomy. Therefore, two problems need to be solved: how to efficiently discover Web services – based on functional and operational requirements – and how to facilitate the interoperability of heterogeneous Web services. In this paper, we present a solution based on ontologies that overcome these problems. We start by illustrating the steps involved in the composition of a workflow. Two of these steps are the discovery of Web services and their posterior integration into a workflow. To assist designers with those two steps, we have devised an algorithm to simultaneously discover Web services and resolve heterogeneity among their interfaces and the workflow host. Finally, we describe a prototype that has been implemented to illustrate how discovery and interoperability functions are achieved.

4.2 INTRODUCTION

E-services have been are announced as the next wave of Internet-based business applications that will dramatically change the use of the Internet (Fabio Casati, Ming-Chien Shan *et al.* 2001). With the development and maturity of infrastructures and solutions that support e-services, we expect organizations to incorporate Web services as part of their business processes. While in some cases Web services may be utilized in an isolated form, it is natural to expect that Web services will be integrated as part of

workflows (Berners-Lee 2001; Fensel and Bussler 2002). Workflow management systems are capable of integrating business objects for setting up e-services in an amazingly short time and with impressively little cost (Shegalov, Gillmann *et al.* 2001). Workflows and Web services play a major role in architectures such as business-to-business (B2B), business-to-customer (B2C), customer-to-customer (C2C), dynamic trading processes (Sheth, Aalst *et al.* 1999), dynamic value chains (Lee and Whang 2001), virtual organizations, and virtual Web organizations (Ulrich 2001).

A workflow is an abstraction of a business process. It comprises a number of logic steps (known as tasks or activities), dependencies among tasks, routing rules, and participants. In a workflow, a task can represent a human activity or a software system. The emergent need of workflows to model e-service applications makes it essential that workflow tasks be associated with Web services. As a result, research is currently being carried out to enhance workflows systems in their support and management of Web services (Shegalov, Gillmann *et al*. 2001).

The modeling of e-services using workflows raises two challenges for workflow systems. First, Web services must be located that might contain (a) the desired functionality and (b) operational requirements needed to carry out the realization of a given task. It is necessary to efficiently discover Web services from the potentially thousands of services available on the Internet. Second, once the desired Web services have been found, mechanisms are needed to (c) facilitate the resolution of structural and semantic differences. This is because the heterogeneous Web services found in the first step need to interoperate with other components present in a workflow host.

(a) The design of traditional workflow applications involves the selection of appropriate tasks with their desired functionality in order to compose a workflow and to establish connections among these tasks (control and data flow). Tasks are selected from a workflow repository (Arpinar, Miller *et al*. 2001; Song 2001) which typically contains only tens to a few hundreds of tasks. Since the number of tasks to choose from is modest,

the process is humanly manageable, not requiring sophisticated search or discovery mechanisms. However, when a workflow is employed to model e-services, the potential number of Web services available for the composition process can be extremely large. Then, we are no longer searching for a task from a set of a few hundred, but we are searching for a service from a set that can potentially contain thousands of Web services. One cannot expect a designer to manually browse through all of the Web services available and select the most suitable ones.

(b) The autonomy of Web services does not allow for users to identify their operational metrics at design time, *i.e.*, before their actual execution. Operational metrics characterize Web services according to their Quality of Service (QoS), which includes their timeliness, quality of products delivered, cost of service, and reliability. When composing a workflow it is indispensable to analyze and compute its overall QoS (Cardoso, Miller *et al*. 2002; Cardoso, Sheth *et al*. 2002; Miller, Cardoso *et al*. 2002). This allows organizations to translate their vision into their business processes more efficiently, since workflows can be designed according to QoS metrics. The management of QoS directly impacts the success of organizations participating in electronic activities. A good management of quality leads to the creation of quality products and services, which in turn fulfills customer expectations and achieves customer satisfaction. To achieve these objectives, one of the first steps is to develop an adequate QoS model for workflow processes, tasks, and Web services. Such a model will allow for the discovery of Web services and for the composition of workflows based on operational requirements.

(c) The information interoperability problems that the composition of workflows involving Web services face are already well known within the distributed database systems community (Sheth and Larson 1990; Kashyap and Sheth 1996; Calvanese, Giacomo *et al*. 1998; Parent and Spaccapietra 1998). To achieve interoperability, it is necessary to address the problem of semantic integration – the identification of

semantically similar objects that belong to different systems and the resolution of their schematic differences (Kashyap and Sheth 1996). When tasks and Web services are put together, their interfaces (inputs and outputs) need to interoperate; therefore, structural and semantic heterogeneity needs to be resolved. Structural heterogeneity exists because Web services use different data structures and class hierarchies to define the parameters of their interfaces. Furthermore, semantic heterogeneity considers the intended meaning of the terms employed in labeling input and output parameters. The data that is interchanged among Web services has to be understood. Semantic conflicts occur when a Web service output connected to another service or task input does not use the same interpretation of the information being transferred. The general approach to semantic integration has been to map the local terms onto a shared ontology. Even though a shared ontology ensures total integration, constructing such an ontology is costly, if not impractical; autonomous systems are required to commit to a shared ontology, and compromises are difficult to maintain when new concepts are added (Rodríguez and Egenhofer 2002).

The main motivation for our work is the need to enhance workflow systems with better mechanisms for e-service composition. More precisely, we target the development of new mechanisms for Web services discovery and integration. Our method is novel and provides a multidimensional approach to Web service discovery and integration using syntactic, semantic, and operational metrics of Web services (Figure 4-1).

In this paper, we describe the composition process of e-workflows and present an algorithm to be employed when designers need to add Web services to an e-workflow. E-services can be orchestrated with hard-coded applications or by using workflows. We call a workflow which manages e-services and possibly traditional workflow tasks an e-workflow. Our approach relies on the use of ontologies to describe workflow tasks and Web services interfaces. Ontologies-based approaches have been suggested as a solution

for information integration that achieves interoperability (Kashyap and Sheth 1994; Uschold and Gruninger 1996).

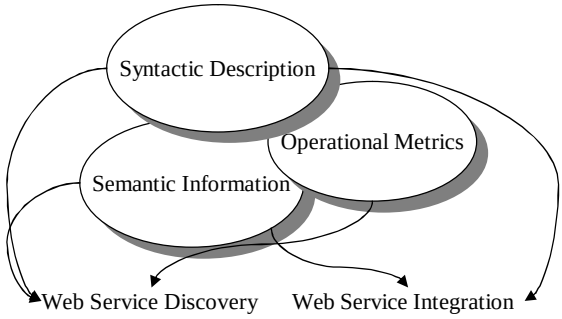

Figure 4-1 – Multidimensional approach to Web Service Discovery and Integration

The discovery and integration of Web services into e-workflows has specific requirements and challenges as compared to previous work on information retrieval systems and information integration systems. In this paper, we describe a methodology with the aim to give a solution to the following objectives and issues:

- Increase the precision of the discovery process. The search has to be based, not only on syntactic information, but also on Web services operational metrics and semantics.
- Tasks and Web services operational metrics need to be represented using a suitable model describing the QoS metrics of (Cardoso, Sheth *et al.* 2002).
- Enable the automatic determination of the degree of integration of the discovered Web services and a workflow host.
- The integration of Web services differs from previous work on schema integration due to the polarity of the schema that must be integrated. The polarity of schema forces an output schema to be connected to an input schema. Furthermore, an input schema needs to have all its input parameters satisfied. When a task or Web service is added to an e-workflow, it is necessary to integrate its input and output

schema with the other tasks already present in the process. The input schema (ns_i) of a new task needs to be integrated with one or more output schema ($s_{o,r}$) of the tasks connected to it ($\{s_{o,1}, s_{o,2}, \ldots, s_{o,n}\} \rightarrow ns_i$). The output schema ($ns_o$) of the new task needs to be integrated with one or more input schema ($s_{i,r}$) of the tasks it connects to ($ns_o \rightarrow \{s_{i,1}, s_{i,2}, \ldots, s_{i,n}\}$). This process does not require a full integration of the schema $\{s_{o,1}, s_{o,2}, \ldots, s_{o,n}\}$ with the schema ns_i. Only the input schema ns_i needs to have its schema fully integrated, *i.e.*, in order to work properly all its (mandatory) inputs need to be mapped to an output belonging to one of the schema $s_{o,r}$. For the integration of the output schema s_o, the schema $\{s_{i,1}, s_{i,2}, \ldots, s_{i,n}\}$ are the ones that need to be fully integrated.

- Previous work (Paolucci, Kawamura *et al.* 2002) on Web service discovery does not address the interoperability problem. Furthermore, the algorithm developed does not address the problem of matching outputs/inputs defined in distinct ontologies. This is a strong limitation. Since Web services are heterogeneous, autonomous, and developed independently, it is desirable to compare and discover Web services that have their schema defined by different ontologies.

This paper is structured as follows. Section 4.3 presents a scenario illustrating the composition of an e-workflow and highlights the difficulties involved. Section 4.4 focuses on the extension of traditional workflow tasks specifications to semantically describe their interfaces, on the specification of Web services, and on the association of a QoS model to specify operational metrics for both tasks and Web services. In section 4.5, we describe the composition process of an e-workflow and the structures that are created and manipulated; these will later be used in the Web service discovery phase. Section 4.6 represents the core of our work; we present an algorithm that takes into account syntactic, operational, and semantic information in order to compute the degree of similarity of a Web service template and a Web service object. The algorithm evaluates the similarity of

its arguments based on their degree of integration. Section 4.7 presents the architecture of the prototype we have developed to demonstrate the concepts introduced in this paper. Section 4.8 discusses related work, and section 4.9 presents our conclusions.

4.3 SCENARIO

A designer is composing an e-workflow to automatically manage the approval of travel authorization requests to conferences. A partial view of the workflow design is illustrated in Figure 4-2. Another interesting example, which could be cast to the e-workflow composition process, is described in (Barbar, Mehrothra *et al.* 1996). The workflow manages the arrangement, cancellation, and postponement of office meetings.

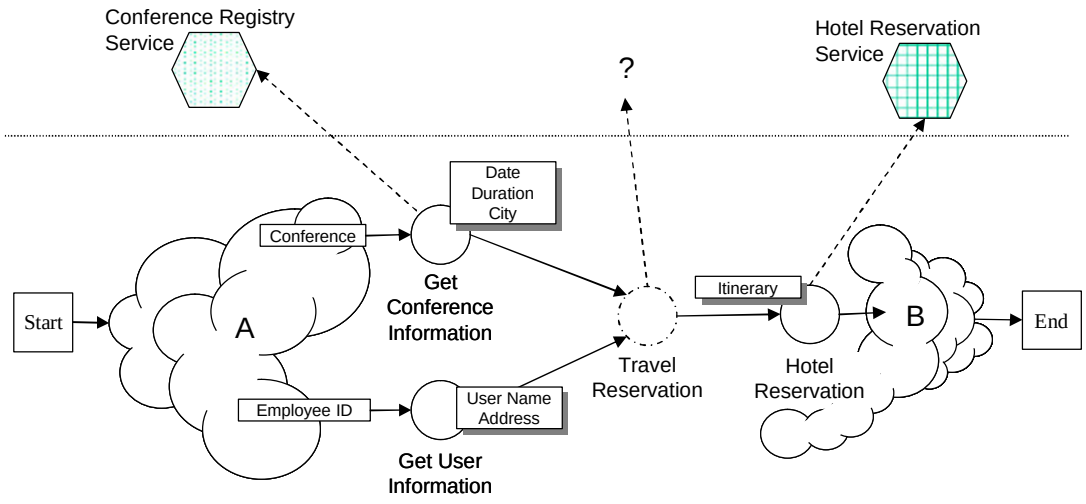

Figure 4-2 – Travel Authorization Request e-Workflow

The e-workflow operates in the following way. When an employee desires to attend a conference, he initializes an instance of the travel authorization request e-workflow. The first part of the e-workflow is the approval process; it is represented by the letter 'A' in the figure. The approval process allows managers to decide if an employee's request will be approved (we have hidden this portion of the workflow for brevity to reduce its complexity.)

If the managers approve the request, the next tasks to be executed are *Get Conference Information*, *Get User Information*, *Travel Reservation*, and *Hotel Reservation*. The *Get Conference Information* task is responsible for obtaining the date, duration, and the city where the conference is being held, based on the conference name. To obtain this information a Web service is chosen and linked to a workflow task. The *Get User Information* task retrieves the employee's name and address based on his ID. The *Travel Reservation* task is responsible for making a travel reservation according to the conference date, duration, city; it is also based on the employee's personal information. Finally, the *Hotel Reservation* task makes the necessary hotel reservation based on the travel itinerary.

Once the tasks involved with the travel and hotel reservation are executed, the portion of the e-workflow represented by the letter 'B' is executed. This part of the e-workflow is responsible for notifying the user of the travel arrangements made for him.

Let us assume that the designer has already placed the tasks shown in Figure 4-2 on the canvas. The e-workflow is almost complete; only the *Travel Reservation* task realization is missing. The designer manually looks for an appropriate Web service by browsing the Internet. This process is time consuming, cumbersome, and tedious. Potentially tens or hundreds of thousands of on-line Web services may be available. Only hundreds provide the desired functionality, and maybe only a handful provides the required operational metrics and interface (*i.e.*, input and output parameters). Furthermore, once a suitable Web service has been found, it needs to be integrated with the tasks already placed in the workflow. The designer needs to manually establish data connections among the new Web service and the tasks already present in the e-workflow, accounting for structural and semantic differences.

4.3.1 E-WORKFLOW COMPOSITION PROBLEMS

In the previous scenario, the workflow designer faces two problems: locating a Web service with the desired functionality and operational metrics to accomplish a specific task and resolving the structural and semantic differences between the service found and the tasks and Web services to which it will be connected (using transitions).

We cannot expect a designer to discover a Web service manually, since potentially thousands of services are available on the Internet. Thus, efficient discovery mechanisms must be available. What makes the e-service vision attractive is the ability to automatically discover the e-services that fulfill users' needs (Fabio Casati, Ming-Chien Shan *et al*. 2001). The discovery of a Web service cannot only be based on its name or description; it also has to account for its operational metrics and its interfaces.

The composition of e-workflows cannot be undertaken while ignoring the importance of operational metrics. Trading agreements between suppliers and customers modeled with e-workflow include the specification of QoS items such as products or services to be delivered, deadlines, quality of products, and cost of service. The correct management of such specifications directly impacts the success of organizations participating in e-commerce and also directly impacts the success and evolution of e-services itself.

Web services can be seen as black boxes, with an input interface and an output interface. Since, when integrated into an e-workflow, a Web service has to interoperate at the interface level with adjacent tasks, the discovery also has to be based on the structural and semantic properties of its inputs and outputs. Once a Web service is found, it is not realistic to expect that its interfaces will perfectly match and interoperate with the hosting e-workflow without additional work. Web services are heterogeneous by nature; we expect the designer will need to manually establish connections among the Web service interfaces and the tasks present in an e-workflow. In our example, the designer is faced

with the problems of manually connecting the outputs of the tasks *Get Conference Information* and *Get User Information* with inputs of the task *Travel Reservation*, and then connecting the outputs of the task *Travel Reservation* with the inputs of the task *Hotel Reservation*. To facilitate this work, a workflow designer should be assisted by mechanisms that suggest the establishment of a connection between outputs and inputs that maximizes the degree of integration.

4.4 WORKFLOW TASKS AND WEB SERVICE TASKS

We rely on the use of ontologies to semantically describe task and Web service interfaces. Semantics have been a strong candidate for increasing the success of information discovery and integration on the Internet; its use has been presented as the next step in the evolution of the World Wide Web (Berners-Lee and Fischetti 1999; Fensel and Musen 2001).

The importance of ontologies is being recognized in research fields as diverse as knowledge engineering, knowledge representation, qualitative modeling, language engineering, database design, information modeling, information integration, object-oriented analysis, information retrieval and extraction, knowledge management and organization, and agent-based systems design (Guarino 1998). Ontologies are introduced as an "explicit specification of a conceptualization" (Gruber 1993). The use of ontologies for the explication of knowledge is a possible approach to overcome the problem of integrating heterogeneous workflow tasks and Web services. In nearly all ontology-based integration approaches, ontologies are used for the explicit description of the information source semantics. Therefore, they can be used to describe the semantics of task interfaces, making their content and function explicit and thus enhancing the integration process.

4.4.1 ONTOLOGIES

An ontology $\Omega_i = \{c_1, \ldots, c_n\}$ contains a set of classes. Each class c_j has an associated set of properties $P_k = \{p_1, \ldots, p_m\}$. Each property has a range indicating a restriction on the values the property can take. An ontology relates more specific concepts to more general ones (from which generic information can be inherited). Such links have been variously named "is a," "subset of," "member of," "subconcept of," "superconcept," *etc*. Such links are used to organize concepts into a hierarchy or some other partial ordering, called a "taxonomy." The taxonomy is used for storing information at appropriate levels of generality and automatically making it available to more specific concepts by means of a mechanism of inheritance. More general concepts in such a partial order are said to subsume more specific concepts, and a more specific concept is said to inherit information from its subsumers. The notion of ontological concepts is very similar to the notion of classes in object-oriented programming.

In our implementation, tasks and Web services interfaces are semantically described by concepts (classes) that are defined in ontologies constructed with DAML+OIL (Horrocks, Harmelen *et al.* 2001). Our approach is not dependent on DAML+OIL; other ontology representation languages could be employed. The DAML+OIL specification enables the creation of ontologies for any domain, and it is a particularly suitable framework that makes the description of services computer-interpretable and shared.

4.4.2 EXTENDING WORKFLOW TASKS SPECIFICATIONS

In most workflow systems, each task is described by several elements which typically include a name, a type, a list of input parameters and output parameters, a short textual description, and a task realization (implementation). A task invocation specifies the number of input parameters that must be supplied for a proper task realization and the number of outputs parameters to hold and transfer the results of the task realization to other tasks. In their simplest form, the input and output parameters can be represented by

attributes, or they can follow an object-oriented model represented by data components. Attributes are specified with an attribute name, a type, and an optional initial value. Examples of built-in primitive types include Boolean, string, byte, integer, and real. Data components are represented by classes composed of a collection of attributes. Classes may form a hierarchy in which inheritance is allowed.

To enhance the integration of tasks and Web services, workflow components need to have their inputs and outputs associated with ontological concepts (classes). This will facilitate the resolution of structural and semantic heterogeneity. Since there is a strong analogy between the attributes and data classes of an object-oriented model and the concepts classes defined in an ontology, the establishment of mappings between the two is facilitated. Figure 4-3 illustrates the establishment of such a mapping.

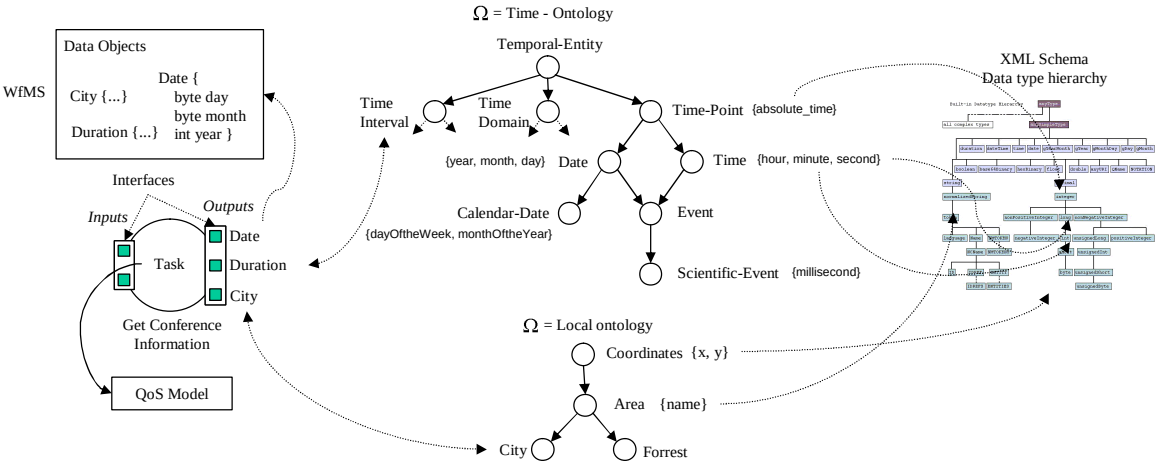

Figure 4-3 – Association of task inputs and outputs with concepts

Each input and output data class parameter of a task is associated with an ontological concept class. We assume a one-to-one mapping between a data class and its associated concept class; *i.e.*, each attribute of a data class must have a corresponding property that belongs to the associated concept class. This assumption can be further relaxed by considering work in schematic heterogeneity (Kashyap and Sheth 1996) and schema mapping (Madhavan, Bernstein *et al.* 2001).

Primitive data types of attributes (such as byte and double) are represented in the ontology by properties which reference data types defined in the XML Schema specification (XMLSchema 2001). It would have been possible to associate primitive built-in data types with ontological concepts or properties. Nevertheless, we have chosen XML Schema because it provides a comprehensive data type hierarchy, which includes unsigned byte, short, decimal, non-negative integer, string, and base 64 binary.

4.4.3 WEB SERVICE SPECIFICATION

The emergence and challenges of e-services have directed the development and creation of mechanisms to support Web services. One fundamental issue is their specification. Two main approaches have been proposed. One of the approaches uses declarative and structured data based purely on syntax, such as WSDL (Christensen, Curbera *et al.* 2001) and XLANG (Thatte 2001). A second approach provides a semantic orientation to the description of Web services. This is the case in the DAML-S specification (Ankolekar, Burstein *et al.* 2001).

Web services are "self-contained, self-describing modular applications that can be published, located, and invoked across the Web" (Tidwell 2000) and therefore are a modern alternative to the specification of workflow tasks. Since they are self-described, the interoperation among independently developed Web services is facilitated. Traditional workflow tasks, such as non-transactional, transactional, and human tasks (Kochut, Sheth *et al.* 1999) can easily be represented or encapsulated with Web services.

As with WSMF (Fensel and Bussler 2002), our approach to e-workflow composition is not dependent on the method chosen to specify Web services. Therefore, any of the specification languages mentioned above can be employed. For the prototype that we have developed we have selected the DAML-S specification; more precisely, we use the Service Profile ontology.

The service profile ontology describes the functionality of a Web service. It tells "what the service does" (Ankolekar, Burstein *et al*. 2001) and is employed to advertise Web services availability and capability. We have decided to use DAML-S because in the same way we did with workflow tasks, we need to establish associations among the inputs and outputs parameters of a Web service with ontological concepts. Since the DAML-S specification semantically describes Web services, there is an explicit association of Web services interface with concepts. In Figure 4-4 we give a partial example of the specification of a Web service using DAML-S.

```
- <profile:input>
- <profile:ParameterDescription rdf:ID="PreferredClass">
  <profile:parameterName>PreferredClass</profile:parameterName>
  <profile:restrictedTo rdf:resource="http://
www.daml.org/2001/06/itinerary/itinerary-ont.daml#class" />
   </profile:ParameterDescription>
  </profile:input>

- <profile:output>
- <profile:ParameterDescription rdf:ID="Itinerary ">
  <profile:parameterName>TripItinerary</profile:parameterName>
  <profile:restrictedTo rdf:resource="
http://www.daml.org/2001/06/itinerary/itinerary-ont.daml#Flight" />
    </profile:ParameterDescription>
  </profile:output>
```

Figure 4-4 – Web service specification using DAML-S

One of the service inputs is the PreferredClass, and one of the outputs is the TripItinerary. Both of them refer to concepts defined in the ontology itinerary-ont.daml.

When using a declarative specification language such as WSDL, there is also the need to associate each input and output with an ontological concept so that they can be semantically described. This may require the extension of the Web service specification language to include additional tags which will be employed to specify the ontology and the concepts associated with input and output parameters.

4.4.4 OPERATIONAL METRICS

The operational metrics of tasks and Web services are described using a QoS model. For us, QoS represents the quantitative and qualitative characteristics of an e-workflow application which are necessary to achieve a set of initial requirements. E-workflow QoS addresses the operational issues of workflows, rather than workflow process functions. Quantitative characteristics can be evaluated in terms of concrete measures such as workflow execution time, cost, reliability, etc. Qualitative characteristics specify the expected services offered by the system such as security and fault-tolerance mechanisms. QoS should be seen as an integral aspect of workflows, and therefore it should be integrated with tasks and Web services specifications.

While the DAML-S specification that we use includes constructs to specify quality of service parameters, such as quality guarantees, quality rating, and degree of quality, the specification does not provide a detailed set of classes and properties to represent quality of service metrics. The model needs to be extended to allow for a precise characterization of each dimension in order to permit the implementation of algorithms for the automatic computation of QoS metrics of processes based on their sub-processes' QoS metrics. Therefore, we have developed our own model.

We have investigated relevant work to determine which dimensions would be relevant to compose a more suitable QoS model for the automatic computation of QoS metrics.

Based on previous studies (Garvin 1988; Stalk and Hout 1990; Rommel 1995), as well as our experience in the workflow domain, we have constructed a model composed of the following dimensions: time, cost, reliability, and fidelity (Cardoso, Sheth *et al.* 2002). Since fidelity is subject to judgments and perceptions, we have decided to omit its specification and analysis in this paper. Nevertheless, a thorough study can be found in (Cardoso, Miller *et al*. 2002).

While in this paper we do not discuss the computation of QoS metrics, comprehensive solutions to the difficult problems encountered in synthesizing QoS for composite services are discussed in detail in Cardoso, Sheth *et al.* (2002). This paper presents a stochastic workflow reduction algorithm and discusses the use of simulation analysis (Miller, Cardoso *et al.* 2002) for computing aggregate QoS properties step-by-step.

4.4.4.1 QoS Dimensions

Based on our model, we have we have developed an ontology for the specification of QoS metrics (for tasks and Web services). This information will allow for the discovery of Web services based on operational metrics and includes the following dimensions:

Time is a common and universal measure of performance. Task response time (T) corresponds to the time a workflow instance takes to be processed by a task. The *task response time* can be broken down into two major components: *delay time* and *process time*. Delay time (DT) refers to the non-value-add time needed in order for an instance to be processed by a task. Process time (PT) is the time a workflow instance spends at a task while being processed; in other words, it corresponds to the time a task needs to process an instance.

Cost (C) represents the cost associated with the execution of workflow tasks. During workflow design, prior to workflow instantiation, and during workflow execution it is necessary to estimate the cost of its execution to guarantee that financial plans are followed. Task cost is the cost incurred when a task or Web service is executed; it can be broken down into two major components: *enactment cost* and *task realization cost.* The enactment cost (EC) is the cost associated with the management of the workflow system and workflow instances monitoring. The task realization cost (RC) is the cost associated with the runtime execution of the task.

Task **Reliability** (R) corresponds to the likelihood that the components will perform when the user demands them. It is a function of the failure rate. Each task structure has an initial state, an execution state, and two distinct terminating states. One of the states indicates that a task has failed or was aborted, while the other state indicates that a task is done or committed (Krishnakumar and Sheth 1995). This QoS dimension provides information concerning the relationship between the number of times the state done/committed is reached, and the number of times the failed/aborted state is reached. To describe task reliability we follow a discrete-time modeling approach. Discrete-time models are adequate for systems that respond to occasional demands, such as database systems. We use the stable reliability model proposed by Nelson (1973), for which the reliability of a task *t* is R(*t*) = 1 - failure rate.

4.4.4.2 DIMENSIONS CHARACTERIZATION

For each dimension, the description of the operational runtime behavior of a task is composed of two classes of information: *basic* and *distributional*.

The basic class associates with each task's QoS dimension the minimum value, average value, and maximum value the dimension can take. For example, the cost dimension corresponds to the minimum, average, and maximum cost associated with the execution of a task.

The second class, the distributional class, corresponds to the specification of a constant or of a distribution function (such as Exponential, Normal, Weibull, or Uniform) which statistically describes task behavior at runtime. The values specified in the basic class are typically employed by mathematical methods in order to compute workflow QoS metrics, while the distributional class information is used by simulation systems to compute workflow QoS.

Table 4-1 shows an example of the specification of QoS metrics for a task from a genomic workflow (Cardoso, Miller *et al.* 2002).

Table 4-1 – Task QoS for a manual task

	Basic class			Distributional class
	Min value	Avg value	Max value	Dist. Function
Time	192	196	199	Normal(196, 1)
Cost	576	576	576	576.0
Reliability	100%	100%	100%	1.0

4.5 THE E-WORKFLOW COMPOSITION PROCESS

The composition of e-workflows differs slightly from the design of traditional workflows. A typical scenario of the composition process is as follows. The designer composes an e-workflow for which several traditional workflow tasks (*e.g.* human, non-transactional, and transactional tasks) and Web service tasks have already been placed and interconnected on the canvas. Tasks with a realization are called grounded tasks (GT). When the designer wishes to add a Web service to the workflow, he starts by creating a service template (ST) – see section 4.5.1 for the formal specification of a ST – which indicates his intention to extend the functionality of the workflow. The ST will be employed later to find an appropriate Web service.

Once a ST is created, it is sent to the Web service discovery module, which returns a set of service object (SO) references that are ranked according to their degree of similarity with the service template. Services can be ranked according to a syntactical, operational, or semantic perspective. The designer then selects the most appropriate Web service to accomplish his objectives (section 4.7 shows an example of the SOs retrieved from the discovery process). The selection automatically associates a realization with the

ST, causing it to change its state to a grounded task. Additionally, a set of data mapping is presented to the designer suggesting a possible interconnection among the newly created task interfaces and the grounded task interfaces.

A ST has five sections that need to be specified:

- The name of the Web service to be found,
- Its textual description,
- Its operational metrics,
- The set of outputs parameters from the grounded tasks that will be connected to SO inputs, and
- The set of input parameters from the grounded tasks that a SO will be connected to.

The construction of a ST is illustrated in Figure 4-5. The outputs of the GTs *Get Conference Information* and *Get User Information* (*Date*, *Duration*, *City*, *User Name*, and *Address*) are employed to construct the outputs of the ST. The input of the GT *Hotel Reservation* (*Itinerary*) is employed to construct the inputs of the ST. The user manually sets the name, description, and QoS model of the Web service to be found.

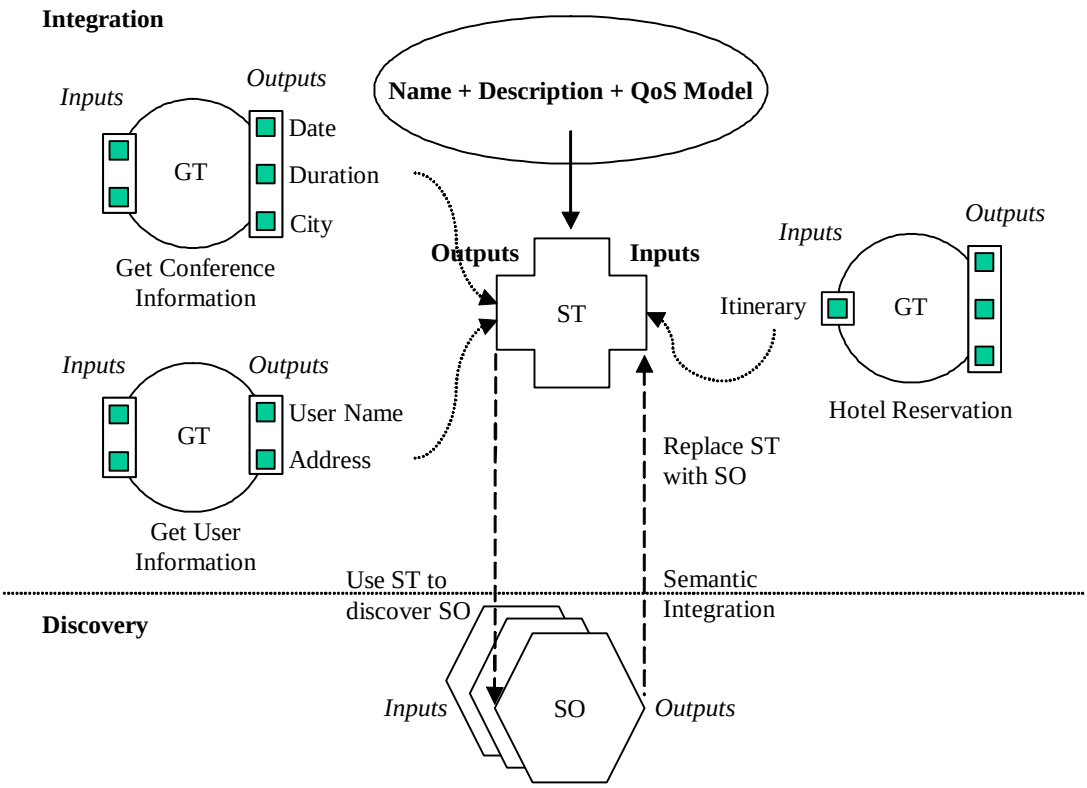

Figure 4-5 – GT, ST, and SO structures

4.5.1 E-WORKFLOW INTEGRATION COMPONENTS

The composition process described in the previous section involved the manipulation of three distinct structures: GT, ST, and SOs. In this section, we formally describe each structure.

Grounded Tasks

Grounded tasks (GT) have a realization and contribute to the achievement of the e-workflow goal. A GT is formally defined as follows:

$$GT(t) = <QoS, Is, Os>$$

Where *t*, *QoS*, *Is*, and *Os* are the name of the task, its QoS, a set of input parameters, and a set of output parameters, respectively. The QoS specification associated with a GT

is to be used by algorithms to synthesize the QoS of workflows based on the QoS metrics of the tasks and the Web services that compose the workflow (Cardoso, Miller *et al.* 2002).

For example, in our initial scenario, the tasks *Conference Registry*, *Get User Information*, and *Hotel Reservation* are grounded tasks. The GT *Conference Registry* has the following structure:

GT ("Get Conference Information") = <{time.max = 50, reliability.avg = 0.95, cost.max = 12.4, cost.max = 21.5}, {"Conference"}, {"Date", "Duration", "City"}>

Please note that the inputs and outputs in this example are associated with ontological concepts.

Service Template

When a designer needs to search for a Web service to be integrated into an e-workflow, a service template (ST) is created. A service template represents the intent of the designer to extend the functionality of an e-workflow, bringing the process closer to its ultimate goal. STs do not have a realization associated with them; they represent a structure or blueprint that the designer uses to indicate the characteristics of the Web service that is needed. A ST is specified as:

ST = <sn, sd, QoS, Os, Is>

Five fields exist: *sn*, *sd*, *QoS*, *Os*, and *Is*. The *sn* variable corresponds to the name of the Web service to be found. We will see later that the name specified does not have to syntactically match exactly with the name of the Web services to be discovered. The *sd*, *qos*, *Os*, and *Is* fields correspond to a textual description, the operational metrics, and a set of output and input parameters, respectively, of the Web service to be found.

The set of output parameters corresponds to the set of the output parameters of the tasks connected to a ST, and the set of input parameters corresponds to the set of the input parameters of the tasks the ST will be connected to. Lets us indicate the GTs to be connected to a ST with the symbol >st, and the GTs that the ST connects to with st<. Then,

$$Os = \bigcup_{gt \in st>} output(gt), Is = \bigcup_{gt \in <st} input(gt)$$

For example, our scenario contains one service template, the *Travel Reservation* template (represented by a dotted circle in Figure 4-2) that holds the following information:

ST = < "Travel_Agency", "An travel agent service that provides flight reservations based on the specification of a flight request", {cost.max=50, time.avg=5},{" Date", "Duration", "City"} ∪ {"User Name", "Address"}, {"Itinerary"}>

Service Object

The service object is a structure that holds the description of a real Web service. As stated earlier, we specified Web services semantically. A SO is formally described as follows:

SO = <sn, sd, QoS, Is, Os>

The structure is composed of five concepts: *sn*, *sd*, *QoS*, *Is*, and *Os*. The fields of a SO have the same meaning as the ones defined in a ST. This makes sense because SOs will be matched against STs.

4.6 MATCHING ST AND SO

The Web service discovery and integration process is carried out by a key operation: the match function. The matching step is dedicated to finding correspondences between a service template and a service object. During the discovery phase, the match function is employed to successively match a ST against a set of SOs, which are possibly advertised in a registry (*e.g.* UDDI). The SOs are ranked based on their degree of similarity and integration with the ST. The user may then select the Web service with the highest degree of similarity and manually solve the schematic differences not already solved by the system.

We have constructed a system which implements the following idea. Given a service template and a set of service objects, the system examines the services and tries to find similarities between a ST and each SO. This is done using syntactic, operational, and semantic information as a way to increase the precision of the match. The system (1) evaluates the degree of similarity between a ST and a SO and (2) provides the means for the interoperability of services through the analysis and suggestion of connections between the SO interfaces that maximize the degree of integration with the ST.

Syntactic Similarity: The syntactic similarity of a ST and a SO is based on their *service names* and *service descriptions*. At this stage, only syntactic information is taken into account, since both fields are simply expressed using a set of words, without attaching any tag of concepts to each one.

Operational Similarity: Syntactic and semantic information allows for the selection of Web services based on their functionality, but without accounting for operational metrics. The operational similarity of a ST and a SO is calculated based on the metrics specified

in their QoS model. The purpose is to determine how close two Web services are, as based on their operational capabilities.

Semantic Similarity: Purely syntactical methods that treat terms in isolation from their contexts are insufficient since they deal with syntactic but not with semantic correspondences, and since users may express the same concept in different ways (Sheth and Kashyap 1992; Lee, Kim *et al.* 1993). Therefore, we rely on semantic information to evaluate the similarity of concepts and properties that define the ST and SO interface. This evaluation will be used to calculate their degree of integration.

4.6.1 SYNTACTIC SIMILARITY FUNCTION

The syntactic similarity of a ST and a SO is calculated with the function *SynSimilarity(ST, SO)*. The similarity computation relies on the *SynNS(ST, SO)* and *SynDS(ST, SO)* functions, and the weights ω_1 and ω_2. The functions *SynNS* and *SynDS* are binary functions that compute the degree of similarity between two service names, and two service descriptions, respectively. The computation is based only on syntactical considerations, and no semantic information is taken into account at this time. Both functions return a real value between 0 and 1, indicating the degree of syntactic similarity. The weights ω_1 and ω_2 are real values between 0 and 1; they indicate the degree of confidence that the designer has in the service name and service description he supplied when constructing a ST.

$$SynSimilarty(ST, SO) = \frac{w_1 SynNS(ST.sn, SO.sn) + w_2 SynDS(ST.sd, SO.sd)}{w_1 + w_2} \in [0..1],$$

$$\text{and } w_1, w_2 \in [0..1]$$

High weight values indicate the designer's confidence in the supplied information. For example, let consider that a user is searching for a service and supplies the service name "Travel Agency" and a service description "Accepts a quote request for air travel." The user his allowed the association of a weight with the service name and with the service description. If the user is not confident about the service description given, the weight ω_2 can be set to a low value, for example 0.20. If the user is certain of the service name given, the weight ω_1 can be set to 0.8. Please note that sum of the weights does not have to add up to 1.

It is not realistic to expect that the majority of users will understand the relationship between information confidence and weighting. In view of the fact that humans often feel awkward in handling and interpreting such quantitative values (Tversky and Kahneman 1974), we have constructed a mapping table that establishes a correspondence between quantitative values and a qualitative scale (Miles and Huberman 1994). Thus, instead of explicitly specifying quantitative values, the designer can optionally select qualitative terms. An example of a mapping table (which can be customized) is expressed in Table 4-2.

Table 4-2 – Confidence Mapping Table

Qualitative	Quantitative
Uncertain	[0.0..0.2]
Hesitant	[0.2..0.4]
Optimistic	[0.4..0.6]
Confident	[0.6..0.8]
Certain	[0.8..1.0]

Several methods can be employed to match service names and descriptions. The similarity of names can be defined and measured in various ways, including equality of name, equality of canonical name representations after stemming and other preprocessing, equality of synonyms, similarity of names based on common sub-strings, pronunciation, and soundex. Service descriptions contain comments in natural language that express the intended semantics of a service. These comments can be evaluated linguistically to determine the similarity between services. The linguistic analysis can be as simple as extracting keywords from the descriptions which are used for synonym comparison, much like names, or it could be as sophisticated as using natural language-understanding technology to look for semantically equivalent expressions.

In our approach, we use "string-matching" as a way to calculate how closely service names and service descriptions resemble each other. The functions $SynNS(n_1, n_2)$ and $SynDS(d_1, d_2)$ evaluate syntactic similarity by considering the number of *q-grams* (Zamora, Pollock *et al.* 1981; Angell, Freund *et al.* 1983; Salton 1988) that their arguments have in common. To achieve a better comparison between two service descriptions we pre-process the descriptions. A common stop list is applied to remove common words with no information value such as "and" and "of" (Fox 1992); words are

also reduced to their stem by removing prefixes and suffixes (Porter 1980), and duplicates are eliminated. Table 4-3 shows the results of two examples of calculating how close two Web service names are.

Table 4-3 – Comparing Web service names

Service Name A	Service Name B	Result
“The Travel Agency”	“Travel Agent”	0.87
“The Travel Agency”	“ An Internet Travel Agent”	0.63

We are not so much interested in introducing a clever function for syntactic similarity, since our work focus on operational similarity, and on semantic similarity and integration, as in showing the importance of considering syntactic information during Web service discovery.

Another popular algorithm that may be considered to compare service names is the edit distance formulated by Levenshtein (1966). For the service description comparison, techniques borrowed from the information retrieval area may also be considered. For example, the frequency-inverse document frequency (Salton 1988) weighting (TF-IDF) has been used in the LARKS system (Sycara, Lu *et al.* 1998) to match heterogeneous agents on the Internet. A very good source of information retrieval techniques can be found in Belew (2000). There is some evidence that combining different ranking methods to yield a new method can improve performance, possibly through capturing the best of the different methods (Losee 1988; Hull, Pedersen *et al.* 1996).

4.6.2 OPERATIONAL SIMILARITY FUNCTION

The operational similarity of a ST and a SO is calculated with the function *OpSimilarity(ST, SO)*. The binary function *OpSimilarity* computes the geometric distance of the QoS dimensions specified in the ST and the ones specified in the SO. The function returns a real value between 0 and 1, indicating the similarity of the operational metrics of its arguments. The closer to the value 1 the result is, the more similar a SO is to a ST.

$$\text{OpSimilarity}(ST, SO) = \sqrt[3]{\text{QoSdimD}(ST, SO, time) * \text{QoSdimD}(ST, SO, cost) * \text{QoSdimD}(ST, SO, reliability)}$$

The distance of two QoS dimensions is calculated using function QoSdimD(*ST, SO, dim*), where *dim* is a dimension. The function calculates the geometric distance of the distance of the individual components making up the dimension *dim* (*i.e.*, the minimum, average, and maximum value the dimension can take) of the ST and of the SO. The distance of two dimension components is called the dimension component distance (*dcd*).

$$\text{QoSdimD}(ST, SO, dim) = \sqrt[3]{\text{dcd}_{\min}(ST, SO, dim) * \text{dcd}_{\text{avg}}(ST, SO, dim) * \text{dcd}_{\max}(ST, SO, dim)}$$

Three *dcd* functions exist: $\text{dcd}_{\min}(ST, SO, dim)$, $\text{dcd}_{\text{avg}}(ST, SO, dim)$, and $\text{dcd}_{\max}(ST, SO, dim)$. The $\text{dcd}_{\min}(ST, SO, dim)$ is defined as follows:

$$\text{dcd}_{\min}(ST, SO, dim) = 1 - \frac{|\min(SO.qos(dim)) - \min(ST.qos(dim))|}{\min(ST.qos(dim))}$$

The definition of the other two functions is similar; the symbol "min" should be replaced with "avg" or "max". The functions min, avg, and max return the minimum, average, and maximum, respectively, of the QoS dimension specified in the argument.

Table 4-4 shows an example of how to compute the distance of two QoS dimensions for the time dimension. The metrics shown are from the task *Prepare Sample* from a genomics process (Cardoso, Miller *et al.* 2002). The results indicate a high similarity between the time dimension metrics of the ST and of the SO.

Table 4-4 – Example on how to calculate the QoS distance for the time dimension

	Min	**Avg**	**Max**
ST	190	197	199
SO	192	196	199
dcd_x(ST, SO, *time*)	$1-\frac{\lvert 192-190\rvert}{190}$	$1-\frac{\lvert 196-197\rvert}{197}$	$1-\frac{\lvert 199-199\rvert}{199}$
QoSDimD(ST, SO, *time*)	$\sqrt[3]{\frac{188}{190}*\frac{196}{197}*1}=0.99$		

4.6.3 SEMANTIC INTEGRATION

Web service integration differs from previous work on information integration due to the number of services involved, the potential number of ontologies employed to describe service interfaces, and the polarity of input/output schema. The polarity of schema forces output schema to be connected to input schema. Furthermore, an input schema needs to have all its input parameters satisfied. This is not required for an output schema. Solutions involving a semiautomatic integration, requiring user input that defines

similarities between terms or semantic interrelations (Hammer, McLeod *et al.* 1994; Kashyap and Sheth 1996; Bergamaschi, Castano *et al.* 1998) are not adequate for the Web service integration problem. It is not realistic to expect the user to provide information about potential mappings or semantic interrelations among terms for each Web service object present in a registry. We desire to develop a mechanism that automatically computes the similarity of two services, efficiently and without human intervention, and that suggests potential mappings between a ST and a SO schema which maximize their degree of integration, thus reducing structural and semantic heterogeneity.

We now present our algorithm to compute the degree of integration of a ST and a SO. This function bases its computation on the input and output parameters of both the ST and the SO.

4.6.3.1 SEMANTIC INTEGRATION FUNCTION

The semantic integration function *DIntegration(ST, SO)* is a binary function that returns the degree of integration between its operators. The operands are a service template (ST) and a service object (SO), and the result is a real value between 0 and 1.

$$DIntegration(\mathrm{ST}, \mathrm{SO}) \in [0..1]$$

The underlying goal of the function is to establish a mapping between the output of the ST (*ST.O*) and the input of the SO (*SO.I*) and a mapping between the output of the SO (*SO.O*) and the input of the ST (*ST.I*) that maximize the degree of integration.

Depending on the data present in a service template, four distinct cases can occur when comparing input and output parameters. The definition of the function *DIntegration* captures these four cases.

$$DIntegration(ST, SO) = \begin{cases} \dfrac{\dfrac{\Pi(ST.Os, SO.Is)}{|SO.Is|} + \dfrac{\Pi(SO.Os, ST.Is)}{|ST.Is|}}{2}, & ST.Os \neq \varnothing, ST.Is \neq \varnothing \\ \Pi(ST.Os, SO.Is)/|SO.Is|, & ST.Os \neq \varnothing, ST.Is = \varnothing \\ \Pi(SO.Os, ST.Is)/|ST.Is|, & ST.Os = \varnothing, ST.Is \neq \varnothing \\ 0, & ST.Os = \varnothing, ST.Is = \varnothing \end{cases}$$

The simplest case occurs when a ST does not specify any inputs or outputs. In this case, the integration degree is evaluated to 0. If a ST only specifies a set of outputs and no inputs, then the function Π(*Os, Is*) is employed to compute the semantic mapping between the outputs *Os* of the ST and the inputs *Is* of the SO. The result of applying the function Π is normalized with respect to the number of inputs being mapped. The rationality of this normalization is that when matching *n* outputs of a task *a* against *m* inputs of a task *b*, we are interested in satisfying all the input requirements of task *b*. A task or Web service always needs to have its mandatory inputs satisfied with data in order to correctly carry out its intended function. Optional inputs are not taken into account. Nevertheless, a designer may explicitly mark an optional input as mandatory if he wishes optional inputs to be considered during the integration evaluation process. The same concept is applied if the ST includes inputs but no outputs.

Finally, if a ST includes both a set of outputs and a set of inputs the mapping function Π is applied to both sets. In this case, we compute the arithmetic mean of the normalized results from the evaluation of function Π. We use the arithmetic mean because we give the same importance to the normalized semantic mapping of the ST outputs with the SO inputs and the normalized semantic mapping between SO outputs with ST inputs.

4.6.3.2 MAPPING INPUTS AND OUTPUTS

The function Π(*Os, Is*), where *Os* is a set of output parameters and *Is* a set of input parameters, computes the best mapping that can be obtained from connecting the outputs of the set *Os* to the inputs of set *Is*.

$$\Pi(Os, Is) = \begin{cases} Max(\Pi(Os - O, Is - I) + \mathrm{p}(O, I)), & Os \neq \varnothing, Is \neq \varnothing, O \in Os, I \in Is \\ 0, & Os = \varnothing \vee Is = \varnothing \end{cases}$$

Please note that the number of mappings established is *Min(|Os|, |Is|)*. Each output *O* of *Os* is matched against each input *I* of *Is*. Their semantic similarity degree is evaluated with function π(*O, I*). Since input/output parameters are associated with ontological concepts (see section 4.4.2), the function π(*O, I*) compares two concept classes represented by *O* and *I*.

$$\mathrm{p}(O, I) = \begin{cases} SemS'(O, I), & \Omega(O) = \Omega(I) \\ SemS''(O, I) / \mid p(I) \mid, & \Omega(O) \neq \Omega(I) \end{cases}$$

The function π(*O, I*) takes into consideration the ontology(ies) associated with the concepts being compared. If the concepts are from the same ontology, *i.e.*, Ω(*O*) = Ω(*I*), the function *SemS'*(*O, I*) is employed to evaluate their similarity; otherwise, if they are from distinct ontologies, *i.e.*, Ω(*O*) ≠ Ω(*I*), the function *SemS''*(*O, I*) is used. We make this distinction since the information available when comparing concept classes from the same ontology has a different nature and structure which is not present when comparing concepts from distinct ontologies. The result of function *SemS''* is normalized with respect to the number of properties of the input concept *I*. As we will see, the evaluation of the similarity of two concepts is based on their composing properties. Once again, the

reason for this normalization is to obtain a measure that reflects the fact that all the mandatory input properties need to have an output property associated with them in order for a task or Web service to work properly.

4.6.3.3 COMPARING OUTPUTS AND INPUTS FROM THE SAME ONTOLOGY

The function *SemS'(O, I)* evaluates the similarity of two concept classes associated with an output (*O*) and an input (*I*), conceptualized within the same ontology. Four distinct scenarios can occur: a) the concepts are the same (*O*=*I*), b) the concept *I* subsumes concept *O* (*O*>*I*), c) the concept *O* subsumes concept *I* (*O*<*I*), or d) concept *O* is not directly related to concept *I* (*O*≠*I*). In the latter case, the concept *O* does not have a parent/child relationship with concept *I*, but both concepts have a parent concept in common.

$$SemS'(O,I) = \begin{cases} 1, & O = I \\ 1, & O > I \\ \dfrac{|\,p(O)\,|}{|\,p(I)\,|}, & O < I \\ Similarity'(O,I), & O \neq I \end{cases}$$

In the first case, which is the simplest, if the two concepts are equal then intuitively their similarity is maximal; therefore, it is evaluated to one. In the second case, if the concept *I* subsumes the concept *O*, their similarity is also evaluated to 1. The similarity is maximal since if an output represented with a concept *O* is a subclass of an input represented with a concept *I* it has at least the same set of properties as *I*. Thus, all input properties have a corresponding output property associated with them. In the third case, the concept *O* subsumes the concept *I* (*O*<*I*). As a result, some properties of the concept *I* may not have an output property associated with them. The similarity is set to the ratio of the number of properties of concept *O* (represented with |*p*(*O*)|) and the number of

properties of concept I ($|p(I)|$). This ratio indicates the percentage of input properties of the SO that are satisfied by output properties of the ST.

In the last case, the concepts O and I are not equal and do not subsume each other in any way. In this case, assessing similarity is a judgment process that requires two "things" to be decomposed into elements in which they are the same and into elements in which they are different (Tversky 1977). Assessing the similarity of concepts is an important process for systems such as information retrieval and information integration. A number of approaches to measuring conceptual similarity between words have been taken in the past. Tversky's feature-based similarity model (Tversky 1977) has been considered as the most powerful similarity model to date (Richardson and Smeaton 1995).

Tversky introduced a general feature-counting metric for similarity called the feature-contrast model. This model is based on the idea that common features tend to increase the perceived similarity of two concepts, while feature differences tend to diminish perceived similarity. Tversky's model claims that feature commonalities tend to increase perceived similarity more than feature differences can diminish it. That is, when assessing similarity we give more credence to those features that concepts have in common than to those that distinguish them. For instance, a SUV (Sport Utility Vehicle) and a sedan are similar by virtue of their common features, such as wheels, engine, steering wheel, and gears, and are dissimilar by virtue of their differences, namely height and the size of the tires. Based on Tversky's model, we introduce a similarity function based on the number of properties shared among two concepts c_1 and c_2. Our similarity function is defined as followed, where the function $p(x)$ retrieves all the properties associated with a concept a and function $|s|$ corresponds to the number of elements in the set s.

$$similarity'(O,I) = \sqrt{\frac{|\ p(O) \cap p(I)\ |}{|\ p(O) \cup p(I)\ |} * \frac{|\ p(O) \cap p(I)\ |}{|\ p(I)\ |}}$$

The *similarity'*(*O*,*I*) function computes the geometric distance between the similarity of the domains of concept *O* and concept *I* and the ratio of matched input properties from the concept *I*.

As an example, let us illustrate the use of function *SemS'(O, I)* for the four cases – a), b), c) and d) – that can occur when connecting an output *O* to an input *I* (see Figure 4-6). In our example, both input and output are conceptualized with concepts from the same ontology, *i.e.*, Ω(*O*) = Ω(*I*) = Time ontology (an example using difference ontologies is given in the next section). The time ontology is not fully represented in Figure 4-6; only the concepts that are employed in our example are shown.

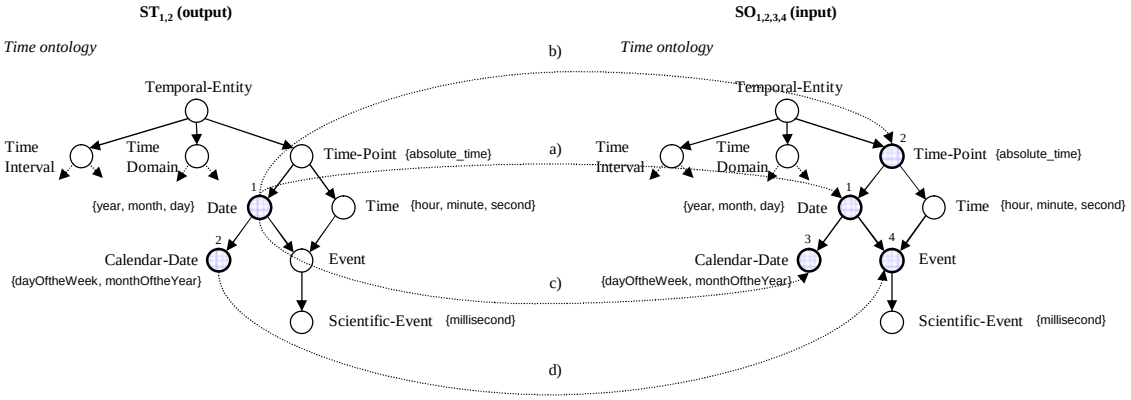

Figure 4-6 – Comparing concepts from the same ontology

Table 4-5 – The four examples illustrated in Figure 4-6.

	Service Template	**Output**		**Service Object**	**Input**
a)	ST_1	Date (1)	→	SO_1	Date (1)
b)	ST_1	Date (1)	→	SO_2	Time-Point (2)
c)	ST_1	Date (1)	→	SO_3	Calendar-Date (3)
d)	ST_2	Calendar-Date (2)	→	SO_4	Event (4)

The four cases that may occur are listed in Table 4-5 and are evaluated as follows:

- In case a), both *O* and *I* are associated with the same concept (*Date*). This is the simplest case. Since the output of the ST_1 matches perfectly the input of the SO_1 the similarity is evaluated to 1.
- In case b), the output *O* is associated with the concept *Date*, and the input *I* is associated with the concept *Time-Point*. Since the concept *Time-Point* subsumes the concept *Date*, the properties of the concept *Date* (the set {absolute_time, year, month, day}) is a superset of the properties of the concept *Time-Point* (the set {absolute_time}). Therefore, the output *O* of the ST_1 can be connected to the input *I* of the SO_2 without any property of *I* being left unfulfilled; there is a direct semantic correspondence and value mapping. All the properties of *I* exist in *O*. As a result, the similarity is evaluated to 1.
- In case c), the output *O* is associated with the concept *Date* and the input *I* is associated with the concept *Calendar-Date*. Since the concept *Date* subsumes concept *Calendar-Date*, the properties of the concept *Date* (the set {absolute_time, year, month, day}) is a subset of the properties of the concept

Calendar-Date (the set {dayOftheWeek, monthOftheYear, absolute_time, year, month, day}). In this case, when the output *O* is connected to the input *I* some properties of *I* are left unfulfilled (the properties dayOftheWeek and monthOftheYear). To indicate this mismatch the similarity is set to the ratio of the number of properties of *O* and the number of properties of *I*, which in this case is $|p(O)|/|p(I)| = 4/6 \approx 0.67$.

- In the last case (d), the output *O* of the ST_2 is associated with the concept *Calendar-Date* and the input *I* of the SO_4 is associated with the concept *Event*. The concept *Event* has the set of properties {absolute_time, year, month, day, hour, minute, second} and the concept *Calendar-Date* has the set of properties {dayOftheWeek, monthOftheYear, absolute_time, year, month, day}. Since the concepts do not have a parent/children relationship, the function *similarity'*(*O*,*I*) is used to compute the geometric distance between the similarity of the domains of concept *Calendar-Date* and concept *Event* and the percentage of input properties that are fulfilled with an output property from *O*. The similarity is evaluated as follows:

$s_1 = p(CalendarDate) =$ {dayOftheWeek, monthOftheYear, absolute_time, year, month, day}

$s_2 = p(Event) =$ {absolute_time, year, month, day, hour, minute, second}

$s_3 = p(CalendarDate) \cap p(Event) =$ {absolute_time, year, month, day}

$s_4 = p(CalendarDate) \cup p(Event) =$ {dayOftheWeek, monthOftheYear, absolute_time, year, month, day, hour, minute, second}

$$similarity'(CalendarDate, Event) = \sqrt{\frac{|s_3|}{|s_4|} * \frac{|s_3|}{|s_2|}} = \sqrt{\frac{4}{9} * \frac{4}{7}} \approx 0.504$$

The result of evaluating the function *similarity'*(*Calendar-Date*, *Event*) indicates a low degree of integration between the concepts *Calendar-Date* and *Event*. On one hand, the concepts show a low similarity according to the feature-contrast model (4/9). On the other hand, only four out of the seven input properties are connected to output properties.

4.6.3.4 COMPARING OUTPUTS AND INPUTS FROM DISTINCT ONTOLOGIES

The problem of determining the similarity of concepts defined in different ontologies is related to the work on multi-ontology information system integration. When the input and output concepts to compare are from distinct ontologies, the evaluation of their similarity is more complex. First, our approach for this problem uses the same rationale that we have exploited earlier to compare input and output concepts from the same ontology without any parent/child relationship. Additionally, we also take into account syntactic similarities among concepts.

Since we compare input and output concept classes based on their properties, the first step is to find the best mapping between output and input concept properties. This objective is achieved using the function *SemS''*(*O, I*), which is very similar to function Π*(Os, Is)* previously defined as being able to find the best mapping between a set of outputs and a set of inputs.

$$SemS''(O,I) = \begin{cases} Max(SemS''(O-o, I-i) + S(o,i)), & O \neq \emptyset, I \neq \emptyset, o \in O, i \in I \\ 0, & O = \emptyset \vee I = \emptyset \end{cases}$$

Each property *o* of output *O* is mapped with a property *i* of input *I*. A property *o* is associated with a property *i* that maximizes the semantic similarity computed, using the function *S(o, i)*.

The function $S(o, i)$ calculates the similarity between a property o and a property i. Three distinct cases are considered: (1) the ontological properties involved are associated with a primitive data type (see section 4.4.2), (2) the properties are associated with concept classes, and (3) one property is associated with a primitive data type, while the other is associated with a concept class. The function $S(o, i)$ is shown below.

$$S(o,i) = \begin{cases} \sqrt[3]{SemDS(d(o),d(i)) * SynS(n(o),n(i)) * SemRS(r(o),r(i))}, & o \text{ and } i \text{ are primitive types} \\ SemDS(o,i), & o \text{ and } i \text{ are concept classes} \\ f(o,i), & \text{otherwise} \end{cases}$$

In the first case, the similarity of the properties is computed based on the geometric distance of (a) the semantic similarity of their domains (*i.e.*, concept classes), (b) the syntactic similarity of their names, and (c) the semantic similarity of their ranges.

a). The semantic similarity of the domains of two properties, $d(o)$ and $d(i)$, is evaluated using function *SemDS(od, id)*, which is based on Tversky's model.

$$SemDS(od,id) = \frac{|\ p(od) \cap p(id)\ |}{|\ p(od) \cup p(id)\ |}$$

When calculating the intersection of sets $p(od)$ and $p(id)$, two elements intersect if their syntactic similarity, using the *q-grams* methodology (see section 4.6.1), is greater than a constant c (we are currently using $c = 0.75$).

b). The syntactic similarity of property names is calculated using the function $SynS(n_1, n_2)$. This function uses *q-grams* to determine the similarity of two property names.

c). The semantic similarity of the ranges of two properties, $r(o)$ and $r(i)$, is evaluated using the function $SemRS(r(o), r(i))$ defined below.

$$SemRS(or, ir) = \begin{cases} 1, & or = ir \\ 1, & or = \text{integer}, ir = string \\ 2/3, & or = \text{long}, ir = \text{integer} \\ 1/3, & or = \text{double}, ir = \text{integer} \\ 1, & or = \text{integer}, ir = \text{long} \\ 0, & \text{otherwise} \end{cases}$$

The function *SemRS*(*or, ir*) indicates the validity and the integration degree that is obtained when an output with a primitive data type dt_a is connected to a particular input with a primitive data type dt_b. This function is automatically created based on the capabilities of the WfMS where the e-workflow being constructed will be enacted. A workflow system that has the competence of making data type conversions (*i.e.*, converting one data type into another) on the data exchanged among tasks can formally define and describe this ability with the customization of function *SemRS*.

For example, if a WfMS can map an output property of task *a*, with range *integer*, to an input property of task *b*, of range *long*, this can be indicated by adding the following entry to function *SemRS*:

1, *or*=integer and *ir*=long

The similarity is maximal, and it is set to 1, since the WfMS can map an *integer* data type to a *long*. When an association between two data types is not valid, the function *SemRS* returns 0. In other situations, it is possible to specify a fuzzy degree of integration

by setting the similarity to a value greater than zero and less than one. For example, let us consider the following entry:

1/3, *or*=double and *ir*=integer

In this case, the WfMS is able to perform a specific data type conversion (*double* to *integer*), but the conversion is not preferred or recommended since a loss of information may occur.

In the second case (2) of function $S(o, i)$, since o and i are concept classes, we use the function $SemDS(o, i)$ to compute their similarity. The function *SemDS* evaluates the similarity of two concept classes only in a shallow fashion. An alternative is to use a deep-based similarity function (*i.e.*, recursively compare subclasses). This can be achieved by substituting the function $SemDS(o, i)$ present in function $S(o, i)$ with the function $SemS''(od, id)/|p(id)|$.

In the third case (3), function $f(o, i)$ is used to calculate the similarity among a property associated with a basic data type and a property associated with a data class. For the definition of this function we rely on the concept of dynamic attributes that has been proposed in (Litwin and Abdellatif 1986) to specify the mappings between different attributes. The idea is to define a function or a set of functions that indiate the possible mappings between a property and a concept class. Examples of such mappings can be found in (Kashyap and Sheth 1993).

Let us illustrate the use of functions $SemS''(O, I)$ and $S(o, i)$ with the example shown in Figure 4-7.

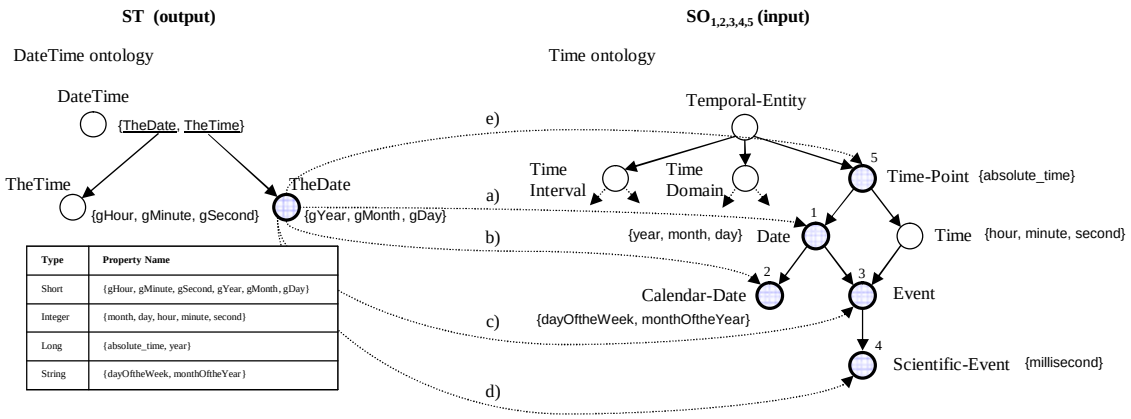

Figure 4-7 – Comparing proprieties referencing primitive data types

To makes the example easier to understand, the ST employed to find a SO only specifies a set of outputs, with no inputs. Furthermore, we carry out the computation of function *SemS''*(*O, I*) for only one of the outputs of the ST (the *TheDate* parameter) and for only one of the SO inputs (the inputs are represented with the indexes 1 through 5 in Figure 4-7). We consider that five SOs ($SO_{1, 2, 3, 4, and\ 5}$) are present in the registry during the discovery procedure. The five cases are shown in Table 4-6.

Table 4-6 – The five examples illustrated in Figure 4-7.

	Service Template	**Output**		**Object Template**	**Input**
a)	ST	TheDate	→	SO_1	Date
b)	ST	TheDate	→	SO_2	Calendar-Date
c)	ST	TheDate	→	SO_3	Event
d)	ST	TheDate	→	SO_4	Scientific-Event
e)	ST	TheDate	→	SO_5	Time-Point

The SO_1 input is associated with the class concept *Date*. The SO_2 input is associated with the class concept *Calendar-Date*. The SO_3 input is associated with the class concept *Event*. Finally, the SO_4 and SO_5 inputs are associated with the concept class *Scientific-Event* and *Time-Point*, respectively.

During the discovery process, the ST is compared with each SO individually. Therefore, the function *SemS''*(*O, I*) is applied five times. In Figure 4-7, the computation of the function between the output of a ST and the input of a $SO_{1..5}$ is represented with a letter (a, b, c, d, or e).

Let us start with the computation of function *SemS''*(*O, I*) to evaluate the degree of integration of the concept class *TheDate* (from the *DateTime* ontology, *i.e.*, the concept Ω(DateTime).TheDate) and the concept class *Calendar-Date* (from the *Time* ontology, *i.e.*, the concept Ω(Time).Calendar-Date). Figure 4-8 shows the mappings carried out by function *SemS''*(*TheDate, Calendar-Date*).

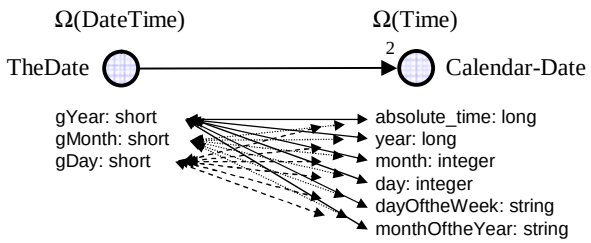

Figure 4-8 – Evaluating the degree of integration

For each connection shown in Figure 4-8, function *S*(*o, i*) is called on to evaluate the degree of integration among two properties. Since in our example the output and input properties of the concept classes *O* and *I* reference primitive data types, function *S* will uniquely use the case (1) described previously. This corresponds to the use of the following function:

$$\sqrt[3]{SemDS(d(o),d(i)) * SynS(n(o),n(i)) * SemRS(r(o),r(i))}$$

Let us trace the computation of *S*(*o, i*) with *o* = ”*gDay*” and *i* = ”*day*”. The function *SemDS* evaluates the similarity of the domains (concept classes) of properties *o* and *i*. The properties ”*gDay*” and ”*day*” have the domain concepts *TheDate* and *Calendar-Date*, respectively, *i.e.*, *d*(*gDay*) = *TheDate* and *d*(*day*) = *Calendar-Date*. Therefore, *SemDS*(*TheDate, Calendar-Date*) is evaluated the following way:

$$p(TheDate) = \{\text{gMonth, gYear, gDay}\}$$
$$p(Calendar\text{-}Date) = \{\text{absolute\_time, year, month, day, dayOfTheWeek, monthOfTheYear}\}$$
$$SemDS(TheDate, Calendar\text{-}Date) = \frac{|\ p(TheDate)\ \cap\ p(Calendar-Date)\ |}{|\ p(TheDate)\ \cup\ p(Calendar-Date)\ |} = 0.5$$

This result, 0.5, indicates that the domains of properties *o* and *i* are somewhat similar, which follows our perception that the concepts *TheDate* and *Calender-Date* are similar.

The second function to be evaluated is *SynS*(*no, ni*). This function computes the syntactic similarity of the property names *no* and *ni*. In our example, the similarity of properties *gDay* and *day* is evaluated to 0.8. This result indicates a close syntactic similarity between the two property names. Other examples of the application of the function *SynS* for the properties involved in our example are:

$$SynS(gDay, dayOfTheWeek) = 0.29$$

$$SynS(gMonth, monthOfTheYear) = 0.44$$

The last function to be evaluated is function *SemRS*(*r*(*o*), *r*(*i*)), which calculates the similarity of the ranges of properties *o* and *i*. For the properties *gDay* and *day*, the following metric is obtained:

$$SemRS(r(gDay), r(day)) = SemRS(short, integer) = 1.0$$

This result indicates that the workflow system that will be enacting the process being constructed supports the connection of parameters of type *short* to parameters of type *integer*. An example of a connection among properties not supported or desired is the following one:

$$SemRS(r(gDay), r(dayOfTheWeek)) = SemRS(short, string) = 0.0$$

Having calculated the functions *SemDS*, *SynS*, and *SemRS*, we can now compute function *S*. The result of evaluating *S*(*gDay, day*) is,

$$\sqrt[3]{0.5 * 0.8 * 1.0} = 0.74$$

Table 4-7 shows the results of applying function *S*(*o*, *i*) to various properties of the concept classes *TheDate* and *Calendar-Date*.

Table 4-7 – Examples of the evaluation of function *S*(*o*, *i*).

o	***i***	***SemDS***	***SynS***	***SemRS***	***S***
gMonth	dayOfTheWeek	0.5	0.12	0.0	0.0
gYear	monthOfTheYear	0.5	0.35	0.0	0.0
gDay	month	0.5	0.0	1.0	0.0
gDay	year	0.5	0.0	1.0	0.0
gDay	day	0.5	0.8	1.0	0.74
gDay	time	0.5	0.0	1.0	0.0
gDay	monthOfTheYear	0.5	0.0	0.0	0.0
gYear	year	0.5	0.86	1.0	0.75
gMonth	monthOfTheYear	0.5	0.44	0.0	0.0
gMonth	month	0.5	0.89	1.0	0.76

Once all the possible mappings between the properties of the output concept class TheDate and the input concept class Calendar-Date are evaluated, the function *SemS''*(*TheDate*, *Calendar-Date*) returns the result shown in Table 4-8 line b). The table also shows the results for all the five cases initially considered in Figure 4-7.

Table 4-8 – Example of computing function *SemS''*(*O,I*).

	ST	**O**	**SO**	**I**	**SemS''(O, I)**
(a)	ST	TheDate	SO_1	Date	2.58
(b)	ST	TheDate	SO_2	Calendar-Date	2.25
(c)	ST	TheDate	SO_3	Event	2.14
(d)	ST	TheDate	SO_4	Scientific-Event	2.05
(e)	ST	TheDate	SO_5	Time-Point	0.00

The function *SemS''*(*O, I*) returns the cumulative degree of similarity of the mappings between two concept classes, *i.e.*, it corresponds to the sum of the weights associated with a bipartite graph constructed with the properties of the output concept and the properties of the input concept. While the function *SemS''*(*O, I*) gives some information relatively to the possible mapping of two input and output parameters, the function does not take into account the number of mandatory inputs that need to be satisfied. Thus, a more significant and useful piece of information is the result obtained from applying function *p*(*O, I*), which normalizes function *SemS''*(*O, I*) with respect to the number of input properties of the concept class *I*. The results of applying function *p*(*O, I*) to our example is shown in Table 4-9.

Table 4-9 – Example of computing function $\pi(O, I)$.

	ST	***O***	**SO**	***I***	p(***O, I***)
(a)	ST	TheDate	SO_1	Date	0.65
(b)	ST	TheDate	SO_2	Calendar-Date	0.38
(c)	ST	TheDate	SO_3	Event	0.31
(d)	ST	TheDate	SO_4	Scientific-Event	0.26
(e)	ST	TheDate	SO_5	Time-Point	0.00

It can be seen that function $p(O, I)$ returns values closer to 1, when the concept classes being compared exhibit a higher degree of similarity. This is the case for the concepts Ω(DateTime).TheDate and Ω(Time).Calendar-Date. When two concepts are not similar the function returns 0, which is the case for the concepts Ω(DateTime).TheDate and Ω(Time).Time-Point.

4.6.3.5 MAPPING OUTPUTS WITH INPUTS

While the algorithm presented does not explicitly show how the mapping between the outputs and inputs of two services which maximize the degree of integration is constructed, this is achieved by keeping track of the best mapping obtained when computing function $\Pi(Os, Is)$ and function $SemS''(O, I)$.

4.7 SYSTEM ARCHITECTURE

The core of our work has already been presented in the previous section, with the description of the algorithm to match a ST against a set of SOs. Therefore, in this section we will only briefly describe the architecture of our prototype. Our system is composed of two main services: registry service and discovery service, as illustrated in Figure 4-9.

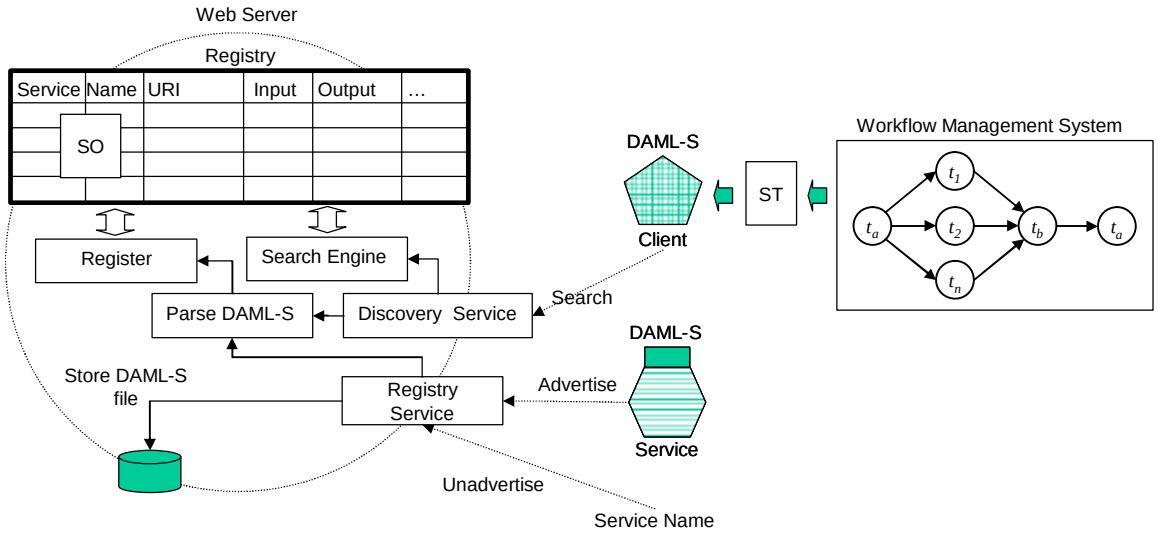

Figure 4-9 – System Architecture

The services available (registry and discovery services) to users and to the WfMS are both implemented using *servlets* and are accessible through HTTP. We are considering extending the access to allow RMI calls.

Suppliers access the registry service to advertise and unadvertise their Web services. To make an advertisement, a supplier registers a DAML-S service object (SO) with the system. To unadvertise a service, the only information necessary is the name of the service.

Clients and customers typically access the system to find Web services previously registered (Figure 4-10). This is achieved by sending a service template (ST) to the system. The service template specifies the requirements about the service to discover. Service templates are described using DAML-S, more precisely by using the *profile.daml* ontology (see section 4.4.3).

Figure 4-10 – Web Service Discovery page.

Once the system receives an advertisement or a discovery message, the SO or the ST received are parsed, using the Jena toolkit (Jena 2002). The Jena DAML API provides support for loading DAML ontologies onto Jena RDF models. Since DAML-S parsers are not yet available, we have built a very simple DAML-S parser on top of the Jena Toolkit to extract specific information from service objects – such as service name, service description, QoS model, inputs, and outputs. The syntactic, operational, and semantic similarity functions will make use of this information.

The information retrieved from parsing a service advertisement is stored in a registry (Figure 4-9). The registry is a service capability table, where service descriptions are added or removed in response to advertised and unadvertised messages. The registry table and its contents are stored in physical memory for fast access.

When the system receives a discovery message (*i.e.*, a ST) from a workflow system for example, it is parsed and matched against the set of SOs registered. The results are ranked according to the criteria specified when the ST was sent to the system (Figure 4-10). Three ranking methods are available, based on: syntactic, semantic, and operational metrics. Better matches are characterized by a score closer to 1. Finally, the ranked candidates are returned to the entity that issued the query. Figure 4-11 shows the results of a query.

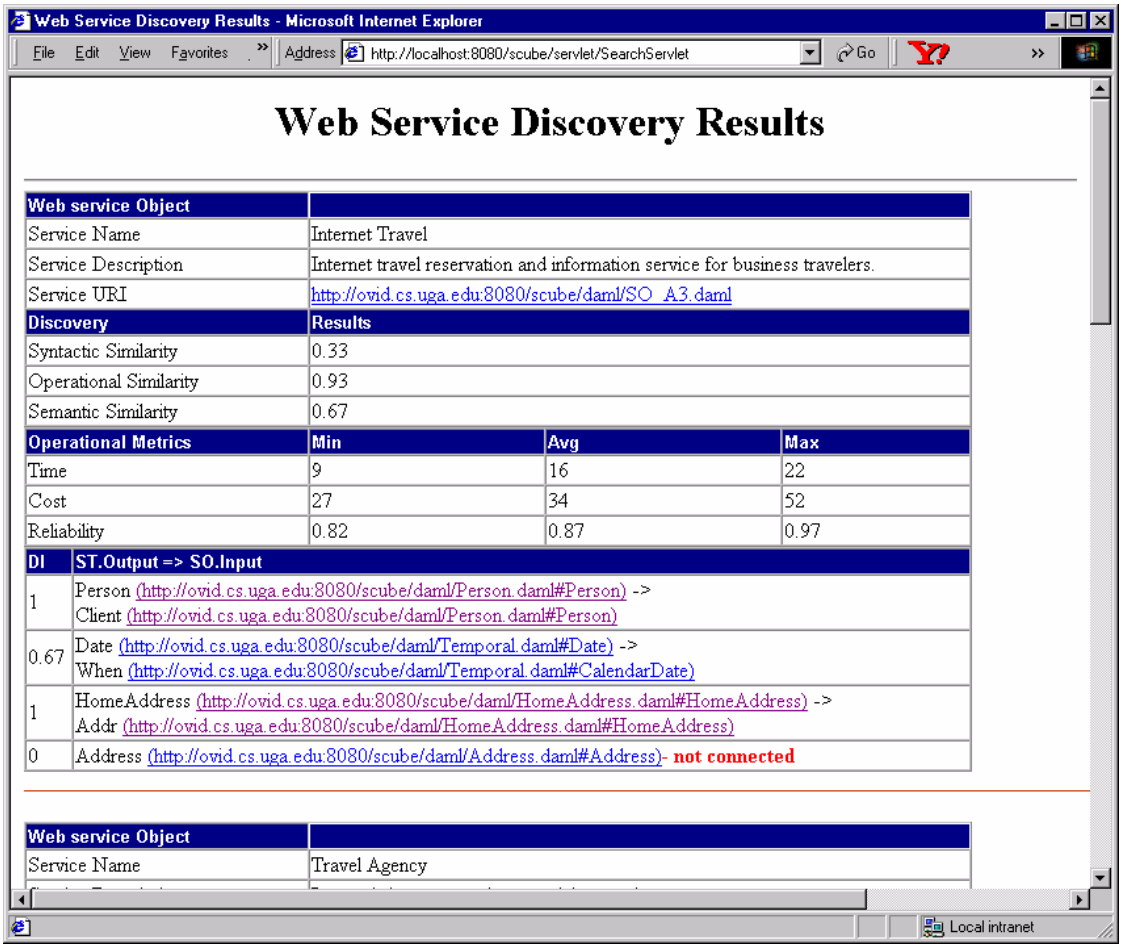

Figure 4-11 – Web Service Discovery Results page

For each SO present in the registry, a detailed information sheet comparing it against the ST is constructed. It includes the results of evaluating the SO against the ST:

syntactically, based on operations, and semantically. Finally, it also includes the suggested data mappings between the ST and the SO (which outputs should be connected to which inputs).

4.8 RELATED WORK

Our work is directly related to ontology-based Web service discovery, search, match, and integration, and indirectly related to information retrieval systems and information integration systems.

The work that most closely relates to ours is described in Paolucci, Kawamura *et al.* (2002). They present an algorithm that deals with the localization of Web services, but they do not address the interoperability problem. Their system also uses the service profile ontology from the DAML-S specification language. Their work considers only the matching of input/output concepts defined by the same ontology. When input/output concepts from different ontologies are matched, the algorithm simply evaluates the match to "fail". This is a limitation. Web services are heterogeneous and autonomous by nature; therefore it is advantageous to compare outputs and inputs that subscribe to different ontologies. The similarity function described is based on the taxonomy of the ontology, accounting for the parent/child relationship between concepts. If the concepts associated with an output and with an input do not have a parent/child relationship the algorithm evaluates the match to a "fail" and does not try to assess a degree of similarity. The algorithm uses the minimal distance between concepts in the taxonomy tree. We believe that a feature-based approach rather than one employing the taxonomy of the ontology achieves better precision in the discovery process. What makes two concepts distinct is not always their distance in a taxonomy tree, but the number of properties in which they are the same and in which they are different. As a last difference, operational metrics of Web services are not taken into account when discovering services.

González-Castillo, Trastour *et al.* (2001) also use DAML+OIL to semantically describe Web services; their approach to the matchmaking of services is based on a subsumption tree. Their algorithm follows a very similar approach to the one taken by Paolucci, Kawamura *et al.* (2002), in the sense that it only accounts for relationships among concepts defined within the same ontology. Their system does not use DAML-S for the description of Web services (the system was developed before its existence). Instead, they have developed their own specification for Web service description, but no notion of inputs and outputs was defined. As a result, the matching of Web services is carried out based on service description, not accounting for inputs and outputs. Their approach does not target the discovery of Web services based on operational metrics, nor does it deal with the Web service integration problem.

Another approach that also uses a specific language to describe service advertisements and requests is the LARKS (Language for Advertisement and Request for Knowledge Sharing) system (Sycara, Klusch *et al.* 1999). The LARKS language can be seen as a precursor of the DAML-S specification. The system uses ontologies defined by a concept language (ITL). Their approach does not provide an automatic solution for the computation of the similarity of concepts defined in distinct ontologies. Furthermore, the technique used to calculate the similarity of ontological concepts involves the construction of a weighted associative network, where the weights indicate the belief in relationships. While they argue that the weights can be set automatically by default, it is clear that the construction of realistically weighted relationships requires human involvement, which becomes a hard task when thousands of agents are available. Their work does not consider the matchmaking of agent-based operational metrics. While the output and input parameters of agents are compared using syntactic and semantic matching methods, the algorithm presented does not provide supply a mapping of potential connections between the outputs and inputs of two agents that yields a maximum degree of integration.

In the information retrieval area, Bejamins and Fensel (1998) present the (KA)2 system, an ontology-based information retrieval system for the World-Wide Web. The system allows a community to build a knowledge base collectively, based on consensual knowledge, by populating a shared ontology. Using the shared ontology, a web-crawler accesses the web pages and uses the ontology to infer answers. The FindUr project (McGuinness 1998) is another initiative in ontology-based information retrieval on the Web. Their work focuses on query expansion and online searching. The use of ontologies has been shown to improve the search from the perspectives of recall and precision, as well as ease of query formation. The OntoSeek (Guarino, Masolo *et al.* 1999) project has also shown that ontologies improve content-based searches. Their work focuses on specific classes of information repositories: yellow pages and product catalogues. Richardson and Smeaton (1995) introduce an approach to information retrieval based on computing a semantic distance measurement between concepts or words and using this word distance to compute the similarity between a query and a document.

Ontologies have been employed as a common basis for information integration. Ontologies allow for the modeling of the semantic structure of individual information sources, as well describing models of a domain that are independent of any particular information source. Several systems have been developed using this solution. Projects include Carnot (Woelk, Cannata *et al.* 1993), InfoSleuth (Bayardo, Bohrer *et al.* 1997), SIMS (Arens, Hsu *et al.* 1996), OBSERVER (Mena, Kashyap *et al.* 1996; Kashyap and Sheth 1998), and COIN (Bressan, Fynn *et al.* 1997). These projects differ from our work in their reduced number of ontologies involved in the integration process and due to their approaches, which require user involvement to manually resolve differences among ontologies. Additionally, a vast amount of the work done is directed to solve schematic differences in multidatabase systems; this does not face the schema polarity problem.

4.9 CONCLUSIONS

In this paper we have presented a set of challenges that the emergence of Web services and e-services has brought to organizations. While in some cases Web services may be utilized in an isolated form, it is normal to expect Web services to be integrated as part of workflows processes. This entails research in two areas. Mechanisms to efficiently discover Web services during an e-workflow (*i.e.*, a workflow managing traditional tasks and Web services) composition process and to facilitate their subsequent integration with the e-workflow host.

We present a methodology and a set of algorithms for Web service discovery based on three dimensions: syntax, operational metrics, and semantics. This approach allows for Web service discovery not only based on functional requirements, but also on operational metrics.

The need to discover workflow components based on operational metrics has a greater importance when Web services are involved, as compared to workflow tasks. The autonomy of Web services does not allow for users to identify their operational metrics at design time, *i.e.*, before their actual execution. The development of mechanisms for the discovery of Web services based on operational metrics allows organizations to translate their vision into their business processes more efficiently, since e-workflows can be designed according to QoS requirements, goals, and objectives.

To facilitate the discovery and posteriori integration of Web service into workflows we propose an approach based on the use of ontologies to describe workflow tasks and Web service interfaces. Ontology-based approaches have already proved to be an important solution to information integration in order to achieve interoperability. During an e-workflow composition, there is a loss of semantics associated with Web service task interfaces because a large part of the domain knowledge a developer employs when deploying a Web service is not present at composition time.

In our work we have devised an algorithm and implemented a prototype to discover and facilitate the resolution of structural and semantic differences during the integration process with an e-workflow. The algorithm uses a feature-based model to find similarities across workflow tasks and Web service interfaces. The system determines and evaluates the best mapping between the outputs and inputs of a SO and the workflow host that yields the highest degree of integration.

We would like to acknowledge Abhijit Patil, Ruoyan Zhang, and Swapna Oundhakar for developing an earlier version of the prototype presented in this work.

4.10 REFERENCES

Angell, R. C., G. E. Freund and P. Willett (1983). "Automatic spelling correction using a trigram similarity measure." Information Processing and Management **19**(4): 255-161.

Ankolekar, A., M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara and H. Zeng (2001). DAML-S: Semantic Markup for Web Services. Proceedings of the International Semantic Web Working Symposium (SWWS). pp. 39-54.

Arens, Y., C. Hsu and C. A. (1996). Knoblock Query Processing in the SIMS Information Mediator. Menlo Park, CA, AAAI Press.

Arpinar, I. B., J. A. Miller and A. P. Sheth (2001). An Efficient Data Extraction and Storage Utility for XML Documents. Proceedings of 39th Annual ACM Southeast Conference, Athens, GA. pp. 293-295.

Barbar, D., S. Mehrothra and M. Rusinkiewicz (1996). "INCAs: Managing Dynamic Workflows in Distributed Environments." Journal of Database Management **7**(1): 5-15.

Bayardo, R. J., W. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh and D. Woelk (1997). InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM Press, New York. pp. 195-206.

Belew, R. K. (2000). Finding Out About : A Cognitive Perspective on Search Engine Technology and the WWW. Cambridge, U.K, Cambridge University Press.

Benjamins, V., D. Fensel and A. Prez (1998). Knowledge Management through Ontologies. Proceedings of the Second International Conference on Practical Aspects Knowledge Management. pp. 5.1-5.12.

Bergamaschi, B., S. Castano, S. D. C. d. Vermercati, S. Montanari and M. Vicini (1998). An Intelligent Approach to Information Integration. First International Conference on Formal Ontology in Information Systems, Trento, Italy, IOS Press, Amsterdam, The Netherlands. pp. 253-268.

Berners-Lee, T. (2001). Keynote presentation on web services and the future of the web. Software Development Expo 2001 Visionary Keynote, http://www.technetcast.com/tnc_play_stream.html?stream_id=616.

Berners-Lee, T. and M. Fischetti (1999). Weaving the Web, The original design and ultimate destiny of the World Wide Web, Harper.

Bressan, S., K. Fynn, C. Goh, M. Jakobisiak, K. Hussein, H. Kon, L. T., S. Madnick, T. Pena, J. Qu, A. Shum and M. Siegel (1997). The COntext INterchange Mediator Prototype. ACM SIGMOD International Conference on Management of Data, Tucson, Arizona. pp. 525-527.

Calvanese, D., G. D. Giacomo, M. Lenzerini, D. Nardi and R. Rosati (1998). Description logic framework for information integration. Proceedings of the 6th Interrnational Conference on the Principles of Knowledge Representation and Reasoning (KR-98). S. C. Shapiro. San Francisco, California, Morgan Kaufmann**:** 2-13.

Cardoso, J., J. Miller, A. Sheth and J. Arnold (2002). "Modeling Quality of Service for Workflows and Web Service Processes." The VLDB Jounal(submitted in May 2002).

Cardoso, J., A. Sheth and J. Miller (2002). Workflow Quality of Service. International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference (ICEIMT/IEMC'02), Valencia, Spain, Kluwer Publishers.

Christensen, E., F. Curbera, G. Meredith and S. Weerawarana (2001). W3C Web Services Description Language, http://www.w3c.org/TR/wsdl.

Fabio Casati, Ming-Chien Shan and D. Georgakopoulos (2001). "E-Services - Guest editorial." The VLDB Journal **10**(1): 1.

Fensel, D. and C. Bussler (2002). The Web Service Modeling Framework. Vrije Universiteit Amsterdam (VU) and Oracle Corporation, http://www.cs.vu.nl/~dieter/ftp/paper/wsmf.pdf.

Fensel, D. and M. Musen (2001). "The Semantic Web: A Brain for Humankind." IEEE Intelligent Systems **16**(2): 24-25.

Fox, C. (1992). Lexical analysis and stoplists. Information Retrieval: Data Structures & Algorithms. W. B. Frakes and R. Baeza-Yates. Englewood Cliffs, NJ, Prentice Hall**:** 102-130.

Garvin, D. (1988). Managing Quality: The Strategic and Competitive Edge. New York, Free Press.

González-Castillo, J., D. Trastour and C. Bartolini (2001). Description Logics for Matchmaking of Services. KI-2001 Workshop on Applications of Description Logics, Vienna, Austria.

Gruber, T. (1993). "A translation approach to portable ontology specifications." Knowledge Acquisition **5**(2): 199-220.

Guarino, N. (1998). Formal Ontology and Information Systems. Proceedings of Formal Ontology and Information Systems, Trento, Italy, IOS Press, Amsterdam. pp. 3-15.

Guarino, N., C. Masolo and G. Verete (1999). "OntoSeek: Content-Based Access to the Web." IEEE Intelligent Systems **14**(3): 70-80.

Hammer, J., D. McLeod and A. Soli (1994). An Intelligent System for Identifying and Integrating Non-Local Objects in Federated Database Systems. 27th International Conference on System Sciences, Honolulu, HI, Computer Society of IEEE. pp. 389-407.

Horrocks, I., F. v. Harmelen, P. Patel-Schneider, T. Berners-Lee, D. Brickley, D. Connolly, M. Dean, S. Decker, D. Fensel, P. Hayes, J. Heflin, J. Hendler, O. Lassila, D. McGuinness and L. A. Stein (2001). DAML+OIL, http://www.daml.org/2001/03/daml+oil-index.

Hull, D. A., J. O. Pedersen and H. Schutze (1996). Method combination for document filtering. Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich, Switzerland, ACM Press, New York. pp. 279-287.

Jena (2002). The jena semantic web toolkit, http://www.hpl.hp.com/semweb/jena-top.html. Hewlett-Packard Company.

Kashyap, V. and A. Sheth (1993). "Schema Correspondences between Objects with Semantic Proximity," Department of Computer Science, Rutgers University, NJ, Technical Report DCS-TR-301.

Kashyap, V. and A. Sheth (1994). Semantics-based Information Brokering. Proceedings of the Third International Conference on Information and Knowledge Management (CIKM), Gaithersburg, Maryland.

Kashyap, V. and A. Sheth (1996). "Schematic and Semantic Similarities between Database Objects: A Context-based Approach." Very Large Data Bases (VLDB) Journal **5**(4): 276-304.

Kashyap, V. and A. Sheth (1998). Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies, Academic Press.

Kochut, K. J., A. P. Sheth and J. A. Miller (1999). "ORBWork: A CORBA-Based Fully Distributed, Scalable and Dynamic Workflow Enactment Service for METEOR," Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia, Athens, GA.

Krishnakumar, N. and A. Sheth (1995). "Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operations." Distributed and Parallel Databases Journal **3**(2): 155-186.

Lee, H. L. and S. Whang (2001). "E-Business and Supply Chain Integration," Stanford University November 2001.

Lee, J., M. Kim and Y. Lee (1993). "Information Retrieval Based on Conceptual Distance in IS-A Hierarchies." Joumal of Documentation **49**(2): 188-207.

Levenshtein, I. V. (1966). "Binary codes capable of correcting deletions, insertions, and reversals." Cybernetics and Control Theory **10**(8): 707-710.

Litwin, W. and A. Abdellatif (1986). "Multidatabase Interoperability." IEEE Computer **19**(12): 10-18.

Losee, R. M. (1988). "Parameter estimation for probabilistic document retrieval models." Journal of the American Society for Information Science **39**(1): 8-16.

Madhavan, J., P. A. Bernstein and E. Rahm (2001). Generic Schema Matching with Cupid. Proceedings of the 27th International Conferences on Very Large Databases, Roma, Italy. pp. 49-58.

McGuinness, D. (1998). Ontological Issues for Knowledge-Enhanced Search. Trento, Italy, IOS Press, Amsterdam, The Netherlands.

Mena, E., V. Kashyap, A. Sheth and A. Illarramendi (1996). OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. Conference on Cooperative Information Systems, Brussels, Belgium, IEEE Computer Society Press. pp. 14-25.

Miles, M. B. and A. M. Huberman (1994). Qualitative data analysis: an expanded sourcebook. Thousand Oaks, California, Sage Publications.

Miller, J. A., J. S. Cardoso and G. Silver (2002). Using Simulation to Facilitate Effective Workflow Adaptation. Proceedings of the 35th Annual Simulation Symposium (ANSS'02), San Diego, California. pp. 177-181.

Nelson, E. C. (1973). "A Statistical Basis for Software Reliability," TRW Software Series March.

Paolucci, M., T. Kawamura, T. R. Payne and K. Sycara (2002). Semantic Matching of Web Services Capabilities. Proceedings of the 1st International Semantic Web Conference (ISWC2002), Sardinia, Italia.

Parent, C. and S. Spaccapietra (1998). "Issues and Approaches of Database Integration." Communications of the ACM **41**(5): 166-178.

Porter, M. (1980). "An algorithm for suffix stripping." Program **14**(3): 130-137.

Richardson, R. and A. Smeaton (1995). "Using WordNet in a Knowledge-Based Approach to Information Retrieval," Dublin City University, School of Computer Applications, Dublin, Ireland, Technical Report CA-0395.

Rodríguez, A. and M. Egenhofer (2002). "Determining Semantic Similarity Among Entity Classes from Different Ontologies." IEEE Transactions on Knowledge and Data Engineering (in press).

Rommel, G. (1995). Simplicity wins: how Germany's mid-sized industrial companies succeed. Boston, Mass, Harvard Business School Press.

Salton, G. (1988). Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer. Massachusetts, Addison-Wesley.

Shegalov, G., M. Gillmann and G. Weikum (2001). "XML-enabled workflow management for e-services across heterogeneous platforms." The VLDB Joumal **10**(1): 91-103.

Sheth, A. and V. Kashyap (1992). So Far (Schematically) yet So Close (Semantically). Proceedings of the MT DS-5 Conference on Semantics of Interoperable Database Systems, Lorne, Australia, Elsvier Publishers.

Sheth, A. P., W. v. d. Aalst and I. B. Arpinar (1999). "Processes Driving the Networked Economy." IEEE Concurrency **7**(3): 18-31.

Sheth, A. P. and J. A. Larson (1990). "Federated database systems for managing distributed, heterogeneous, and autonomous databases." ACM Computing Surveys **20**(3): 183-236.

Song, M. (2001). RepoX: A Repository for Workflow Designs and Specifications. M.Sc. Department of Computer Science, University of Georgia, Athens.

Stalk, G. and T. M. Hout (1990). Competing against time: how timebased competition is reshaping global markets. New York, Free Press.

Sycara, K., M. Klusch, S. Widoff and J. Lu (1999). Dynamic Service Matchmaking Among Agents in Open Information Environments. SIGMOD Record. A. Ouksel and A. Sheth**:** 47-53.

Sycara, K., J. Lu and M. Klusch (1998). "Interoperability among Heterogeneous Software Agents on the Internet," The Robotics Institute, Carnegie Mellon University, Pittsburgh, USA October 1998, pp. 35.

Thatte, S. (2001). XLANG: Web Services for Business Process Design. Microsoft, Inc., http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm.

Tidwell, D. (2000). Web services - the Web's next revolution. IBM, http://www-106.ibm.com/developerworks/webservices/.

Tversky, A. (1977). "Features of Similarity." Psychological Review **84**(4): 327-352.

Tversky, A. and D. Kahneman (1974). "Judgement under uncertainty: Heuristics and biases." Science **185**: 1124-1131.

Ulrich, F. (2001). "The Concept of Virtual Web Organisations." Electronic Journal of Organizational Virtualness: 43-64.

Uschold, M. and M. Gruninger (1996). "Ontologies: Principles, methods and applications." Knowledge Engineering Review **11**(2): 93-155.

Woelk, D., P. Cannata, M. Huhns, W. Shen and C. Tomlinson (1993). Using Carnot for enterprise information integration. Second International Conference on Parallel and Distributed Information Systems. pp. 133-136.

XMLSchema (2001). XML Schema Part 2: Datatypes. W3C Recommendation 02 May 2001, http://www.w3.org/TR/xmlschema-2/.

Zamora, E., J. Pollock and A. Zamora (1981). "The Use of Trigram Analysis for Spelling Error Detection." Information Processing and Management **17**(6): 305-316.

CHAPTER 5

CONCLUSIONS

Organizations operating in global and competitive markets require a high level of quality of service (QoS) management. The use of workflow systems to automate, support, coordinate, and manage business processes enables organizations to reduce costs and increase efficiency. However, workflow systems should be viewed as more than just driving forces of automation or mechanization. They should be used to reshape and re-engineer the way business is done. One way to achieve a continuous process of improvement is to view and analyze workflow processes from a QoS perspective. This allows processes to be designed and adapted according to quality of service constraints drawn from organizational goals and strategies.

New trading models, such as e-commerce, bring a new set of challenges and requirements that need to be explored and answered. In such processes, trading agreements between suppliers and customers include the specification of QoS items such as products or services to be delivered, deadlines, quality of products, and cost of service. The correct management of such QoS requirements directly impacts the success of the organizations participating in e-commerce and also directly impacts the success and evolution of e-commerce itself. The composition of Web services, and therefore of workflows, cannot be undertaken while ignoring the importance of quality of service measurements. A good management of QoS leads to the creation of quality products and services; this in turn, fulfills customer expectations and achieves customer satisfaction. The good management of QoS becomes increasingly important when workflow systems

are used in new organizational and trading models, such as in virtual organizations and e-commerce activities that span organizational boundaries.

While QoS management is of a high importance to organizations, current WfMSs and workflow applications do not provide adequate features to support QoS. Three research areas need to be explored. First, a good theoretical QoS model is necessary to formally specify and represent QoS metrics for workflow tasks. Second, algorithms and mechanisms are necessary to synthesize workflow QoS based on the QoS metrics of the tasks that compose the workflow. Third, experimental workflow systems need to be developed to identify the challenges and difficulties associated with the implementation of QoS management.

Chapter 2 discusses and explains the importance of quality of service management for workflow and workflow systems. Based on our experience in developing of workflow applications for various domains and with emergent workflow requirements, we present a QoS model. This model allows for the description of workflow components (tasks) from a quality of service perspective; it includes four dimensions: time, cost, reliability, and fidelity. The use of QoS increases the added value of workflow systems to organizations, since the non-functional aspects of workflows can be described.

Once the QoS of workflow components is specified, the QoS of workflows can be automatically computed. This feature is important, especially for large processes, which in some cases may include hundreds of tasks. For this reason, we present a mathematical model and describe how a simulation system can be used with a workflow system to carry out efficient workflow QoS simulations. Both approaches enable a predictive computation of workflows QoS based on QoS estimates for tasks. The mathematical model formally describes a set of formulae for computing QoS metrics among workflow tasks. Based on these formulae, we have developed an algorithm (SWR algorithm) to automatically compute the overall QoS of workflows. The algorithm applies a set of

reduction rules to a workflow, until only one task remains which represents the QoS for the entire workflow.

To test the validity of our QoS model and of our mathematical model we have deployed a set of production workflows in the area of genetics (Fungal Genome Resource Laboratory). We have executed several workflow instances, and the generated QoS data have been collected and analyzed. Analysis of the data has corroborated our initial hypothesis, which states that our QoS model and mathematical model provide a suitable framework for predicting and analyzing the QoS of production workflows.

Having developed a QoS theoretical model and an algorithm to compute workflow QoS, the next step was the implementation of these new concepts in a workflow system. For this purpose, we have selected the METEOR system developed at the LSDIS Lab. Chapter 3 describes in detail the modifications and extensions necessary for workflow systems components to support QoS management. Modifications are necessary for some components, including the enactment system, the workflow builder (or designer), the monitor, the code generator, the repository, the workflow model, and the task model. Additionally, new components need to be implemented, such as a QoS estimator module to create QoS estimates for tasks and probabilities for transitions. The monitor needs an additional interface so that runtime tasks QoS metrics are propagated and logged into a database for data processing purposes.

Chapter 4 discusses and the set of challenges that the emergence of Web services and e-services has brought to organizations. While in some cases Web services may be utilized in an isolated form, it is normal to expect Web services to be integrated as part of workflows processes. This entails research in two areas. Mechanisms to efficiently discover Web services during an e-workflow (*i.e.*, a workflow managing traditional tasks and Web services) composition process and to facilitate their subsequent integration with the e-workflow host.

We present a methodology and a set of algorithms for Web service discovery based on three dimensions: syntax, operational metrics, and semantics. This approach allows for Web service discovery not only based on functional requirements, but also on operational metrics.

The need to discover workflow components based on operational metrics has a greater importance when Web services are involved, as compared to workflow tasks. The autonomy of Web services does not allow for users to identify their operational metrics at design time, *i.e.*, before their actual execution. The development of mechanisms for the discovery of Web services based on operational metrics allows organizations to translate their vision into their business processes more efficiently, since e-workflows can be designed according to QoS requirements, goals, and objectives.

To facilitate the discovery and posteriori integration of Web service into workflows we propose an approach based on the use of ontologies to describe workflow tasks and Web service interfaces. Ontology-based approaches have already proved to be an important solution to information integration in order to achieve interoperability. During an e-workflow composition, there is a loss of semantics associated with Web service task interfaces because a large part of the domain knowledge a developer employs when deploying a Web service is not present at composition time.

In our work we have devised an algorithm and implemented a prototype to discover and facilitate the resolution of structural and semantic differences during the integration process with an e-workflow. The algorithm uses a feature-based model to find similarities across workflow tasks and Web service interfaces. The system determines and evaluates the best mapping between the outputs and inputs of a SO and the workflow host that yields the highest degree of integration.

APPENDIX A

THE DNA SEQUENCING WORKFLOW

A.1 INTRODUCTION

The life sciences research laboratory at the University of Georgia specializes in genomics, a discipline that investigates biological problems by examining entire genomes, or a large number of genes, at one time. Genomic projects involve highly specialized researchers and laboratory personnel, sophisticated equipment, and an intense generation and computation of data. The characteristics of the human and technological resources involved are often geographically distributed; they require a sophisticated coordination infrastructure to manage not only laboratory personnel and equipment, but also the flow of data generated.

The research laboratory has realized that to be competitive and efficient it must adopt a new and modern information systems infrastructure. A first step was taken in that direction with the adoption a workflow management system to support its laboratory processes. The adoption of a workflow system has enabled the logic of traditional laboratory processes to be captured in a workflow schema. In the context of our research on quality of service, we have developed a workflow application to manage the DNA sequencing process. This workflow has been used to study, validate, and demonstrate the applicability of our QoS model.

In the next sections, we give a brief introduction to genomics and describe the DNA sequencing workflow application we have developed.

A.2 INTRODUCTION TO GENOMICS

A genome represents a complete set of instructions for making an organism. It is a detailed map that contains the master blueprint for a cell or an organism. Packed into nearly every body cell there is a complete copy of the human genome, *i.e.*, all the genes that make up the master blueprint for building a living human.

A cell or organism is composed of deoxyribonucleic acid (DNA) and protein molecules, which are organized into structures called chromosomes (Figure A-1). The DNA is a double-stranded linear chain of nucleotide bases from the set composed of adenine, thymine, cytosine, and guanine (or A, T, C, G, respectively). The nucleotides can be thought of as an alphabet, as their sequence along the DNA strand encodes instructions for building proteins.

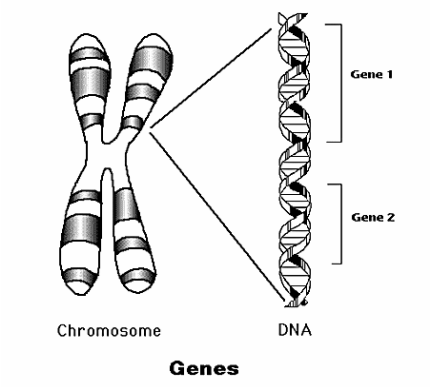

Figure A-1 – Chromosome, DNA, and Genes.

Nucleotide base pares are organized into units called chromosomes. All genes are arranged linearly along the chromosomes and contain roughly equal parts of protein and DNA. The DNA molecules are among the largest molecules known.

A gene is a specific sequence of nucleotide bases whose sequence carries the information required for constructing a single protein. Proteins provide structure to cells, tissues, and enzymes, which catalyze all of the cellular biochemical reactions.

A major task in genomics is determining the DNA sequence of a genomic region (Hall, Miller *et al*. 2000). The first draft of the entire sequence of the human genome has been recently reported (Lemonick 2002). The next step in genomics is in determining how individuals differ from one another at the genetic level. Genetic differences between individuals underlie differences in susceptibility to most diseases and in the response to drug therapy.

A.3 DNA SEQUENCING WORKFLOW DESCRIPTION

The DNA Sequencing workflow manages a set of tasks to carry out particular activities necessary to sequence DNA. The workflow is responsible for: managing the preparation of a sample, cloning, sequencing, assembly, sequence processing, and processing the results obtained. A graphic representation of the workflow is illustrated in Figure A-2.

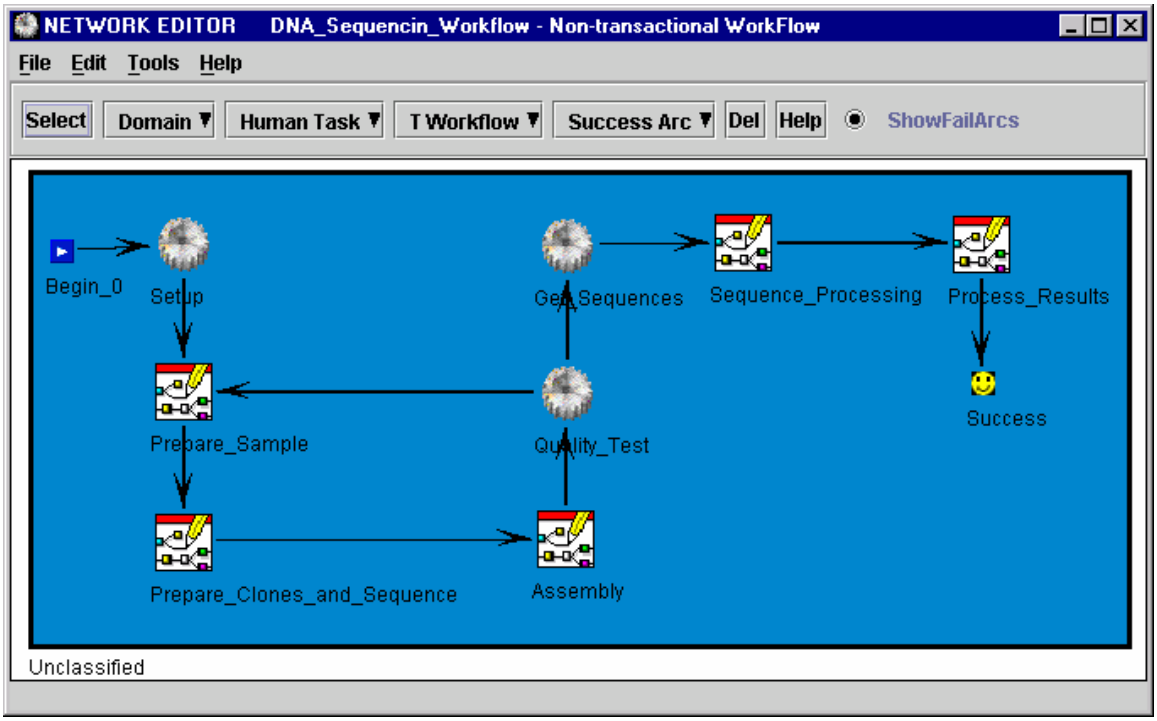

Figure A-2 – The DNA Sequencing workflow.

In the next section, we describe the main tasks that compose the DNA Sequencing workflow.

A.3.1 SETUP AND PREPARE SAMPLE

The *setup* task is relatively simple; its objective is to initialize general information describing the samples being sequenced.

The prepare sample task consists of isolating DNA from a biological sample. The samples can be prepared using a variety of protocols, which when carried out rigorously will ensure DNA that is of high quality. A high quality DNA sample will yield more accurate sequencing results. The quality of the DNA sample or template is one of the most critical factors in DNA sequencing.

This task is accomplished using the "shotgun" or random sequencing approach method. This entails cloning small genomic DNA fragments into a series of plasmid clones. Shotgun sequencing has the advantage that it requires no prior knowledge of the DNA sequence and puts no limitations on the size of the DNA to be sub-cloned. In shotgun sequencing, the target DNA is fragmented by enzymatic digestion or by shearing it into a large number of fragments, from a few hundred to a few thousand nucleotide pairs. Each piece is then inserted into bacteria to establish clones.

In this technique, the DNA of specific genes is removed from the desired sequences and placed into bacteria. Once a specific sequence of DNA has been isolated and placed into the bacteria, it is allowed to replicate itself, producing millions of clones that contain the DNA sequenced being used for the research project. When the bacteria replicate, they also replicate the gene. When the desired number of clones is obtained, the bacteria is separated from the inserted piece of DNA understudy. The colonies that grow are ones that contain the DNA that will be used later to translate the protein.

The task *prepare sample* is composed of 16 individual, manual steps executed sequentially. We give a brief description of each step:

Inoculate (1). The step inoculate consists of growing a culture of 20-ml LB tubes in 50-ml conical tubes and letting them shake overnight.

Miniprep. The step miniprep takes the inoculated culture, and using an enzyme, removes any genomic contamination (e.g. *E. coli*).

Inoculate (2). This second inoculation grows a culture of 500 µl of a 20-ml LB tube and inoculates 250 ml of LB that will shake overnight.

Maxiprep. The step maxiprep takes the inoculated culture, and using an enzyme, removes any genomic contamination (e.g. *E. coli*), as well as any bacterial cells which were required to grow the cDNA.

Nebulize DNA. In this step, a nebulizer containing buffered DNA solution and glycerol is placed in an ice-water bath and subjected to argon gas at a pressure of 8 psi for 2.5 minutes. The pressure induces the fragmentation of the DNA.

Ethanol Precipitation. An ethanol precipitation is carried out to clean up the DNA.

End Repair Nebulized DNA. Since nebulized DNA fragments usually contain single-stranded ends, the samples need to be repaired prior to ligation.

Agarose Gel Electrophoresis. In this step, the DNA samples are placed on a gel that is subjected to electrophoresis. The fragments in each sample are allowed to migrate to separate, adjacent lanes in the gel. The pieces of DNA in each lane become separated by size, the smaller pieces traveling farther in the gel than the larger ones. The separation of the pieces in the lanes is then visualized by long wave UV light and photographed with a computerized system; this reveals the distance traveled by the pieces in each lane.

Cut Out Bands. The gel bands of interest are excised with a sterile razor blade. The sizes of the bands of interest are in the 1-2 kb and 2-4 kb size range.

Purify DNA from Gel Fragments. In this step, the gel fragments are purified with a High Pure PCR kit by means of column centrifugation, and the "freeze and squeeze" method by ethanol precipitation is carried out.

Ligation. The recipient vector, which usually contains an antibiotic resistance gene, is cut so that it is ready to receive an insert. Both are joined together to form a recombinant DNA molecule. DNA ligase seals the breaks between the inserted, end-repaired DNA fragments and the cloning vector.

Electroporation. The recombinant molecules are introduced into a host, in our case a bacteria XL-1 blue competent (various types of organisms can be used). The bacteria take up the DNA by electroporation (*i.e.*, an electric shock).

Ethanol precipitation. The DNA is clean up again by ethanol precipitation.

Put Cells in Plates. In this step, the cells are plated on selective media containing an antibiotic that will kill any cells that have failed to take up the vector carrying the antibiotic-resistant gene. Two types of colonies arise. One is the white colony, containing the insert, and the other is the blue colony, which is missing the insert.

Pick Colonies. Finally, a robot picks 384 colonies from the plates and puts them into two 96-well microtitre plates. The cells will grow overnight in a solution of carbanicillin, LB, and glycerol.

A.3.2 PREPARE CLONES AND SEQUENCE

The sequencing task is composed of three tasks (Figure A-3): *prepare clones*, *sequencing*, and *base calling*.

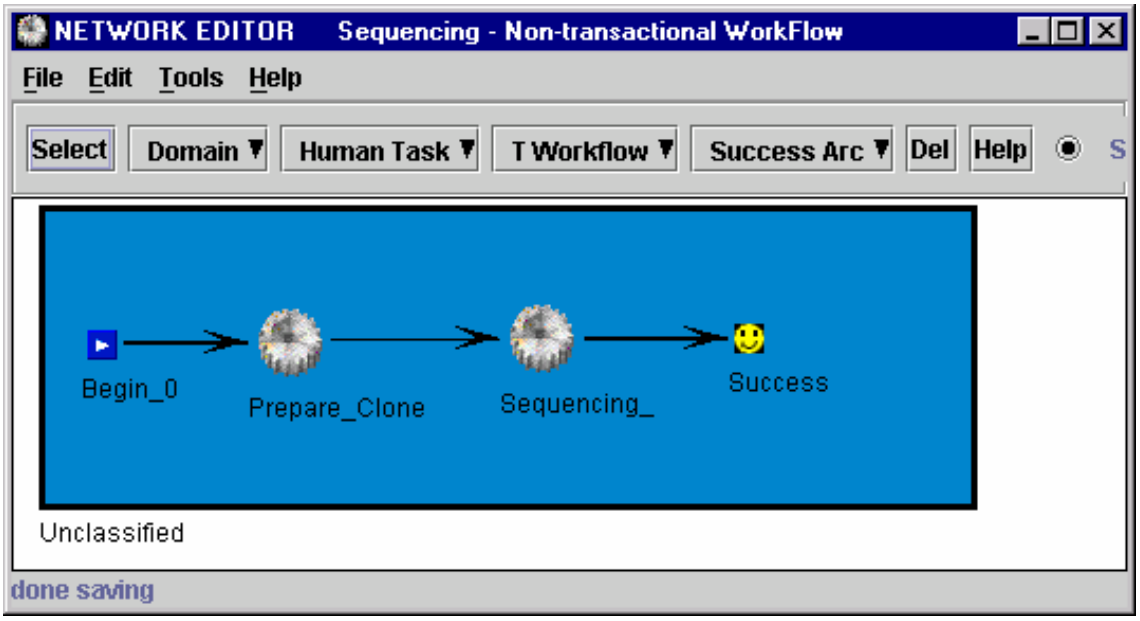

Figure A-3 – Prepare Clones and Sequence.

A.3.2.1 PREPARE CLONE

During the realization of this task, specific regions of the genome are isolated from the materials generated in the previous step. This task is composed of three steps: *grow clones*, *template purification*, and *sequence reaction*.

Grow Clones. In this step the plates are replicated into deep well plates.

Template Purification. Alkaline lysis and ethanol precipitation are used to purify the cells in the deep well plates.

Sequence Reaction. In this step a Polymerase Chain Reaction (PCR) is done on the cloned DNA. PCR allows the DNA to be selectively amplified from specific sites in order to isolate specific sequences of DNA which are of interest for further research. The isolated DNA sequences are inserted into a molecule of bacterial DNA called a plasmid. When the bacteria replicate, so do the plasmids, thereby replicating the sequence of interest.

A.3.2.2 SEQUENCING

The task *Sequencing* consists of loading the deep well plates onto a DNA sequencer in order to read each biochemical "letter" (A, G, C or T) of a cloned DNA fragment. The output is composed of decoded segments of 100 to 750 nucleotides in length (*e.g.* sequence AGGCATTCCAG.…) The development of instruments for automated DNA sequencing has dramatically increased the output of individual laboratories. Unlike manual sequencing methods, which generally use a radioactive label and visualize the banding pattern by autoradiography, automated sequencers use a scanning laser to detect DNA fragments labeled with fluorescent dyes. Considerable time and labor savings can be obtained with automated sequencing – through increased capacity, immediate data acquisition, and automatic data entry. The DNA sequence information is detected and signals are sent to a computer to be collected by data collection software. This data is then processed and interpreted by specific analysis software.

A.3.2.3 BASE CALLING

This task processes and interprets the data files (in ABI format) obtained from the sequencing task. The files are analyzed, peaks are identified, and base calls and associated quality values are generated. This task uses PHRED (Ewing and Green 1998) to carry out the base calling.

A.3.3 ASSEMBLY

This task is responsible for the assembly of the fragments resulting from the *prepare clones and sequence* task. The sequence assembly reconstructs the sequence of a clone from readings made from shorter DNA fragments which were generated from the clone.

To carry out the assembly the PHRAP application (PHRAP 2002) is used. PHRAP exhaustively compares all readings against all other in a pairwise fashion, ranking each potential reading-pair overlap according to its alignment score. Starting with the highest scoring pair, it then takes each pair in turn and constructs contigs by using a greedy algorithm.

After the sequence assembly, additional reading and editing is usually required to raise the accuracy of the sequence to a level that is considered adequate. This process involves the assembly of additional sequences and the visual inspection of assembly discrepancies. To this end, the Consed (Gordon, Abajian *et al*. 1998) application is used.

A.3.4 QUALITY TEST

To improve the quality of the DNA Sequencing workflow, it is advantageous to detect any contamination that may have occurred to the sequences. Clones grown in bacterial hosts and managed by humans are likely to have contamination, such as *escherichia coli* (*E. coli*). A quick and effective way to screen for the *E. coli* contaminant is to compare a DNA sequence against its gemone. For *E. coli*, this task is made easier with the availability of its full genome; the comparison can be done using the program BLASTN (Altschul, Gish *et al*. 1990). The goal of the *test quality* task is to detect *E. coli* contamination after the *assembly* task.

A.3.5 GET SEQUENCES

Get sequences is a simple task that downloads the sequences created in the assembly step, using the FTP protocol.

A.3.6 SEQUENCE PROCESSING

The goal of the *sequence processing* task is to identify macromolecules with related structures and functions in the DNA sequences. Given a new DNA sequence, a scientist will typically compare the sequence to a repository of known sequences (*e.g.*, Swiss-Prot or GenBank), using one of a number of computational biology applications for comparison. This search is used to determine whether a newly-sequenced DNA has already been published in the literature, and, if not, to give some hint of its putative function by searching for related sequences. This task is itself composed of two tasks which identify macromolecules with related structures and functions: the *SP BLAST* and *SP FASTA* tasks.

The *SP BLAST* task uses the BLAST (Basic Local Alignment Search Tool) application to provide a method for rapid searching of nucleotide and protein databases. The application uses heuristic algorithms to identify similar sequences in databases. It first identifies very short, exact matches between the query sequence and the database sequences. Subsequently, the best short matches from the previous stage are extended to see if more similarity can be found. The programs classify each search with a score reflecting the degree of similarity between the query sequence ("probe") and the compared ("subject") sequence. If the degree of similarity is strong, then the two sequences may share a homologous relationship, and the new sequence may be assigned potential biological functions that can be tested in the laboratory or classified into a functional family.

Other programs, with the same objective as BLAST, include FASTA (FASTA 2002) and those that implement the Smith-Waterman algorithm (Smith and Waterman 1981). The programs perform sequence comparisons which typically generate a series of scores that estimate the degree of similarity between a subject sequence and a set of target sequences. The BLAST and FASTA programs also generate an estimate of the

probability that the relationship between two sequences could have been observed by chance. A very low probability indicates that the relationship between two sequences is highly significant, whereas a relatively high probability suggests that the relationship may be due to random chance. The *SP FASTA* task uses the FASTA for sequence comparisons.

Many of the most interesting relationships between sequences occur in the boundary zone, where the significance of the relationship between two sequences is not clearly resolved by a particular algorithm. To resolve this situation, the comparisons can be defined as a series of similarity tests, using several different algorithms and several parameterizations from each – from fast heuristic methods such as BLAST, to slower, more rigorous methods such as Smith-Waterman.

A.3.6.1 PROCESS RESULTS

After obtaining the data from the *sequence processing* task, the results are processed. The results are stored, e-mailed, and a report is created (Figure A-4).

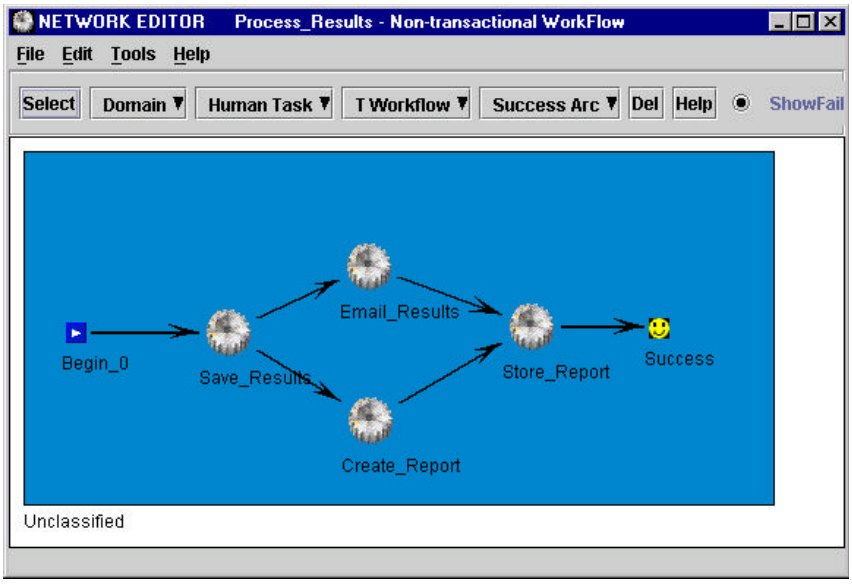

Figure A-4 – Process Results.

The *store results* task stores the data generated by the previous task in a database. As the name indicates, *e-mail results* is responsible for electronically mailing the sequencing results to the persons concerning with this process, such as researchers and laboratory technicians. The *create report* task creates a technical report containing formatted information which describes the results obtained. Finally, the *store report* task stores the report in a persistent storage site.

A.4 ACKNOWLEDGEMENTS

We would like to thank Dr. Jonathan Arnold, director of the Fungal Genome Resource laboratory, from the Department of Genetics (University of Georgia) for helping us to construct the DNA Sequencing workflow.

A.5 REFERENCES

Altschul, S. F., W. Gish, W. Miller, E. W. Myers and D. J. Lipman (1990). "Basic local alignment search tool." Journal of Molecular Biology 215: 403-410.

Ewing, B. and P. Green (1998). "Base calling of automated sequencer traces using Phred II: error probabilities." Genome Research 8: 186-194.

FASTA (2002). FASTA home site: http://fasta.bioch.virginia.edu/fasta/.

Gordon, D., C. Abajian and P. Green (1998). "Consed: a graphical tool for sequence finishing." Genome Research 8: 195-202.

Hall, D., J. A. Miller, J. Arnold, K. J. Kochut, A. P. Sheth and M. J. Weise (2000). "Using Workflow to Build an Information Management System for a Geographically Distributed Genome Sequence Initiative," LSDIS Lab, Department of Computer Science, University of Georgia, Athens, GA, Technical Report.

Lemonick, M. D. (2002). The Genome Is Mapped. Now What. Time Magazine. July 3: 24-29.

PHRAP (2002). The Phred/Phrap/Consed System Home Page. http://www.phrap.org/.

Smith, T. F. and M. S. Waterman (1981). "Identification of common molecular subsequences." Joumal of Molecular Biology 147: 195-197.