Service-Oriented Workflow

Article · January 2008

Source: DBLP

CITATIONS
5

READS
6,651

1 author:

Youakim Badr
Pennsylvania State University
175 PUBLICATIONS **1,642** CITATIONS

SEE PROFILE

Service-Oriented Workflow

Youakim BADR
National Institute of Applied Sciences – Lyon
Laboratory of Informatics for Enterprise and Production Systems
LIESP, INSA-Lyon, F-69621, FRANCE
youakim.badr@insa-lyon.fr

Journal of Digital Information Management

ABSTRACT: *The advantages of Information and Communication Technologies (ICT) have forced many enterprises to reconsider the way their business processes are undertaken. Among various Information Technologies, the Workflow Management System greatly improves the efficiency of business processes. The challenge for Workflow Systems relies on their ability to align ICT and the processes of production and business. They should support the strategic policy of the enterprise and quickly respond to on-demand business and opportunities. Unfortunately, they suffer from some lacks and limitations due to their monolithic architectures. Most Workflow Management Systems are dependent on technology platforms and partially conform to standards. We attempt throughout the present paper to propose the next generation of Workflow architectures. We conceive workflow systems as a set of services, which are generic and platform-independent. Services are self-contained and fully-customized. Workflow services are assembled from computational components (i.e. web services), which are loosely coupled. The new architecture is split into layers to guarantee a high level of abstraction, each of which is managed by a crucial Web Service known as the orchestrator service.*

Categories and Subject Descriptors
D.2.11 [Software Architecture]; D.2.12 [Interoperability]: H.4 [Information System Applications]; Workflow management

General Terms
Workflow management, Software architecture, Interoperability and standards

Keywords: Workflow management, Interoperability, Standards, Software Architectures, Integrated environment, Design Tools and Techniques.
Received 27 June 2006; Revised and accepted 15 Sep. 2007

1. Introduction

Today's business enterprises must deal with global competition, reduce the cost of doing business, and rapidly develop new services and products. To address these requirements enterprises must constantly reconsider and optimize the way they do business and change their Information Systems to support the agility of their business processes. The emergence of Information Technologies (IT) has forced many enterprises to reconsider the way they carry out their business processes. Among the various Information Technologies, the Workflow Management System, which is widely recognized as an effective tool, greatly improves the efficiency of business process. Workflow [1] is a concept closely related to the reengineering and the automation of businesses and information processes. Workflow describes the activities of business processes at a conceptual level and captures control flows and information flows between these activities. In addition, it captures process requirements such as human skills, software applications and resources. From the viewpoint of Computer Science [2], workflow is the computerized facilitation or automation of a business process, in whole or in part, by the management of the sequence of work activities and the use of appropriate human and IT resources.

Nowadays, a great number of commercial Workflows and academic prototypes [2, 3, 14, 21, 42, 40] have been available on the market and in literature. The Internet, the mobile communication and the Enterprise Information Systems (i.e. ERP and Supply Chain systems) have rapidly become a growing area gaining more and more substantial attention not only in the interrelation of electronic business applications (B2B, e-commerce, B2C) but also in cross-enterprise collaborations. The challenges for workflow management systems stem from their ability to align Information and Communication Technologies (ICT) and on-demand businesses. Evolution of technology should not make legacy enterprise applications hard to maintain. Instead, they should encourage cooperation with partner applications beyond the enterprise's boundaries. Workflow makes a clear separation between business logic and the enterprise's resources (i.e. software, human skills and/or machines) and it correlates the enterprise's policy and adequate resources by using business process definition languages. We have listed several works [4, 6, 46, 21] regarding the definition of the business process languages, platforms for business collaborations between enterprises, the interoperability between legacy applications and their remote execution by external applications. Few works consider the architectures of workflow management systems. Due to the crucial role of workflows, their architectures have to evolve and cope with ICT. They should take into consideration new business paradigms. Unfortunately, this trend does not shape current workflow architectures. Many lacks and various limitations arise around their architectures. We state the following problems:

1.1. Architecture Design

Early workflow management systems were monolithic, they are considered as black boxes. Extensions of their functionalities were difficult and sometimes incompatible with workflow design. Most of today's workflows are centralized and based on a 2-tier architecture (Client/Server). The robustness of workflows depends on the server side; during the maintenance or the upgrade period, the system is inaccessible. By using components and persistence objects, workflow architectures become modular, easy to maintain and distributed over the intranet. The development platforms of components belong to vendors (J2EE, .NET, CORBA, etc) and thus there are not natively compatible.

1.2. Reference Model

Another problem arises from the interoperability between workflows and the applications invoked by the workflow engine during the execution of business processes. Web Services technology seems to be a promising solution for interoperability problems. It operates as a neutral interface to invoke applications. Still, it is too far away to respect a common reference model. The WfMC [7] initiative attempts to propose a reference model for the workflow community. As we notice, many workflows either conform partially to the reference model or they do not agree with it at all.

1.3. Enterprise Dependency

Even if some workflows are generic, they depend on the enterprise's organizational structure, legacy applications and business strategy. These workflow management systems need to be customized (i.e. using configuration files) in order to support production and business processes within the boundary of the enterprise. The new age of the electronic business applications (B2B, e-commerce, B2C, e-market) has radically changed the management of workflows. E-business applications have become dependant on the collaboration of cyber-partners,inter-workflow interoperability and an enterprise's agility.

1.4. Specific Platform Dependency

Workflow management systems are strongly but not exhaustively dependant on 1) Programming languages (i.e. Java, C++, etc), 2) Operating systems (MS Windows, Linux, Unix, Mac OS, etc), 3) Networking protocols (i.e. protocols based on TCP/IP), 4) Remote Call protocols (IIOP, RPC, RMI, etc) and 5) Database management systems (Oracle, MySQL, SQL-server, etc). It is clear that a platform-independent architecture of workflows empowers development and facilitates the enactment of workflows without vendor dependency.

These limitations and lacks associated with workflow management encourage us to design a new architecture for conceiving and developing workflow engines. The new architecture should respond to the agility of the enterprise's organization and on-demand business and it should take full advantages of ICT. We outline our problematic with the following statement:

> *How can we develop a distributed, adaptive and platform-independent Workflow Engine based on standards and a Common Reference Model?*

Throughout the paper, we attempt to answer this question. Thus, we propose a generic architecture for workflow management systems based on layers of loosely-coupled web services. We particularly illustrate the design of the workflow engine, which relies on a centric-service called the orchestrator that guarantees agility. We conceive the architecture with the WfMC standard [7] in mind. Web services are platform-independent and distributed.

The remainder of this paper is organized as follows. Section 2 provides an overview of related works. Section 3 introduces preliminaries for service-oriented architecture (SOA), a definition of service as well as the orchestration of services. Section 4 presents the generic workflow management system through the three layers of services. Section 5 discusses in detail the design of the workflow engine based upon SOA. Related services to the engine are illustrated as well. Section 6 stresses on the implementation of the five interfaces of the WfMC reference model as web services. Section 7 illustrates the orchestration of engine services and, finally, Section 8 and 9 describe the prototype implementation and concludes the paper.

2. Related Works

The academic community has the longest track record of involvement with workflow which dates back from the late seventies [24], [25]. During the last decades workflow has been the focus of intense activity in terms of commercial workflow systems, standards, and research [8, 9, 10, 11, 12]. Many commercial systems have been introduced to support workflow management systems [21]. The Business Workflow Architecture which is supported by SAP [40], the Lotus Workflow of IBM [41], TIBCO InConcert [42] and i-Flow of Fujitsu are just a few examples of commercial workflows. Despite their many features, most commercial WFMSs have a number of significant limitations due to their lack of evolving from centralized computing environments (i.e. within the firm's boundaries) to a new distributed environment (i.e. cross-firms). Due to their dependency on vendor technologies, they also suffer from supporting integration and interoperability among loosely-coupled components corresponding to heterogeneous, autonomous, and/or distributed legacy and new systems.

In workflow literature, business process languages play an important role in the workflow area. Since they are used to implement activities of processes, various languages and standards have been developed (i.e. XPDL, YAWL, BPEL and BPMN [27, 28, 29, 30]). In addition, Web Services provide a means to formally specify business processes and interaction protocols [20]. Hence, web services lead to a range of process languages which include, and not exhaustively, Business Process Execution Language for Web Services (BPEL4WS [31]), Web Services Choreography Interface (WSCI [32]), Business Process Management Language (BPML [33]), XLANG [34], Web Services Flow Language (WSFL [35]). A detailed comparison is given in [19]. Even though Business Process languages allow invoking heterogeneous and distributed applications, workflow systems remain centralized and based on client/server architecture. Business and organizational processes, however, are no longer relying on pre"specified or well"known actions. They become complex collaborative tasks requiring the integration of partner applications.

A number of research projects address the design of workflow architectures [36, 37]. WIDE (Workflow on Intelligent and Distributed database Environment) is an ESPRIT project of the European Commission [23], which emphasizes the integration with external databases and provides strong transaction support by the coordination between a global and multiple local transaction managers. Micro-workflow architecture [22] attempts to bridge the gap between the type of functionality provided by current workflow systems and the type of workflow functionality required in object-oriented applications. It focuses on customizing workflow features and integrating with other systems. The COMBINE project [26] follows a similar approach to support model-driven development of enterprise systems - using Components. The project introduces cohesive component concepts independent of any particular technology (e.g. J2EE/EJB, COM or CORBA) that address fine-grained components applicable throughout the development lifecycle.

The author in [13] proposes yet another architecture to meet the new, emerging requirements for workflow systems, such

as inter-organizational collaborative support. The architecture is based on a set of service components, the components are mainly implemented by web services. The architecture enables the automatic execution of Web services for both workflow system management and workflow activity execution. According to the requirements that we have stated in the introduction, this architecture does not conform to the WFMC reference model. The five interfaces are not respected, which makes the workflow isolated and it thus cannot communicate with associated applications and or other systems outside the enterprise's boundary.

WISE (Workflow-based Internet SErvices) [14] has both a runtime component and a development environment associated to it. WISE is organized around three service layers: database services, process services and interface services. A close view of the architecture shows that services are tightly coupled and the consequence between these services depends on the communication between them. It is not easy to replace a service with another and to dynamically change the interaction between web services.

CrossFlow [15] proposes a contract based inter-organizational workflow system to control the execution of business services offered by different organizations. The extended workflow management system is tailored for supporting standard cooperation for a supply chain in a static virtual enterprise environment. Although the architecture fits the network of distributed enterprises well, it does not respect standards such as the WfMC model or communication protocols (XML-SOAP).

As we notice, the challenge for workflow management systems stems from their ability to align information systems (IS) and new business opportunities. In other words, The IS has to support the strategic policy of the enterprise and quickly respond to a new business opportunity by reorganizing the internal processes with fewer efforts and without additional cost. Information systems should no longer be a cost-centre for firms, where the investment is expensive and the return on investment is low. Workflow management systems, which have been studied, show some lacks and limitations such as interoperability between the workflows and their supported applications, the absence of a common reference model and their dependence on technologies. They implement rigid hierarchical models of task coordination over which the workflow system administration centrally has total or significant authority. This approach has obvious organizational difficulties as well as a deep impact on architectural models and system design options. Some limitations of traditional workflow architectures are due to the fact that the enactment system is implemented in a central server. This server contains a repository of workflow definitions, assets, resources and drives process execution and synchronization as well as task distribution and management.

A distributed, adaptive and platform-independent Workflow Engine based on standards and a Common Reference Model serves as a foundation to support the openness enterprise for B2B electronic business applications.

3. Service-Oriented Architecture (SOA)

Over the last four decades, the practice of software development has gone through several different developmental methods. Each methodological shift was made in part to deal with greater levels of software complexity. The way we have managed complexity is to continuously invent coarser-grained constructs, such as *functions*, *classes*, and *components*. Technologies such as EJB, .NET and CORBA are effective ways of implementing components. The method of component-based development has allowed developers to create more complex, higher quality systems faster than ever before. A *Service-Oriented Architecture* (SOA) [16] is an architecture that has *special properties* and is made up of components and interconnections that stress location transparency and interoperability. The term *service* has been used to indicate the ability to make a remote method call. A service is a behavior that is provided by a component for use by any other component based only on the *interface contract*. A service has a network-addressable interface and may be dynamically discovered and used. It mainly stresses interoperability. The term service is centric in SOA. With the introduction of *Web Services* [17], there has been a renewed interest in Service-Oriented Architectures (SOA). Web Services and SOA are ultimately about designing and building systems using heterogeneous network-addressable software components. Thus, Web Services have given the term *service* more prominence.

3.1 The Service

A service is defined as an application component providing a particular function in a specific context of use. A service is self-contained and self-describing and uses open protocols to communicate and be executed by other services (i.e. service consumers). In general, a service has five important properties:

1-Loose Coupling: a service cannot call or execute another. The call mechanism is dedicated to a special service called the orchestrator. Thus, the loose coupling property is associated with the independence of the service activation against the implementation of technology.

2-Remote Execution: a service exposes a unique interface to describe its usage. Service activation is completely independent of the programming language and the operating system of its consumer.

3-Asynchronous Communication: the services communicate in asynchronous mode, which means that the service will not wait for a reply message in order to continue its execution.

4-Contract-Based: each service provides a contract describing its input and output messages, the transport or the binding protocols and a list of operations to be invoked by the service consumer.

5-Conforming to the SOA pattern: the conception of SOA architecture consists in dividing the functions of an application into small services attached to packages of object-oriented classes. Each package is equivalent to a *category* as defined by Grady Booch [18]. A category is just a business object (i.e. components) in the enterprise vocabulary; each category has a unique façade (interface) to access its services. Figure 1 illustrates two services which are represented by categories of components (C_1 and C_2). In the left side of the figure, the services are not loosely coupled and isolated whereas the figure on the right shows the indirect interaction of services through a special component known as Orchestration.

3.2 Services Orchestration

An important part of SOA architecture is the function of the orchestrator. The orchestrator guarantees that the services are not coupled tightly. As each service is *stateless*, the orchestrator does not only take the responsibility of calling services in a particular sequence, but also the responsibility

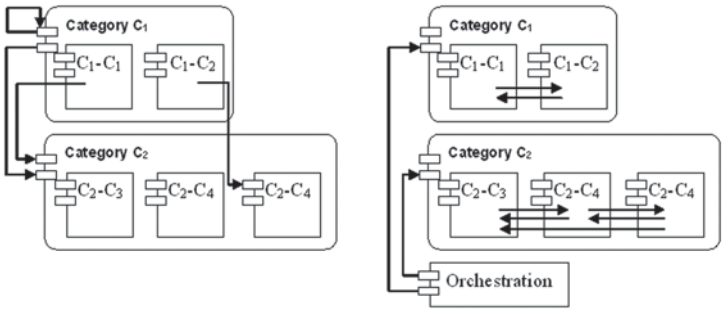

Figure 1. SOA Patterns

of managing the execution context such as maintaining the information between services managing transactions as well as human-machine interaction.

Figure 2 represents graphically the orchestrator's functionalities. Thanks to the orchestrator, the assembly of services is flexible and can be easily modified to globally change the *function* of the underlying services.

4. Workflow System Design Based on Service-Oriented Architecture

We propose a general architecture for a workflow management system based on the assembly of services. The architecture is first modeled as modular components. Each component is encapsulated by a web service. Thus the services reify components and take full advantages of the web services stack. The architecture also respects the service-oriented approach (SOA), which allows for an agile and flexible reorganization of the web services. Agility is guaranteed thanks to an orchestrator. The orchestrator is the only web service which can invoke other web services. By such, web services are loosely-coupled and thus the logic behind their collaboration is encoded in the orchestrator. The design of the workflow engine is based on the WfMC standard [7]. It is for this reason that we encapsulate the five interfaces into web services as well.

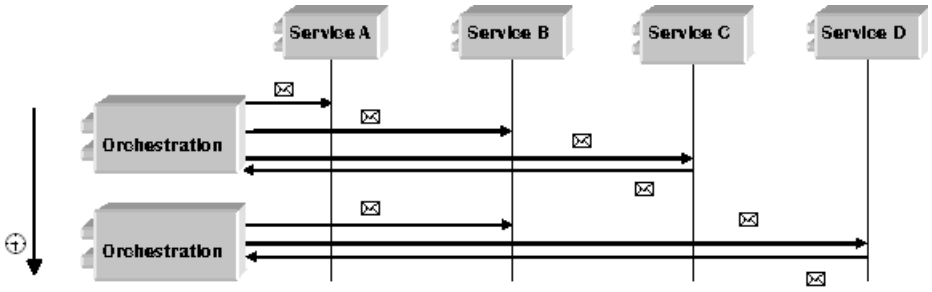

Figure 2. The orchestration of loosely-coupled services

We present in the following paragraphs the workflow system and particularly discuss the workflow engine, which is based on service-oriented architecture. Hereafter, we commonly refer to the workflow engine by WF-WS (Workflow based on Web Services).

4.1. General View of Workflow Systems

Workflow systems are not only composed of machines or information processes but users who are also important components of the system. Thus *workflow* views software and computers as tools used to assist users in managing stages of work process.

From the view point of a "workflow management system," there is a need for log service, monitoring and administration, exception handle, interface, components orchestration, and databases. However, in our WF-WS case, we concentrate our work to cope with workflow engine architecture. The engine provides an interface for the users to design and execute business processes. During the process definition, we define tasks and control flow as well as pre-conditions, resources, and human roles or system applications, etc... Once the process is defined using a process definition language, we load it into the workflow engine and generate a process instance to be executed. In the same way, the process instance generates work items associated with conditions and triggers. Every prepared task will be scheduled by the scheduler and then registered into the work list. Before performing a work item, the workflow engine should verify that the necessary resources are available and before launching the execution if the work item is activated by the trigger. In this case, the work item becomes an activity.

In Figure 3 we illustrate the generic architecture of the workflow management system based on web services which executes a business process of web services (i.e. BPEL4WS). The architecture is mainly divided into three layers assembled from web services.

4.1.1 The Workflow Management System Layer

We propose a workflow management system based entirely on web services. This layer provides services for monitoring,

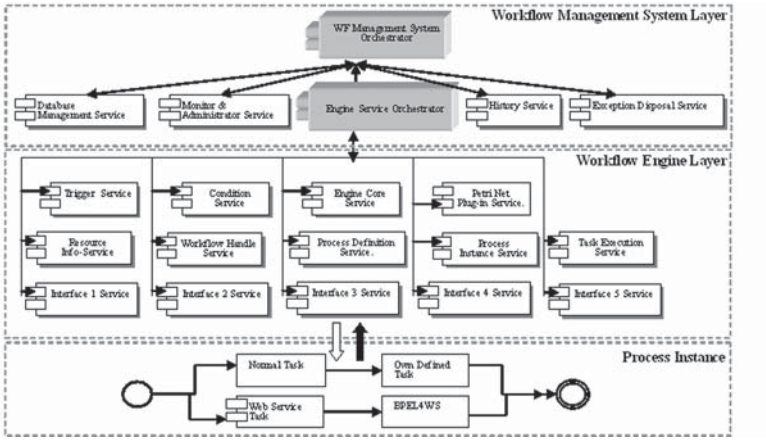

Figure 3. The general architecture of the workflow management system

administrating, logging, database management and so on. We outline two special web services: the workflow engine orchestrator and the workflow management system orchestrator. The former is responsible for the orchestration and execution of workflow engine web services. It organizes the sequence of all operations in the engine layer. The latter is responsible for the orchestration and execution of the workflow management services including the workflow engine orchestrator.

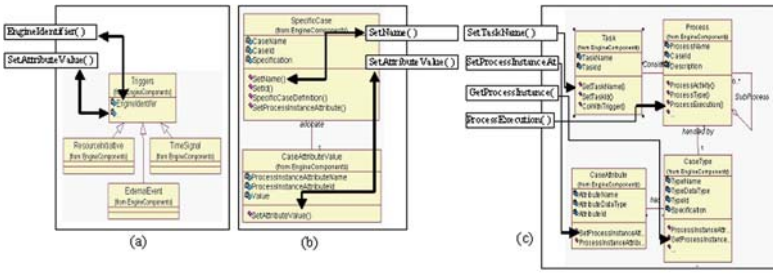

Figure 4. (a) Trigger Service, (b) Process Instance Service. (c) Process Definition Service

4.1.2 The Workflow Engine Layer

The engine layer is the heart of workflow management; it consists of a minimal set of web services providing basic operations such as: Process Instance Service, Task Execution Service, Process Definition Service, Trigger Service, Condition Service, Engine Core Service, Workflow Handle Service, and Resource Information Service. The aggregation of engine functionalities is defined by the engine orchestrator. By such, an existent service can be removed, a new service can be added and/or different operations can be invoked based on the execution context. The engine offers flexibility and is open and distributed over the network.

4.1.3 The Process Definition - Instance Layer

The third layer represents the business process definition and its instance for an eventual execution. The business process consists of tasks, a control flow, conditions and resources. The tasks could be local to the context of the enterprise or result from web services invoking remote services. BPEL4WS is an example of a process definition language where the tasks are invoked by web services.

5. The Workflow Engine Based Upon Web Services

In our work, we pay attention to workflow engine architecture; the development of the overall architecture is behind the scope of this master. We explain in detail the construction phase of web services that commonly interact to provide for a functional and agile engine.

The approach of the conception is guided by the WfMC reference model, which motivates us to encapsulate the five proposed interfaces in web services.

In Figure 5 we illustrate the diagram of classes that we conceive to model a generic workflow engine. The next step in our work consists of grouping the classes into (Grady Booch) categories. As we will explain in the following paragraphs, each category has a façade or an interface and will be encapsulated by web services.

Process Instance Service: According to a general process definition language, the process instance service generates a specific case to perform by the process once case attributes have given relevant values (Figure 4(b)).

Task Execution Service: An important step in the execution of the workflow engine relies on the preparation of resources and the scheduling of work items in the trash (work item list handler). The aim of this service is to execute an activity corresponding to a work item by receiving a trigger. The list of work is maintained by the Workflow Handle Service. During the execution of this service the orchestrator takes into account the value of the Specific Case registered in Process Instance Service.

Process Definition Service: The process definition service allows the creation of business processes. This service can be invoked by a third party application or another web service (Figure 4(c)). It is completely dependant on Process Definition Languages, such as WPDL, XLANG, Petri Net. As we design a general workflow engine, we can imagine a different implementation of this service albeit with the

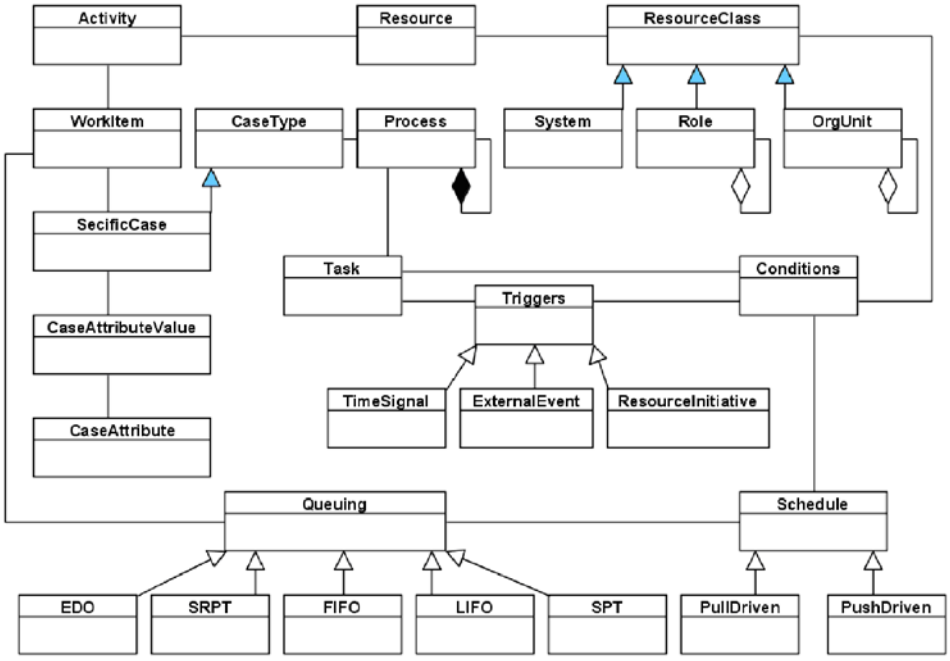

Figure 5. UML diagram class of the workflow engine

same interface, this is possible because the interface respects the WfMC reference model. Still, the inner diagram of classes presents a specific definition language. We can imagine that this service is a plug-in easy to integrate into a generic framework. In this context, the designer of a specific process definition can connect his/her tool to graphically create processes and then load them into the workflow engine. The process can be expressed by a directed graph of tasks, and a list of conditions and resources are associated with the tasks. For a specific case, the task becomes a work item and we refer to its performance as an activity.

Trigger Service: The Trigger service provides operations to manage the triggers (Figure 4(a)). By such, different triggers are implemented, for example the *resource initiative* (i.e. a user action). However, other forms of triggering are also possible: an *external event* (i.e. the arrival of an EDI message) or reaching a particular *time signal* (i.e. the generation of a list of orders at 6 o'clock). Of course, we can easily extend or specialize the set of triggers without side effect on other services.

Condition Service: Similarly, we delegate the management of condition-associated tasks to a special service called a *condition service*. Before a task can be performed as part of a specific case, it must fulfill certain conditions. A condition is therefore a requirement that must be occurring before an activity can be carried out.

Engine Core Service: The function of the Engine Core Service hinges on allocating a selected work item to an appropriate resource. There are two ways in which this can be done:

- The workflow engine matches work items and resources. Within pre-set conditions, the workflow engine can choose which resource performs each work item. The resource is thus itself unable to choose. As soon as it has finished performing one activity, it is given a new work item. We refer to this as push-driven: the engine "*pushes*" work items onto resources.
- The resources themselves match work items and resources. In this scenario, it is the resources that take the initiative. Each has studied the work items that it is able to carry out. It then chooses one. We call this pull-driven: the resources "pull out" work items and all "eat" from the same basket of work items.

Workflow Handle Service: The objective of a workflow system is to complete work items as quickly as possible. Indeed, a hold up affecting work items can result in the case as a whole taking longer. In order to transform work items into activities, two decisions always need to be made:

1. In what order are work items transformed into activities?
2. With which resources are the activities carried out?

The order can be important when selecting a resource. Conversely the choice of a resource can affect the order in which work items are transformed into activities.

The workflow handle service applies to select a particular order. In particular, the service implements various queuing disciplines for production management that are used in factories such as First-In, First-Out, Shortest Processing Time and Earliest Due Date.

Resource Information Service: The resource information service manages the resources in a workflow management system. We mainly distinguish three categories of resources: Role, System, and Organization Unit. The roles designate human resources such as operators and clients whilst the system resource presents an application or a production machine. According to our general architecture, the resource information service can be supported or extended by web services of enterprise applications. The workflow engine is thus scalable.

6. Workflow Engine Interfaces

We conceive workflow engine architecture accordingly to the WfMC reference model. After studying WfMC documentations [7], we identify the five interfaces proposed by the consortium and organize their functions and attributes into classes (see Figure 6). The classes are encapsulated to rebuild corresponding web services. Moreover, interfaces are

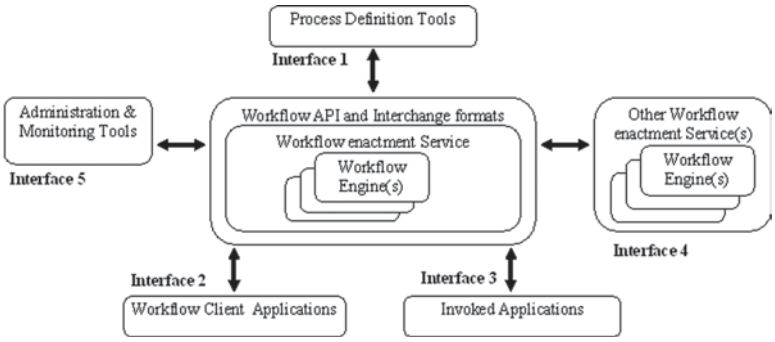

Figure 6. Workflow Interfaces (source Workflow Management Coalition (1995))

commonly standardized and accepted by most of the pure plays of workflows. They provide a gateway to uniformly communicate with the workflow engine. In the following sub-paragraphs we briefly describe the operations provided by each interface. Table 1 illustrates samples of API proposed by the WfMC.

Interface 1 Web Service (process definition tools)

The 1st interface service provides the link between the tools designed for creating and modifying workflow definitions and the workflow enactment service. It controls the system connection. In addition, it inquiries the workflow for process definitions and resource classifications.

Interface 2 (workflow client applications)

Through the 2nd interface service, users have access to a work list handler service and enactment services. In addition, the service produces states of cases and work items. The states are also controlled through this service (i.e. start, interruption, hang-up, executing, etc).

Interface 3 (invoked applications)

When some tasks of a process need to invoke an application outside of the workflow engine, they directly activate the workflow enactment service through the 3rd interface service and open interactive applications the current work item of the work list handler.

Interface 4 (other workflow enactment services)

The interoperability and communication between workflows become the bottleneck to solve with the recommendations and standards of WfMC. The 4th interface service handles the basic operations to exchange between autonomous workflow systems. Through this interface, workflows transfer cases, resources and work items.

Interface 5 (administration and monitoring tools)

The 5th interface manages the administration and the monitoring tools of the workflow system. Through this service, the executing process should be managed, well-supervised and the events should be recorded in log files.

7. The workflow engine orchestrator

The Orchestrator is considered to be the "soul" of the entire workflow engine. It controls web services in a loosely-coupled fashion. The workflow is completely agile, adaptable and scalable. The orchestrator coordinates distributed autonomous web services. Any service can be substituted by another. Existent web services published by an enterprise's applications can be easily integrated into the existent workflow engine or create a new workflow engine by assembling services. In these situations, the orchestrator is the central service that one should be modified. The orchestrator

WAPI Connection Functions	**WAPI Activity Status Functions**	**WAPI Process Status Functions**

`WMConnect()`
`WMDisconnect()`
	`WMOpenActivityInstancesList()`
	`WMFetchActivityInstance()`
	`WMCloseActivityInstancesList()`
	`WMGetActivityInstance()`
	…
		`WMOpenProcessInstancesList()`
		`WMFetchProcessInstance()`
		`WMCloseProcessInstancesList()`
		`WMGetProcessInstance()`
		…

WAPI Process Control Functions	**WAPI Worklist Functions**	**WAPI Activity Control Functions**

`WMOpenProcessDefinitionsList()`
`WMFetchProcessDefinition()`
`WMCloseProcessDefinitionsList()`
`WMFetchProcessDefinitionState()`
`WMChangeProcessDefinitionState()`
`WMCreateProcessInstance()`
`WMStartProcess()`
`WMTerminateProcessInstance()`
`WMOpenProcessInstanceStatesList()`
`WMFetchProcessInstanceState()`
`WMChangeProcessInstanceState()`
`WMFetchProcessInstanceAttribute()`
`WMAssignProcessInstanceAttribute()`
…
	`WMOpenWorkList()`
	`WMFetchWorkItem()`
	`WMCloseWorkList()`
	`WMGetWorkItem()`
	`WMCompleteWorkItem()`
	`WMOpenWorkitemStatesList()`
	`WMFetchWorkitemState()`
	`WMCloseWorkitemStatesList()`
	`WMChangeWorkitemState()`
	`WMReassignWorkItem()`
	`WMFetchWorkItemAttribute()`
	`WMGetWorkItemAttributeValue()`
	`WMAssignWorkItemAttribute()`
	…
		`WMOpenActivityInstanceStatesList()`
		`WMFetchActivityInstanceState()`
		`WMCloseActivityInstanceStatesList()`
		`WMChangeActivityInstanceState()`
		`WMOpenActivityInstanceAttributesList()`
		`WMFetchActivityInstanceAttribute()`
		`WMCloseActivityInstanceAttributesList()`
		`WMGetActivityInstanceAttributeValue()`
		`WMAssignActivityInstanceAttribute()`
		…

Table 1. Interfaces 2 and 3 of WfMC reference model

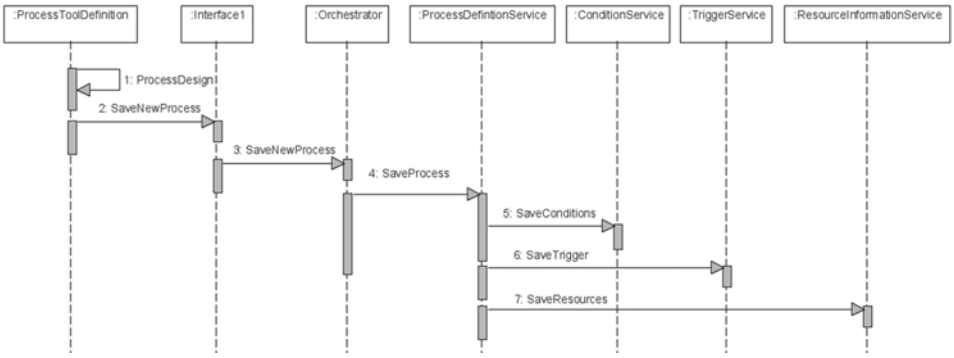

Figure 7. The orchestration operation for saving a new process

manages the context of the execution of services. It defines the order of execution and maintains the messages exchanged through execution.

The workflow engine orchestrator is implemented as is a web service. Each operation provides a particular orchestration function, for example an orchestration operation to execute a task. The orchestrator's role is crucial in our workflow architecture. The same importance is given to the orchestrator of workflow management systems, where the workflow engine orchestrator is considered as a web service to integrate with subsidiary web services in the workflow management layer. For the sake of this paper's length, we have illustrated an example of an orchestration operation to save a new process definition in the workflow engine. As shown in Figure 7 the operation is presented graphically with a UML sequence diagram

8. Implementation

We have validated the proposed architecture by the implantation of a sample prototype as illustrated in Figure 8. The prototype is meant to show how the above-mentioned ideas can be implemented. As the goal is to test ideas and not to build a complete Workflow Management System. The current version of the prototype does not contain a complete implementation of all services. However it contains implementations of the Workflow Service Layer and the Engine Service Orchestrator. The prototype is fully implemented in Java using Eclipse version 3.2.0 [5] and Web Tools Platform (WTP). The orchestrator is built on top of the ActiveBPEL™ engine [38], which is capable of executing process definitions created for the Business Process Execution Languages (i.e. BPEL4WS 1.1 specification and the WSBPEL 2.0 standard). The visual editor to manipulate workflow processes and the orchestration of loosely coupled services is built upon the Eclipse Modeling Framework (EMF) and Graphical Editing Framework (GEF) [39] which provide many features for a framework for manipulating models and creating graphical editors. The implementation of the Service Oriented Workflow prototype is approximately 30,000 lines of program code and scripting.

We conduct experimentation to test the performance of the prototype and to determine the agility of the architecture; Services of the Workflow Engine Layer can run on a single Tomcat 5.5 container server. The Workflow Engine Layer also

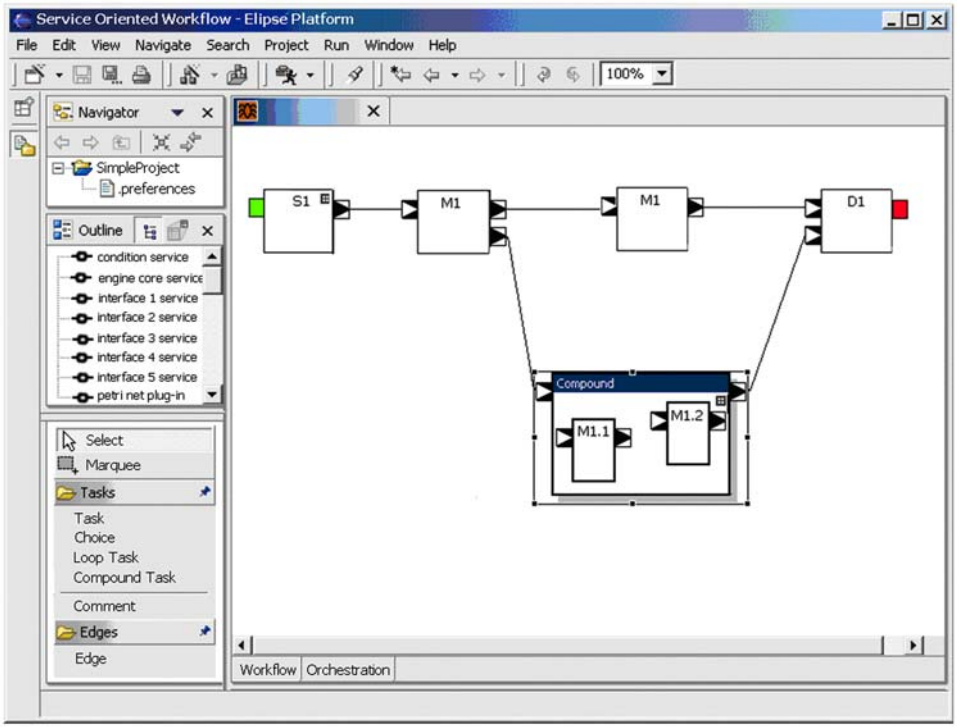

Figure 8. A screen snapshot of the prototype

allows its services to spread over distributed web container servers (i.e. in the case of cross-organization). In this case the orchestrator, which based on BPELWS, successful invokes them. Another advantage of the architecture consists of updating the code of a particular service with a new version or just replacing the current code with the code of a new algorithm. In these cases, input and output of the service as well as its exchange messages are unaffected since web services are agnostic and their interfaces are compliant with the WfMC standard. In the Trigger Service, for example, a new optimized version of the "First In First Out" algorithm replaces the old one or it can be substituted by a different algorithm such as Last In First Out (LIFO) algorithm without side-effects on the global architecture.

Several research initiatives attempt to conceive innovative workflow architectures [44, 45]. The Service–Oriented Workflow is close in spirit to the workflow architecture studied in the COMBINE project [26]. Despite the component-based approach the design is not compliant with WfMC standards. The Micro-workflow architecture presented in [22] relies on the object-oriented paradigm and depends on Smalltalk programming language [43]. Instead the Service-Oriented workflow is independent from technology platforms and relies on weak coupling of self-contained services.

Conclusion

This paper proposes a new architecture for conceiving and developing workflow management systems. The new architecture responds to new business paradigms and takes full advantage of Information and Communication Technology. It allows for the development of a distributed, adaptive and platform-independent Workflow Engine based on standards and the WfMC Common Reference Model. Hence, the architecture relies on autonomy and distributed and self-contained web services. The assembly of the services is centered in the orchestrator, which reifies an instance organization of the workflow and provides specific functionalities. We validate our architecture based on the Service-oriented approach and web services by implementing a preliminary prototype using Eclipse development platform. In future research, many issues need to be further investigated in regards to this architecture, including support for transactional workflows, versioning of workflow processes, security, resource management and communication and, lastly, the implementation of a global architecture based entirely on web services and the orchestrator which coordinates loosely-coupled services of each layer.

References

[1] Van der Aalst, W., Van Hee, K. (2004), Workflow Management Models, Methods, and Systems, Cloth edition, March 2004, 384 pp.

[2] Fischer, L. (2006), Workflow Handbook 2006, published in association with the Workflow Management Coalition (WfMC), 2006, 320 pp.

[3] Diimitrios, G., Hornick, M., Sheth, A. (1995): An Overview of Workflow Management - From Process Modeling to Workflow Automation Infrastructure, in: Journal on Distributed and Parallel Database Systems, Kluwer Academic Publishers, Boston, 3 (2), April 1995.

[4] Zur Muehlen, M., Becker, J. (1999), WPDL State-of-the-Art and Development Perspectives of a Meta-Language. In: Proceedings of the 1st KnowTech Forum, September 17-19 1999, Potsdam 1999.

[5] Eclipse Project (2006), Web Tools Platform, http://www.eclipse.org/webtools/ [last visited July, 2006].

[6] Sheth A. (1995), An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, Athens, 1995.

[7] Workflow Management Coalition (1995), The Workflow Reference Model. Document Number WFMC-TC-1003, 19-Jan-95, 1.1, Workflow Management Coalition, 1995.

[8] Zur Muehlen, M., Rob A. (2002), Embedded vs. Autonomous Workflow: Putting Paradigms into Perspective Excellence in Practice Volume IV: Innovation and Excellence in Workflow and Knowledge Management, Future Strategies, 2002, pp. 49-58.

[9] Yu J., Buyya R. (2005), A Taxonomy of Workflow Management Systems for Grid Computing, Report-no: GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, 2005

[10] Choi, J., Kwon, Y., Ryu, S., Jeong C. (2005), Workflow Management System Based on Service Oriented Components for Grid Applications. Springer, 2005.

[11] Bastin T., Savarimuthu R (2005), Integrating Web Service with Agent Based WMS IEEE, 2005.

[12] Jian Cao, Ming Li, Shengsheng Zhang, and Qianni Den. (2004) Composing Web Services Based on Agent and Workflow, Springer 2004.

[13] Xiaohui Z., Chengfei L., Yang Y. (2004). Web Service Based Architecture for Workflow Management Systems, LNCScience, Vol 3180, Jan 2004, 34-43 pp.

[14] Lazcano A., Schuldt, H., Alonso G., Schek H (2001). WISE: Process based E-Commerce. IEEE Data Bull, 24(1) (2001) 46 -51 pp.

[15] Grefen P., Aberer, K., Ludwig, H., Hoffner, Y (2001). "CrossFlow: Cross-organizational workflow management for service outsourcing in dynamic virtual enterprises," Special Issue on Infrastructure for Advanced E-Services, vol. 24, no. 1, 2001.

[16] Erl, T. (2006). Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall, 2006, 792 pp.

[17] Eric Newcomer (2002). Understanding Web Services: XML, WSDL, SOAP, and UDDI, Addison-Wesley Professional; 1st edition (May 13, 2002), 368 pp.

[18] Booch, G. (2005). Unified Modeling Language User Guide, Addison Wesley, June 2005, 475 pp.

[19] Peltz, C. (2003), Web services orchestration. a review of emerging technologies, tools, and standards., Technical report, Hewlett-Packard Company.

[20] Leymann F., Roller D., Schmidt M.-T. (2002), Web services and business process management, IBM Systems Journal 41(2), 198-211.

[21] Muehlen M. (2004): Workflow-based Process Controlling. Foundation, Design, and Application of workflow-driven Process Information Systems. Logos Verlag, Berlin 2004. 315 pages. ISBN 3-8325-0388-9.

[22] Manolescu D. (2000). Micro-Workflow: A Workflow Architecture Supporting Compositional Object-Oriented Software Development. PhD thesis, Computer Science Technical Report UIUCDCS-R-2000-2186. University of Illinois at Urbana-Champaign, October 2000, Urbana, Illinois.

[23] Casati F., Grefen P., Pernici, Pozzi, G. Sánchez G. (1997). WIDE workflow model and architecture, 1997. http://www.sema.es/projects/WIDE/Documents/

[24] Hammer M., Gerry Howe W., Kruskal V., Irving W (1977). Very high level programming language for data processing applications. Communications of the ACM, 20(11):832–840, November 1977.

[25] Ellis C., Gary J (1980). Computer science and office information systems. ACM Computing Surveys, 12(1):27 – 60, March 1980.

[26] The Combine project Web overview (2003), http://www.opengroup.org/combine/

[27]. WFMC. Workflow Management Coalition Workflow Standard (2002): Workflow Process Definition Interface – XML Process Definition Language (XPDL) (WFMC-TC-1025). Technical report, Workflow Management Coalition, Lighthouse Point, Florida, USA, 2002.

[28] Van der Aalst W.M.P., ter Hofstede A.H.M. (2005). YAWL: Yet Another Workflow Language. Information Systems, 30(4):245-275, 2005.

[29] Andrews, T., Curbera, F., Dholakia, H., Goland Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S. (2003); Business Process Execution Language for Web Services, Ver. 1.1, 2003

[30] White, S; (2003) Business Process Modeling Notation Working Draft (1.0) http://www.bpmi.org/bpmn-spec.esp

[31] Andrews T., Curbera F., Dolakia H., Goland J., Klein J., Leymann F., Liu K., Roller D., Smith D., Thatte S., Trickovic I., Weeravarana S. (2003). Business Process Execution Language for Web Services, 2003.

[32] W3C. Web service choreography interface 1.0 (2002). http://www.w3.org/TR/wsci/

[33] BPML.org, Business process modeling language (2002). http://www.bpmi.org/, 2002.

[34] Thatte S. (2001) Microsoft Corporation. Xlang: Web services for business process design. http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

[35] Leymann F. (2001), Web Services Flow Language, http://www-306.ibm.com/software/solutions/webservices

[36] Yongyi X. , Weishi Z., (2003) Component-Based Workflow Architecture of a Distributed Software Process Management System," qsic, p. 204, Third International Conference On Quality Software, 2003

[37] Balasooriya J., Joshi J., Prasad S., Navathe S. (2006), "A Two-Layered Software Architecture for Distributed Workflow Coordination over Web Services, pp. 933-934, IEEE International Conference on Web Services 2006

[38] ActiveBPEL Open Source Engine Project (2007), http://www.active-endpoints.com/active-bpel-engine-overview.htm, 2007

[39] Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework, IBM Redbooks (2007), http://www.redbooks.ibm.com/, 2007

[40] Rickayzen A., Dart J., Brennecke C., Schneider M.(2002), Practical Workflow for SAP - Effective Business Processes using SAP's WebFlow Engine, Galileo Pr Inc (July 27, 2002), 552 pages, ISBN-10: 159229006X

[41] Lotus Workflow Overview, IBM (2007), http://www.ibm.com/lotus/workflow, 2007

[42] TIBCO, InConcert (2002). http://www.tibco.com/products/in_concert/, 2002

[43] Cincom visualworks documentation. Cincom Systems, Inc. Available on theWeb from http://www.cincom.com/visualworks/documentation.html.

[44] Yongyi, X. Weishi Z. (2003) Component-Based Workflow Architecture of a Distributed Software Process Management System. QSIC 2003: 204-210

[45] Rajesh K., Lan Z., Taezoon P., Sebastien G., (2007) A Web Service-Enabled Distributed Workflow System for Scientific Data Processing, 11th IEEE International Workshop on Future Trends of Distributed Computing Systems, pp. 7-14, 2007

[46] Hodges R., Fisher W., (2005), Integrated Management & Workflow: Software Directory Columns, Gartner Gartner Research documents, 2005.