
Towards a workflow management system for service oriented modules

Leonardo Salayandía

The Pan American Centre for Earth and Environmental Studies,
University of Texas at El Paso,
El Paso, Texas 79968-0518, USA
E-mail: leonardo@utep.edu

Ann Q. Gates*

Department of Computer Science,
University of Texas at El Paso,
El Paso, Texas 79968-0518, USA
E-mail: agates@utep.edu

*Corresponding author

Abstract: With the emergence of service-oriented architectures (SOAs), workflow management systems are being used to create applications from service compositions. Because service compositions may extend across platforms, domains, and virtual organisations, classical implementations of workflow management systems must be extended to address interoperability and reliability. In particular, there is a need to define interoperable and scalable communication mechanisms to support workflows over loosely-coupled environments and to provide support for runtime verification to increase user confidence in workflow behavior. This paper presents a road map towards creating a scientific workflow management system that leverages SOA and runtime verification technologies.

Keywords: Service Orientation (SO); Service-Oriented Architecture (SOA); scientific workflows; Web Services (WS) composition; workflow patterns; constraint specification; runtime verification.

Reference to this paper should be made as follows: Salayandia, L. and Gates, A.Q. (xxxx) 'Towards a workflow management system for service oriented modules', *Int. J. Simulation and Process Modelling*, Vol. x, No. x, pp.xxx-xxx.

Biographical notes: Leonardo Salayandía received his MS in Computer Science from the College of Engineering at the University of Texas at El Paso (UTEP) in 2002. He is pursuing a PhD in Computer Science from UTEP, and is currently employed as a Research Specialist for the Pan American Centre for Earth and Environmental Studies funded by the National Aeronautics and Space Administration (NASA). He is also involved in the NSF-funded GEON project that is building cyber infrastructure for the geosciences. His research interests are in service orientation and software development practices.

Ann Q. Gates received the BS in Mathematics from the College of Science and the MS in Computer Science from the College of Engineering at the University of Texas at El Paso (UTEP). She received the PhD in Computer Science from the College of Arts and Sciences at New Mexico State University. She is currently employed as a professor at UTEP. Her research interests are in software-fault monitoring and grid computing. She is a member of the NSF-funded GEON research team that is building the cyber infrastructure for the geosciences. She is an IEEE-CS Certified Software Development Professional and a member of the IEEE-CS Board of Governors.

1 Introduction

With the emphasis on increasing resource usage across organisations and the emergence of Web Services (WS) technologies, SO is rising as the main alternative to the development of distributed, loosely-coupled software. The term SO refers to the level of abstraction in which functionality is specified. In particular, SO is an approach for analysis, design, and development of modules that support principles such as reusability, loose coupling, abstraction, and separation of concerns (Erl, 2005). The more familiar term SOA is used to describe “the policies, practices, and frameworks that enable application functionality to be provided and consumed as sets of services” (Sprott and Wilkes, 2004).

Application developers from business and scientific domains are using WS to implement systems based on the SOA paradigm. Web service technologies provide the necessary mechanisms to expose shareable resources (service-oriented modules that provide data and functionality) over the network and allow the resources to be consumed by users across heterogeneous platforms, enhancing interaction across organisations. As the number of shareable resources increases and as service consumer applications grow in complexity, there is a need to define formal venues for connecting services and supporting interoperability of interfaces among services.

Workflow technology is one approach that has been pursued to address the problems of composition and interoperability of services. A workflow specifies a group of activities and their order of execution with respect to each other, where an activity is considered an atomic unit of work (van der Aalst et al., 2003). In a scientific setting, a workflow usually is driven by data flow, and activities represent automated services, e.g., a dataset transformation module. In other words, an activity is triggered by data streams flowing as input and the activity requires limited or no user interaction.

Workflow management systems provide functionality to access reusable modules, support specification of workflows, and manage workflow execution. Implementing a workflow management system that hosts effective workflows over service-oriented modules presents the following challenges:

- maintaining the interoperability goals of SO
- allowing users with limited technical background to compose workflows from independent, third-party modules
- giving users the confidence that the workflow execution results in the intended behaviour.

The paper is organised as follows. Section 2 provides background information and related work on workflow management systems, and Sections 3–5 describe methods to address the challenges described above. In particular, Section 3 describes an approach to implement communication mechanisms among compute-bound distributed modules (services with limited user interaction requirements) that support the interoperability goals of SO; Section 4 describes functionality necessary to guide the user in composing workflows; and Section 5 describes an approach for verifying workflows at runtime. The paper concludes with a scenario that illustrates the application of the presented concepts and a summary.

2 Background

This section provides an overview of workflow management systems and presents related work in service composition.

2.1 Workflow management systems

Workflow management systems provide functionality to access reusable modules, support specification of workflows, and manage workflow execution. Systems like the Kepler Scientific workflow management system (Ludäscher et al., 2005) provide a stand-alone module that incorporates all of the aforementioned functionality. Other systems like the IBM Domino suite (Lotus Development Corporation, 1999) extend the basic setting to a client-server environment. In a stand-alone environment, the user downloads and installs the management system on his or her local platform. The client-server environment, on the other hand, consists of a central framework that is hosted on a high-end server platform. The central framework is responsible for maintaining the workflow specifications, executing the workflow activities, and maintaining their state. Users connect to the central framework through client or web applications in order to create a workflow specification and monitor its execution. Because different users can connect to a single central management system, the client-server setting has the additional benefit of enhancing team collaboration for the management and execution of workflows. The client-server setting is well-suited for domains where multiple-user interaction is a key feature, while the stand-alone environment facilitates experimentation on a controlled, local environment, a feature that is desirable in a scientific domain.

The range of reusable modules that may be involved in the workflow is limited by the connectivity interfaces that the workflow management system supports. Typically, a workflow management system utilises a registry of available modules that the user can plug into a workflow. The registry may be internal or external. Maintaining an internal registry of modules provides the benefits of assuring that a module is compatible with the workflow framework and that the framework has sufficient information about the module's interface to guarantee interoperability with other registered modules. In other words, if a module does not meet the interface requirements of the workflow management system, then the module cannot be included in the internal registry. For example, the Kepler system includes an internal library of 'actors' that provide the functionality that can be plugged into Kepler workflows. An external registry presents an alternative that is more aligned to SO. The registry presents clients with a description of

the functionality provided by a given module and the interface requirements imposed by that module. This description serves as a contract between the module and the client of the module. Also, it allows third-party modules to be added to the registry without restrictions imposed by the workflow management system. In the case of an external registry, interoperability between modules is achieved through standardisation of communication protocols.

2.2 *Related work*

The Kepler Project (2005) is a cross-project collaboration that includes several organisations that are developing modules to be used in scientific workflow applications. Kepler is a stand-alone system based on Java and the Ptolemy II framework (2005). Kepler/Ptolemy II use the notion of ‘actors’ as an abstraction for the modules that are plugged into a workflow. The main limitation of this work is that it does not scale well to distributed environments or to functionality provided by third-party modules. The METEOR-S project (Verma et al., 2005) proposes an automatic process generation using planning based on quantitative and non-quantitative process constraints. This work assumes a well understood business model which allows the workflow management system to make decisions on behalf of the human user. Although METEOR-S shows promise for business domain applications, the methods used for automation would counter the flexibility of exploration requirements of a scientific domain. The GridMiner project (2005) is focused on scientific knowledge discovery through the execution of the composition of services that incorporate the basic elements of the knowledge discovery process. The user constructs a script that customises the knowledge discovery process to a particular scenario and a workflow engine executes the script, resulting in the composition of the knowledge discovery services. Although the workflow management component of the GridMiner project can potentially be generalised to workflow composition (Kicking et al., 2003), GridMiner falls short on abstracting the technical details, requiring the user to learn a scripting language and have a computer programming background. Finally, there have been numerous proposals of frameworks aimed at service composition. Some examples include the Grid Service Flow Language (GSFL) (Krishnan et al., 2002) from the Globus Alliance organisation, XCAT project (2005) from Indiana University Extreme Lab, and the Business Process Execution Language for WS (BPEL4WS, 2003) proposed by industry leaders such as IBM, Microsoft, among others. Although these frameworks form a constituent part of a workflow management system, these frameworks do not contemplate user-friendly interfaces that would permit average users to access reusable modules, support specification of workflows, and manage workflow execution.

3 **Interoperable communication mechanisms**

Before discussing communication mechanisms that support interoperability, it is necessary to understand the types of data used in workflows and to describe the issues surrounding workflow execution with respect to data flow. This section begins by elucidating these issues.

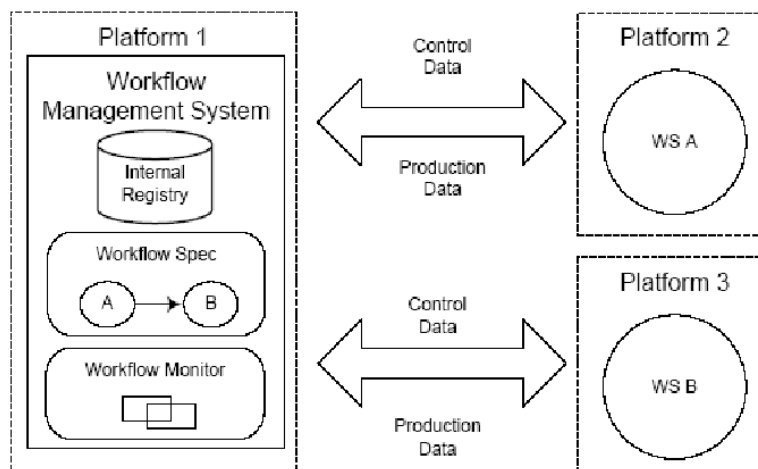
Workflow execution involves two kinds of data: control data and production data (van der Aalst, 2003). The sole purpose of control data is to manage how services relate

to one another, e.g., as specified in the execution control patterns described later in this paper. In addition, control data may initiate or terminate user-related operations, such as monitoring the workflow execution. Production data, on the other hand, does not depend on workflow management. Rather, it refers to the data that is produced by the services involved in the workflow. While control data interacts with the central framework in order to present the user with feedback information, it is not necessary to communicate production data back to the central framework between workflow services unless the user explicitly makes a request, e.g., to monitor data.

A problem with using service-oriented modules in a workflow that was originally designed for a local domain is that communication bottlenecks can arise if all data produced by workflow services are routed through the platform that hosts the central workflow management system. This issue is not a problem when dealing with services that are hosted within a local platform or organisation, since fast interconnections exist; however, it does present a problem when dealing with remotely hosted services. The problem may be aggravated in a scientific setting, where workflows can produce heavy data traffic between services. Moreover, workflow implementations that were originally designed for a local domain do not take full advantage of an SOA setting because the usability of services is limited by the communication capabilities of the platform that hosts the workflow management system.

Figure 1 shows an implementation of a workflow management system that includes an internal registry of modules and provides monitoring capabilities for the execution of workflows. In this example, the user implements a workflow specification by selecting references to Web Services A and B (WSA and WSB, respectively) from the internal registry. Once workflow execution starts, WSA and WSB are instantiated through proxy stubs that are hosted on the workflow management system (cf. Platform 1). The proxy stubs are responsible for bridging communication between Platform 1 and WSA and WSB, which are remotely hosted on Platforms 2 and 3 respectively. Since the workflow management system is the sole initiator of workflow services, all communication of control and production data is routed through it. As a result, WSA and WSB never interact directly with each other.

Figure 1 A workflow management system intended for local domain modules adapted to distributed service-oriented modules



The emergence of SOA technologies and the standardisation of XML-based messaging protocols like the Simple Object Access Protocol 1.1 (SOAP, 2000) have facilitated the development of basic distributed applications across heterogeneous platforms connected through the Internet. The SOAP 1.1 messaging protocol has limitations, e.g., it is difficult to develop standard-compliant SOA applications that include stateful entities and that require asynchronous communication patterns.

Because of the customisable nature of XML and the SOAP messaging protocol in particular, attributes can be defined on the protocol to address any limitations. Customising the messaging protocol for a particular application, however, may result in reduced reusability of resources by entities that are not privy to the enhancements. Other techniques to overcome the SOAP messaging protocol limitations can be applied at the transport level. For example, in order to address stateful entities, the programmer may choose a traditional web development technique that consists of appending an HTTP request to include a session ID number. This binds the SOA application to a specific transport protocol and creates a loss in interoperability.

Standardisation of the messaging mechanisms is essential in order to maintain the interoperability of SOA applications. The community has responded with proposals for standards such as Web Services Addressing (WS-Addressing, 2004) and the Web Services Resource Framework (WSRF, 2004) that overcome the current limitations of WS. WS-Addressing proposes mechanisms to reach end-point references through the SOAP protocol that are flexible enough to accommodate asynchronous communication patterns. WSRF consists of a family of specifications that enable access to stateful resources through WS conventions.

The WS-Addressing standard introduces two constructs to the SOA XML-based protocols:

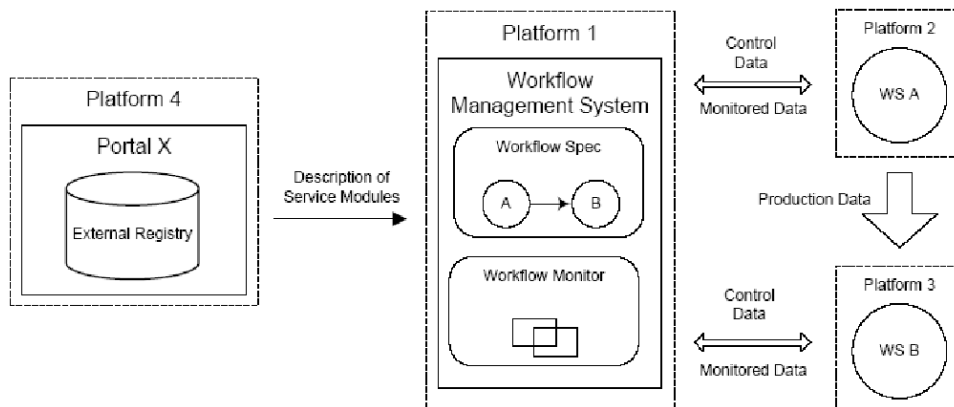
- the end-point-reference construct that conveys information required to reference a service resource
- the message-information construct that defines message characteristics that extend the communication patterns that can be employed, asynchronous communication in particular.

An example that illustrates the benefits of using these constructs is given in Figure 2. The workflow management system in this example allows the user to query and extract description details of third-party service modules from an external registry. Furthermore, given the support for asynchronous communication mechanisms, WSA is able to communicate with WSB using the aforementioned constructs, reducing communication costs and promoting interoperability. With this implementation, the platform that hosts the management system does not have to deal with the heavy production data traffic that may result from the activities involved in a scientific workflow. As a result, the system requirements on the workflow management system's platform (cf. platform 1 in Figure 2) are not as demanding as implementations comparable to the one illustrated in Figure 1. Gannon et al. (2002) explore similar communication mechanisms in the context of the XCAT framework, which is a Common Component Architecture implementation.

Event-driven messaging is commonly used for inter-object communication, so when the occurrence of a defined event is encountered, an action is triggered that alerts other related objects. In SOA, event-driven messaging is being standardised in the form of a publish/subscribe mechanism based on the observer pattern proposed by

Gamma et al. (1994). Web Services Notification (WS-Notification, 2004) is the standard that is defining the message interfaces that allow a web service to publish topics and to have clients subscribe their interest in order to get notified at appropriate times. With such mechanisms in place, it is possible to have the workflow management system receive notification messages of events of interest published by service modules and, as a result, the user or monitor can be given feedback as a workflow is executing.

Figure 2 A workflow management system that employs asynchronous communication mechanisms over SOA



4 Workflow composition

In order for a scientific workflow management system to be effective, it must include functionality that allows non-technical users to:

- locate and access modules of interest
- plug them into their particular projects.

The issues related to this functionality are discussed next.

4.1 Module discovery

Typically, a workflow management system utilises a registry of available modules that the user can plug into a workflow. Registries assist the user in ‘discovering’ modules to use on a workflow. With an internal registry, modules may be categorised into intuitive subclasses, allowing the user to navigate through the registry to find modules of interest. On external registries, the categorisation of modules is done at the host level. For example, an external registry could be hosted through a portal that provides services related to the particular interests of the user. The user may browse the web across different portals to find modules of interest.

In a service-oriented environment, the workflow management system is required to provide functionality that permits the user to find appropriate modules that have been developed by third-party organisations and abstract the essential details of modules in order to integrate them to a workflow. As a result, a workflow management system designed to take advantage of an SOA environment would delegate the functionality of module discovery to external registries that are maintainable by communities of interest. Otherwise, maintaining an internal registry of modules as part of the workflow management system would limit the potential range of reusable modules available over the web and would place the registry maintenance burden on a reduced group of developers and administrators.

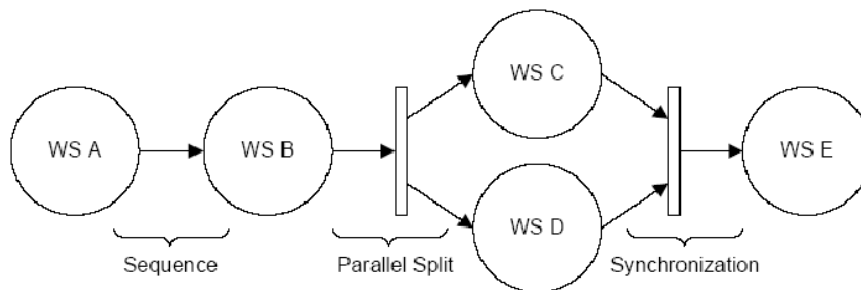
As the use of SOA continues to become ubiquitous, communities of interest are forming cyber infrastructures hosted over virtual organisations and presented as portals over the web. One such example from the Earth science community is the GEOsciences Network (GEON, 2005). The vision of GEON is to create a cyber infrastructure of resources that will facilitate the interdisciplinary collaboration of Earth science efforts. As community contributions increase the resources managed by GEON and as ontologies are fully developed to support advanced search capabilities, the GEON portal will become *the* repository of resources for the Earth sciences.

Other research approach the discovery and interoperability problems from the standpoint of semantic-aware applications (Berners-Lee et al., 2001; Denny, 2002).

4.2 Order of execution and filtering mechanisms

Once individual modules have been identified, the user composes a workflow by specifying the order of execution and the interaction between modules. With the intention of improving usability, the workflow management system should provide intuitive constructs to specify order of execution and provide functionality that allows the user to map inputs to outputs between modules. Figure 3 shows a workflow specification that exemplifies three of the basic workflow patterns described by van der Aalst et al. (2003). First, the Sequence workflow pattern is illustrated by WSA and WSB where, as WSA terminates WSB is enabled. Second, the Parallel Split pattern is shown by WSB, WSC and WSD where, as WSB terminates, WSC and WSD are both enabled. Finally, the Synchronisation pattern is illustrated by WSC, WSD, and WSE where both WSC and WSD must terminate before WSE can be enabled.

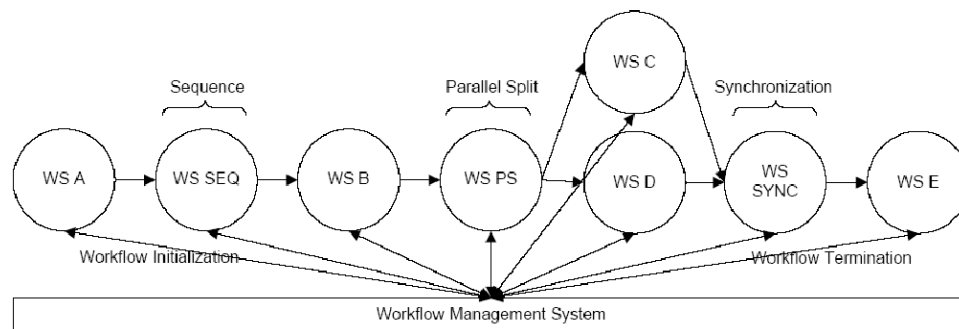
Figure 3 A workflow specification with three basic workflow patterns



By providing basic constructs or patterns for workflow composition along with a complementary graphical user interface that allows the user to read the interaction requirements of the modules and map inputs to outputs, users should be able to specify order of execution and interaction between modules without an extensive technical background. The proposed approach is to have specialised services that can serve as intermediaries between service modules to implement workflow patterns and facilitate the filtering mechanisms. Such services would be controlled by the workflow management system and would be instantiated based on the workflow specification defined by the user. After the user chooses all the services that are to be part of the workflow, the management system provides the user with tools to specify the filters needed to ensure that message requirements of connecting services are met. The management system will then automatically customise an interfacing web service, which has the task of maintaining the appropriate composition control specified in the workflow and processing messages as necessary between interacting services.

Consider the scenario depicted in Figure 4. This diagram represents an implementation of the workflow specification previously discussed (cf. Figure 3) with the aid of three intermediary WS, the Sequence web service (WSSEQ), the Parallel Split web service (WSPS), and the Synchronisation web service (WSSYNC). WSSEQ filters the message received and generates a new message request to WSB with an instruction to reply to WSPS, the next workflow pattern indicated in the specification. The WSPS service receives the response message from WSB and filters the message accordingly to produce proper request messages to WSC and WSD. WSPS generates a request message for each parallel split service and has the service reply to the next intermediary service in the queue, i.e., WSSYNC. In the case of WSSYNC, the service waits until it receives the two response messages from WSC and WSD and formulates the next request message to WSE with an instruction to reply back to the workflow management system, which indicates the termination of this workflow execution.

Figure 4 Implementation of a workflow specification with intermediary Web Services



It is important to note that the services in the workflow need not know anything about their interaction with other services. For example, WSA is not aware that its results are utilised by WSB. This is an essential characteristic resulting from employing SOA, where services are developed and maintained by independent virtual organisations and the only thing in common between their implementations is their adherence to SOA standards, such as SOAP and WS-Addressing.

5 Verification of workflow execution

The development and consumption of service modules in SOA result in additional complexity not existing in other classical distributed environments: the developer does not know the context in which the service module will be used, and the consumer does not know how a service module was developed. Determining whether a service-oriented workflow yields the intended results requires viewing the workflow from several aspects: the correctness of the individual services that comprise the workflow; the correctness of the data; and the correctness of the specified ordering of service execution. Standard verification techniques are not sufficient. As suggested by Tsai et al. (2005), new software engineering techniques are needed to target the development, testing, and publishing processes of a service module prior to being offered to clients. Given the dynamic nature of the SOA environment, continuous evaluation of service modules is one approach for achieving a level of trust. As workflows move towards SOA, the runtime verification of service modules becomes even more crucial. The user must have assurance that service modules employed behave as expected, that retrieved data is correct, and that the integration of service modules yields correct results. A workflow management system can play an important role in providing such assurance using runtime verification techniques.

Numerous researchers have studied runtime verification. A comprehensive survey has been conducted by Delgado et al. (2004). Efforts that specifically target monitoring of service modules have focused mainly on quality-of-service issues, e.g. (Gardner et al., 2004; Tierney, 2004; Deutsch et al., 2004). In addition, Robinson (2003) is investigating monitoring of service modules for quality of service, providing a stable infrastructure for monitoring at the service level. Preliminary results of this research have shown that it is possible to monitor low-level requirements failure, data violations, workflow anomalies, performance degradation, and other non-functional requirements at the service level.

Runtime verification in an SOA environment is limited because the service client only has access to the external, static interface of the service module. This is sufficient when monitoring the integration of service modules, i.e., the user can attach pre- and post-conditions to verify service module interaction. Monitoring a service module beyond pre- and post-conditions requires an ‘observable service’, i.e., one that contains hooks or probes that identify points of interest in the service module. The probes can be based on events associated with change in state of variables and user-defined checkpoints. Checkpoints are points in the code that signal the beginning or end of a task being modelled by the code. The service-module developer is responsible for identifying observable variables and checkpoints. With this approach, developers of the service can specify properties on the service, e.g., properties that capture assumptions and limitations of the service. The user of the service module can select the developer-defined properties to be monitored, and he or she can add new properties. This approach assumes that scientists have developed ontologies to define vocabularies, capture attributes and axioms of entities, and identify relationships among these entities. Because the services are primarily domain specific, it is conceivable that the developer will be able to map variables to pre-defined and well-understood concepts.

To realise runtime verification of workflows, the workflow management system must include a runtime monitor, and it must have an interface that assists non-technical users with the following functionality:

- accessing the probes, checkpoints, and properties of selected service modules
- specifying pre- and post-conditions on services that check the communication between service modules
- supporting the specification of formal properties, e.g., using tools such as Specification Pattern System (SPS) (Dwyer et al., 1998) or Property Specification Pattern System (Prospec) (Mondragon and Gates, 2004; Mondragon, 2004; Salamah et al., 2005).

6 A workflow scenario

The following scenario describes the interaction between a user and the proposed workflow management system to construct, execute, and monitor a workflow. When appropriate, implementation details are included to demonstrate the feasibility of the system.

In this scenario, the user starts by looking for appropriate service modules that may be used in the desired workflow application. The user searches particular communities of interest by browsing through their portals and attains description information of service modules in the form of URI's that reference Web Service Description Language (WSDL) documents. Services WSA, WSB and WSC are identified at this stage.

The user launches the workflow management system and initialises it by starting a new workspace on which the user creates three new objects that represent the service modules of interest. The user customises each of the objects by completing the WSDL properties of the objects to match the collected URI's. The workflow management system fetches the WSDL descriptions of each of the objects and customises the graphical representation of the objects to include the input and output fields offered by each given service, as well as additional observable variables that the service may offer for monitoring purposes.

The user decides that the workflow will consist of a sequential execution of A, followed by B and ending with C. The user connects the object representations of the services with connectors that represent the Sequence workflow pattern. Each connector represents an intermediary service associated with the workflow pattern. The workflow management system displays a window with the output fields of A on the left and the input fields of B on the right, allowing the user to customise the intermediary service. The workflow management system abstracts message formatting requirements from the user's view. The user can then choose to do direct mappings between output and input fields or to specify some mathematical manipulation as part of the interaction between A and B. The user chooses to specify post-conditions on the output variables of WSA to verify that the data meets the specified conditions, and the user specifies a pre-condition on the input variables of WSB to ensure that the data does not become corrupted during transmission.

After the user finishes specifying the interactions for all the modules involved, the user takes advantage of the observable variables offered by WSC to monitor its internal behaviour. As a result, the user refers to object C and the description of its observable variables. The user decides to specify properties on observable variable C-1 to determine whether service C is operating as expected during workflow execution. Since property

specification normally requires knowledge of formal verification techniques, the workflow management system provides functionality to guide the specification process through a graphical user interface such as one provided by Prospec.

Once the workflow specification is complete, the user initiates workflow execution by applying the appropriate command on the workflow management system. The system launches intermediary services hosted on preconfigured high-end platforms to implement the specified workflow connectors employed and to enforce the specified filtering processes. When workflow execution begins, a runtime-verification module is also launched. The runtime verification module provides feedback to the workflow management system when a property is violated. One possible implementation of the runtime verification module is MaC (Kim et al., 2004), a functional prototype that instruments the system with a filter and generates an event recogniser and a checker based on the monitoring constraints specified by the user.

7 Summary

SOA is becoming omnipresent in many application domains. This paper describes a workflow management system that supports workflows built on service-oriented modules. Although there are different frameworks that provide the base functionality required to execute such workflows, a workflow management system goes beyond these frameworks by incorporating functionality that allows non-technical users to access reusable modules, support specification of workflows, and manage workflow execution.

Implementing an effective workflow management system over SOA presents the following challenges:

- maintaining the interoperability goals of SO
- allowing users with limited technical background to compose workflows from independent, third-party modules
- giving users the confidence that the workflow execution results in the intended behaviour.

This paper presented methods that can be implemented by standards-based protocols of the SOA domain in order to achieve composition of independent services and provide efficient and reliable solutions to the user. A workflow communication mechanism was described that utilises the WS-Addressing standard and that allows independent services to be composed into an interoperable application. Furthermore, the WS-Notification standard was presented, which can be employed to implement event-driven communication between service modules and the workflow management system to enhance runtime verification of workflows.

Composing a workflow application from independent services raises the issue of discovery of services and interoperability. This paper suggests the use of external registries, possibly provided by portals developed by domain experts from a community of interest, for service discovery. Also, functionality is described that permits non-technical users to assemble independent service modules into workflows and to manually set filters between interacting services.

Since SO hides the implementation details from the user, the question of trusted services is raised. WS Testing, Reliability Assessment and Ranking (WebStrar, 2005) is addressing this issue by proposing new software engineering techniques that have the objective of giving a reliable measure of trustworthiness to the user. The approach described in this paper uses run-time monitoring to support verification of software with respect to a set of user-defined properties. Monitoring workflow execution provides an extra line of defence against faulty services and data that may be inherent in the service or caused by unreliable networks and malicious behaviour. The runtime verification of workflows, using properties chosen by the user, will likely improve the user's confidence on the composed application.

Acknowledgement

This work is funded in part by the National Aeronautics and Space Administration (NASA) under grant NASA NCCS-498 through the Pan American Centre for Earth and Environmental Studies (PACES), and the National Science Foundation (NSF) through grant NSF EAR-0225670. The authors would like to acknowledge the recommendations of three anonymous referees.

References

- Berners-Lee, T., Hendler, J. and Lassila, O. (2001) 'The semantic web', *Scientific American*, Vol. 234, May, pp.34–43.
- BPEL4WS (2003) *Business Process Execution Language for Web Services, Version 1.1*, <http://www-128.ibm.com/developerworks/library/specification/ws-bpel>.
- Delgado, N., Gates, A. and Roach, S. (2004) 'A taxonomy and catalog of runtime software-fault monitoring tools', *IEEE Transactions on Software Engineering*, Vol. 30, No. 12, December, pp.859–872.
- Denny, M. (2002) *Ontology Building: A Survey of Editing Tools by Michael Denny*, O'Reilly's XML.com at <http://www.xml.com/pub/a/2002/11/06/ontologies.html>.
- Deutsch, A., Sui, L. and Vianu, V. (2004) 'Specification and verification of data-driven web services', *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Paris, France, pp.71–82.
- Dwyer, M.B., Avrunin, G.S. and Corbett, J.C. (1998) 'Property specification patterns for finite-state verification', *Proc. 2nd Workshop on Formal Methods in Software Practice*, Clearwater Beach, Florida, pp.7–15.
- Erl, T. (2005) *Service Oriented Architecture: Concepts, Techniques, and Design*, Prentice-Hall, Englewood Cliffs, New Jersey.
- Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1994) *Design Patterns*, October, Addison Wesley Professional Computing Series, Berkley, CA.
- Gannon, D., Ananthakrishnan, R., Krishnan, S., Govindaraju, M., Ramakrishnan, L. and Slominski, A. (2002) *Grid Web Services and ApplicationFactories*, <http://www.extreme.indiana.edu/xcat/publications/AppFactory.pdf>.
- Gardner, M.K., Deng, W., Markham, T.S., Feng, W. and Reed, D.A. (2004) 'A high fidelity software oscilloscope for globus', presented at the *GlobusWorld Conference*, January.
- GEON (2005) *GEON: Cyberinfrastructure for the Geosciences*, <http://www.geongrid.org>.

- GridMiner project (2005) <http://www.gridminer.org>.
- Kepler Project (2005) <http://kepler-project.org>.
- Kickingger, G., Hofer, J., Brezany, P. and Tjoa, A.M. (2003) 'Workflow management in GridMiner', *3rd Cracow Grid Workshop*, Cracow, Poland, October 27–29.
- Kim, M., Kannan, S., Lee, I. and Sokolsky, O. (2004) 'Java-MaC: a run-time assurance tool for java programs', in Havelund, K. and Rosu, G. (Eds.): *Proc. the Runtime Verification Workshop*, Vol. 55, No. 2, pp.97–104.
- Krishnan, S., Wagstrom, P. and von Laszewski, G. (2002) *GSFL: A Workflow Framework for Grid Services*, <http://www.globus.org/cog/papers/gsfll-paper.pdf>.
- Lotus Development Corporation (1999) *Domino Workflow Automating Real-World Business Processes*, white paper, September, <ftp://ftp.lotus.com/pub/lotusweb/eibu/dominoworkflow.pdf>.
- Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J. and Zhao, Y. (2005) 'Scientific workflow management and the Kepler system', *Concurrency and Computation: Practice and Experience, Special Issue on Workflow in Grid Systems*, John Wiley & Sons Ltd., Vol. 18, No. 10, pp.1039–1065.
- Mondragon, O. (2004) *Elucidation and Specification of Software Properties through Patterns and Composite Propositions to Support Formal Verification Techniques*, PhD Dissertation, the University of Texas at El Paso, May, El Paso, Texas, USA.
- Mondragon, O. and Gates, A. (2004) 'Supporting elicitation and specification of software properties through patterns and composite propositions', *Intl. Journal Software Engineering and Knowledge Engineering*, Vol. 14, No. 1, pp.21–41.
- Ptolemy Project (2005) <http://ptolemy.eecs.berkeley.edu>.
- Robinson, W. (2003) 'Runtime monitoring of web service requirements', *Proc. 11th IEEE Int'l Requirements Engineering Conference*, pp.65–74.
- Salamah, S., Gates, A.Q., Roach, S. and Mondragon, O. (2005) 'Verifying pattern-generated LTL formulas: A case study', *Proc. 12th Int'l SPIN Workshop*, San Francisco, CA, USA, pp.200–220.
- SOAP (2000) *Simple Object Access Protocol 1.1*, <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>.
- Sprott, D. and Wilkes, L. (2004) 'Understanding Service-Oriented Architecture', *Microsoft Architect Journal*, January, Microsoft Corporation, Microsoft Developer Network, January 2004, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnmaj/html/aj1soa.asp>.
- Tierney, B. (2004) 'Grid troubleshooting with the NetLogger toolkit', presented at the *GlobusWorld Conference*, January.
- Tsai, W.T., Chen, Y. and Ray, P. (2005) 'Specification-based web services verification and validation', *Proc. the 10th IEEE Int'l Workshop on Object-oriented Real-time Dependable Systems*, February, pp.163–171.
- van der Aalst, W., ter Hofstede, A., Kiepuszewski, B. and Barros, A. (2003) 'Workflow patterns', *Distributed and Parallel Databases*, Vol. 14, No. 3, July, pp.5–51.
- Verma, K., Gomadam, K., Sheth, A.P., Miller, J.A. and Wu, Z. (2005) *The METEOR-S Approach for Configuring and Executing Dynamic Web Processes*, Technical Report, <http://lsdis.cs.uga.edu/projects/meteor-s/>.
- WebStrar (2005) *Web Services Testing Reliability Assessment, and Ranking*, <http://asusrl.eas.asu.edu/srlab/projects/webstrar>.
- WS-Addressing (2004) *Web Services Addressing*, <http://www.w3.org/Submission/ws-addressing>.
- WS-Notification (2004) *Web Services Notification*, <http://www-106.ibm.com/developerworks/library/specification/ws-notification>.
- WSRF (2004) *Web Services Resource Framework*, <http://www.globus.org/wsrfl>.
- XCAT Project (2005) <http://extreme.indiana.edu/xcat/index.html>.