Scientific Workflow: A Survey and Research Directions

Adam Barker and Jano van Hemert

{a.d.barker,j.vanhemert}@ed.ac.uk
National e-Science Centre, University of Edinburgh, United Kingdom

Abstract. Workflow technologies are emerging as the dominant approach to coordinate groups of distributed services. However with a space filled with competing specifications, standards and frameworks from multiple domains, choosing the right tool for the job is not always a straightforward task. Researchers are often unaware of the range of technology that already exists and focus on implementing yet another proprietary workflow system. As an antidote to this common problem, this paper presents a concise survey of existing workflow technology from the business and scientific domain and makes a number of key suggestions towards the future development of scientific workflow systems.

1 Introduction

Service-oriented architectures are a popular architectural paradigm for building software applications from a number of loosely coupled, distributed services. By loosely coupled we mean that the client of a service is essentially independent from the service itself. When a client (which can be another service) makes an invocation on a remote service, it does not need to concern itself with the inner workings (for example, what language it is written in) to take advantage of its functionality. Loosely coupled services are often more flexible than traditional tightly coupled applications; in a tightly coupled architecture, the different components are bound to one another, sharing semantics, libraries and often state; making it difficult to evolve the application. As services are independent from one another they offer a greater degree of flexility and scalability for evolving applications. Although the concept of service-oriented architectures is not a new one, this paradigm has seen wide spread adoption through the Web services approach, which makes use of a suite of standards such as XML, WSDL and SOAP, to facilitate service interoperability.

Web services in their vanilla form provide a simple solution to a simple problem, things become more complex when a group of services need to coordinate together to achieve a shared task or goal. This coordination is often achieved through the use of workflow technologies. As defined by the Workflow Management Coalition [1], a workflow is the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant (a resource; human or machine) to another for action, according to a set of procedural rules.

Workflow provides the glue for distributed services, which are owned and maintained by multiple organisations. Although originally a concept applied to automate repetitive tasks in business, there is currently much interest in the scientific domain to automate distributed experiments. The plethora of competing workflow specifications, standards, frameworks and toolkits from both the business and scientific community cause researchers and engineers to decide it is safer to reinvent the wheel by implementing yet another proprietary workflow language rather than investing time in existing workflow technologies. As a response to this re-occuring problem, this paper provides a concise survey of existing workflow technology from the business and scientific domain, and uses this to make a number of key suggestions.

2 Business workflow technology

As workflow technology was first adopted by the business community, this has led to a space crowded with competing specifications, from opposing companies, some of which have risen to the top, superseding others. Before discussing the standards it is important to note that there are two main architectural approaches to implementing workflow; service orchestration and service choreography.

Service orchestration refers to an executable business process that may interact with both internal and external services. Orchestration describes how services can interact at the message level, with an explicit definition of the control and data flow. Orchestrations can span multiple applications and/or organisations and result in long-lived, transactional processes. A central process always acts as a controller to the involved services and the services themselves have no knowledge of their involvement in a higher level application.

The *Business Process Execution Language (BPEL)* [2] is an executable business process modelling language and currently the current de-facto standard way of orchestrating Web services. It has broad industrial support from companies such as IBM, Microsoft and Oracle. Industrial support brings concrete implementations, tools and training. Recent efforts from the Open Middleware Infrastructure Institute UK (OMII-UK) have resulted in an open-source graphical editor, called BPEL Designer [3].

Other languages have been developed but have not been as widely adopted by the community. *Yet Another Workflow Language (YAWL)* [4] is based on the rigourous analysis of workflow patterns, a particular type of design pattern. YAWL aims to support all (or most) of the workflow patterns and has a formal underpinning based on Petri-nets. The language is supported by an an Open Source implementation [5] and has some industrial support. *XML Process Definition Language (XPDL)* is a format standardised by the Workflow Management Coalition (WfMC) to interchange Business Process definitions between different workflow products like modelling tools and workflow engines. *WfMOpen* [6] is an open-source J2EE based implementation of a workflow engine as proposed by the Workflow Management Coalition (WfMC) and the Object Management Group (OMG), WfMOpen uses XPDL as input.

Service choreography on the other hand is more collaborative in nature. A choreography model describes a collaboration between a collection of services in order to achieve a common goal. Choreography describes interactions from a global perspective, meaning that all participating services are treated equally, in a peer-to-peer fashion. Each party involved in the process describes the part they play in the interaction. Choreography focuses on message exchange, all involved services are aware of their partners and when to invoke operations. Orchestration differs from choreography in that it describes a process flow between services from the perspective of one participant (centralised control), choreography on the other hand tracks the sequence of messages involving multiple parties (decentralised control, no central server), where no one party truly owns the conversation.

The *Web Services Choreography Description Language (WS-CDL)* [7] is an XML-based language that can be used to describe the common and collaborative observable behavior of multiple services that need to interact in order to achieve a shared goal. WS-CDL describes this behavior from a global or neutral perspective rather than from the perspective of any one party. WS-CDL is designed to sit on top of the Web services interface language, WSDL. WSDL focuses on capturing message types, while WS-CDL is about capturing behaviour. A user models a choreography from a global perspective, then each service will have to be programmed by a developer in such a way that they talk to one another, and in doing so, enforce the constraints of the choreography. WS-CDL supersedes the Web Service Choreography Interface (WSCI), although the language is defined by a W3C specification, at the time of writing no implementations exist and interest in the specification has dwindled.

3 Scientific workflow technology

The concepts of workflow have recently been applied to automating large-scale science (or e-Science), coining the term scientific workflow [8]. Business workflow tools look more like traditional programming languages, and are in general pitched at the wrong level of abstraction for scientists to take advantage of. Instead, scientists require higher level tools, which enable them to plug together problem solving components to prove a scientific hypothesis. A scientific workflow attempts to capture a series of analytical steps, which describe the design process of computational experiments. Scientific workflow systems provide an environment to aid the scientific discovery process through the combination of scientific data management, analysis, simulation, and visualisation.

Scientific and business workflows began from the same common ground. Both communities have overlapping requirements, however they each have their own domain specific requirements, and therefore need separate consideration. Scientific work is centred around conducting experiments, therefore a scientific workflow system should mirror a scientist's conventional work patterns by allowing them to apply their methodology over distributed resources. The workflow system should allow the same information to be shown at various levels of abstraction, depending on who is using the system. A high level of abstraction should be presented to the scientist, who we assume knows nothing, about the underpinnings of service composition. The elements of the workflow should be in the context of the appropriate scientific domain and allow the scientist to validate a hypothesis. The process of constructing a workflow that achieves this validation will generally be built in an incremental manner as opposed to the business oriented approach where a workflow will be designed and then implemented. It is therefore essential that the workflow language and scientific workflow system can support this kind of *user-driven*, *incremental*, *prototypical* approach to workflow composition. As the validation of scientific hypotheses depend on experimental data, scientific workflow tends to have an execution model that is dataflow-oriented, where as business workflow places an emphasis on control-flow patterns and events.

Workflows in the scientific community involve the transportation and analysis of large quantities of data between distributed repositories. Scientists will have to schedule and fund the use of expensive resources and cannot afford for a workflow to fail half way through the execution cycle. It is therefore desirable that the systems that support them are robust and dependable. In addition, they should support incremental workflow construction and must be able to run detached from the user console.

As a consequence of the lengthy, iterative design process, workflows become a *valued commodity* and a source of intellectual capital. The output of workflows or workflows themselves may be used as a basis for future research, either by the scientists who generated the data, or colleagues in a related field. This methodology is consistent with the usual practise of non-computational labs. These workflows should be reused, refined over time, and shared with other scientists in the field. Scientific workflows must be fully reproducible. In order for a workflow to be reproduced, provenance information must be recorded that indicates where the data originated, how it was altered, and which components and what parameter settings were used. This will allow other scientists to re-conduct the experiment, confirming the results.

Similar to the business domain, the space of scientific workflow systems has become crowded with different languages and frameworks allowing scientists to automate tasks through a workflow. Below we provide a survey of the most popular workflow systems.

Taverna is an open-source, Grid-aware workflow management system; it provides a set of transparent, loosely-coupled, semantically-enabled middle-ware to support scientists that perform data-intensive *in-silico* [9] experiments on distributed resources. Taverna is implemented as a service-oriented architecture, based on Web service standards. Provenance [10] plays an integral part in Taverna, allowing users to capture and inspect details such as who conducted the experiment, what services were used, and what the results of services provided. Taverna uses a proprietary language, the Simple Conceptual Unified Flow Language or SCUFL [11] for short. The SCUFL language is a high level XML-based conceptual language allowing a user to define a workflow through groups of local or remote services which are connected with data links (providing data flow) and control links (allowing coordination of services not connected through data flow). The Taverna workbench depends on the FreeFluo engine [9].

Kepler [12] is an open-source scientific workflow engine with contributors from a range of application-oriented research projects. Kepler is built upon the Ptolemy II system [13] based at the University of California at Berkeley, which is a mature dataflow-oriented workflow architecture. In Kepler, the focus is on actor-oriented design. Actors are re-usable independent blocks of computation, such as: Web services, database calls etc. They consume data from a set of inports and write data to a set of outports. A group of actors can then be wired together by introducing a mapping from outports to inports. A novel feature in Kepler allows the actor communication (dataflow) concerns to be separated from the overall workflow coordination, which is defined in a separate component called a director. This separation allows a workflow model to be run with different execution semantics, such as synchronous dataflow and process networks. Kepler provides a large variety of computational models inherited from the Ptolemy II system and uses the proprietary Modelling Markup Language.

Triana [14] is an open-source problem solving environment and a test application for the GridLab project [15]. It is designed to define, process, analyse, manage, execute and monitor workflows. The toolkit allows users to compose workflows graphically by dragging programming components called units or tools onto a workspace; connectivity is achieved by wiring components together using data and control links. Triana can distribute sections of a workflow to remote machines through a connected peer-to-peer network. Triana supports multiple languages by allowing different workflow readers/writers to be plugged in, including: Web Services Flow Language (WSFL), Directed Acyclic Graph (DAG), Business Process Execution Language (BPEL) and Petrinet formats.

Planning for Execution in Grids or Pegasus [16] is a framework which maps scientific workflows onto distributed resources such as a Grid. Abstract workflows designed by a domain scientist are independent of any resources they will be executed on, this allows a scientist to focus on workflow design rather than having to decide which physical resources to use. Pegasus then attempts to find a mapping of the tasks to the available resources for execution at runtime through the use of Artificial Intelligence planning techniques. Pegasus uses the proprietary language, DAX at the abstract level which is an XML representation of DAG.

GridNexus [17] is a graphical system for creating and executing scientific workflows in a Grid environment. GridNexus allows the user to assemble complex processes involving data retrieval, analysis and visualisation by building a directed acyclic graph (DAG) in a visual environment. The graphical user interface (GUI) of GridNexus, like Kepler is based on Ptolemy II from UC Berkeley. Once a scientist has designed a workflow using the GUI editor it is translated into the proprietary XML-based language, JXPL. Importantly GridNexus separates the GUI from the execution of the workflow, hence once constructed a workflow (described using JXPL) can be executed locally or remotely.

DiscoveryNet [18] is an EPSRC funded project to build a platform for scientific knowledge discovery from the data generated by a wide variety of high throughput devices at Imperial College, London. DiscoveryNet takes a service-oriented view and provides a concrete implementation firstly for scientists to plan, manage and share knowledge discovery and secondly for service providers to publish data mining and data analysis software components. The Discovery Process Markup Language (DPML) is a proprietary XML-based language which describes process as dataflow graphs, a user can compose a workflow by wiring together nodes (which represent datasets and functions) as a directed acyclic graph.

Other projects worth referencing are the *Bioinformatics Workflow Builder Interface (BioWBI)* [19], an IBM Web-based environment for constructing and executing workflows in the Life Sciences community, *GridBus* [20], a Grid-aware workflow execution environment, *Imperial College e-Science Networked Infrastructure (ICENI)* [21], and *Magenta* [22], an open-source, decentralised Web services composition tool. The *Workflow Enactment Engine Project (WEEP)* aims to implement an easy to use workflow enactment engine for WS-I/WSRF services using WS-BPEL 2.0 (Web Services Business Process Execution Language) as a formalism to process and execute workflows, which currently focus on data mining tasks [23].

4 Discussion

Through our experience of evaluating workflow frameworks, this paper makes the following observations for improvements and future research concerning usability, sustainability and tool adoption:

Collaboration is key. With so many tools available, it is essential that researchers from multiple domains form collaborative groups to meet and discuss the common, overlapping requirements. Collaboration is necessary in order to prevent implementing highly specific, hardly used features, as well as preventing reinvention of any wheels.

Use a conventional scripting language instead. There is increased interest and hype surrounding workflow technology and often it is oversold to domain scientists. Existing scripting languages (Perl etc.) have received much investment, both in the form of training and in the development of features such as debugging, extensive add-on libraries and integrated editors. These features have yet to be included in workflow systems. Workflow systems essentially have to start from scratch, re-implementing each feature for a specific framework.

Do not implement another workflow language. As demonstrated by our survey of workflow technology, implementing yet another workflow language is the last thing that researchers should be doing. There are many well developed and well supported frameworks, which researchers should first investigate.

Abstract is not abstract enough. “Research your domain” may seem like an obvious statement, but it is a fact that the day-to-day tools that scientists use differ vastly from one domain to the next. Experience with working alongside domain scientists has taught us that wet laboratory biologists are uncomfortable using even the most abstract workflow tools currently available. Problem solving in terms of services is outside the normal pattern of thinking. On the other end of the spectrum, physics researchers are comfortable with command-line tools and are used to thinking about problems in terms of programming. Tools need to be tailored to the domain instead of being built by computer scientists for computer scientists. Research into intelligent, abstract editors should be a priority for the scientific community, preliminary examples can be seen through the Taverna and Pegasus frameworks.

Stick to standards. It is still unclear if the processes required of a scientific domain can be captured using business workflow technology such as BPEL. To this end, does it make sense to create languages specifically for scientific workflow, like SCUFL which is used in Taverna? BPEL has become the de-facto standard workflow language, supported by industrial strength software implementations by major vendors. With this level of support, from the perspective of the user it makes sense to stick to standards instead of tying oneself to a proprietary workflow language. If software development and tool support terminates on one of the proprietary frameworks, workflows will need to be re-implemented from scratch. Instead, standards need to be viewed in a different way, research needs to be targeted at providing powerful abstraction mechanisms for languages such as BPEL and providing integrated tool support. These mechanisms should allow scientists to model workflows from higher levels of abstraction and automatically translate them into BPEL processes that will run on any BPEL workflow engine. BPEL designer, which is part of OMII-UK software stack, is an initial step at providing these abstractions.

Portal-based access. Through our experience, even if workflow tools are made available to domain scientists they often cannot and will not download and install them. These tools need to be made available through portals, taking advantage of thin client technology such as AJAX.

It is clear from our survey of existing workflow technology that the space is starting to get crowded with competing alternatives. Business workflow specifications and standards are the result of rival companies negotiations. With these concrete standards come industrial strength software. Scientific workflow on the other hand is a relatively new area of interest, each project has its own unique set of requirements and often develops yet another proprietary language and workflow execution engine.

To encourage the adoption of existing tools, this paper has presented a concise survey of business and scientific workflow technologies. Furthermore, we have pointed out important research directions that make the existing technologies more suitable within the context of scientific workflow.

References

1. Hollingsworth, D.: The Workflow Reference Model. Workflow Management Coalition. Document Number tc00-1003 edn. (1995)
2. The OASIS Committee: Web Services Business Process Execution Language (WS-BPEL) Version 2.0 (2007)
3. Wassermann, B., et al.: Sedna: A BPEL-based environment for visual scientific workflow modelling. Workflows for eScience - Scientific Workflows for Grids (2006)
4. van der Aalst, W., ter Hofstede, A.: Yet another workflow language. Information Systems **30**(4) (2005) 245–275

5. van der Aalst, W.M.P., et al.: Design and Implementation of the YAWL system. In: Proceedings of the 16th International Conference on Advanced Information Systems Engineering (CAiSE'04). (2004) `http://sourceforge.net/projects/yawl/`.
6. Lipp, M.: The Danet Workflow Component V2.1. (2007) `http://wfmopen.sourceforge.net/`.
7. Kavantzas, N., et al.: Web Services Choreography Description Language Version 1.0 (2005)
8. Deelman, E., Gil, Y.: Workshop on the Challenges of Scientific Workflows. Technical report, Information Sciences Institute, University of Southern California (2006)
9. Oinn, T., et al: Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics **20**(17) (2004) 3045–3054
10. Zhao, J., et al: Annotating, linking and browsing provenance logs for e-Science. In: 1st Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data, Sanibel Island, Florida, USA. (2003)
11. Oinn, T., et al: Delivering Web Service Coordination Capability to Users. In: WWW2004, New York. (2004) 438–439
12. Ludascher, B., et al.: Scientific workflow management and the kepler system. Concurrency and Computation: Practice and Experience **18**(10) (2005) 1039–1065
13. Buck, J.T., Ha, S., Lee, E.A., Messerschmitt, D.G.: Ptolemy: A framework for simulating and prototyping heterogeneous systems. Journal of Computer Simulation **4** (1994) 155–182
14. Taylor, I.J., et al.: Distributed P2P Computing within Triana: A Galaxy Visualization Test Case. In: 17th International Parallel and Distributed Processing Symposium (IPDPS 2003), IEEE Computer Society (2003) 16–27
15. Allen, G., et. al: Enabling Applications on the Grid: A GridLab Overview. International Journal of High Performance Computing Applications: Special Issue on Grid Computing: Infrastructure and Applications **17**(4) (2003) 449–466
16. Deelman, E., et al.: Pegasus: A framework for Mapping Complex Scientific Workflows onto Distributed Systems. Scientific Programming Journal **13**(3) (2005) 219–237
17. Brown, J.L., et al.: GridNexus: A Grid Services Scientific Workflow System. International Journal of Computer Information Science (IJCIS) **6**(2) (2005) 72–82
18. Rowe, A., et al.: The Discovery Net System for High Throughput Bioinformatics. Bioinformatics **19**(1) (2003) 225–231
19. Siepel, A.C., et al.: An integration platform for heterogeneous bioinformatics software components. IBM Systems Journal **40**(2) (2001) 570–591
20. Buyya, R., Venugopal, S.: The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report. In: Proceedings of the First IEEE International Workshop on Grid Economics and Business Model. (2004) 19–36
21. Mayer, A., et al.: Meaning and Behaviour in Grid Oriented Components. In: Lecture Notes in Computer Science. Volume 2536., Springer-Verlag Berlin Heidelberg (2002) 100–111
22. Walton, C., Barker, A.D.: An Agent Based e-Science Experiment Builder. In: Proceedings of The 1st International Workshop on Semantic Intelligent Middleware for the Web and the Grid, European Conference on Artificial Intelligence (ECAI). (2004)
23. Janciak, I., Kloner, C.: Workflow enactment engine project v1.0 (2007) `http://weep.gridminer.org/`.