

On Using BPEL Extensibility to Implement OGSI and WSRF Grid Workflows

March 2005

Aleksander Slomiski
Department of Computer Science
Indiana University

Abstract

This paper discusses the benefits and challenges of using BPEL4WS in Grid environments. In particular we look on how BPEL4WS built-in extensibility can be used to facilitate execution of BPEL based workflows in OGSI and WSRF based Grids.

Introduction

Workflows have a long history dating from the work of Skip Ellis and Michael Zisman in Xerox Parc on “Office Automation Systems” in the 1970s [Ellis77]. What made workflows attractive then is still true today: ability to describe a process in a way that makes it easy to understand, change, execute, and monitor. The Workflow Management Coalition [WfMC] was established as a non-profit, international organization of workflow vendors, users, analysts and university/research groups with a mission to promote and develop the use of workflow. WfMC defined a workflow as “*The automation of a business process, in whole or parts, where documents, information or tasks are passed from one participant to another to be processed, according to a set of procedural rules*”. This definition can certainly be extended to non business processes such as scientific workflows. In particular, WfMC formally specified Workflow Management System (WFMS) as “*a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications*”. Above properties of a Workflow Management System make it a desirable tool for scientific problem solving environments and portals which need to manage and integrate disperse resources and scientific applications. Therefore it is interesting to look at business workflows to determine to what extent they can be used in scientific environments.

In the business world Web services have recently gained popularity as a new approach to integrate Internet business resources by using XML and open protocols (such as HTTP and SOAP.) This is a similar task to what Grids in scientific environments tried to accomplish: one definition of Grid is that Grid is a system that “*coordinates resources that are not subject to centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service*” [WhatIsGrid]. The use of Grids in scientific environments differs from business environments. For example, the typical scientific Grid will need to handle large amounts of data, deal with sharing resources such as scientific instruments or sensors, and make available to users applications that were written in scientific programming languages that are not popular in business environments (for example Fortran.)

Combining Web services and Grids is a promising way to leverage existing work in both business and scientific environments. In particular this integration was looked into at Global Grid Forum [GGF] and, as a result, the Open Grid Services Infrastructure (OGSI) was recently proposed. OGSi defined the Grid Service interface that every OGSi grid service must implement. This interface provides lifecycle methods and it allows accessing Grid service state exposed as XML-based Service Data Elements (SDEs.) Service Data Elements provide a simple to use and standardized way to discover and manipulate public state of any grid service. Additionally OGSi defined a set of optional interfaces such as OGSi Factory. This interface provides a well defined and extensible way to create grid service instances. Each grid service instance is identified by Grid Service Handle (GSH), which provides a binding between a grid service stable name, and a set of Grid Service References (GSR). A Grid Service Reference contains everything necessary to contact the grid service but a GSR is possibly short lived and a grid service over its lifetime may have multiple GSRs. GSR can be thought as a permanent name for a service (URL) that will never change whereas GSR is a short lived address to current location of the service and is valid only until it expires. When GSR expires a new GSR needs to be found. Standard OGSi services, such as OGSi Registry, provide capability to find valid GSRs for GSHs and to find GSHs for grid services that provide desired functionality (such as *find GSHs for grid services that can run a weather simulation in next 20 minutes.*) Multiple GSHs and GSRs that identify a grid service can be combined together with a description of the service interface to form an OGSi service locator. The OGSi service locator is a useful abstraction as one grid service may have more than one GSH/GSR but nevertheless is accessible through the same interface.

In 2004 OGSi specification was refactored into a set of WS Resource specifications called WS-Resource Framework [WSRF]. This change addresses one of the main concerns about modularity of OGSi specification: now users can decide which WS-Resource Framework specifications to use. Additionally WSRF moves away from object oriented approaches to services (GSH/GSR is like an object reference) to explicitly distinguish between the “service” and the stateful “resources” acted upon by that service. From a technical point of view the biggest change is that the GSH, GSR and service locators are replaced by WS-Addressing [WSA] Endpoint References (so called EPRs). Endpoint References are very similar to OGSi service locators but apart from the service address and its interface they can contain XML properties. Those XML properties (called “Reference Properties”) can be used to externalize some state associated with Web service. WS-Addressing Endpoint Reference is like a traditional

address in postal service: a message is placed in a SOAP Envelope labelled with WS-Addressing Endpoint Reference labels (headers) that indicate where the message should go when it is travelling through intermediary Grids services. The other change is that the OGSi Factory interface is no longer present in WSRF and is replaced by a factory-like creation pattern.

We expect that workflows in Grids will be as important as they already are in business environments. Considering dependency of OGSi and WSRF on Web services specifications created for business usage it is natural to expect that business workflows and, in particular, business workflow languages designed for Web services can be used in Grids. The Business Process Execution Language for Web Services Version 1.1 [BPEL4WS], in short BPEL, is an example of such a business workflow language. In last few years BPEL has become a leading candidate for the standard workflow language for business oriented Web services. BPEL replaced IBM's WSFL and Microsoft's XLANG languages and is currently in process of standardization in OASIS [WS-BPEL].

BPEL is a combination of graph oriented (WSFL) and procedural (XLANG) workflow languages and provides a convenient language to express majority of workflow processes. There are some remaining issues about BPEL language expressiveness and soundness of theory behind BPEL design [Wohed02, Aalst03] but they should be worked out during the standardization process in OASIS.

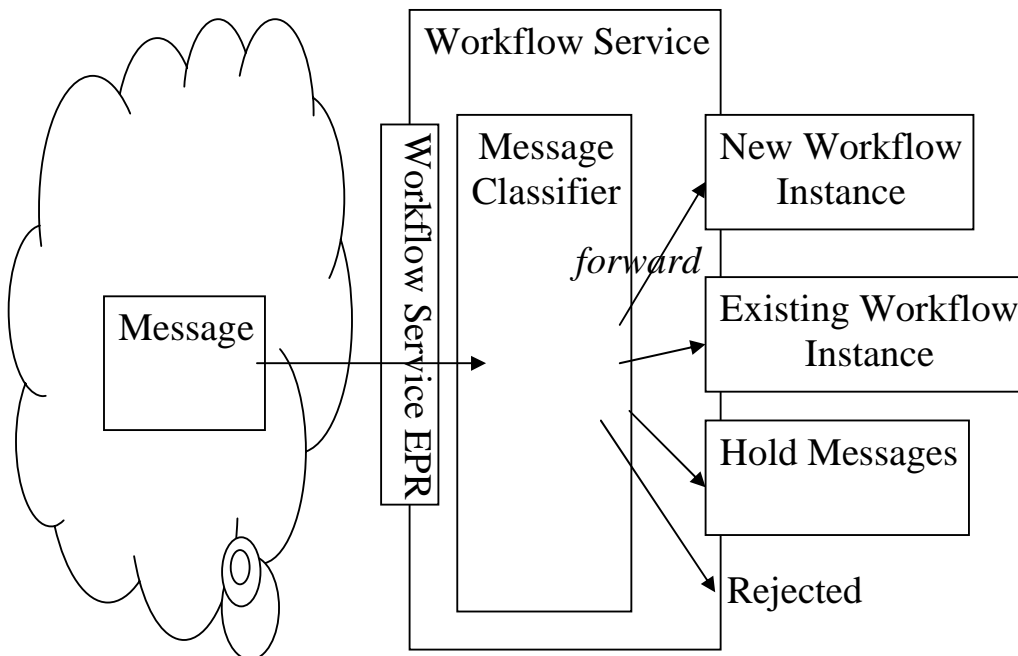


Figure 1. Critical part of a BPEL workflow engine functionality is ability to route message to appropriate workflow instance, create a new workflow instance, hold message for later processing (ahead of time arrival), or reject messages that are incorrect.

What makes BPEL a very attractive candidate to use in Grid environments is a very strong support for Web services. A BPEL workflow is designed to act as a Web service and it can easily interact with existing Web services to combine them into a new service. BPEL workflow engine acts as a Web service that receives messages,

determines their destination, and depending on workflows execution state, sends messages to other Web services. Figure 1 shows possible outcomes of handling an incoming message inside a BPEL workflow engine.

BPEL uses the WS-Addressing Endpoint References to simplify passing addresses of Web services participating in a workflow. One very important advantage of using BPEL in Grids is that it can easily and seamlessly interact with standard Web services (that may not be part of Grids) and grid services such as provided by OGSi and WSRF. However to leverage this one must be careful not to change BPEL in incompatible way and map standard BPEL semantics to grid environment in the most natural way possible.

In this paper we will now describe our experience with using BPEL in our environment. Our target environments for BPEL workflows are OGSi and WSRF based Grids. By using a well supported workflow standard, such as BPEL, we hope to leverage existing and future investments made by the enterprise IT industry in the BPEL infrastructure to the advantage of large scale scientific Grids. Even though BPEL was not designed for Grids we are using BPEL extensibility to meet our requirements.

By making the decision to base our approach on an existing workflow language (such as BPEL) and to use **only** the extensibility mechanisms provided by BPEL we are limiting ourselves from other possible ways to execute Grid workflows. An alternative approach would be to make changes to BPEL language itself (simplifying or removing some constructs, adding completely new syntax etc.) but by doing it we would make a new BPEL-*derived* workflow description language that is incompatible with the standard BPEL. While doing this may be required in some situations, we believe compatibility and interoperability may require any BPEL-derived workflows to be mapped to standard BPEL. Unfortunately such mappings are typically not reversible and some information may be lost in the mapping so we want to avoid doing mappings as much as possible.

Integrating OGSi and BPEL

Workflow functionality is typically provided by a Workflow Engine. A Workflow Engine is defined by WfMC as “A *software service or "engine" that provides the run time execution environment for a process instance*”. Therefore to successfully use BPEL in OGSi environments it is necessary to specify expected behaviour of a BPEL engine in OGSi based Grids.

When integrating BPEL with OGSi we can leverage the extensibility mechanism present in BPEL: BPEL does not specify any deployment information. Instead BPEL workflow describes how to interact with services that are consuming or producing messages described in WSDL port types. Such services when used in BPEL workflow are called *partners*. We can use this extensibility mechanism to treat OGSi services as BPEL partners. Additionally BPEL does not mandate how BPEL engine must make workflow functionality available except that BPEL engine must expose a workflow service with WSDL port types used in BPEL workflow. This allows us to implement the BPEL engine as an OGSi service. Not specifying deployment information is a

very powerful mechanism that allows using BPEL workflows in multiple environments as long as basic Web service abstractions, such as WSDL port types, are present.

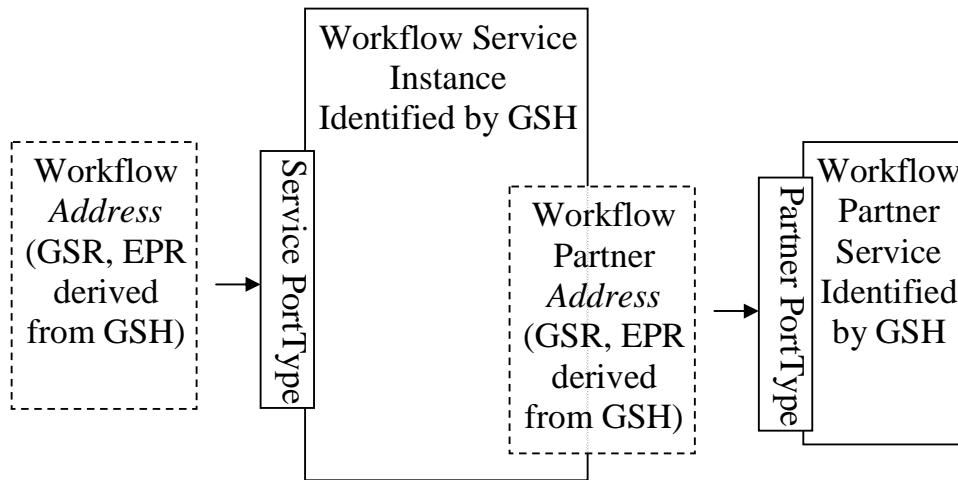


Figure 2. Each partner in workflow is bound to existing service identified by GSH or EPR. Also workflow itself is a service and is identified by GSH or EPR. During workflow execution one GSH may resolve to multiple GSRs.

OGSI services are based on Web services standards but provide additional capabilities that extend typical Web services. Those additional capabilities (and their associated semantics) need to be well defined for BPEL workflows to use OGSi services and to make BPEL workflows usable by OGSi clients.

The biggest problem that needs to be solved is a possibility that some of OGSi services may not exist until they are created by an OGSi factory. Even though a GSR corresponds to a WSDL document there may be a need to use multiple GSRs during a grid service instance lifetime (see Figure 2). Therefore it is necessary during an OGSi service lifetime to track GSR expiration times and to use GSH to retrieve valid GSRs. This is an additional functionality that must be added to a BPEL engine or to its supporting environment. For example proxy service may be used to transparently create an instance of OGSi grid service, to track GSH/GSR mappings, and to forward messages to current location of grid service instance.

BPEL uses WS-Addressing Endpoint References to establish locations of Web services. BPEL engine must map GSHs, GSRs, and service locators to a something equivalent to WS-Addressing endpoint reference to be able to interact with OGSi services. This is not difficult as OGSi service locator (containing service GSHs, GSRs, and service port types) corresponds to WSA endpoint reference (address, port type.) Those reference properties can be used to encapsulate GSHs, GSRs, and service port types. However this mapping will not work in case when OGSi service locator acting as WSA endpoint reference needs to be passed to non-OGSi service used in BPEL. Such service will not be able to use mapped endpoint reference to invoke OGSi service unless there is a proxy service that translates between non-OGSi Web service call into OGSi service calls. Maintaining such proxy service is an additional burden and may not be possible for services that are discovered during workflow

execution. When interacting with a non-OGSI services that already support WSA Endpoint References thus it may happen that such references may not be mapped to OGSi service locators (as XML reference properties and policy elements are lost in the mapping.) An OGSi service will not be able to access service identified by such mapped service locator because the non-OGSI service may need all information that was present in WSA Endpoint Reference. Again solution to this is to provide a proxy service that will maintain mapping between the service locator and the WSA endpoint reference and forward incoming messages to the service identified by the endpoint reference.

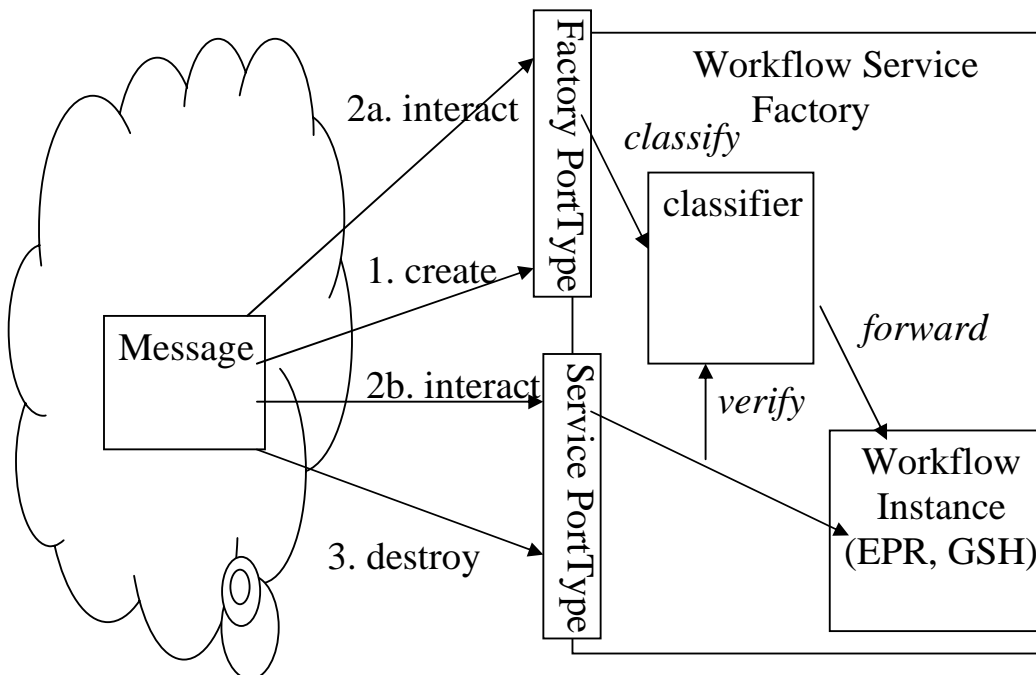


Figure 3. Hybrid BPEL workflow engine that supports both direct creation of workflow instances with factory (1,2b,3) and indirect creation of workflow instances when they are needed to be created (2a).

A related problem appears when a BPEL workflow instance may be used by OGSi and non-OGSI clients. Each BPEL workflow instance may have multiple Endpoint References identifying it (each endpoint reference corresponds to a role in which the workflow instance is interacting with a partner web service.) Those Endpoint References should be used by clients to interact with the workflow instance as in Figure 3 where we have a workflow engine that supports both correlations based interactions where a workflow instance is created on demand (2a) and direct interactions with workflow instances that is created (1) and then uniquely identified by EPR, GSH, or similar mechanism (2b). Workflow engine must be able to distinguish between lifecycle messages and workflow specific message (1 and 2b) and may need as well to have special handling for destroy messages sent directly to workflow instance identified OGSi GSH/GSR (2b and 3).

An amount of information in an Endpoint Reference may be not sufficient to identify the workflow instance that should be used to process an incoming message. That is because BPEL may use parts of message independent from transport or WS-Addressing to route message to workflow instance. BPEL has a mechanism called

correlation set that can be used to indicate which parts of a message are to be used to find workflow instance to process it. That means that even though an Endpoint Reference may be in some cases mapped to an OGSi service locator (when an endpoint reference has no Reference Properties or WS-Policy elements), still such service locator may not be enough to identify the workflow instance as there is a need to know a current value of the correlation set. This is a semantic difference: OGSi clients will expect that service locator is enough to unambiguously identify the workflow service instance but that may not be case when BPEL correlation sets are used and needs to be known when sending different messages to the workflow instance. The solution to this is to make sure that mappings between endpoint references and service locators used in BPEL engine are reversible. BPEL engine will need to provide for each workflow instance one or more GSHs and maintain corresponding GSRs even if correlation sets are used (possibly assigning a unique GSH for each correlation set.) Those GSHs must identify exactly one workflow instance without need to use correlation sets.

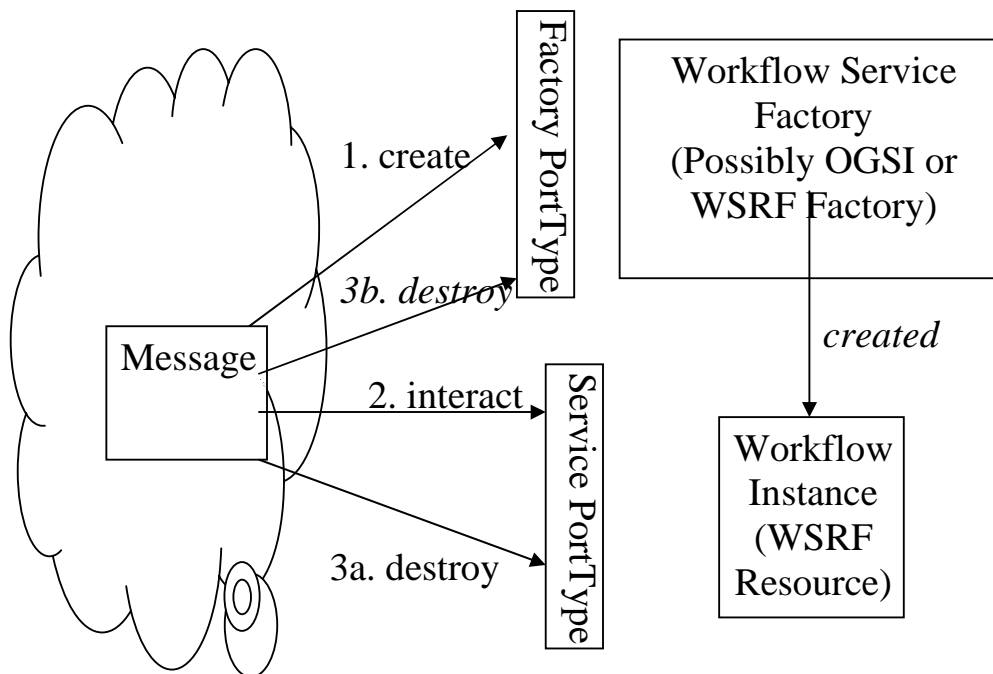


Figure 4. Timeline for workflows that use explicit lifecycle.

When giving each BPEL workflow instance one or more GSHs we have to make sure that each workflow instance implements the OGSi Grid Service interface. That means that a workflow instance needs to support access to OGSi Service Data Elements (SDEs). This does not mean that BPEL engine needs to create a completely new grid service instance with its own SDEs but that a workflow instance must *look* like an OGSi grid service instance to OGSi clients. Service Data Elements provide a convenient way to expose XML based information about workflow state and are a very good tool to provide a workflow instance monitoring capability to OGSi clients. As part of OGSi Grid Service interface workflow instance must handle lifecycle requests (such as termination requests.) Because workflow instances are typically managed by the a BPEL workflow engine, the simplest solution is for a BPEL implementation to pass such request from a workflow instance to the BPEL engine as the BPEL engine is already maintaining workflow instances lifecycles.

OGSI services that create new service instances are expected to implement the OGSi Factory interface (see Figure 4.) Implementing OGSi Factory interface in BPEL engine is difficult as a BPEL engine already has to support a workflow creation pattern that does not require an explicit factory interface. BPEL specification allows that some incoming messages may be identified to create a new workflow instance (such message contains correlation set that resolves to none of existing workflow instances.) Implementing OGSi Factory interface in BPEL engine means that a second creation mechanism is supported (besides one described in BPEL specification.) Moreover workflow instances created by using OGSi factory can not be described in BPEL (BPEL require at least one message to create workflow instance) *unless* we allow for two stage creation as pictured in Figure 4. In this figure we have a step-by-step diagram showing workflow instance creation that use explicit lifecycle: workflow instance is created with an unique address (or identifier) such as EPR or GSH (step 1), Web service sends messages to workflow instance (step 2), when workflow instance is no longer needed it is destroyed by sending special message to workflow instance (step 3a) or to workflow engine (step 3b). If workflow instance is not used for long time then destruction may be handled automatically by workflow engine (for example when lease expired.)

In the first stage a BPEL workflow instance is created by using an OGSi Factory interface implemented by BPEL engine. At this stage the workflow instance is created but not “activated”. It remains in this state until a first message arrives and then BPEL workflow instance becomes active. Such two stage creation process allows to configure a workflow instance (for example with security, performance, reliability or other QoS requirements) before it is used.

Deciding to make a BPEL Engine that supports only the OGSi Factory to create workflow instances would make such workflow engine no longer compatible with BPEL specification therefore such workflow engine could no longer be called a BPEL engine. Therefore OGSi BPEL engine should handle both OGSi and non-OGSI clients by implementing OGSi Factory interface and allowing standard BPEL workflow instance creation (see Figure 3).

Fortunately OGSi Factory interface is optional. However if BPEL engine does not implement OGSi Factory interface then it is difficult to integrate BPEL based workflows into OGSi based Grids that expect to use OGSi factories to create grid service instances including grid workflow instances.

BPEL Integration with WS-Resource Framework

When comparing the complexity of integrating BPEL with OGSi to the complexity of integrating BPEL with WSRF it is obvious that the BPEL integration is much easier with WSRF. WSRF does not require that every grid service implements the Grid Service interface. Moreover WSRF is now using the same WS-Addressing specification that is used in BPEL. Therefore we no longer need to map between GSHs, GSRs, service locators and WS-Addressing Endpoint References when using

BPEL in WSRF based Grids. Those are very positive changes and show a great amount of alignment between WSRF and BPEL.

WSRF has no longer a direct equivalent of OGSi Factory. Instead a WS-Resource factory is defined to be a pattern. In this pattern a WSRF factory is any Web service that returns in a response message WS-Addressing Endpoint Reference that identifies newly created WS-Resource. That means that for BPEL engine it is no longer necessary to implement an OGSi Factory interface to create workflow instances and workflow instance can naturally return its own endpoint reference. This greatly simplifies integration of BPEL workflows in WSRF enabled Grids as one of the major problems of trying to map BPEL to OGSi was that BPEL already supports a workflow creation pattern that does not require a factory as typically first message sent to BPEL engine implicitly creates workflow instance if it did not exist. Moreover this pattern allows for more fine grained interactions with BPEL as one BPEL workflow instance may have **multiple** Endpoint References corresponding to different roles it has when interacting with partners that are Web and Grid services.

By using WSRF there is also no longer a requirement that each workflow instance created by BPEL engine must implement the OGSi Grid Service port type. That means that there is no longer a need to delegate lifecycle from a BPEL workflow instance to the engine but it is possible (see Figure 3). An additional advantage is that BPEL workflow instances that are created outside of WSRF Grid can participate with WSRF enabled BPEL workflows (the distinction between non-OGSi and OGSi service that have to implement Grid Service interface is no longer present.)

WSRF provides a convenient way to expose a BPEL workflow instance state by using WS-Resource Properties. This combined with WS-Notification is a very powerful mechanism to make BPEL workflows easy to monitor. In OGSi Service Data Elements (SDEs) had similar function but OGSi Notification interface was not as powerful as WS-Notification (for example there was no notion of topics or brokers.)

Towards Full Integration of BPEL in OGSi and WSRF Grids

We have described how BPEL can be integrated with OGSi and WSRF Grids however there is more that needs to be considered to have BPEL workflows fully integrated with Grids. BPEL workflows may need to be able to interact with Grid services that may not be yet exposed as Web services (such as Globus Toolkit 2 based services.) When running multiple BPEL workflows it becomes very important to be able to monitor and manage BPEL workflows. Last but not least, Web and Grid services may fail so BPEL engine needs to be reliable and to be able to cope with failures especially in case when workflows need to be running for very long periods of time.

In this section of the paper we describe issues that apply to running any grid application and thus apply to any grid workflow management service implementation. We highlight challenges and sketch some ideas about possible solution.

Facilitating BPEL integration with existing Grid Services: BPEL Pseudo Partners

When writing BPEL workflows it is very useful to provide a set of services that are **always** available. One way to do this is to use BPEL partner abstraction. BPEL partners are services that workflow needs to use and typically are mapped to Web (or Grid) services. However it is possible to map some BPEL partners to locally implemented services as long as a WSDL port type for a partner is provided. In Figure 5 there is an example of a BPEL workflow that interacts with many partners some of them may be Web services, WSRF services or other BPEL workflows. Additionally some partners may not be bound to actual Web service but instead bound to WSDL that describes local Java class, components, GridFTP services, or local WS-Notification publishing service.

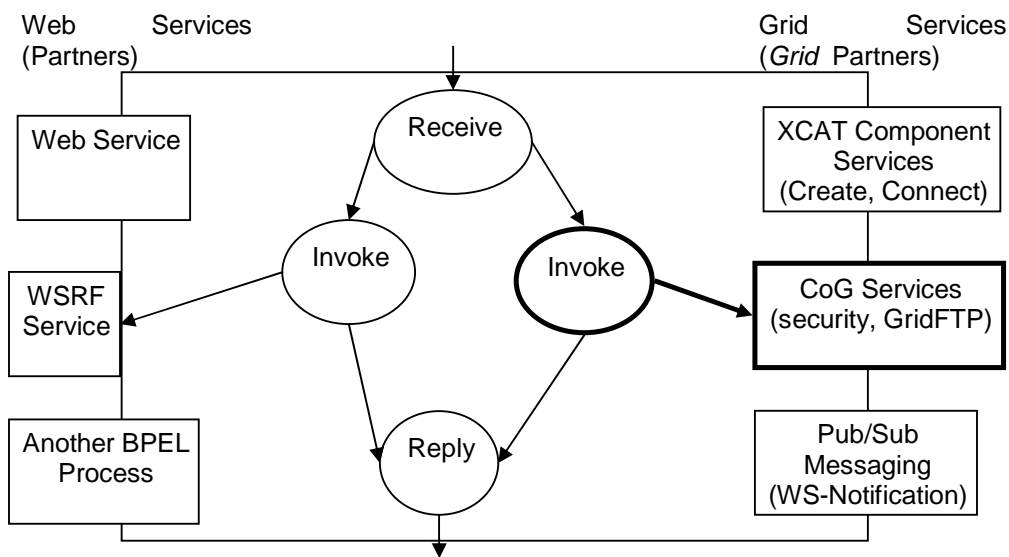


Figure 5. Example of a very simple workflow that uses WSRF service and accesses Grid functionality (such as GSI security credentials) through “pseudo partner” interface.

We call services without network endpoint *pseudo* partners as they are not accessible as “real” Web services but still implement WSDL port type. Tools such as Web Services Invocation Framework [WSIF] can be used to implement pseudo partners as they allow invoking services described in WSDL that are available locally, for example a service invocation may be mapped invoking a method in a Java class. Pseudo partners can provide services such as support for large data files transfers using GridFTP [GridFTP], Reliable File Transfer (which is part of Globus Toolkit 3), component management XCAT [XCAT3], or interacting with Condor. It is important to realize that, if necessary, pseudo partners could be implemented by Web services but by doing this we are limiting performance gains available when coupling set of common services in the same machine where the workflow engine is running.

BPEL Workflow Monitoring

Monitoring and managing of a large number of workflows is an additional and important aspect that goes beyond a workflow execution but it is very important for any Grid workflow system. As BPEL is an XML based workflow language, it is natural to describe a workflow state in XML. By using XML to represent a workflow instance state it is easier to create a workflow monitoring Web Service that can be used by workflow clients to track workflows execution progress. It is easier because when a workflow state is available in XML format then it can be exposed as OGSi Service Data Elements or WSRF Resource Properties in a straightforward manner.

When using scientific workflows to coordinate execution of large parameter sweeps it becomes very important to be able to monitor large number of workflow instances and, at minimum, to be able to stop them when execution is not proceeding correctly. If workflow monitoring can be integrated with existing monitoring capabilities it will help to give a more complete picture of workflow execution progress at any given moment. Monitoring can be enhanced by a workflow execution history service that keeps track of workflow state changes. Such service could be very helpful when trying to determine what was happening to a workflow instance and it is critical when tracking data provenance.

Fault Tolerance and Reliability

It seems that fault tolerance is the most important issue when trying to run BPEL workflows that needs to interact with real Web or grid services. Longer the running time of the workflow, the higher probability that some of the services used will fail. Therefore an implementation of a workflow engine for scientific applications must be able to deal with situations when grid services are down or are not working correctly (for example too slow.) To cope with such problems it is necessary to define additional QoS properties that are not part of BPEL. For example it may be possible to configure properties of a grid service to indicate that it is idempotent (so retrying is allowed), supports transactions, or that an alternative service can be used. Additionally if services used by BPEL workflow support some form of reliable messaging (such as WS-ReliableMessaging) then delivering of messages can be delegated to runtime library increasing workflow runtime resilience to network problems. This requirement is not specific to Grid environments and it is shared with other domains in which Web services are used so it is reasonable to expect that future BPEL implementations will have a reliable messaging (and other form of reliability) implemented and directly usable in Web services based Grid environments.

Very Long Running Grid Workflows

Fault tolerance becomes very important when there is a need to support very long running workflow instance (days, weeks, or longer.) For long running workflow the main challenge is to be able to deal with errors gracefully. For example consider a

situation in which a workflow instance was running for days but since workflow was started there were made incompatible changes to grid services that the workflow instance is using. In this case it is important to be able to instruct the workflow instance to use alternative services (if available) or dynamically modify the workflow instance to be able to interact with new services. Such "healing" actions are difficult to implement and in some cases such "ad hoc" modifications to the workflow instance may not be possible. There is already a substantial academic and industrial research work done on facilitating such "ad hoc" modifications (for example [Casati98, Liu98]) and it is important to see what can be leveraged when using BPEL.

Long running workflows requires also that BPEL engine supports persistence so workflow instances can be saved to a stable storage to protect against a situation when workflow engine is stopped (when for example machine is restarted.) In some cases it is not possible to restore workflow state. For example if workflow instance already accepted HTTP connection and did not send response. In such case when the network connection is lost (for example when machine is restarted) it is not possible to restore TCP connection and send response back. This example emphasizes an importance of asynchronous messaging as such messaging does not require to keep connections open for long time and a connection to send message can be easily re-established when each workflow participant has its own network endpoint. However this problem is not the problem that is specific to BPEL but it is shared by all Web services that use HTTP. However it is important to mention that BPEL workflow description does not make any assumptions about connection availability. Instead BPEL engine is typically layered on top of messaging module that will map workflow semantics to underlying messaging medium (such as request-response HTTP or one-way HTTP with WS-Addressing and WS-ReliableMessaging.)

Another problem that comes up when running workflows with a long time to complete is a need to maintain security credentials (or capabilities) for long time. Typical Globus grid proxy certificate has lifetime between few hours and few days and it becomes difficult in more complex workflows to provide a renewed grid proxy certificate when it is needed for weeks (or months.)

Supporting Large Data and Streaming

One of the most important and currently unresolved problems for using workflows is dealing with large files that are used in scientific application. We see that there is a strong need to support large data sets and in particular data streaming as part of a workflow. This is ongoing work and we think that only future can show which approach is the best and it is more likely that many approaches will be used and will need to be integrated in scientific workflows.

As a first step in this direction we think that making easy to access current tools and APIs used for data handling is important. Currently we consider using "pseudo" partners (see previous section) to provide such integration with GridFTP (including managing grid security proxy certificates.) However in future it would be very beneficial to investigate possibility of more abstract notions of "data stream" or "data links" that would allow creating direct peer-to-peer data connections between services, possibly by leveraging previous work such as described in [GSFL].

Conclusions and Future Work

As part of conclusions we would like to answer questions that were proposed for “Workflow Execution and Runtime Considerations“ panel. Our answers are for the case when a business workflow (such as BPEL4WS) needs to be applied to Grid. That means that such issues like adapting business workflow to Grid environments, integration with Grid services and resources, and handling large data sets become very important.

- *“What are the appropriate execution models for Grid workflows?”*
We think that BPEL extensibility and flexibility is sufficient to be an execution model for Grid and Scientific workflows.
- *“How does this differ from commercial workflow execution?”*
In this paper we demonstrated that a business workflow specification such as BPEL can be adapted to OGSi and WSRF Grids and to what extent it can be done. We identified also main challenges when considering a seamless integration of BPEL with Grid service (such as data handling, streaming, and security.)
- *“What Web and Grid Services are required for a reliable robust workflow execution?”*
We think that Web services related specifications such as WS-ReliableMessaging can help to achieve a reliable workflow execution. Persistent workflows, asynchronous messaging, monitoring, and improved security models are needed to support running reliable workflows that take very long to complete.

We hope that in this paper we provided convincing arguments for a claim that BPEL can meet requirements for Grid workflow though there is still a lot of work needed before BPEL can become a natural component of grids.

We plan to work on a BPEL prototype implementation that is a vehicle to test our approach in practice. As the first priority we concentrate on making BPEL workflows easy to monitor and easy to integrate with WSRF.

References

- [Aalst03] "Don't go with the flow: Web services composition standards expose" W.M.P. van der Aalst. IEEE Intelligent Systems, Jan/Feb 2003
- [Casati98] "A discussion on approaches to Handling Exceptions in Workflows". F. Casati. Workshop on Adaptive Workflow Systems Conference on Computer Supported Cooperative Work (CSCW), Seattle, USA, November 1998.
- [BPEL4WS] "Business Process Execution Language for Web Services Version 1.1" <http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>
- [GGF] The Global Grid Forum <http://www.ggf.org>
- [Liu98] "Automating Handover in a Dynamic Workflow Environment" C.Liu, M. Orłowska and H.Li. Proceedings of 10th International Conference, CAiSE'98, pp. 158-171. June 1998, Pisa, Italy. Lecture Notes in Computer Science, Vol. 1413, Springer.
- [Ellis77] "Representation, Specification and Automation of Office Procedures" PhD thesis, University of Pennsylvania, Warton School of Business, 1977
- [GSFL] "GSFL, A Workflow Framework for Grid Services" S. Krishnan, P. Wagstrom and G. von Laszewski, SC2002, Baltimore, MD, 11-16 Nov. 2002, (Poster)
- [GridFTP] "GridFTP: Protocol Extensions to FTP for the Grid," W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, and S. Tuecke, Mar 2001. <http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf>
- [OPERA] "Flexible Exception Handling in the OPERA Process Support System" C. Hagen, G. Alonso, International Conference on Distributed Computing System 1998s
- [WhatIsGrid] "What is the Grid? A Three Point Checklist" Ian Foster, Argonne National Laboratory & University of Chicago, GRIDtoday, <http://www.gridtoday.com/02/0722/100136.html>
- [WfMC] Workflow Management Coalition <http://www.wfmc.org/>
- [Wohed02] "Pattern Based Analysis of BPEL4WS" P. Wohed, W.M.P. van der Aalst, M. Dumas, A.H.M. ter Hofstede, QUT Technical report, FIT-TR-2002-04, Queensland University of Technology, Brisbane, 2002
- [WS-BPEL] OASIS Web Services Business Process Execution Language TC http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
- [WSA] "Web Services Addressing" <http://www-106.ibm.com/developerworks/webservices/library/ws-add/>
- [WSIF] Web Services Invocation Framework <http://ws.apache.org/wsif/>
- [WSRF] WS-Resource Framework <http://www.globus.org/wsrfl/>
- [XCAT3] "XCAT3: A Framework for CCA Components as OGSA Services" S. Krishnan, and D. Gannon Accepted for publication to HIPS 2004, 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments. April 2004.