

On Using BPEL Extensibility to Implement OGSi and WSRF Grid Workflows

25 January 2004

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.

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 even today is in demand: ability to describe a process in a way that makes it easy to understand, change, execute, and monitor. 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 integrating Internet business resources by using XML and open source 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. Work on this idea was performed in 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 access to Grid service public state exposed as XML-based Service Data Elements (SDEs). Service Data Elements provide simple to use and standardized way to discover and manipulate 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. Standard OGSI services such as OGSI Registry provide way to find grid services GSH and map them to valid GSRs. Multiple GSHs and GSRs that identify a grid service can be combined together with a description of service interface to form an OGSI service locator. The Service locator is a useful abstraction as one grid service may have more than one GSHs/GSR but nevertheless is accessible through the same set of interfaces.

In recent weeks OGSI specification was refactored into set of WS Resource specifications called WS-Resource Frameworks [WSRF]. This addresses one of the main concerns about modularity of OGSI specification: now users can decide which WS-Resource specifications to use. Additionally WSRF moves away from object oriented approaches to services (GSH/GSR is like 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. Additionally the OGSI Factory interface is no longer present and is replaced by a factory-like creation pattern.

We expect that workflows in Grids will have a role as important as they already have in business environments. Considering relationship of OGSI and WSRF with Web services it is natural to expect that business workflows and, in particular, business

workflow languages designed for Web services can be leveraged 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 such standard workflow language. 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) 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 as part of a standardization process in OASIS.

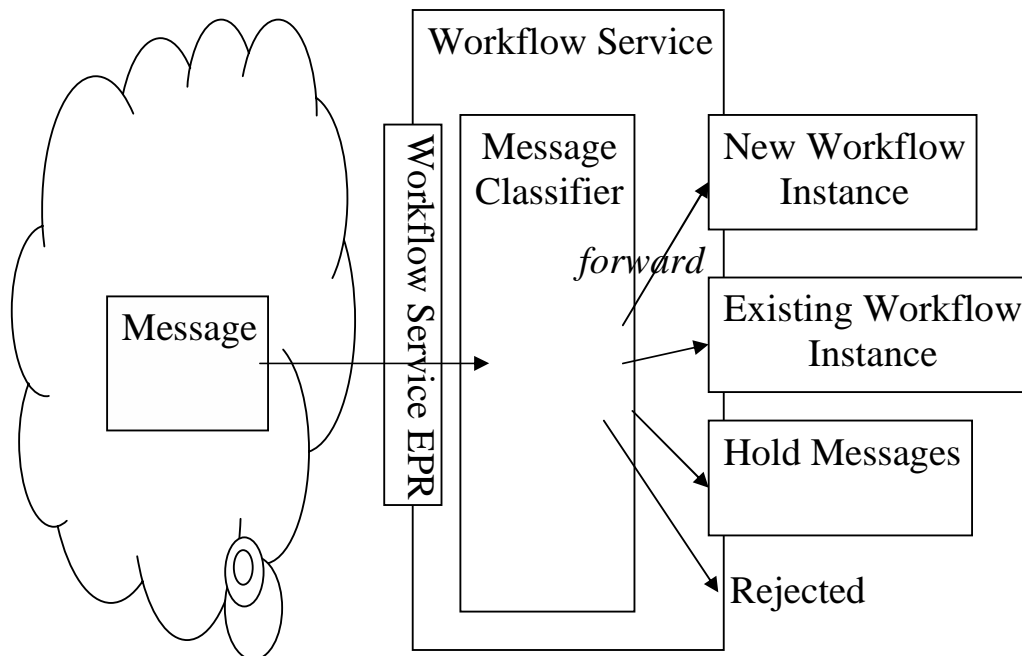


Figure 1. Critical part of BPEL workflow engine functionality is ability to route message to appropriate workflow instance, create a new workflow instance, hold message for later processing (ahead of time arrival), or reject messages that are incorrect.

What makes BPEL a very attractive candidate to use in Grid environments is its 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, determine their destination, and depending on workflows execution send messages to other Web services (see Figure 1 for more details).

In particular BPEL uses the WS-Addressing notion of endpoint references to simplify passing address of Web service participating in workflow. One very important advantage of using BPEL in Grids is that it can easily and seamlessly use both typical Web services (that are not part of Grids) and grid services such as provided by OGSF 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 environment for BPEL workflow management is 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 use standard 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 (simplifying by 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 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 bi-directional we want to avoid them as much as possible.

Integrating OGSi and BPEL

Workflow functionality is typically provided by Workflow Engine. 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.

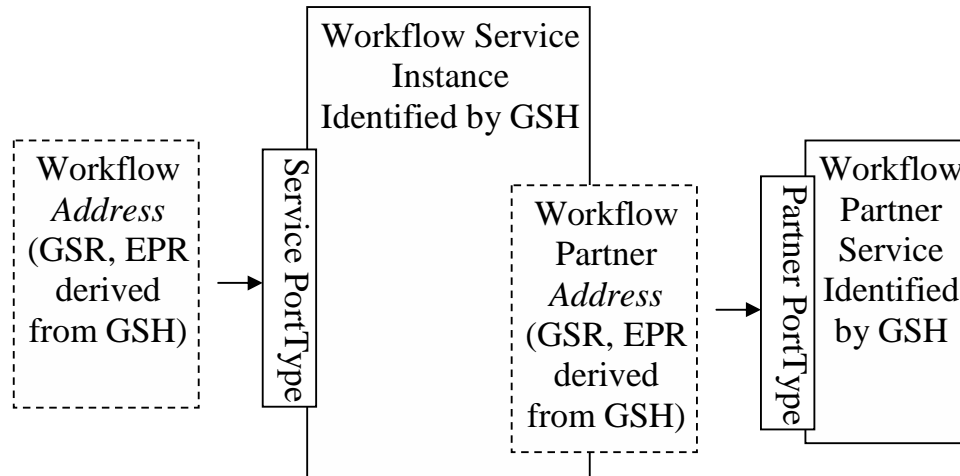


Figure 2. Each partner in workflow is bound to existing service identified by GSH or EPR. Also workflow itself is a service and is identified by GSH or EPR. During workflow execution one GSH may resolve to multiple GSRs.

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 in and to make BPEL workflows usable by OGSI clients.

The biggest problem that needs to be solved to use OGSI services is the fact that an instance of OGSI services may not exist until it is created by an OGSI factory. Even though a GSR corresponds to a WSDL document there may be need to use multiple GSRs during grid service instance lifetime (see Figure 2). Therefore it is necessary during OGSI service lifetime to track GSR expiration times and 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 location of used 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, XML reference properties can be used to contain 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 WS-Addressing endpoint references then such WSA endpoint references can not be successfully mapped to OGSI service locators (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.

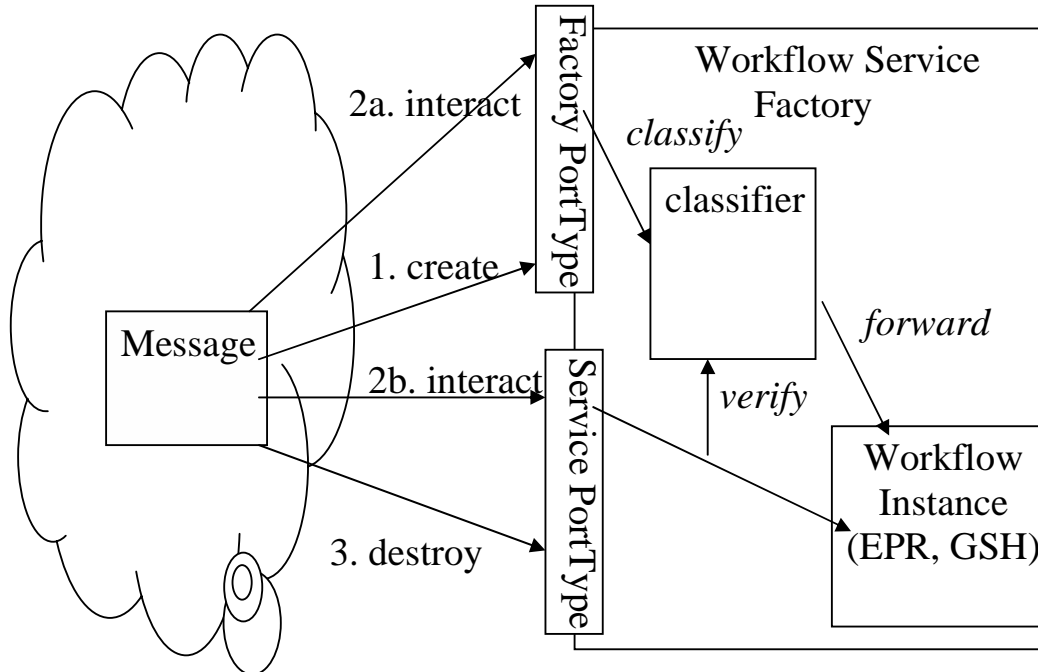


Figure 3. Workflow engine that supports both correlation based interactions (2a) and direct interactions with workflow instances uniquely identified by EPR, GSH, or similar mechanism (2b). Workflow engine must be able to distinguish between lifecycle messages and workflow specific message (1 and 2b) and may need as well to have special handling for destroy messages sent directly to workflow instance identified OGSIS GSH/GSR (2b and 3).

A related problem appears when a BPEL workflow instance is used (invoked) by OGSIS or non-OGSI clients. Each BPEL workflow instance may have multiple endpoint references identifying it (each endpoint reference corresponding 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 (see Figure 3). However amount of information in such endpoint reference may be not sufficient to identify the workflow instance that should be used to process an incoming message. That is because BPEL uses WSDL extensions to indicate which parts of the incoming message are to be used to correlate message with workflow instance to process it (those message parts form a *correlation set* that is used to identify a target workflow instance). That means that even though an endpoint reference may be in some cases mapped to an OGSIS service locator (when an endpoint reference has no XML reference properties or WS-Policy elements), still such a service locator may not be enough to identify the workflow instance as there is a need to know current values of the correlation set. This is a semantic difference: OGSIS 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 invoking the workflow instance. The solution to this is to make sure that mappings between endpoint references and service locators used in BPEL engine are bi-directional. BPEL engine will need to provide for each

workflow instance one or more GSHs and maintain corresponding GSRs. Those GSHs must identify exactly one workflow instance without need to use correlation sets.

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 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 BPEL workflow engine, the simplest solution is for a BPEL implementation to pass such request to BPEL engine as BPEL engine is already maintaining workflow instances lifecycles.

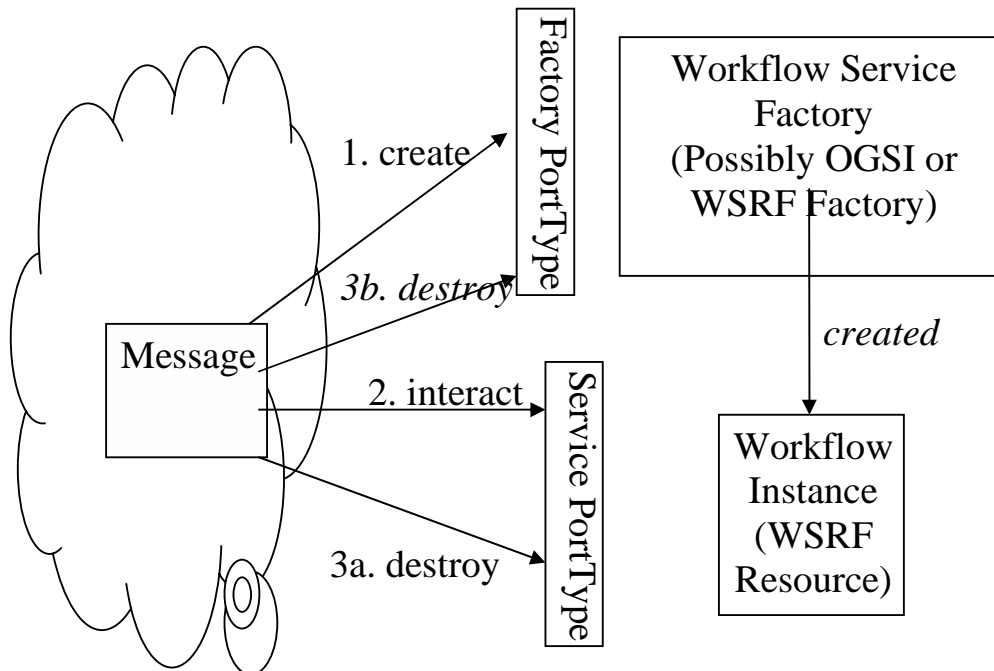


Figure 4. Timeline for workflows that use explicit lifecycle. Workflow instance is created with an unique address (or identifier) such as EPR or GSH (step 1), Web services send messages to workflow instance (step 2), when workflow instance is no longer needed it is destroyed by sending special message 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).

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 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 workflow instances.

Implementing OGSi Factory interface in BPEL engine means that second creation mechanism, besides one described in BPEL specification, needs to be supported. 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 (see Figure 4). 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 so it may be useful in some situations.

Trying to make a BPEL Engine support 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.

Fortunately OGSi Factory interface is optional. However if BPEL engine does not implement OGSi Factory interface then BPEL engine will not be very well integrated into OGSi based Grids.

WS-Resource Framework Integration Improvements

When comparing complexity of integrating BPEL with OGSi to integrating BPEL with WSRF it is obvious that this task is much easier when using WSRF. WSRF does not require that every grid service implements 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 just a pattern. In this pattern a WSRF factory is any Web service that returns in a response message WSA EndpointReference 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. 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 (typically first message sent to BPEL engine implicitly creates workflow instance). Moreover this allows for more fine grained interactions with BPEL as one BPEL workflow instance may have multiple endpoint references corresponding to different roles it takes when interacting with partner 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. 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).

If WS-Addressing becomes ubiquitous then using WSRF with other non-WSRF Web Services should also be easy. Otherwise use of a proxy service that will add WS-Addressing headers to SOAP messages is necessary. Because BPEL has the correlation set mechanism it is no required to use of WS-Addressing to interact with BPEL engine but WS-Addressing is required when interacting with WS-Resources in WSRF Grids.

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. And last but not least, Web and Grid services may fail so BPEL engine needs to reliable and to be able to cope with failures especially in case when workflows needs to be running for very long periods of time.

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 define a new kind of BPEL partner. 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 (see Figure 5). Such *pseudo* partners are not represented by real Web services and may be implemented as a part of BPEL runtime. 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 provided by Java library. 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.

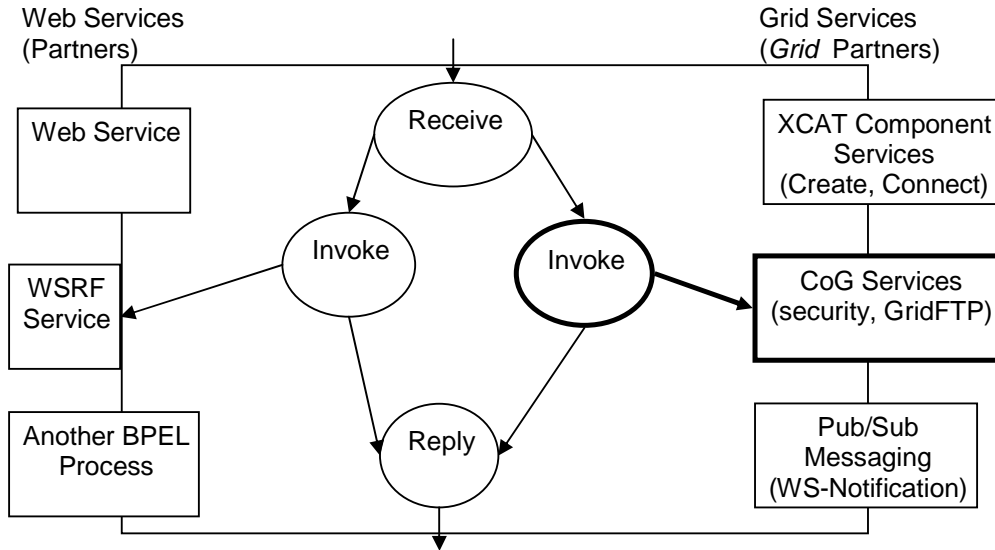


Figure 5. During workflow execution BPEL workflow may interact with many partners some of them may be Web services, WSRF services or other BPEL workflows. Additionally some partners may not be bound to actual Web service but instead bound to WSDL that describes local Java class, components, GridFTP services, or local WS-Notification publishing service.

BPEL Workflow Monitoring

Additional aspects of Grid workflow systems that reach beyond workflow execution are monitoring and managing of large number of workflows. As BPEL is an XML based workflow language, it is relatively easy to describe a workflow state in XML. By using XML to represent a workflow instance state it is easy to create a workflow monitoring Web Service that can be used by workflow clients to track workflows execution. If a workflow state is available in XML format then it can be exposed as OGGI Service Data Elements or WSRF Resource Properties.

When using workflows to coordinate 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. Such service is very helpful when trying to determine what was happening to a workflow instance.

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 a workflow engine must be able to deal with situations when grid services are down or are not working correctly (for example too slow). In this case 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 of 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 as 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 it is important 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 workflow instance state may not be possible. There is already 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 in BPEL context.

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 restarted (when for example machine was 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 network connection is lost (for example when machine is restarted) it is not possible to restore TCP connection and send response back (that underlines importance of asynchronous messaging as it does not require to keep connections open for long time). 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.

Supporting Large Data and Streaming

We see that there is a strong need to support large data sets and in particular data streaming as part of a workflow. As first step in this direction we think that seamless integration and making as easy as possible to access current tools 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 is enough extensible and flexible to be used to execute Grid 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 differences in particular challenge of seamless integration with Grid service (such as data handling, streaming, and security).
- *"What Web and Grid Services are required for 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 made convincing arguments for a position that BPEL can meet requirements for Grid workflow though there is still a lot of work needed to make BPEL a native 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/wsrp/>
- [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.