OGC Testbed-14

BPMN Workflow Engineering Report

Table of Contents

Publication Date: 2019-02-11

Approval Date: 2018-12-13

Submission Date: 2018-10-31

Reference number of this document: OGC 18-085

Reference URL for this document: http://www.opengis.net/doc/PER/t14-D026

Category: Public Engineering Report

Editor: Sam Meek

Title: OGC Testbed-14: BPMN Workflow Engineering Report

OGC Engineering Report

COPYRIGHT

WARNING

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR’s sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to

indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Chapter 1. Summary

This Engineering Report (ER) presents the results of the D146 Business Process Modeling Notation (BPMN) Engine work item and provides a study covering technologies including Docker, Kubernetes and Cloud Foundry for Developer Operations (DevOps) processes and deployment orchestration. The document also provides the beginning of a *best practices* effort to assist implementers wishing to orchestrate OGC services using BPMN workflow engines. As with previous investigations into workflow engines, the implementation described within utilizes a *helper class*, which is a bespoke implementation of some of the best practices. Work in future testbeds on workflows should include a compelling use case to demonstrate the power of service orchestration.

1.1. Requirements & Research Motivation

Workflows have long been a topic of interest for the OGC and have a corresponding *Workflows DWG*. Previous testbeds have focused on Business Process Execution Language (BPEL) that gained some traction, but has a set of fundamental problems to wide-spread adoption. BPMN is a language that addresses many of the issues associated with BPEL. The International Organization for Standardization (ISO) has approved BPMN as ISO/IEC 19510:2013 standard. The Testbed-13 Workflows ER described a rudimentary implementation a BPMN engine that sought to enable execution of remotely authored BPMN documents via a transactional web processing service. The test was successful, however, it was noted that there is no best practices for orchestrating OGC services using BPMN as the orchestration language. This ER provides a description of the BPMN best practices and corresponding implementation including issues with the approach and identified gaps for future research.

1.2. Prior-After Comparison

Prior to execution of this testbed, there were several areas for investigation for workflows based activities within the OGC, this testbed has answered these questions as well as producing a set of best practices for future workflows applications.

1.3. Recommendations for Future Work

Throughout this piece of work, several recommendations for future work have been identified. Testbed-14 sought to, among other things, generate a set of best practices for orchestrating OGC services using BPMN. However, there are several outstanding work items that need to be addressed, potentially in future testbeds.

- **Checking for likelihood of workflow completion success prior to execution.** Currently, workflows are executed *blind*, i.e. the user has no indication of whether the workflow will successfully execute or fail at some point. This is particularly important for long running processes, where the workflow could fail after several hours. The supporting documentation outlined in the Testbed-13 workflows report is a suitable starting point for this endeavor and potentially use a semantic registry.
- **Managing the BPMN to Web Processing Service (WPS) process transformation.** The BPMN document for this Testbed was supplied to the client implementer as a template for population.

However, there is not currently a way within OGC to map WPS to BPMN. It maybe the case that utilization of BPMN in WPS should be tightly coupled to the WPS 3.0 standard. As a start, the related OGC Testbed-14: WPS-T Engineering Report (OGC 18-036) describes a transactional extension for WPS 2.0 and recommendations for a process deployment profile for BPMN

- **Explore more sophisticated security encoding options.** In this testbed, security was handled by inserting the OAuth2 token into the BPMN document, which is translated into the HTTP header when WPS are executed. Future testbeds should explore access to different federations using different security models as well as finding methods to obfuscate security tokens. Some of the considerations are discussed in the related OGC Testbed-14: Federated Clouds Engineering Report (OGC 18-090r1). Additionally, security tokens have an expiry time. If a process is long running, then tokens may expire and cause the workflow to fail. A pre-testing or authentication process could allow a session to be initiated and refreshed if necessary whilst the workflow is running. A suggested solution to this problem is to use HTTP headers with multiple scopes.
- **Understand how to enforce removal of secured resource access to unauthorized services.** It is currently possible to extract resources from a secured service and pass them on to an unsecured service, methods for controlling this should be explored.
- **Error catching in BPMN.** The mechanisms of executing workflows are very complex by nature. If an error is thrown by a service, then it can be difficult to trace the source and report the error to a user in a recognizable fashion. Error catching and reporting specific to OGC services should be passed through the workflow engine and reported to the user in a human readable fashion.
- **Create a suitably complex, relevant and motivating use case for future Testbeds.** In Testbed-14 and with Testbed-13, the use cases for workflows are simplistic and have focused on the mechanics of getting software up and running. Future testbeds should have a sufficient number and range of services to orchestrate to utilize the BPMN language as well as an end goal to take advantage of aspects such as parallel processing, swimlanes, decision gates and compensation events.

1.4. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Contacts

Name	Organization
Sam Meek	Helyx SIS

1.5. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Chapter 2. References

The following normative documents are referenced in this document.

- OGC: OGC 06-121r9, OGC® Web Services Common Standard [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2]
- OGC: OGC 17-021, OGC Testbed-13 Security Engineering Report [http://docs.opengeospatial.org/per/17-021.html]
- OGC: OGC 17-029r1, OGC Testbed-13 Workflows Engineering Report [http://docs.opengeospatial.org/per/17-029r1.html]
- OGC: OGC 14-065r2 OGC Web Processing Service 2.0 Corrigendum 2 [http://docs.opengeospatial.org/is/14-065/14-065.html]
- ISO: ISO 19510:2013, Information technology - Object Management Group Business Process Model and Notation [https://www.iso.org/standard/62652.html]
- ISO: ISO 12651-2:2014(en), Electronic document management - Vocabulary - Part 2: Workflow management [https://www.iso.org/standard/42673.html]

Chapter 3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard OGC 06-121r9 [https://portal.opengeospatial.org/files/?artifact_id=38867&version=2] shall apply. In addition, the following terms and definitions apply.

- workflow

  ```
  automation of a process, in whole or part, during which electronic documents,
  information or tasks are passed from one participant to another for action,
  according to a set of procedural rules (source: ISO 12651-2:2014(en)).
  ```

- workflow engine

  ```
  software service or "engine" that provides the run time execution environment for a
  process instance (source: ISO 12651-2:2014(en)).
  ```

3.1. Abbreviated terms

- BPEL

  ```
  Business Process Execution Language
  ```

- BPMN

  ```
  Business Process Modeling and Notation
  ```

- WCPS

  ```
  Web Coverage Processing Service
  ```

- WFS

  ```
  Web Feature Service
  ```

- WMS

  ```
  Web Map Service
  ```

- WMTS

```
Web Map Tile Service
```

- WPS

```
Web Processing Service
```

- WPS-T

```
Transactional Web Processing Service
```

Chapter 4. Overview

Section 5 introduces the ER and the corresponding component task within the NextGen Thread in the OGC Testbed-14.

Section 6 provides a review of motivating work for producing this iteration of the BPMN workflow engine and the rationale for the Docker, Kubernetes and Cloud Foundry study at the end of this document.

Section 7 provides an overview and discussion of the BPMN 2.0 standard, the language used to express workflows in the BPMN workflow engine component.

Section 8 outlines best practices for orchestrating OGC services using BPMN.

Section 9 describes the component implementation for the D148 BPMN Engine.

Section 10 consists of the Docker, Kubernetes and Cloud Foundry study required as part of this ER.

Section 11 concludes the document.

Chapter 5. Introduction

This OGC Engineering Report (ER) describes the work carried out in the Testbed-14 initiative for utilizing BPMN workflows and provides the results of a study into technologies including Cloud Foundry, Docker and Kubernetes. The foundational work for this document is the Testbed-13 Workflows ER, which describes many of the grounding concepts and the state-of-the-art in OGC compliant workflows. This document is in turn based upon the practices outlined in [1], motivating document for using Business Process Modeling and Notation 2 (BPMN2, styled as *BPMN*) within OGC, and [2] that provides a preliminary implementation. Additionally, there are explorations of workflows using technologies such as BPEL in OGC, notably from Testbed 8.

Testbed-13 provided many successes for the authors of the ER and design and implementation of the system described within. The Testbed-13 Workflows concept demonstrator enabled execution of workflows using BPMN and a Transactional Web Processing Service (WPS-T), however, the demonstrator required a *helper class*, which is bespoke to the Camunda implementation of BPMN. The workflows work in Testbed-14 proposes an OGC Best Practices methodology for executing workflows and presents a demonstrator implementation of these best practices in the form of a similar helper class. Due to the available services for this testbed, the best practices are presented as implementation recommendations, not utilization of the BPMN language for orchestrating OGC services, which should be explored with a suitable use case in future testbeds. Additionally, it is the objective of this ER to identify solutions to some of the shortcomings of the Testbed-13 approach to workflows and to describe solutions.

The main issues cited with the BPMN helper class in Camunda was the utilization of data inputs and outputs, as each implementation handles this aspect differently. Additionally, BPMN provides standardized methods for managing data both in flux and at rest that were not utilized within Testbed-13 work. BPMN concepts such as *Service Tasks* also provide a mechanism to input data via direct input or reference, however there is also the concept of *data objects*, which seek to model data inputs and outputs in a formal way. This document aims to address ambiguity and contradictions between the standard and available software whilst providing guidance to implementers on OGC best practice. This document reports on a demonstrator implementation of the stated draft OGC Best Practice and the results of a study on Cloud Foundry, Kubernetes and Dockers with their applicability to the OGC and the wider geospatial domain.

Chapter 6. Review of motivating work

This section contains an overview of the existing OGC work in workflows. It is noted that work has also been done outside the OGC that has either been reviewed in the publications addressed here, or is deemed out of scope for this ER.

6.1. Testbed-13

There are two publications from OGC, notably Testbed-13, that are considered in scope for motivating work for this ER:

- 17-021 OGC Testbed-13 Security ER [3].
- 17-029r1 OGC Testbed-13 Workflows ER [4].

6.1.1. OGC Testbed-13 Security ER

Security continues to play an important role in the development of OGC standards, practice and doctrine. Implementation of security is also a requirement for all work done in the *Next Generation Services* thread in Testbed-14. Therefore, encoding security into a workflow request is a requirement for this work.

The three motivating use cases outlined in the Testbed-13 Security ER are:

1. Use Case 1: Dominating Privileges - the situation where the user has more privileges than the computer system being used. Access attempted in this way will result in a security violation.
2. Use Case 2: Tunneling Proxies - passing of credentials is likely to be interrupted as a proxy constitutes a new connection to the resources.
3. Use Case 3: Identity Mediation - mediation between different security models is required in a chained workflow, as it is likely different services will use different authentication procedures and providers.

This section briefly discusses the considerations from the ER that are relevant to the work presented here.

OAuth-enabled Web Processing Service

As part of the Testbed-13 Security ER, an included work item was how to enable WPS to be authenticated using OAuth tokens. Following on from this, Testbed-14 includes security to authenticate workflows in all scenarios.

Figure 1 outlines the abstract OAuth flow including the resource owner, the authorization service, and the resource server. In the flow, the client requests access to the resource via the resource owner. The owner grants authorization to the client who then requests a token from the authorization server using the grant from the resource owner. The authorization server then provides the client with an access token that is sent to the resource server to gain access to the resource.

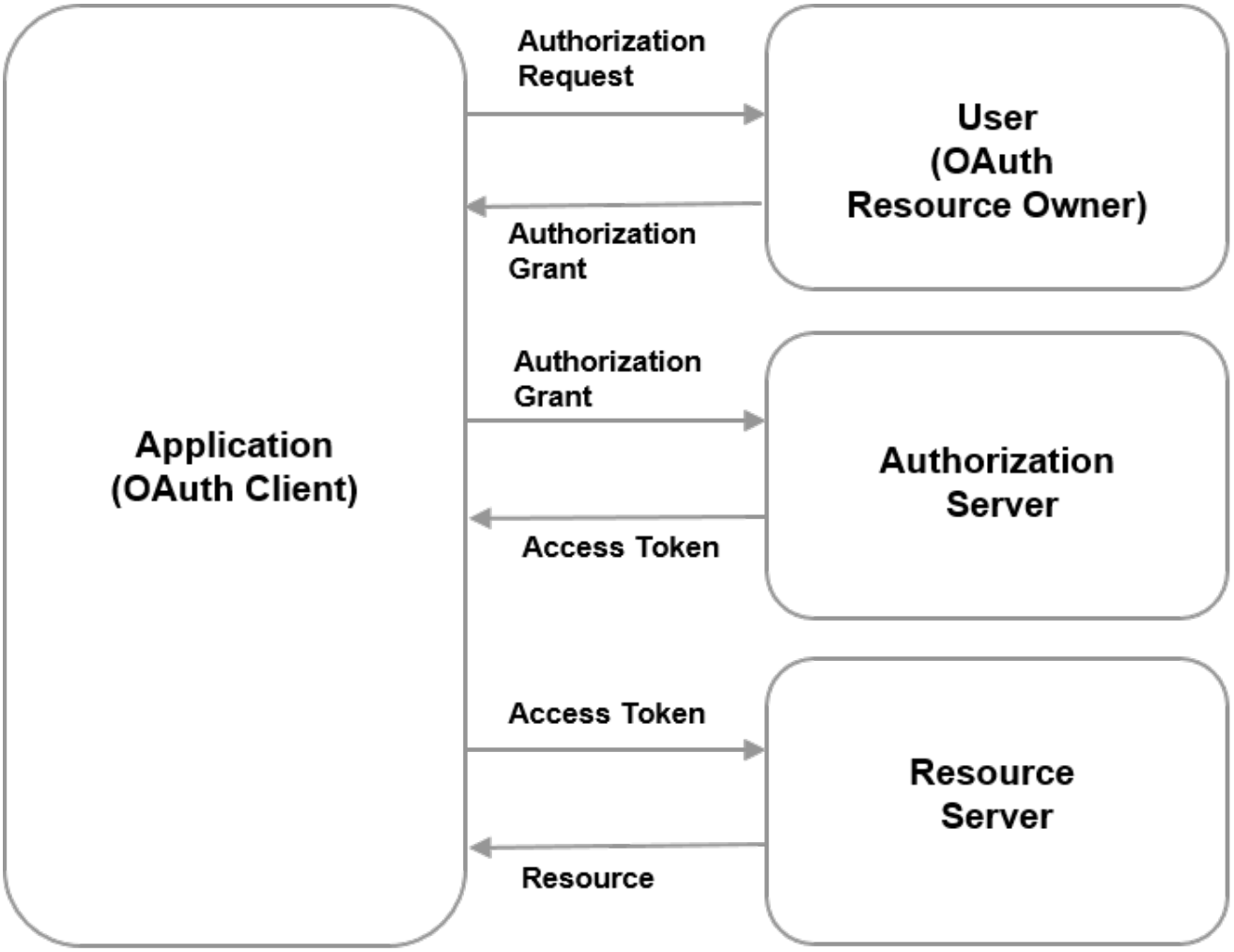

Figure 1. High-Level Testbed-13 Abstract OAuth authorization flow

A key aspect from this ER is the Authorization Code Grant Flow, which is the complete flow described in Figure 1. In this scenario, credentials are granted prior to resource access requests, therefore removing the requirement to register the application. Authorization can be done using a private authorization server or, in the case of Testbed-13, it is the Auth0 service (https://auth0.com). One of the motivating requirements for this ER is to *explore the use of code grant flow* in workflows. This requirement will be addressed later in the document.

6.1.2. OGC Testbed-13 Workflows ER

The Workflows ER provided an account of the work done in Testbed-13 workflows and included information on workflow construction and execution, an overview of workflow engines, transactional web processing services for accepting BPMN workflow documents, data quality services, conflation services and clients for constructing an executing the workflows. The scenario from Testbed-13 is shown in Figure 2 where the workflow client aspect stored existing workflows in a Catalog Service for the Web (CSW) and submitted the workflow to the workflow engine for execution. The workflow engine then orchestrated and executed the three WPS services to produce a result. The Web Feature Service (WFS) is utilized as a data repository within the workflow, but not formally by the workflow engine as the data is passed to the first WPS in the chain through configuration of WPS parameters.

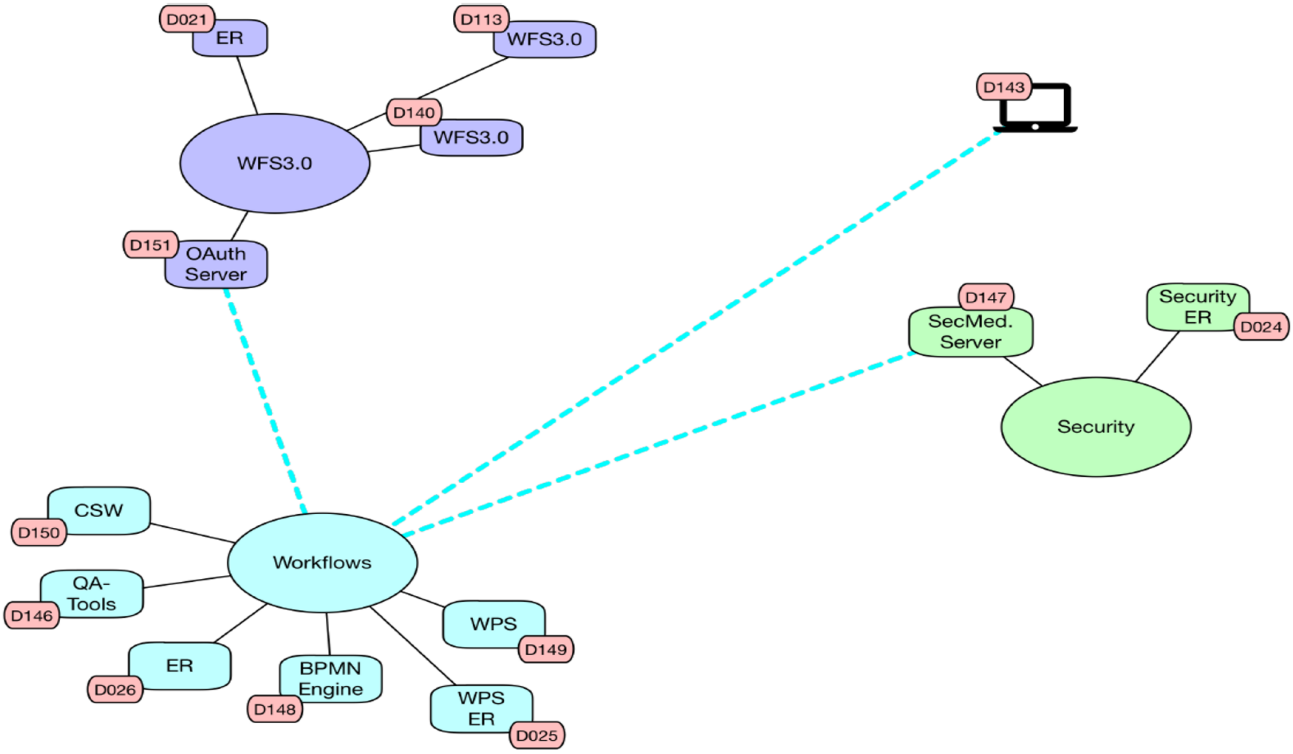

Figure 2. High-Level Testbed-14 Next Generation work items overview.

Data objects

The workflow engine has no knowledge of the concept of *data objects*, either in the formal BPMN definition or informally as part of service tasks, as data endpoints are passed as string references via a Service Task parameter. Although this method is acceptable in terms of the BPMN standard, data objects should be modeled formally within a BPMN diagram where the implementation supports it. This represents one of the challenges addressed in the ER, as BPMN has a well-defined set of methods for handling data objects whether at rest or in flux. Understanding how to de-couple data objects from processing capability so that they can be reused, if necessary, within a workflow is also an objective of this ER and is discussed in detail later in the document. One of the issues mentioned in Testbed-13 was the encoding and decoding of data being a hurdle to interoperability as BPMN workflow engines tend to handle the encoding and decoding procedures differently. The best practices outlined in this ER accept this as being part of an implementation standard and does not seek to mandate any specific encoding or decoding procedure, but instead rely on the capabilities of the orchestrated services to deal with data inputs and outputs. The workflow engine should remain neutral in its data format requirements and be used to orchestrate existing capability rather than introducing new processing capability. Alternatively, if orchestrated services do not contain a common format, then the workflow engine may call upon a partner WPS to perform the translation.

The Testbed-13 scenario and users

The scenario outlined in the Testbed-13 Workflows ER consists of two use cases and corresponding user groups. These are:

- The expert/workflow configurator.
- The user/workflow executor.

These distinctions are not necessarily mutually exclusive, as it is envisaged that users will be able to construct or at least tweak existing workflows to their requirements. Likewise, the expert may be required to execute workflows that either they or another expert has created. In the scenario, the expert creates and configures the workflow via querying a CSW for processes that fit their needs. They then compose the workflow and configure all of the parameters including the variables and end points. They then send the composed document to the WPS-T, which calls the workflow engine for execution. The cataloged processes are protected by an OAuth Security Service. Once authorization takes place, then the workflow engine WPS harvests the processes and data and the workflow is constructed (Figure 3).

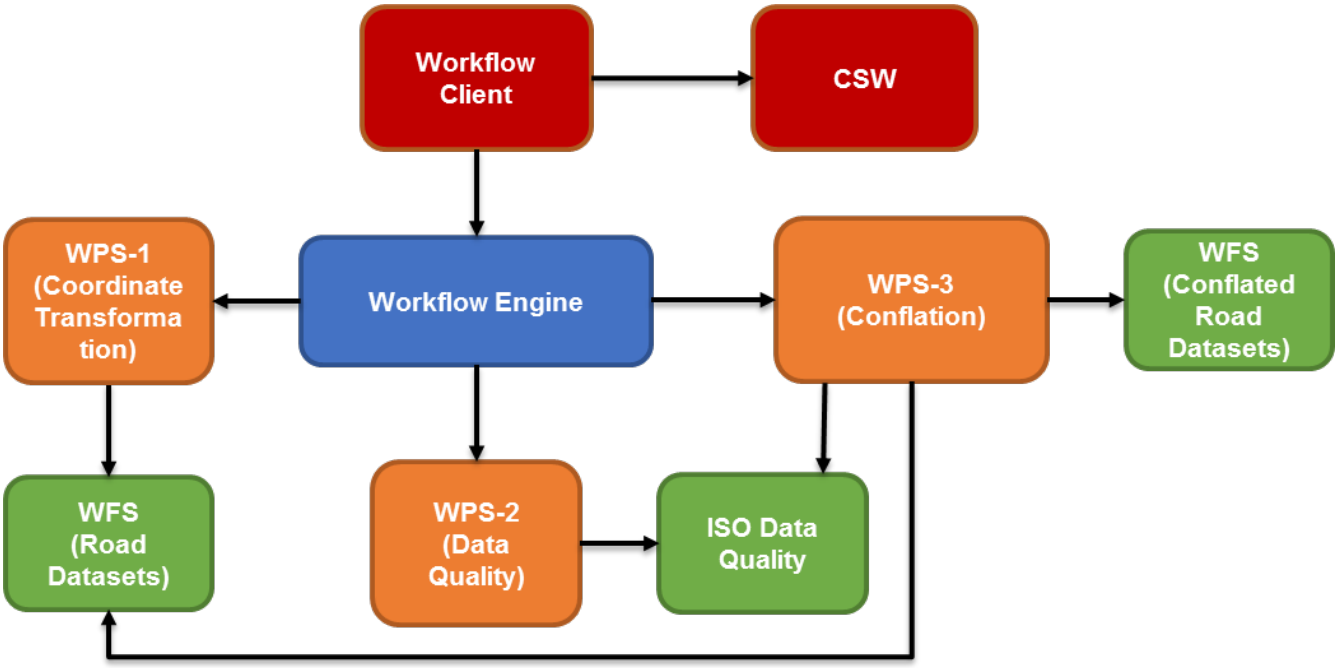

Figure 3. High-Level Testbed-13 Workflow Overview

A notable issue with this use case is that there is no workflow execution and no testing whether the workflow is valid prior to making it available in the catalog. Therefore, the expert (or user for that matter) does not know whether the workflow will execute in the given scenario, and also does not know the scope that the workflow will operate within. For example, if the user (non-expert) retrieves the workflow, executes and then receives an error, then this will likely prove problematic for issue resolution. A method to address this is to use the *getErrors()* call within the WPS to return any errors to the user. However, the mechanism for returning errors to the WPS is not yet determined and will likely be represented by well-defined error catching mechanisms from within the BPMN standard.

Decoupling error reporting and handling from the WPS is an approach that fully utilizes the BPMN standard whilst ensuring an implementation is service agnostic for maximum flexibility.

The capabilities of the workflow engine have been somewhat enhanced in the Testbed-14 version of the engine as it no longer requires the user to *setup* a workflow and then execute it with a set of parameters. Instead the BPMN document can be submitted and executed by the workflow engine regardless of whether the workflow engine has prior knowledge of what is to be executed. This added functionality is due to the choice of base implementation used rather than anything to do with the BPMN language. This has implications for security, as a completely unsecured workflow engine service would in theory be susceptible to misuse; as it could be instructed to execute anything a client sent it. This issue is resolved in the demonstrator by only allowing execution of a

single workflow *type* and variable passing within a well-defined scope.

Workflow engine configuration

The Testbed-13 workflow engine was based on Camunda fronted by a WPS-T to enable access to the standardized OGC calls as well as a new *insertProcess* call that enables a process to be inserted into the WPS using the HTTP POST operation. A diagram of the workflow component setup can be found in Figure 4.

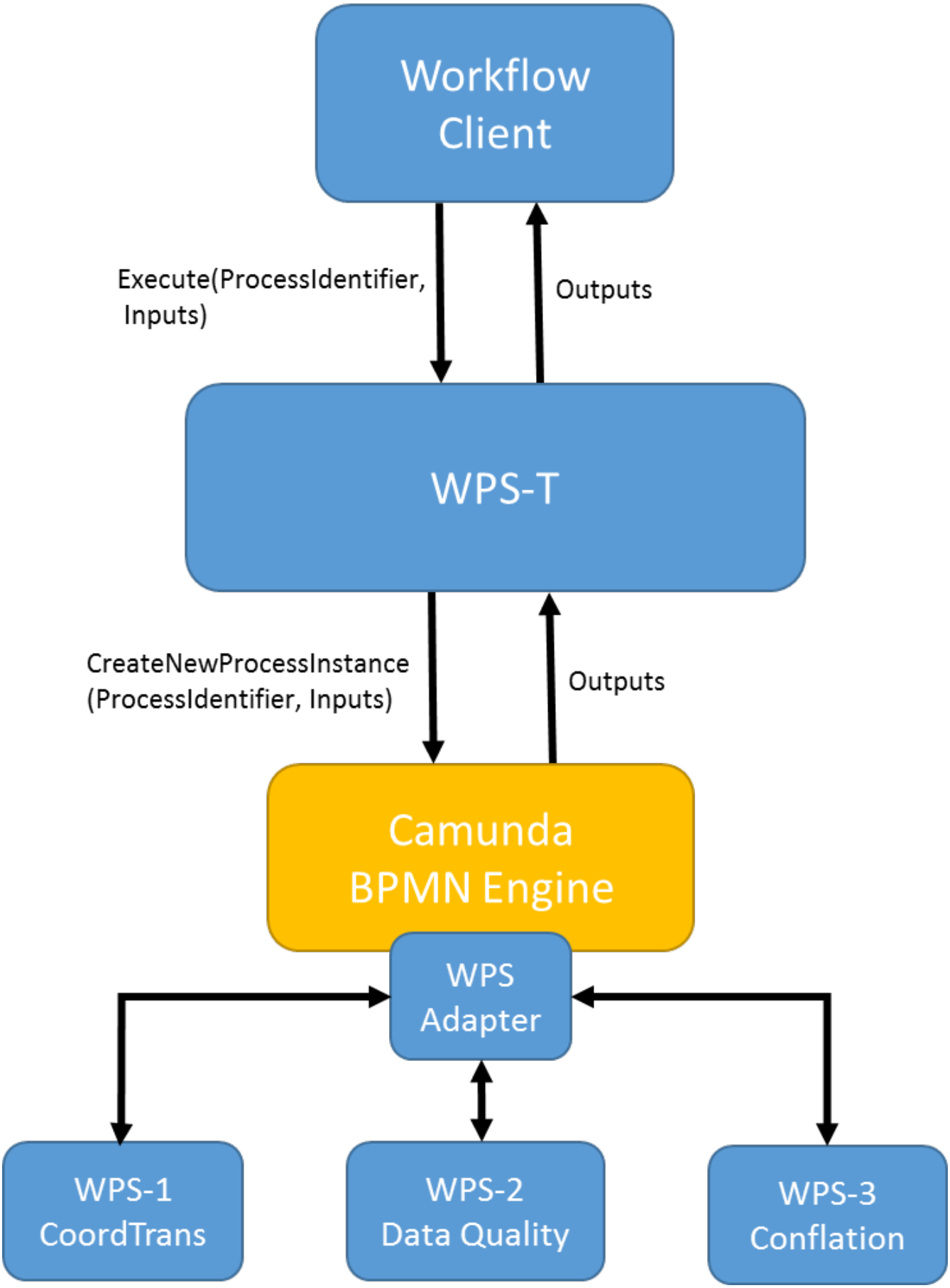

Figure 4. High-Level Testbed-13 Workflow Components

Fronting a workflow engine with a WPS enables standardized calls to be used to execute workflows. However, it should be noted that it introduces a level of verbosity that is not found when using the raw workflow engines, many of which are already fronted by a Representational

State Transfer (REST) Application Programming Interface (API). Additionally, the construction of a *helper class* is required for workflow engines to understand the different data models and types relevant to *geoprocessing*. In the Camunda example, the helper class parsed data by utilizing a GeoTools internal construct called a *FeatureCollection*. This concept of a collection of features is prevalent throughout geographic data endpoints for vector data, however, coverages are delivered in a number of file formats that should be considered when defining *best practices* for OGC workflows. The jBPM BPMN workflow engine uses a combination of the Git protocol and a REST API to submit and execute workflows. The detail of this is discussed in a later section.

Response to recommendations from Testbed-13

There are several recommendations that were made at the culmination of the ER, many of which have made it into requirements for the work described in this ER. Briefly, the relevant workflow recommendations are described below with the responses to the recommendations including actions taken:

- Investigate how client functionality to communicate with OGC web services can be made usable by different workflow engines. This recommendation is the cornerstone to the efforts made in this ER, understanding how BPMN as a *language* relates to OGC services with a focus on data models will likely solve re-usability and verbosity issues in workflows whilst enabling interoperability between workflow engines beyond demonstrators in Testbed-13 and 14. This is initiated by returning to the language, making recommendations in the form of best practices and then demonstrating these best practices in an implementation. The solution to this requirement outlined here is to submit the BPMN document wholesale and then implement functionality via the *helper class* to register work items and produce the result. Note that the beginning of a BPMN language mapping exercise is started, but was not tested due to the restricted WPS services made available.
- Investigate a common approach to handle inputs and outputs defined in BPMN. The data modeling aspect of interoperability will help towards realizing the first recommendation in this list. BPMN essentially has two data concepts, the first is a *Data Object*, which models data in flux, and a *Data Store*, which models data at rest. OGC services have a subtle, but important distinction in that data are supplied as *services*. Inputs to WPS and other services are provided as *parameters*, rather than existing as standalone objects. In truth, BPMN allows for references to data to be passed as a parameter. However, data objects should be utilized to model the *flow* of data between services. Reconciliation between these two concepts is described later in the document. It should be noted that OGC architectures do not really account for *data objects* as such, instead it deals with *services* and more recently, *resources*. These concepts are covered by the BPMN *service task* and *data objects* should not be used to refer to external resources, but they can be used optionally to orchestrate processes *internally*.
- Investigate process discovery mechanisms. Process discovery in Testbed-13 was done in a catalog, however the processes were presented to the client via an *identifier*, which is suitable for a user who understands what all of the processes do (an extreme case of the *expert user* use case in the Testbed-13 Workflows document), but potentially problematic for other users. It has been suggested that the user is able to discover processes from the catalog using metadata, however a profile for WPS processes has not been agreed upon. There are ISO standards that are likely to fit the requirements including ISO 19119 - Geographic Information - Services, however further discussion is required. Process discovery in Testbed-14 is done through the

CSW that stores the registered WPS processes, it is also populated with BPMN workflows described as WPS processes. Therefore, discovery should be completed in the recognized OGC way, through a CSW.

- Investigate OAuth Code Grant Flow for dynamic authorization. OAuth for authorization has been gaining traction within the OGC, along with SAML and some other technologies. This recommendation largely follows some of the work completed in the COBWEB project where the concept of a *session* in terms of security was discussed [1: https://cordis.europa.eu/project/rcn/105504/reporting/en]. Essentially, a session can exist at three different levels within a workflow:

  1. The process level.
  2. The workflow execution level.
  3. The workflow composition level.

These three levels offer different security grains in ascending order. Security at the process level is a useful model if processes require different levels of access to execute. In the Testbed-13 setup, it was prudent to deny access at the workflow composition and orchestration phase, rather than upon execution. The workflow execution level means that the user authenticates as soon as the workflow is executed and the session exists for the duration of the workflow execution. This concept notably lends itself to synchronous rather than asynchronous processes. The workflow composition level means that the user authenticates in a federated fashion, i.e. once they are authenticated, then they have access to everything. This scenario appears to be most suited to the motivating use cases for this ER.

For Testbed-14, it was decided that the workflow engine would simply pass the security information in the WPS execute requests generated during workflow execution. Although this is a simplified use case, it shows the beginnings of truly secured workflows with options for resources sitting in different federations that should be explored in a future Testbed. In terms of the levels of security, this is an example of authentication at the process level.

- Investigate security encoding aspects in BPMN.

Encoding security in a BPMN document is likely to require a best practices approach, as the BPMN standard does not mention security in any meaningful way beyond a normative reference. However, the standard is flexible enough to be able to support security, even if it is carried as a parameter in a service task. It is possible to extend the standard to enable further execution semantics, however the support for extensions in implementations is unknown and a strategy for implementing security in an extension is somewhat of an undertaking. In this ER, authorization tokens are passed along with the http header information in an execute request generated by the workflow engine. In this instance, the BPMN engine is acting as a pass through for security tokens as authentication to processes and data will have been done prior to execution via request of a token. A more thorough examination of security models will likely be the work of future Testbeds and will span more than just a piece on workflows.

6.2. Other documents

Documents outside the Testbed initiative have acted as motivating work for this ER and ERs from previous Testbeds. The two documents to be considered are:

- A BPMN solution for chaining OGC services to quality assure location-based crowdsourced data [1]
- BPMN 2.0 for Orchestrating OGC Services Discussion Paper [5]

These two documents created the foundation for bringing BPMN into the OGC as a language for expressing service orchestration. The use case for OGC service orchestration has always focused on the data quality domain, this is potentially due to the requirement for configured, finely grained services to establish data *fitness for purpose*. Although a suitable use case for orchestration, there are several points highlighted in the documentation that are considered:

- Configuring processes is time consuming and complicated and generally only undertaken by an expert.
- Workflow orchestration and execution rely on external services that the engine may not be able to validate remotely, therefore errors within the workflow engine might be uncontrollable and cause the workflow engine to fail.
- Security was not considered within these documents either in terms of the workflow engine security beyond basic authentication or in terms of secured services.
- The output mainly consisted of metadata, which meant that the input data was not changed during the workflow. Ramifications of altering the data during the workflow are discussed later in this document.

Previous testbeds have attempted workflows using Business Process Execution Language (BPEL), which provided some success, but suffered with the following fundamental issues:

- Lack of a standardized graphical interface.
- Issues with execution due to lack of correct WSDL bindings.
- Complexity and skills required to execute.
- A general lack of uptake.

Therefore, BPMN was put forward as an alternative as it addresses the stated issues and has full interoperability with BPEL as conversion can be done losslessly. There have also been other suggestions to enable orchestration of services within OGC, notably to return to the WPS standard and make orchestration and chaining fundamental to the standard, rather than optional.

6.3. Open questions from Testbed-13

Testbed-13 made recommendations that have in turn been reiterated as *requirements* in Testbed-14, however there are some notable omissions from this set that are discussed in this section. These suggestions and observations are generated from a combination of prior experience from projects such as Citizen Observatory Web (COBWEB) [http://cobwebproject.eu], LandSense [http://landsense.eu] and the other noted documents in this space.

6.3.1. Testing for workflow validity prior to execution

When a user constructs a workflow for execution they currently do not know if the workflow will run or fail. This problem is exacerbated when there are workflow processes with dependencies on

inputs and outputs. There are in general two types of processes that should be considered here:

1. Processes that do not change the input data - unordered. These processes are likely to consist of information gathering algorithms (for example, processes that generated metadata, as seen in the data quality WPS in Testbeds-12 and 13). Here the input data does not change in the process, and all processes following a process of this type do not need to consider the order that processes are executed, i.e. the processes can be executed in any order to produce the same result.
2. Processes that do change the input data - ordered. In this scenario, the order that processes are executed in will likely provide differing results, null results or an error. For example, there are two processes:
    a. process 1 - filters a dataset according to input variables
    b. process 2 - selects data within a distance of a set of input points

If a workflow were constructed where process 1 is followed by process 2, this could potentially produce a resultant data set than if process 1 followed process 2. A likely occurrence is that if the filtering or selection criteria are two strict then the first process in the chain would produce a *null* result. This would then result in an error message and workflow failure.

Managing the ordering and production of results for processes is an on-going issue that is likely to remain unsolved in this Testbed as it is out of scope. However, solutions should be considered to mitigate, be it through sampling, some sort of semantic validity of workflows and/or suitable error messaging to make the user aware of process failures and a reason for the failure. Construction of some OGC specific error messaging should be considered in future Testbeds.

6.3.2. Data modeling

As mentioned previously, the data modeling aspects of this ER are key to the motivating requirements and the best practices endeavor. BPMN has the concept of *data objects* and *data stores*. The former is for modeling data in flux and the latter is for modeling data at rest. OGC services and in particular; WPS take data as parameters either by reference or as a raw type. When orchestrating OGC services, ideally this should be de-coupled and data objects passed by reference as a BPMN data object or store; rather than as an input parameter to a WPS process. The data object as a reference could then be mapped to a WPS process parameter therefore making the data object reusable throughout the workflow. Additionally, this method of mapping data objects enables users to graphically model data flows as well as process flows within the BPMN workflow diagram. Generally, the BPMN constructs for data modeling are suited to internal orchestration of data and references to data, rather than providing methods to access external data. BPMN also does not have a formal method for querying internal process variables post process completion, i.e. workflows are self-contained, therefore, getting data in and out of workflow is defined by the services that form part of the orchestration effort.

6.3.3. Helper class construction

The implementation of demonstrators supporting the workflow engine work items have had to implement a *helper* class. In this implementation, the helper class is simplified to act as a WPS client, thus offloading the capabilities required for executing OGC services to resources that already

have it.

The issue with geospatial data structures is experienced when workflows are executed synchronously, as the data are retrieved from an executed process and then passed to the next process. This could potentially be mitigated through passing of links in an asynchronous fashion. However, this would restrict the functionality of the system as it would make demands upon it that are not OGC compliant. The data inputs and outputs point relates back to the concept of data objects, which has been covered in previous sections. This helper class by default passes the results as a reference from one process to another, thus removing the requirement for the workflow engine to constantly be passing data back and forth from the engine to the next service in the chain. This approach does have a drawback in that every service in the chain is acting as both a client and a server. However, this is preferable to putting data load onto the workflow engine as it could easily form a bottleneck for processing and data.

A consideration for future work in this area is to have a system that tests for data compatibility within the workflow to make a decision on the approach to take, that is, passing by reference or passing by value. This would ensure full data compatibility throughout the workflow while making the most efficient use of computational resources.

Chapter 7. BPMN 2.0 discussion

BPMN 2.0 is an Object Management Group (OMG) standard ratified in 2011 and designed primarily to model business processes through a graphical interface. In addition to graphical modeling, it is also able to execute services and model human interjection into an automated process. It also contains typical systems type operations such as error catching and messaging and multiple exit paths from a system. Due to its flexibility and simple user interface, it can be utilized by both novices and experts to communicate a workflow.

As BPMN is a normative standard, it builds upon rules to ensure harmonization across implementations. This section does not recreate the normative documentation, it instead reviews some of the aspects that are of consequence to providing an OGC best practices effort that can be implemented across BPMN workflow engines via the aforementioned *helper class*.

7.1. Activities

BPMN contains the concept of *Activities*, these are essentially how the standard models work being undertaken within the workflow either in an automated fashion, or by manual intervention by a user. There are several *Tasks* within the *Activities* grouping that have relevance to orchestrating OGC services, these are:

- Service tasks
- Send tasks
- Receive tasks
- Script tasks

7.1.1. Service tasks

BPMN describes Service Tasks as those done by software. Essentially the user configures a service task with parameters required by the external service which is then executed as part of the workflow on behalf of the user. A key aspect of a service task is that it is synchronous, therefore the workflow will wait for a result before continuing. This can of course cause problems if a web service times out or is non-responsive. This can be mitigated using an *Event* which is described in a following section.

From an OGC perspective, Service tasks are best used for orchestrating OGC web services that are synchronous, i.e. the workflow engine configures a process, executes the process and then passes the result onto the next process. This requires the workflow engine to understand spatial data structures and concepts, this is expressed in the *helper class* for the software package being used.

Service tasks should be used for *processing* rather than internal *data management* as these are best described in *data objects* and *data stores* which are discussed later and used for internal workflow data management. Therefore, OGC services such as WPS and Web Coverage Processing Service (WCPS) are best represented using Service tasks. Note that service tasks are different from *Script* tasks that execute a simple script upon execution. Script task could feasibly be used to interact with OGC services. However, the script task is designed as more of a catch-all to execute a script within the workflow rather than to interact with *well-defined* services.

7.1.2. Send tasks

Send tasks are used to send messages between processes and swim lanes. These are generally used to initiate another part of the workflow via sending a message to do so. This type of task does not have the ability to await a response from the service task and is completed as soon as the message is sent. A typical use case for a send task is to initiate a branch of the workflow that is separate from the *primary* swim lane. As with the data modeling aspects of BPMN, send tasks refer to *internal* workflow orchestration rather than messaging external services that are orchestrated using the *service task*.

7.1.3. Receive tasks

Receive tasks are the Send task counter-part and will await a message sent (usually from a send task) before initiating. This task is usually used at the beginning of a swim lane to initiate tasks after instantiation. The task is considered complete as soon as the message is received. As with *send* tasks, these are designed for internal orchestration of workflows.

7.1.4. Multiple task instances and looping

BPMN has the ability to model multiple instances of a task in a workflow as well as using loops for iterative execution of a task. These can be used as normal in OGC services, as the two examples are simply executing instances of a process with a given pattern.

7.2. Swim lanes and pools

Swim lanes and Pools are ways of organizing a BPMN diagram to describe *collaboration* between entities. In the case of OGC services, swim lanes can be used to model the location of different processing entities. For example, should the workflow engine have access to different WPS instances, then each of these instances could be represented by a Swim lane to model result passing between the different services.

7.3. Events

There are several start events that can be used to orchestrate OGC services. Start events define how a process is initiated. It might be via a manual process (i.e. someone calling the workflow from a REST endpoint), or it might be via a message received from another process or third-party application. For the most part, the standard BPMN guidance documentation is sufficient. However, there are some specific aspects of BPMN where extra notation is necessary.

BPMN does not differentiate conceptually between events that can start and end a process. For example, it is possible to both start and end a workflow using a *message* event or any other type of event for that matter. In addition to start and end events, there are also intermediate events that may or may not interrupt a workflow according to certain conditions.

7.3.1. Messages and signals

Message start events are those that initiate a workflow based upon some external event. A typical example of this is that the workflow is called by an external process. This is in contrast to a

standard start event where the workflow is called by a client interacting with the API interface of the workflow engine. Additionally, the Message event start does not discern between users or calling systems, it simply executes having received the message. In contrast to this is the signal event, where the workflow is awaiting a specific signal to initiate. This is likely to be a specific client or execution pattern rather than the general one that messaging is suited to. Signal events also have more flexibility for communicating aspects other than simple error messages, as is expected with the Message event. Likewise, an end event will act in the same way by throwing either a signal or message to a chosen service to end the workflow. Note that an end event is different than the *email* event, which is specifically setup to email results to a configured account.

Intermediate events for messages and signals are triggered in much the same way as the start and end events, however they are triggered before the workflow has completed.

Signaling functions are available from outside the workflow during execution and are often used to *hold* the workflow until an external task has been completed. This has crossovers with the *Human Task* aspect of BPMN, which relies on an external actor to provide information to allow the workflow to continue.

7.3.2. Error

Error events capture and report on errors based upon a condition. For example, a workflow error might be that a process has failed for some reason, be it a coding error or unexpected behavior experienced due to the input parameters. Error messaging is a catch-all in terms of BPMN, but should be used to communicate the errors from processes or data objects. Typical uses of an Error event include; mis-configuration of processes, reporting a lack of authentication for services and data type conflicts. Error catching should be used to report back to the workflow executor in human readable language. An example of where this is useful is to watch for a null result from an external process and report back accordingly, rather than simply throwing a language specific error.

7.3.3. Compensation

It is difficult to predict how workflows will behave with different sets of input data and parametrization of processes. For example, a workflow that executes for one set of data may fail for another set. This could be due to an *Error* in the typical sense (i.e. there is something wrong with one of the processes) or it could be that there is no error and the processes executed correctly but it failed for another, data dependent reason.

Considering a workflow that has two dependent processes expressed as Service tasks, with the second process configured to use the outputs from the first process. If the first process produces an output that contains zero or a null result, then it is likely that the second process would fail. Although the workflow may fail at this point, this should not be considered an *error* as everything in the workflow acted exactly as it should. A compensation event tells the workflow how to behave in this eventuality; it might execute a message to the user informing them of the reason for the event being triggered, alternatively it may provide an alternative execution path for results that conform to the compensation requirements.

Orchestrating services that include compensation is prudent when the workflow is complex with many branches. It may also be considered an extension of the error event instead of reporting an

error, it activates an alternative path to *compensate* for the error.

7.4. Data Modeling

The data modeling aspect of BPMN is key to creating best practices as it is one of the aspects that differ between implementations. As alluded to in the review sections of this document, providing a suitable method of modeling data from OGC services will increase the interoperability of workflow engines, remove verbosity from business processes, and enable re-usability of data end points within workflows. BPMN has four ways of describing data with a workflow:

- Data objects
- Data inputs
- Data outputs
- Data stores

7.4.1. Data objects

Data objects describe data requirements for activities to be performed. The data object element can represent a singular object or a collection of objects. The life-cycle of the data object is implicitly tied to the containing process or sub-process. In fact, a data object in its raw form cannot exist without a containing process or sub-process. Data object references are related to data objects in that they enable reference to the same data object across a workflow. The advantage with data object references is that they are state-aware across the diagram where as data objects do not have a state and following the life cycle of the containing process. Data objects also have a class like hierarchy of access, much like variable scope. A process containing sub-processes may all have data objects and references. However, their accessibility is dependent on their location in the process/sub-process hierarchy. Figure 5 and 6 show the BPMN representation for a data object and data object collection.

Figure 5. High-Level BPMN Data Object

Figure 6. High-Level BPMN Data Object Collection

7.4.2. Data stores

A data store is used when data will persist beyond the life cycle of the containing process. Data stores also have the concept of a *reference*. However, the reference to the data store is simply a

convenient way to access the data store without replicating the icon as the reference, for all intents and purposes is the same as the store. It is not appropriate to represent OGC services as data stores, as a data store refers to a traditional database. Data from OGC services is requested via a query to a standardized interface, which may refer to a set of databases or indeed, other services behind the scenes. It is more likely that access to data should be treated like any other service task in BPMN as it is essentially performing the same operation.

Chapter 8. OGC Service Orchestration with BPMN 2.0 Best Practices

This section outlines the draft *best practices* for orchestrating OGC services using BPMN. The main focus of these best practices is:

- Representation of *processes* and *tasking* within BPMN.
- How to configure processes.
- Representation of workflow outputs.
- Security.

8.1. Tasking and Activities

BPMN contains several methods of representing processes and tasking in the form of *Activity Tasks*. There are several options for tasking that enable the workflow to perform functions such as:

- Sending and receiving information.
- Tasking personnel.
- Running scripts.
- Implementing business rules.
- Executing services.

OGC processing tasks through services such as WPS and WCPS should therefore be represented as a **Service Task**, as they are designed to represent executable processes.

8.2. Managing data

Understanding how to manage data objects across the OGC services suite and as part of a workflow is a key requirement for this piece of work. As mentioned previously, there are two main *types* of data objects that need to be represented:

- Data that persists beyond the life-cycle of a workflow.
- Data that exists in the life-cycle of a workflow only.

These different data objects map well to the BPMN concept of a *data store* and a *data object*. However, there is a conceptual difference between the data object in BPMN and passing parameters. The data object should only exist according to the life-cycle of a process. Therefore, a process that includes an input from a previous process's output should be modeled separately as *data outputs* and then *data inputs*.

- Data that is to be utilized beyond the life-cycle of the workflow via exposure as *global processes*.
- Internal mapping of data should be represented by a data object where supported and mapped where required.

- Where data formats are compatible within a workflow, data should be passed by reference.
- Where data formats are incompatible within a workflow, data should be passed by value and encoded in an appropriate format.

8.2.1. Use of a data store

A data store is used for a persistent database, which is rarely the case in OGC services. Instead data are posted and accessed through services as the data architecture behind the services is opaque. Therefore, a data store should not be used to invoke an OGC service, as they are completely different concepts but maybe used to represent a local or remote traditional database storage.

8.2.2. Use of a data object

The data object is used to graphically represent the data inputs and outputs from a single Task. The BPMN specification describes usage of the data object be restricted to a single Task and as such, the data object is destroyed with the completion of the Task. Data objects should be used in the following circumstances:

- Describing relevant inputs and outputs of singular Tasks.

Additionally, data objects should be used to represent certain data types, notably, complex types that include:

- Extensible Markup Language (XML)/ Geography Markup Language (GML).
- JavaScript Object Notation (JSON), GeoJSON, JSON for Linked Data (JSON-LD).
- Shapefiles.
- Any other complex type.

This is done to enable visualization of the *data flow* within a workflow and between Service Tasks.

Data object states

By convention, all data objects have *states* that are referenced after the name of the data object and communicated via square braces ([]). Data objects in OGC do not require a specific state to be accessed, as the data are all provided through interaction with service tasks. For example, there has been no use case presented so far for workflows that simply requests data, as the objective of the BPMN workflow is to provide processing orchestration with data request orchestration a bi-product.

Data object collections

Collections of data objects are simple representations of one to many data objects within a single state. Collections of data objects are not represented in a service or resource-based architecture as the data objects are generally provided from a single WFS or WCS interface. What happens behind the scenes in these queries is largely up to the implementer, therefore requesting data objects is largely the work of a process. A related concept is passing multiple instances of an input to a WPS, in this instance, data object collections may be used.

8.2.3. Service Task Parameters

In addition to input data, processes and service tasks also often contain parameters for configuring each service. Unlike data objects, these parameters should be configured directly in the Service Task and not represented by a data object. Simple types typically consist of the following:

- String
- Integer
- Double
- Float
- Boolean

The nature of OGC services is to utilize *complex data inputs* and therefore a *helper class* has to be created for each implementation. This section does not describe in detail the implementation practices for creating helper classes as it is envisaged that BPMN does not support external data types natively, but instead uses a common internal method of dealing with data objects. Currently, the data parsers are defined by the processing services, as it is likely that this is the entry point for getting data. As mentioned previously, the BPMN engine has no processing capability of its own but relies on the services that it is orchestrating for processing and data parsing power. Therefore, all outputs from a WPS are represented as an *Object* type. This allows the output of a WPS, be it as a FeatureCollection or reference, to be successfully handled by the workflow engine. It is possible to define the input and output types as their *actual* types if required. The disadvantage of using Object types is that no validation is done by the workflow engine on the content of an input or output parameter.

Currently, all image types should be referred to via a reference since imagery is not supported natively within BPMN or via the helper class. But as with vector data, services that can support imagery should be used to process and parse that imagery.

Service Tasks for accessing processing services

Accessing a processing service via a service task is done by passing the variables required to execute the process from the BPMN document to the execute document, irrespective of the specific WPS. A WPS process generally requires the following to execute:

- WPS Uniform Resource Locator (URL) - the location of the WPS.
- Process description - the specific WPS process to execute.
- Authentication information (optional).
- Variables - the required variables to execute the process.

The variables required to execute the process are defined in the BPMN document.

Chapter 9. Description of demonstrator implementation using jBPM

This section outlines the component design and implementation for the BPMN workflow engine component. The design is based off of the overall architecture and the statement of work provided by the sponsors. In general terms, the implementation is designed to reflect the *best practices* described in this ER with respect to the scenario or set of use cases. The best practices however are intended to be independent of the BPMN engine used.

9.1. Workflow engine Helper classes

The implemented workflow helper class is deployed in jBPM, a Java-based BPMN workflow engine. The service utilizes the jBPM Workbench stack that provides a REST interface for remote execution. This interface is sent a BPMN document by the WPS-T that was in turn orchestrated by the client. The BPMN engine executes the document and then returns the result (an output or an error to the WPS-T that passes it onto the client).

jBPM works with Custom Work Items to orchestrate services and these work items have definitions that describe the inputs, outputs and data types. Once these are described, they are selectable within the web application interface. Enabling the re-usability of work items in jBPM is not a requirement for this piece of work. Therefore the helper class provides the BPMN engine with knowledge of spatial data structures rather than performing functions such as setting up the user interface.

The helper class essentially acts as a WPS client that creates the process request for each WPS from the submitted BPMN document. It is then the job of the workflow engine to map the outputs from one process to the input process that requires it. This is done through a simple local variable definition or, if supported, via a data object.

9.2. Security implications

Throughout the Testbed process, there has been discussion as to the model of security that should be used to enable differential access to services according to access credentials of a user. There have been two main scenarios discussed to handle security in the workflow engine, these are as follows:

1. Encoding the security information in the BPMN document.
2. Passing security information including tokens in the HTTP request header.
    - Encoding the security information directly into the BPMN document. This offers the advantage of differential access to services, potentially in separate federations. However, it also creates several points of concern including:
        - The aforementioned security concerns of the sponsor and others.
        - The ability to extract data from a *protected* service and then insert it into an *unprotected* service by way of result passing within the workflow engine.

- The increased complexity of modeling security in this way.
- The usage of different types of security. Putting an access token in the BPMN document maybe suitable for OAuth2, but it is unclear how this transfers to other types of security.
- Exposing security information such as access tokens outside of the HTTP header may be considered a security risk.

The implications of different concepts of security are described in Figure 7. There are three concepts expressed within the diagram, the first is where the BPMN engine sits outside of the security federation and requires credentials represented by the solid line. This scenario offers the advantage of allowing access to the BPMN workflow engine by outside actors, but requires credentials to access the federated, protected services. The approach adopted in Testbed-14 largely conforms to this. A second scenario is described by the dashed line where the workflow engine is inside the federation. This offers the advantage of not requiring credentials to be passed to the workflow engine as the user is already authenticated and therefore has access to the services. A disadvantage with this approach is that the workflow engine is potentially closed to users that are not authenticated and therefore does not offer differential access to services. A third scenario is a workflow that includes services that are both inside and outside the federation represented by the WPS-3 object in the diagram. In this instance, it is possible for the BPMN engine to pass protected data out of the federation to a service that is not authenticated to see that data, therefore posing a potential security risk.

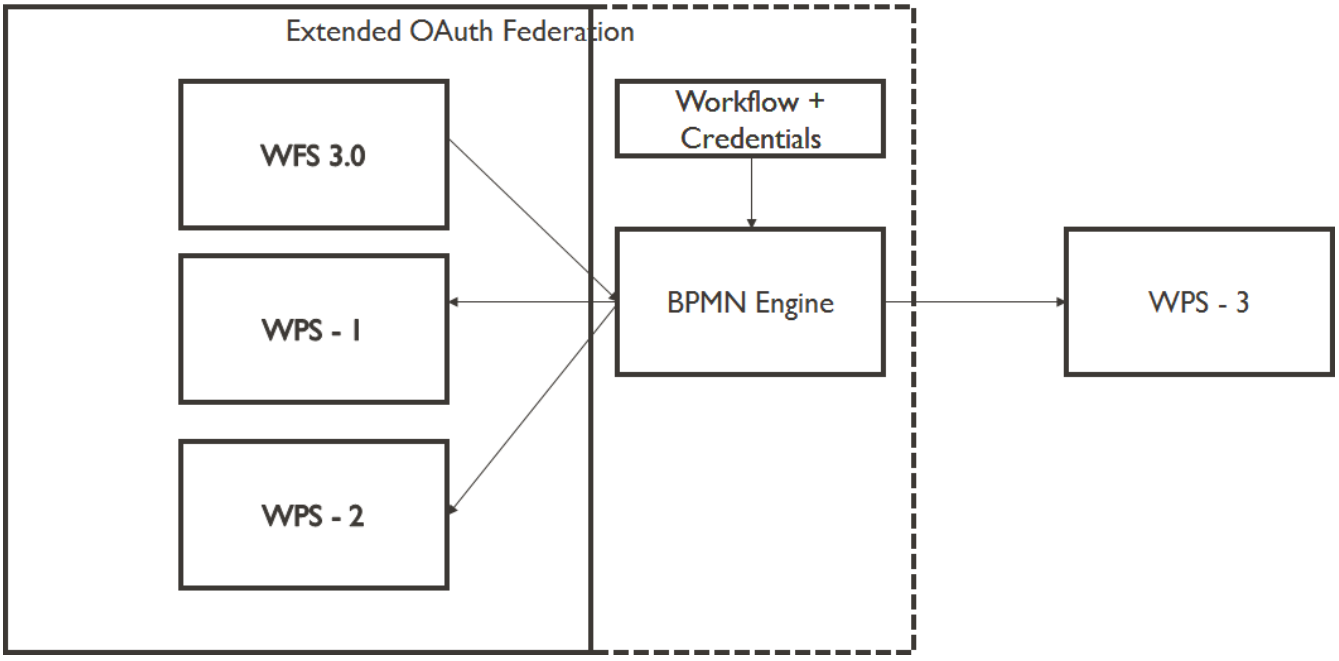

Figure 7. High-Level BPMN Engine security possibilities

Although these issues exist, in Testbed-14 it has been decided that the security token would be passed as part of the BPMN document describing each service. This allows differential access to security models and enables the workflow engine to deal with secured and unsecured services in the same workflow.

9.3. Component design

The BPMN workflow engine component works in a similar manner to the workflow engine component, that is, an open source implementation of a workflow engine coupled with a *helper*

class to enable the workflow engine to understand geographic data constructs. As the workflow engine is used to request either data or processing capability, the workflow engine does not contain any data parsing capabilities (beyond that implemented in the helper class), but instead relies on the external services orchestrated to perform this functionality. Adopting this approach solves the data encoding/decoding issues that arose in the Testbed-13 Workflows implementation and report.

Relying on external services to perform data encoding and decoding keeps the workflow engine agnostic and it therefore does not make any demands on the services being orchestrated. However, the workflow engine still requires an internal construct to be able to manage geographic information internally, this is accomplished through the helper class, which is likely to require some language specific components.

The open source BPMN engine used in this implementation is jBPM (https://www.jbpm.org/) created and maintained as an open source product by Redhat. jBPM has several components including a development suite and a web application that includes a visual front end for orchestrating services. In terms of the OGC implementation, there are three aspects of the jBPM to consider:

1. The workflow engine and helper class.
2. The REST API for remote execution.
3. The User Interface (UI).

The objective of this work item is to produce a helper class for the workflow engine, therefore that will be the focus of the implementation. Orchestrating OGC services using the jBPM user interface is not in scope for this piece of work (although it is functional), the BPMN documents are created by the client and then sent to the WPS-T, which is in turn handed off to the workflow engine to actually to the service orchestration and execution. The workflow engine is fronted by a WPS which provides a standards based interface for the WPS-T to call. The WPS contains a process called *ExecuteWorflow* that accepts a BPMN document as a parameter.

The UI aspects of jBPM are mentioned here as they require customization and configuration within the web application to be fully functional. That is, should the user wish to see their custom work items appear in the UI, then they have to define them as part of the Work Item Definition (WID) file. This can be accomplished remotely, but is deemed out of scope for this Testbed. A full explanation of this can be found in the draft OGC 16-091 BPMN 2.0 for Orchestrating OGC Services Discussion Paper.

9.4. Helper class design

As mentioned previously, the helper class is a simple implementation that is compiled and added to the jBPM library priory to deploying the web application (WEB-INF/lib/<name of jar>.jar), this enables the web application to have access to geospatial constructs. As jBPM is written in Java, the geospatial library used is GeoTools (www.geotools.org), and the specific class used to hold and manipulate geospatial information from within the workflow engine is *FeatureCollection*. There are variants of the FeatureCollection such as *SimpleFeatureCollection*, which deals specifically with simple features (i.e. flat data structures). However, these are not implemented as specific constructs as the FeatureCollection can handle any variants or derivatives of this construct.

jBPM requires two types of class to execute WPS processes and by extension, WFS data. Workflows

will usually have a processing component to them, therefore parsing of data objects should be handled by the processing services with the outputs from those processing service mapped to data objects. This means that a workflow that only contains data inputs and outputs will not make use of the helper class as a data object will likely only be passed as a reference between BPMN Activities. The data are only parsed when it is necessary to do so as it needs to be processed. The two classes that need to be implemented are as follows:

- WPS client - to create execute documents, send execute requests, and parse and transmit the results of the execution. The WPS client is 2.0 compliant.
- Custom work item handler - this class acts as the interface between the WPS and the workflow engine. To keep it generic, the work item handler for a WPS has the following variables:
  - The WPS URL (for example *http://localhost:8080/wps/*)
  - The WPS process description (for example *echo.process*)
  - A hashmap defined as <String>, <Object> - this is a generic class for handling process specific variables. The Object data type allows for casting in the WPS client.
  - An OAuth *Bearer token*, optional overall but required if the process being called is secured.

These variables are then exposed in each of the workflow service tasks in the BPMN document for population. If the workflow engine interface is being used for orchestration, then it is prudent to configure the workflow engine to take default values for the WPS URL and the process description, as these can be gathered from the WPS services exposed in the workflow. In Testbed-14, however, the workflow documents are configured prior to submission and execution and therefore default values are not required.

The helper class is designed to be generic as it is able to deal with any WPS with inputs. Additionally, processes have to be registered with the workflow engine and maybe submitted in a BPMN document without registration. This means that the workflow engine should act as a receptacle of a BPMN workflow and not put any further burden on the user to register processes.

9.4.1. Security in the helper class

This implementation does security by first testing whether the capabilities of the service requires security, it does this by getting the capabilities of a service and looking for <Constraint> tags within the advertised operations.

It is assumed that if the constraint tags are missing or are empty, then there are no security constraints on the service. If there are no security constraints, then the service execution happens as normal without the inclusion of a bearer token in the request. However, if there are found to be constraints, they are currently assumed to be OAuth2. The Bearer token is currently held in the process variables for the BPMN workflow. Originally it was considered to have a single security federation with the OAuth Bearer token inserted into the header of the request to the WPS that fronts the BPMN workflow engine. However, it quickly became apparent that this was an over-simplification of the requirements for even the simplest workflows due to:

- The lack of available services to produce a simple demonstrator workflow.
- The recognition that security is prevalent throughout many services, therefore a generic BPMN workflow engine needs to be flexible enough to utilize multiple tokens.

- The BPMN engine has its own security requirements, one of which is encoding the credentials of the executor in the header of the REST calls to create the container that can execute the workflow. Therefore, passing all credentials in the header becomes problematic.

The helper class then takes the security token as an input and inserts it into the HTTP header of the execute request for the relevant WPS service. If the token is valid then the WPS executes as normal.

9.5. Remote execution of BPMN documents

jBPM acts as a Git repository, this is implementation specific, and simply a way of pushing a remotely constructed BPMN document to a running instance of jBPM. Therefore, the workflow engine has a WPS facade in front of it to:

- Receive the BPMN document.
- Create the Git repository on the jBPM instance.
- Execute the workflow.

These functions are performed using a combination of Git calls made in jGit (https://git-scm.com/) and then calls to the jBPM REST API within the WPS process.

It is noted that other BPMN workflow engines perform this task in a different way, i.e. by directly allowing access to the remote execution aspect of the workflow as seen in the Camunda implementation in Testbed-13.

The WPS contains a single process to execute the workflow with a parameter to receive the BPMN document. This approach replicates some of the work done in the WPS-T aspect of the architecture for simplicity and to assist other implementers. The process description for the *ExecuteWorkflow* WPS process is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:Execute
    xmlns:wps="http://www.opengis.net/wps/2.0"
    xmlns:ows="http://www.opengis.net/ows/2.0"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/wps/2.0
http://schemas.opengis.net/wps/2.0/wps.xsd"
  service="WPS" version="2.0.0" response="document" mode="sync">
    <ows:Identifier>testbed14.workflows.ExecuteWorkflow</ows:Identifier>
    <wps:Input id="BPMNDocument">
        <wps:Reference mimeType="text/xml"
xlink:href="http://meekbaa1.miniserver.com/dl/testbed-14-gait-process.bpmn"/>
    </wps:Input>
    <wps:Output id="WorkflowResults" transmission="value"/>
</wps:Execute>
```

9.5.1. Receiving the BPMN document and creating the Git repository

The WPS is there as a simple way for a client to execute a workflow by executing a WPS process. As mentioned, jBPM uses Git as a method of remotely updating projects within the web application. Therefore, the WPS process simply commits the BPMN document to the relevant project within the jBPM instance. In the testbed, the jBPM project is already setup and the change committed is the workflow document received by the WPS. jBPM projects contain supporting files beyond the workflow document such as project configuration that includes registration of service task work items and deployment variables. Although these can be configured remotely via Git, they are not in the Testbed due to the added complexity.

This highlights an issue where jBPM requires more than a BPMN document to perform executions as there are several supporting files used in the registration process. Therefore in the testbed, projects are updated with changes to the BPMN document and these usually include updates to variable values for the WPS processes executed by the workflow engine.

9.5.2. Executing the workflow

jBPM offers a REST API as a method of executing workflows remotely, however this is a multi-step process. The steps are as follows:

1. Check for container deployment of existing workflow project and if necessary undeploy the container.
    - Deploy the new container with the updated BPMN document.
    - Create a new instance of the BPMN process using the deployed container - note that a container has the ability to execute several workflows.
    - Execute the instance of the workflow and get the workflow execution status - this does not correspond to the results of the workflow. it is a set of messages that respond to whether the workflow has executed correctly or not.

Executing jBPM remotely requires basic authentication, which is passed in the header information. Additionally, content-type must also be set appropriately (all examples here are XML, but the system has been tested using JSON).

All of this is managed through the REST API with the following example requests.

Remove existing container

Deleting an existing deployment is required to remove any existing container deployment of a project. Note that containers need to be rebuilt when the underlying project is changed via Git and example REST request using HTTP DELETE is as follows:

- http://localhost:8080/kie-server/services/rest/server/containers/Testbed_14_secured_1.0.0/

This process is called regardless of whether there is a running container or not. In an operational system, it is likely that existing containers would have to be checked for running processes prior to removal.

Deploy new container

The new container is deployed with the updated project using HTTP PUT operation:

http://localhost:8080/kie-server/services/rest/server/containers/Testbed_14_secured_1.0.0/

```
<kie-container container-id="Testbed_14_secured_1.0.0">
  <release-id>
    <artifact-id>Testbed_14_secured</artifact-id>
    <group-id>com.myspace</group-id>
    <version>1.0.0</version>
  </release-id>
</kie-container>
```

This process simply rebuilds the project using the updated BPMN documents that have been committed via Git.

Execute the process

Process execution via REST has the ability to take variables as arguments for execution. However, all variables are described in the submitted BPMN document and execution without further arguments is therefore sufficient. This endpoint is executed using an HTTP POST operation with a blank body.

http://localhost:8080/kie-server/services/rest/server/containers/Testbed_14_secured_1.0.0/processes/Testbed_14_secured_1.0.0.testbed-gait-process/instances

When called, this endpoint returns a *process ID* to the client for reference.

Retrieve the results

After the process has completed (in this case it is run synchronously), the results are made available via the following HTTP GET call:

http://localhost:8080/kie-server/services/rest/server/queries/processes/instances/<processID>/variables/instances

The results are then returned to the calling WPS and inserted as WPS output in the WorkflowResults variable.

9.6. Architecture

The architecture has changed a little from the Statement of Requirement (SOR) as the main addition is the WPS that sits in front of the workflow engine. It is felt that including a WPS on the workflow engine side is simpler than attempting to write a WPS process for the WPS-T that is under the control of another vendor. This way, a standard interface is provided for anyone to execute the BPMN workflow engine beyond the WPS-T, therefore adhering to a *loosely coupled* architecture. The crude revised architecture can be found in Figure 8.

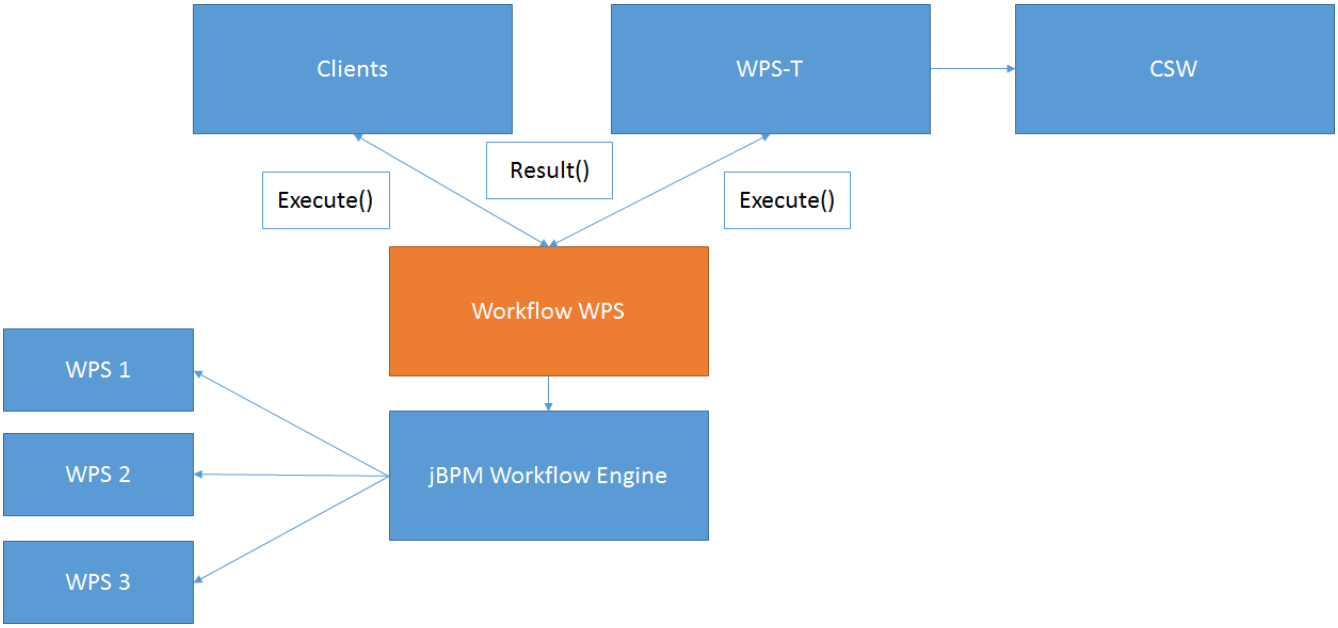

Figure 8. The revised architecture with the addition of a WPS in front of the workflow engine for simple execution

The behavior of the system requires implementation of the use cases found in Figure 9. The Actors on the system are the WPS-T and the *generic client*. The generic client is included to show that the revised interfaces are standards based and therefore any client can interact with the WPS and therefore the workflow engine. The use cases involve receiving and parsing a BPMN document to orchestrate the processes. This is not a simple command as it involves treating the workflow web application as a Git repository, this enables dependencies to be incorporated into a workflow command (through Maven), but also requires aspects such as authentication.

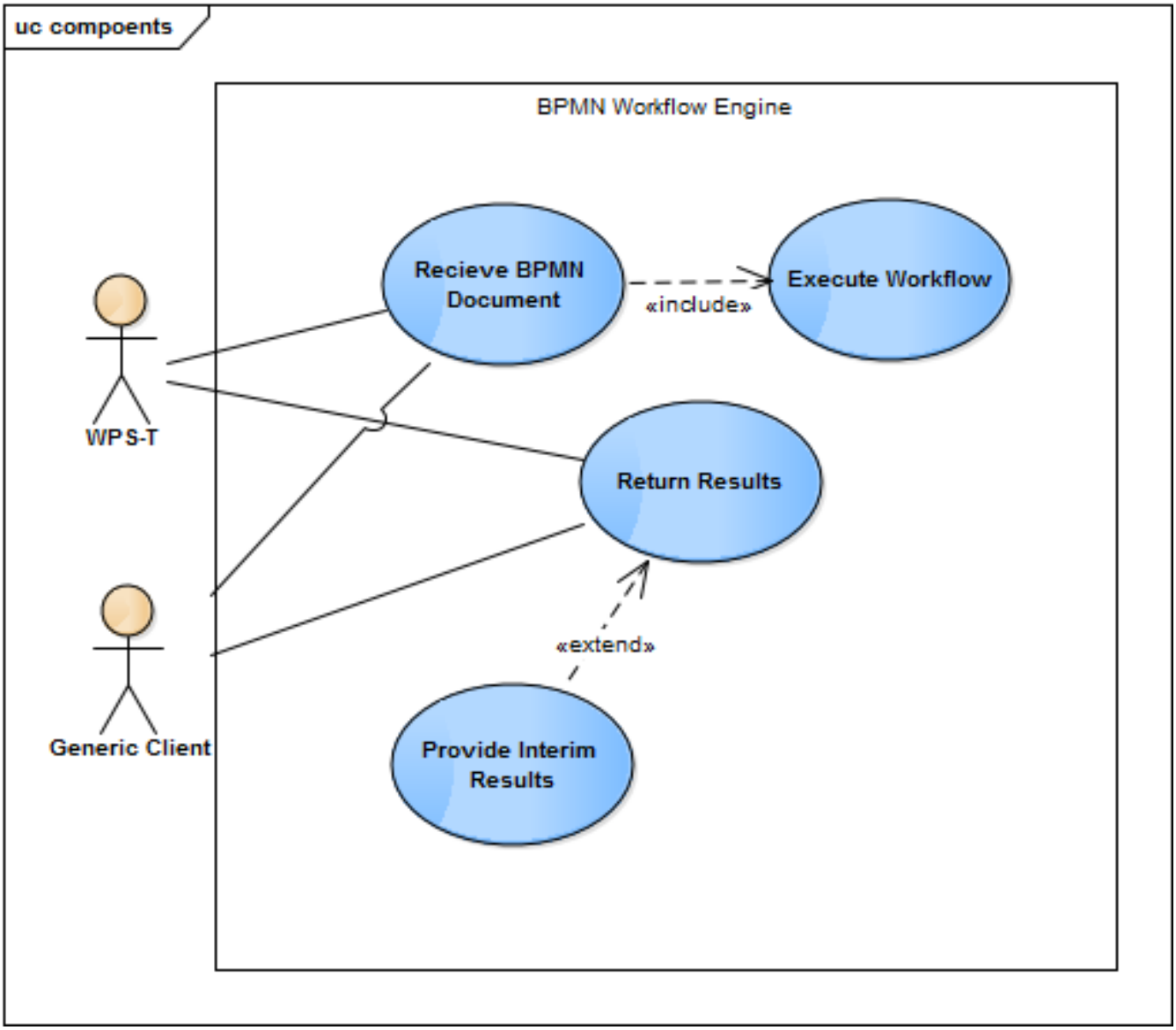

Figure 9. A UML use case diagram for the BPMN Workflow Engine

The sequence of executions is much like that seen in Testbed-13. The sequence diagram for the implementation is found in Figure 10. The system receives the BPMN diagram and executes the workflow. Access to services is checked as the access token is simply passed in the header of a POST request. The BPMN system is not doing any token authentication, it simply passes the token into the execute document for the WPS services to initiate a session. If the user is not authenticated, then the BPMN service receives an error that is passed onto the client. After the workflow has executed, there are two possible outcomes, either the workflow executes all the way to the end, or it does not. If it does, then a result is generated and passed to the client. If it does not then an error message is passed to the client.

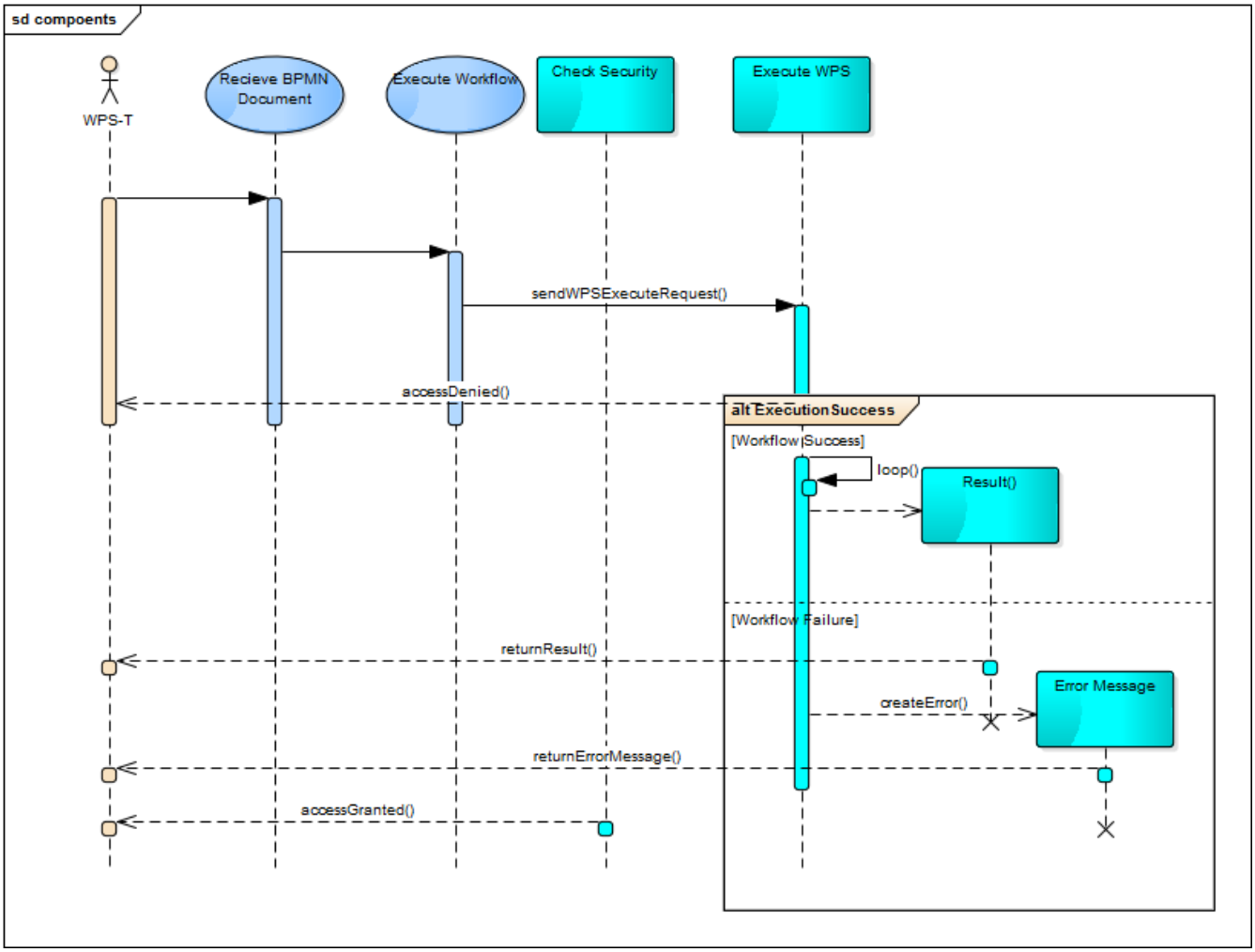

Figure 10. A UML sequence diagram for the BPMN Workflow Engine Execute Workflow use case

jBPM is an implementation of a BPMN workflow engine and has its own methods of committing workflows to be executed, deploying containers, executing tasks and retrieving results. Figure 11 describes the jBPM specific sequence for receiving a BPMN document, committing the document to the workflow engine and performing the execution. jBPM uses the Git protocol to create and update projects. The specific REST calls used are described in the previous section. The *local* machine contains a Git project that is changed according to a newly submitted BPMN document through the WPS, the changes are then committed to the local Git repository and then the changes are pushed to the remote Git repository, i.e. the jBPM web application. This requires authentication that is hard coded into the WPS process as a temporary solution. After the project is committed, old versions of the deployment have to be undeployed, this is done through a simple REST call. Then, a new version is deployed and built, and the process is executed by a further REST call and assigned a task number. This enables multiple instances of a process to be run sequentially or simultaneously. It is possible to inject variables into the REST call, however, this is not required in this use case because all of the variables updates are included in the submitted BPMN document. After the process is completed, the results are made available by a further REST call with the previously assigned task number used as the lookup.

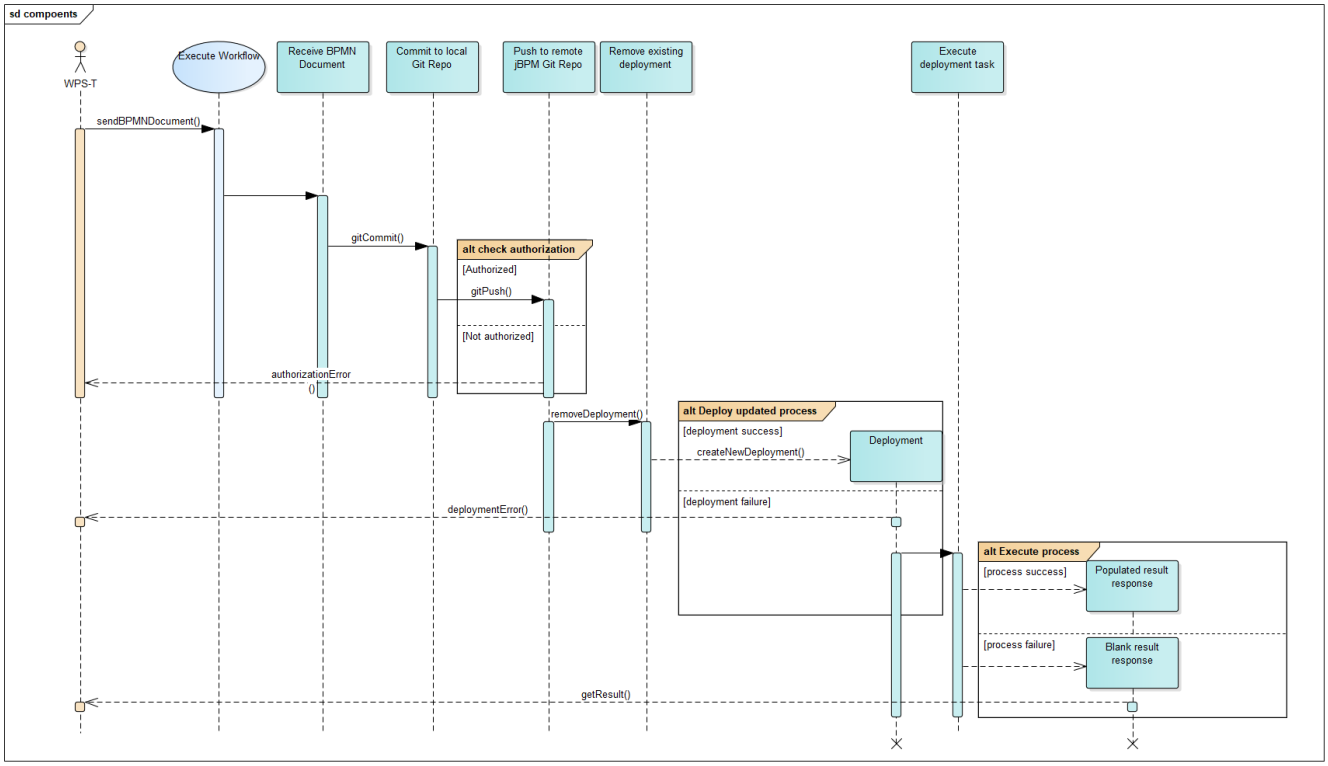

Figure 11. A UML sequence diagram to show jBPM deployment and execution procedure.

9.7. Scenario

Unlike previous Testbed initiatives on workflows, this piece of work has no grounding scenario and no services to execute as part of the CFP. Therefore, a use case for workflow execution has been created in collaboration with the sponsors and thread partners. The BPMN engine will be responsible for executing a data quality workflow using the National Geospatial Intelligence Agency (NGA) Gait tools [2: https://github.com/ngageoint/Geospatial-Analysis-Integrity-Tool] fronted by a WPS facade.

The workflow has been setup to include two processes:

1. NGA Gait Tools process:

```
<wps:ProcessOfferings
    xmlns:wps="http://www.opengis.net/wps/2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ows="http://www.opengis.net/ows/2.0"
xsi:schemaLocation="http://www.opengis.net/wps/2.0
http://schemas.opengis.net/wps/2.0/wps.xsd">
    <wps:ProcessOffering processVersion="1.0.0" jobControlOptions="sync-execute async-
execute" outputTransmission="value reference">
        <wps:Process>
            <ows:Title>
A simple demonstrator process to execute a workflow
</ows:Title>
            <ows:Abstract>
Takes a BPMN document, executes the workflow and produces the result
</ows:Abstract>
            <ows:Identifier>testbed14.workflows.ExecuteWorkflow</ows:Identifier>
            <wps:Input minOccurs="1" maxOccurs="1">
                <ows:Title>
The BPMN Document to execute (this process only allows for workflows with known
handlers)
</ows:Title>
                <ows:Identifier>BPMNDocument</ows:Identifier>
                <wps:ComplexData
                    xmlns:ns="http://www.opengis.net/wps/2.0">
                    <ns:Format default="true" mimeType="text/xml"/>
                </wps:ComplexData>
            </wps:Input>
            <wps:Output>
                <ows:Title>The results from the workflow engine</ows:Title>
                <ows:Identifier>WorkflowResults</ows:Identifier>
                <wps:ComplexData
                    xmlns:ns="http://www.opengis.net/wps/2.0">
                    <ns:Format default="true" mimeType="text/xml"/>
                </wps:ComplexData>
            </wps:Output>
        </wps:Process>
    </wps:ProcessOffering>
</wps:ProcessOfferings>
```

1. A sample process that returns the data given to it, although it performs no processing on the data, it demonstrates:
    a. That processes can be chained.
    b. That a workflow engine can work with both secured and unsecured processes.
    c. The workflow engine working with data generated with different WPS.

```
<wps:ProcessOfferings
    xmlns:wps="http://www.opengis.net/wps/2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ows="http://www.opengis.net/ows/2.0"
xsi:schemaLocation="http://www.opengis.net/wps/2.0
http://schemas.opengis.net/wps/2.0/wps.xsd">
    <wps:ProcessOffering processVersion="1.0.0" jobControlOptions="sync-execute async-
execute" outputTransmission="value reference">
        <wps:Process>
            <ows:Title>storage.geoserver.GetWFSData</ows:Title>
            <ows:Identifier>storage.geoserver.GetWFSData</ows:Identifier>
            <wps:Input minOccurs="1" maxOccurs="1">
                <ows:Title>Input WFS Data</ows:Title>
                <ows:Identifier>inputWFSUrl</ows:Identifier>
                <wps:ComplexData
                    xmlns:ns="http://www.opengis.net/wps/2.0">
                    <ns:Format default="true" mimeType="application/x-zipped-shp"/>
                    <ns:Format mimeType="text/xml; subtype=gml/3.1.1"
schema="http://schemas.opengis.net/gml/3.1.1/base/feature.xsd"/>
                    <ns:Format mimeType="text/xml; subtype=gml/3.1.0"
schema="http://schemas.opengis.net/gml/3.1.0/base/feature.xsd"/>
                    <ns:Format mimeType="application/json"/>
                </wps:ComplexData>
            </wps:Input>
            <wps:Output>
                <ows:Title>Vector Data</ows:Title>
                <ows:Identifier>vectorData</ows:Identifier>
                <wps:ComplexData
                    xmlns:ns="http://www.opengis.net/wps/2.0">
                    <ns:Format default="true" mimeType="application/x-zipped-shp"/>
                    <ns:Format mimeType="text/xml; subtype=gml/3.1.1"
schema="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
                    <ns:Format mimeType="text/xml; subtype=gml/3.1.0"
schema="http://schemas.opengis.net/gml/3.1.0/base/feature.xsd"/>
                    <ns:Format mimeType="application/json"/>
                </wps:ComplexData>
            </wps:Output>
        </wps:Process>
    </wps:ProcessOffering>
</wps:ProcessOfferings>
```

The BPMN document used in Testbed-14 chains the two processes above and is what was used in the Technology Integration Experiment (TIE) testing aspects of the workflow. The BPMN document was to demonstrate the security token parameter being changed to show transactional BPMN document execution, it can be found in the annex.

jBPM has two different environments for orchestrating and executing processes, the development environment based upon the Eclipse Integrated Development Environment (IDE), and the web application which is a set of standalone modules running on JBOSS Wildfly 11. There are several

differences between the two and they are not particularly interoperable and both have been included in this document to show the mapping of data objects. The web application does not support data object icons, but the Eclipse IDE environment does. From a BPMN perspective, both methods of mapping variables produce valid BPMN (both conceptually and via a validator), but the formal mapping of objects is preferred.

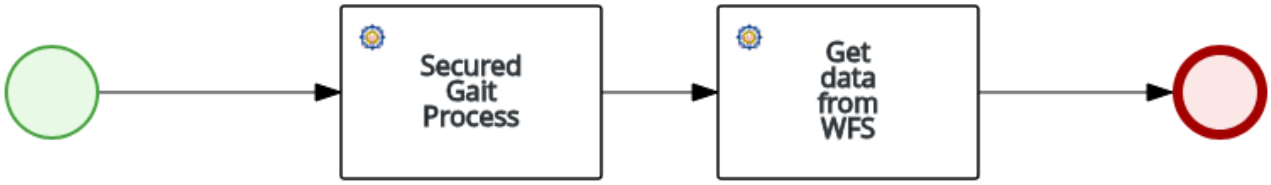

Figure 12. The Testbed-14 workflow shown in the JBPM Console.

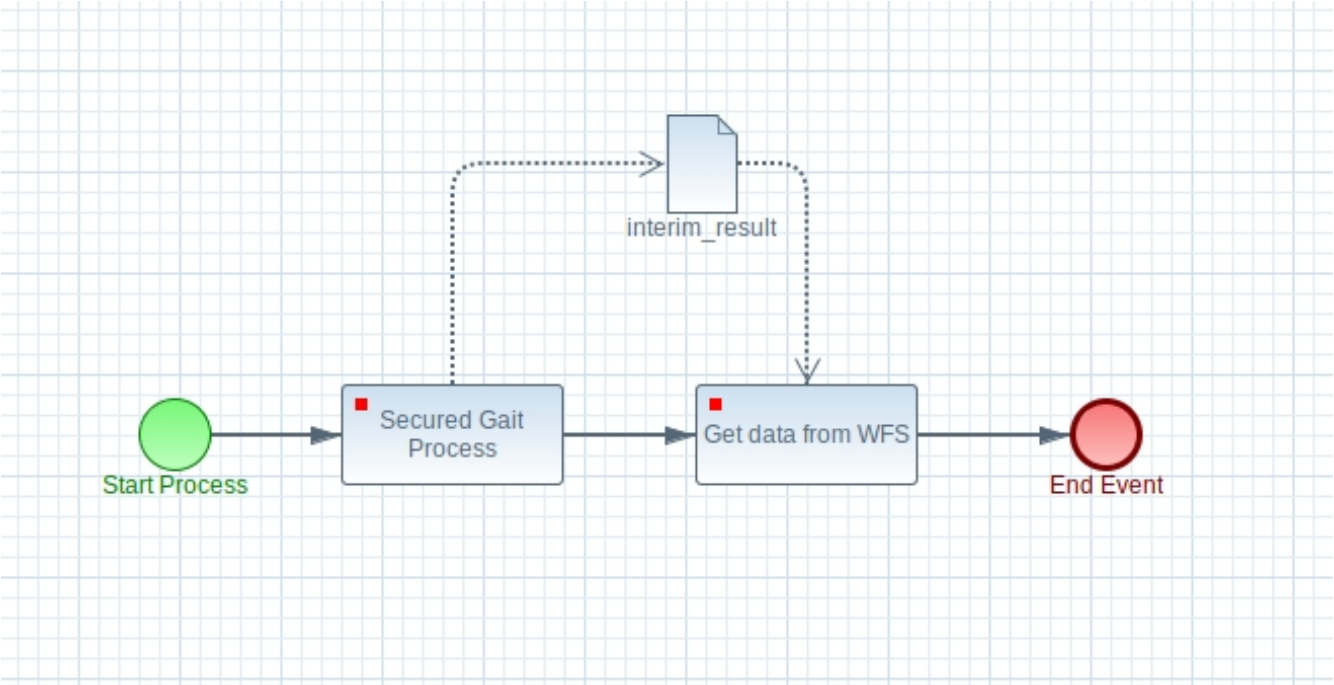

Figure 13. The Testbed-14 workflow shown in the Eclipse development environment.

9.8. TIE Results

The workflow engine WPS produces the following after a successful execution.

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:Result
    xmlns:wps="http://www.opengis.net/wps/2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wps/2.0
http://schemas.opengis.net/wps/2.0/wps.xsd">
    <wps:JobID>cfcaccc5-2789-4eb3-b518-03d4c9c14d66</wps:JobID>
    <wps:Output id="WorkflowResults">
        <wps:Data mimeType="text/xml">
            <variable-instance-list>
                <variable-instance>
```

```
                    <name>initiator</name>
                    <old-value/>
                    <value>admin</value>
                    <process-instance-id>4</process-instance-id>
                    <modification-date>2018-09-06T12:33:47.967Z</modification-date>
                </variable-instance>
                <variable-instance>
                    <name>QACR</name>
                    <old-value/>
                    <value>https://testbed.dev.52north.org/tb14-d146-
secured/service/wps?request=GetOutput&amp;version=2.0.0&amp;service=WPS&amp;id=d0634e5
4-4626-4481-b6db-aa0cc05694fcQACR_file.d1cd3458-67a3-45ab-9a35-14beb33bdaf2</value>
                    <process-instance-id>4</process-instance-id>
                    <modification-date>2018-09-06T12:33:55.679Z</modification-date>
                </variable-instance>
                <variable-instance>
                    <name>AttError</name>
                    <old-value/>
                    <value>https://testbed.dev.52north.org/tb14-d146-
secured/service/wps?request=GetOutput&amp;version=2.0.0&amp;service=WPS&amp;id=d0634e5
4-4626-4481-b6db-aa0cc05694fcattribution-errors.afd65f49-d63a-4cf5-aec9-
ee2ceb8e71a3</value>
                    <process-instance-id>4</process-instance-id>
                    <modification-date>2018-09-06T12:33:55.680Z</modification-date>
                </variable-instance>
                <variable-instance>
                    <name>ConReport</name>
                    <old-value/>
                    <value>https://testbed.dev.52north.org/tb14-d146-
secured/service/wps?request=GetOutput&amp;version=2.0.0&amp;service=WPS&amp;id=d0634e5
4-4626-4481-b6db-aa0cc05694fccondition-reports.d32eb836-978a-4bc6-bce8-
ef65b7375d1c</value>
                    <process-instance-id>4</process-instance-id>
                    <modification-date>2018-09-06T12:33:55.681Z</modification-date>
                </variable-instance>
                <variable-instance>
                    <name>ConsolLN</name>
                    <old-value/>
                    <value>https://testbed.dev.52north.org/tb14-d146-
secured/service/wps?request=GetOutput&amp;version=2.0.0&amp;service=WPS&amp;id=d0634e5
4-4626-4481-b6db-aa0cc05694fcconsolidated1LN.918dccf2-1849-41f1-abe7-
96cbe71f2a1e</value>
                    <process-instance-id>4</process-instance-id>
                    <modification-date>2018-09-06T12:33:55.681Z</modification-date>
                </variable-instance>
                <variable-instance>
                    <name>ConsolPT</name>
                    <old-value/>
                    <value>https://testbed.dev.52north.org/tb14-d146-
secured/service/wps?request=GetOutput&amp;version=2.0.0&amp;service=WPS&amp;id=d0634e5
4-4626-4481-b6db-aa0cc05694fcconsolidated1PT.5195483c-5f78-4ff7-a1ec-
```

```
83a0e23cfe43</value>
                    <process-instance-id>4</process-instance-id>
                    <modification-date>2018-09-06T12:33:55.681Z</modification-date>
                </variable-instance>
                <variable-instance>
                    <name>Result</name>
                    <old-value/>

<value>http://54.218.149.72:8000/wps/RetrieveResultServlet?id=16ef4f9d-8f8d-4e31-aedf-
efaace7afe7dvectorData.a08a2f7e-b5eb-48cd-90c3-7b5b584c35ad</value>
                    <process-instance-id>4</process-instance-id>
                    <modification-date>2018-09-06T12:33:57.138Z</modification-date>
                </variable-instance>
            </variable-instance-list>
        </wps:Data>
    </wps:Output>
</wps:Result>
```

Verification of the transactional approach is demonstrated through submission of the Bearer Token taken from the Security Mediation Service developed elsewhere in the thread. The outputs of the workflow are simply wrapped in a catch-all WPS output. Each of the individual results are WPS references that can be parsed by any compliant OGC service.

9.9. Shortcomings of the approach

Although successful in demonstrating the approach to secured workflows, there are a few shortcomings that will be described in this section.

- Managing security and tokens. Currently, the tokens are encoded in the BPMN document. This enables differential, fine grained access to different services. However, there are several problems:
  - Security tokens are encoded as plain text in the BPMN document. This represents a security risk as they can be easily read and reused if they are intercepted.
  - Workflows can include secured and unsecured services. This means that results from a secured service can be pushed to an unsecured service, this also represents a security risk.
- Allowing access to the workflow engine.
  - The workflow engine in this Testbed is locked down to a specific set of processes, that is, the process described can have its parameters changed, but submitting completely new processes is currently not possible. This is because further implementation is required to enable this, and it would be a security risk to have an open transactional offering.
  - The workflow engine offers remote access via a REST API and requires basic authentication in the header to execute anything. Currently, this access is hard coded into the WPS process that fronts the workflow engine. In future iterations of the system, the user should be authenticated with the workflow engine either prior to execution or via a secure method.
- Returning results to the client.
  - The workflow engine returns variables that are exposed according to orchestration and

configuration. These can be called by external clients (currently, the workflow WPS) and returned. However, the results are a *dump* of all available external variables from the workflow engine and are not configurable in the current setup.

- For convenience, the workflow engine is fronted by a WPS (separate from the WPS-T work item in the thread), which contains a single process (ExecuteWorkflow) that takes a single parameter as an input (the BPMN document) and returns a single result (the workflow output variable dump). This approach enables simple access to the workflow engine, but requires any calling client to parse the dump to extract the individual workflow results. This is a trivial task, but has the following conceptual issues:
  - Different workflow engines may return results in a different way.
  - Parsing out results on the server side will require a new WPS process to be created for each new BPMN document submitted. This is potentially a job for the WPS-T.

Chapter 10. Docker, Kubernetes and Cloud Foundry

This section provides the results of a short study looking at Cloud Foundry and including the specific technologies of Kubernetes and Docker. It complements the Federated Clouds ER [6] in this Testbed.

10.1. Introduction

Remit from the SOR

" Kubernetes, Docker and Cloud Foundry A discussion has recently arisen around geospatial services (here "services" more in the sense of geospatial functions and processes, but not necessarily as in "Web services") and their availability in modern distributed architectures. Cloud platforms such as Cloud Foundry or OpenShift and container platforms such as Docker or Kubernetes lose their distinct borders as they integrate and interweave, partly by adding services such as e.g. Docker Swarm and Compose, partly by extending their functionality beyond their original purpose. The Cloud Foundry Diego project integrates Docker into the Cloud Foundry PaaS. Docker again can be integrated into various infrastructure tools such as Kubernetes, but plans native support for Kubernetes itself. Remains the question what the geospatial community needs to do in order to support those integrated solutions in the best way. The Workflow Engineering Report (D026) shall contain a study on these aspects and answer questions such as:

- What functionality needs to be provided as a software service?
- What is a platform function that shall be provided?
- Where are the borders between these two?

The study shall serve as a starting point for future architecture discussions. It does not need to be comprehensive in the sense that all tools or architectural aspects need to be analyzed."

10.2. Motivating technologies

The particular technologies of interest as mentioned previously are Docker and Kubernetes, this section provides a short overview of what these technologies are and their relevance to the geospatial domain in terms of OGC services. The section is organized according to a hierarchy with finely grained microservices technology described in the first section to enterprise level deployments on clouds in the final section with Cloud Foundry. This list can also be considered in terms of the complexity of each of the tools as containers are generally self-contained with minimal external dependencies whereas Cloud Foundry works with all of the technologies beneath it in the hierarchy.

10.2.1. Microservices

Microservice architecture is a way of working that encourages development and deployment of small pieces of functionality in a *loosely coupled* and *finely grained* architecture. Loose coupling enables interoperability between components where small pieces of functionality can be developed

and replaced without compromising the functionality of the entire system. Loose coupling of microservices also relies on lightweight protocols to perform communication between microservices and clients. Developing microservices lends itself to small teams working in parallel on different pieces of system functionality. This provides an efficient use of resources and enables that no one module of a system becomes over-complicated and ultimately un-maintainable.

Another aspect of microservice development and delivery is the use of Agile methodologies and continuous integration/delivery practices. Although not a focus of this study, continuous delivery is a software engineering process (DevOps) that has sought to utilize microservices architecture by developing capability though short cycles (*sprints* in Agile Scrum terminology) which means that releases are performed regularly with minimum viable products.

10.2.2. Docker

Containerization and micro-services have become fashionable in modern architectures, or at least are reported to have. Indeed, many large organizations still maintain legacy systems that do not implement any sort of micro-services architecture at all, therefore, these technologies are still pipeline considerations for many. Docker is a popular method of deploying capability using *container* technology. Containers may be thought of as very stripped-down virtual machines (VM) that generally run inside virtual machines in the loose sense. Containers package everything required for the contained service to run within the container image, enables agility, simple deployment of services and removes reliance on system wide libraries to be made available to enable software to run. Some other advantages of containers are as follows:

- Firewalling of individual applications/services for enhanced security.
- Efficient use of system resources as multiple containers can be spun up onto a single VM environment (Kernal sharing).
- Containers can be rapidly deployed and un-deployed at scale as system load requires.
- Containers can be orchestrated using technologies such as Kubernetes and Docker Swarm.

It should be noted that there is other container technology available including Singularity (https://singularity.lbl.gov/) and Turbo (https://turbo.net). However, Docker appears to be the technology of choice for many.

10.2.3. Kubernetes

Containerization and microservices architectures are designed to make efficient use of computational resources whilst providing a stripped-down version of only the required functionality. As mentioned previously, this is in contrast to previous methods of setting up entire virtual machines for the purposes of deploying one or two applications whilst requiring only a small subset of an application's functionality. A drawback of this approach to deploying resources is that microservice architectures can quickly become complex and unmanageable without a further software solution to orchestrate and maintain containers. Kubernetes is one such open source product that provides orchestration and management for containerized systems.

Kubernetes reports to act as a:

- Container platform.

- Microservices platform.
- Portable cloud platform.

Although Kubernetes does container orchestration, it is marketed as a service that removes the need for container orchestration via removal of centralized control and utilization of pathways to move an entire system from its current state to the desired state.

10.2.4. Cloud Foundry

Cloud Foundry (CF in this section) is a piece of software for managing infrastructure, services and applications. It is designed to take the work out of managing infrastructure to enable the focus to be shifted to application and service development. Essentially, CF sits at the top of the Docker→Kubernetes→CF stack and is focused on the macro-management of systems, rather than the detail of specific deployments. The software is designed to manage a system that is infrastructure and approach agnostic, working with a combination of localized VMs as well as IaaS providers such as AWS and Azure to remove the specifics of deploying applications and services on those specific platforms. This allows developers to focus on development rather than deployment.

CF is designed to manage infrastructure in a way that allows for redundancy, outages and scaling. It does this by installing applications on multiple running VMs that can be upscaled and downscaled as required. Load balancing is accomplished at via three different routes:

1. BOSH - a tool designed to configure and run VMs directly onto physical infrastructure.
2. Cloud Controller - the software that runs applications and other processes on the VMs. This is where traffic load balanced according to demand.
3. Router - this actually does the routing of the traffic from the outside world. This is accomplished through a Bulletin Board System (BBS) that services advertise via and the Router periodically queries to maintain an up-to-date list of what services are available where.

Diego

Diego is a new orchestration manager that distributes tasks and application processes and it is a self-healing container management system. One of Diego's advantages is that it attempts to maintain the *correct* numbers of instances running in *cells* to avoid network failures. The correct number of container instances is defined partially by the request for resources, but it is obviously also constrained by the resources that are available. Diego utilizes *Garden*, which is an isolated, containerization layer. One of Diego's responsibilities in continuity, that is, if a resource becomes overloaded or crashes, then Diego should *self-heal*, and make further resources available to complete the Tasks at hand. In addition to these features, Diego also allows for SSH access to running containers and therefore application instances. Within Diego, *Garden* extracts the necessary metadata from a Docker image, mounts the image, builds the container and executes the commands from a Task if required. Diego also has a concurrent process running to check the health of the container at period times.

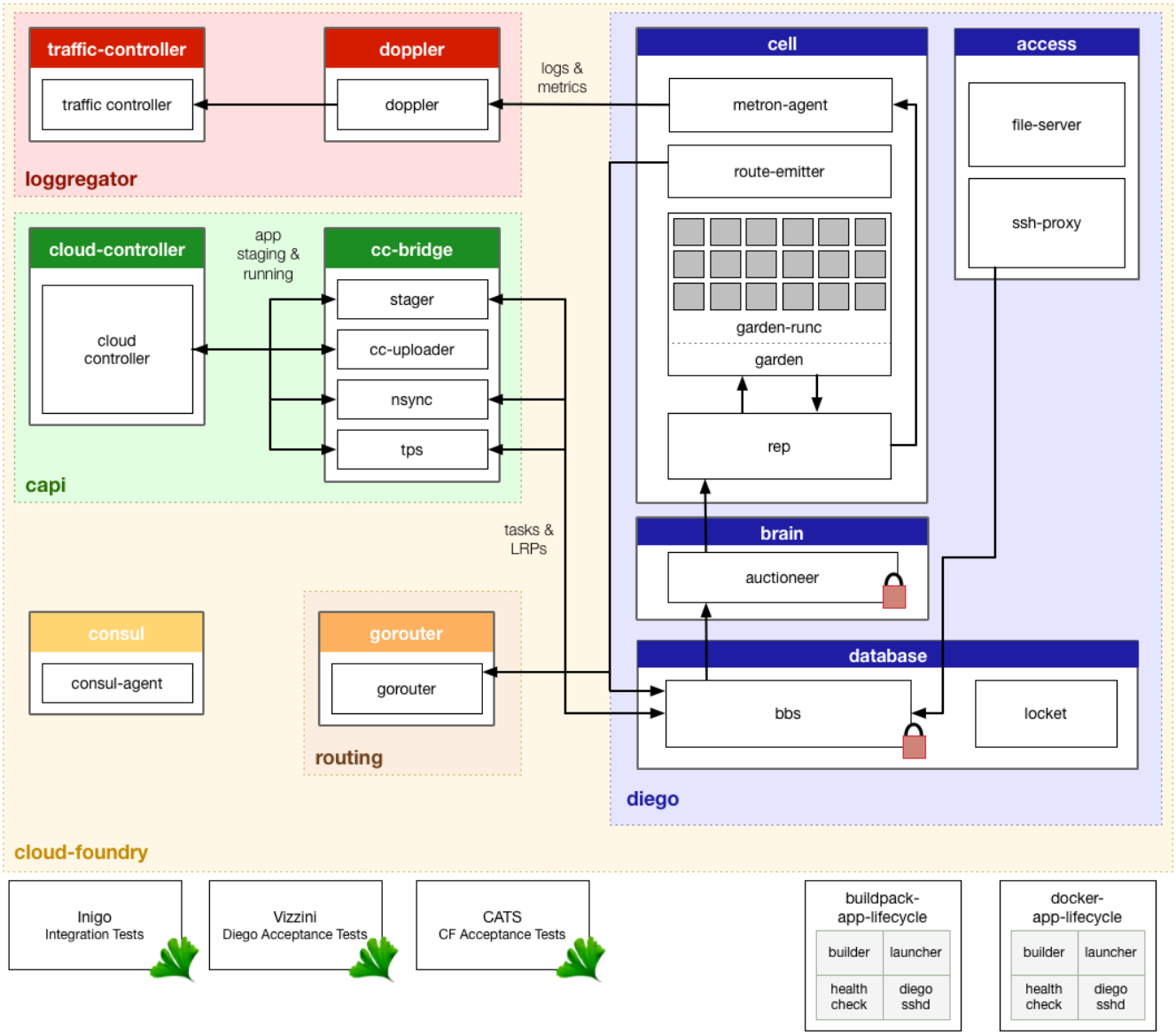

Figure 14. Diego Flow and Architecture Digaram

Cloud Foundry Process

CF utilizes *Diego* in the process of making resources available for a process. It is also used as an *app management system* that sends a request to the BBS API, the public facing aspect of the service. Clients send data to the BBS with a work description. Diego then makes decisions about distribution and lifecycle enforcement of a job based upon the submitted characteristics. The BBS then contacts the *Auctioneer* system to create an *Auction* based upon the resources requested by the job (usually an LRP - Long Running Process). The Auctioneer polls the resource offerings for information including *state* and available resources. The BBS then matches the available resources with the task requests and takes corrective action on the resources if required.

The Rep is a piece of software that *bids* on tasks to be completed. It does this by accepting the request from the Auction, creates a *Garden Container* (essentially an isolated containerized environment) and executes the work described in the Task. Once the container has been created in the local Garden, the Rep broadcasts *cells* to the rest of the system and instructs what containers should be created on behalf of Cloud Foundry. The *Converger* periodically checks the state of the BBS used by Diego executors and cleans as appropriate. During the execution process, Diego streams real-time logs for Tasks and LRPs to the *Loggerator*. Additionally, each LRP instance is

registered with GoRouter (the router for a Cloud Foundry instance) for routing of external web traffic to the task.

10.2.5. Implications of Cloud Foundry in the Geospatial Domain

Geospatial data has always required a processing overhead due to the nature of the complex operations that can be performed using spatial techniques. However, with recent trends including Volunteered Geographic Information (VGI), crowdsourcing and collection of three-dimensional (3D) point clouds, data volumes have grown-significantly. Processing of these datasets is a challenge due to their size and complexity. Up until recently, OGC architectures have assumed a one-to-one relationship between server and interface, that is, one interface exposes resources on one server instance. These new forms of big data require a different approach to resource management to enable the most efficient usage.

Services or resources?

OGC services have been traditionally based upon a *Service Oriented Architecture* (SOA). For example, *WFS* is a *view on a database*, that is, there is a database on a server and a WFS sits in front of the database to provide access to the database contents in a standardized manner. Other OGC data services such as WCS work in a similar manner although they may refer to a directory structure, although still focused on a single server. Processing services within the OGC have worked on a similar assumption that the service is running on a single machine, or more recently, a single container. Although not explicitly stated, often an OGC service is tightly coupled to the server that it is providing an interface for.

There has recently been a move within the OGC to OpenAPI (Swagger) and REST interfaces within the OGC. This has been spearheaded by the WFS Standards Working Group (through work on WFS 3.0) who have chosen to revolutionize the standard by moving to an entirely different framework. Although WFS 3.0 maintains the spirit of WFS, it has removed a lot of the calls that were part of previous versions of the standard. There are no longer calls such as:

- GetCapabilities.
- GetFeature.

Instead the REST calls are based upon HTTP verbs:

- GET
- PUT
- POST
- HEAD
- PATCH
- DELETE

Additionally, resources are exposed via different endpoints, which in turn determine the function of the API path, these endpoints are then called using one of the above http methods depending on the capabilities of the endpoint.

Cloud Foundry in OGC architectures

Cloud Foundry is essentially a server-side application that contains the functions described in the above paragraphs. The potential applicability of Cloud Foundry to OGC services is enabled by the switch to this resource-based architecture. Whereas with service-based architectures, the assumption is one server, one interface. With resource-based architectures, no such assumption is asserted. Therefore it is feasible to have an API sitting in front of many servers or container-based architecture and exposing the resources without the user necessarily understanding the architecture behind the API. This approach is seen in Artificial Intelligence (AI), with services such as Clarifai (https://clarifai.com/) and crowdsourced weather information (https://www.wunderground.com/weather/api/), but rarely seen in OGC web services.

From an architectural perspective, Cloud Foundry could be used behind the scenes of an API such as WFS 3.0 to manage resources required by the requesting client. OGC web services could scale elastically according to demand and load could be balanced across resources.

Long Running Processes (LRPs) are an area that geographic information services have found problematic to account for using current architectures. WPS 2.0 uses asynchronous processing to create and manage LRPs. Although this solves problems such as client time-outs, it does not re-address resourcing associated with long running processes and just deploys a process onto the server in the same way that any other process is deployed. In contrast, Cloud Foundry has modules and methods specifically devoted to registering and resourcing long running processes through its bulletin board interface. This means that the correct resources can be allocated via Cloud Foundry according to the long running process.

Docker Swarm

Docker Swarm provides many of the same functions as Kubernetes as it provides clustering and scheduling for Docker containers. Container management is automated through Docker Compose, which defines multicontainer applications and Swarm itself, which consolidates Docker hosts into a cluster and enables container orchestration and cluster management. A difference between Swarm and Kubernetes is the manner in which they orchestrate, Swarm has a focus on specific resources such as services, whereas Kubernetes utilizes co-located groups of containers.

Docker Enterprise Edition

Docker EE is a platform that enables simultaneous running of both Docker Swarm and Kubernetes, which enables developers and operations teams to use both interchangeably without committing to one. Resources in Kubernetes and Swarm are presented as shared and interchangeably and worker nodes can be set to work with Kubernetes only, Swarm only or in a mixed mode. Although useful for development, mixing Kubernetes and Swarm in mixed mode is not recommended in the current release.

10.2.6. Discussion

CF and technologies like it are built to cope with processing large quantities of data across elastic architectures and resource management systems with resources spread across different cloud providers, local machines, virtual machines and containers. From an OGC perspective, many of the use cases put forward for testbeds do not require this level of behind-the-scenes resource

management, as the use cases are for demonstrator deliverables. Utilization of CF and associated technologies falls into the operational sphere, where timely result return is critical to mission success. The movement to a resource-based architecture (at least in WFS 3.0) is a move that mirrors the move to REST based architectures in many of the non-geospatial (and geospatial for that matter) domains and is timely in terms of CF. The new resource-based architectures do not assert or assume a one-to-one mapping between resources and interfaces, or indeed that resources are co-located. Resource-based architectures provide an API to access the contained resources that may or may not route through to local servers.

CF has potential in the geospatial domain as it enables scalable, elastic resource management using a variety of providers. This coupled with the movement to resource-based architectures and APIs offers the OGC the opportunity to create or revise a suite of standards that support more modern, real-world operational practices.

Chapter 11. Conclusion

The implementation described in this ER has demonstrated several new possibilities for workflows in the OGC. BPMN can be used to orchestrate OGC services, this has been shown through this document and the Workflows ER from Testbed-13 using two different BPMN workflow implementations. Some of the achievements include:

- Differential access to security within the workflow.
- A fully *independent* workflow engine loosely coupled to the rest of the federation.
- Best practices for developing BPMN helper classes.
- An informative study on emerging DevOps orchestration technologies for use in Geo.

Appendix A: XML Schema Documents

BPMN document used in TIE testing.

```
<bpmn2:definitions
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.omg.org/bpmn20"
    xmlns:bpmn2="http://www.omg.org/spec/BPMN/20100524/MODEL"
    xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
    xmlns:bpsim="http://www.bpsim.org/schemas/1.0"
    xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
    xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
    xmlns:drools="http://www.jboss.org/drools" id="_gY3O0KdlEeiaoq0Pfrgi7g"
xsi:schemaLocation="http://www.omg.org/spec/BPMN/20100524/MODEL BPMN20.xsd
http://www.jboss.org/drools drools.xsd http://www.bpsim.org/schemas/1.0 bpsim.xsd"
exporter="jBPM Process Modeler" exporterVersion="1.0"
targetNamespace="http://www.omg.org/bpmn20">
    <bpmn2:itemDefinition id="_AttErrorItem" structureRef="Object"/>
    <bpmn2:itemDefinition id="_ConReportItem" structureRef="Object"/>
    <bpmn2:itemDefinition id="_QACRItem" structureRef="Object"/>
    <bpmn2:itemDefinition id="_ConsolLNItem" structureRef="Object"/>
    <bpmn2:itemDefinition id="_ConsolPTItem" structureRef="Object"/>
    <bpmn2:itemDefinition id="_ResultItem" structureRef="Object"/>
    <bpmn2:itemDefinition id="__5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_TaskNameInputXItem" structureRef="String"/>
    <bpmn2:itemDefinition id="__5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_bearerTokenInputXItem" structureRef="String"/>
    <bpmn2:itemDefinition id="__5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_input-
featuresInputXItem" structureRef="String"/>
    <bpmn2:itemDefinition id="__5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_processDescriptionInputXItem" structureRef="String"/>
    <bpmn2:itemDefinition id="__5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_wpsURLInputXItem"
structureRef="String"/>
    <bpmn2:itemDefinition id="__5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_QACR_fileOutputXItem" structureRef="Object"/>
    <bpmn2:itemDefinition id="__5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_attribution-
errorsOutputXItem" structureRef="Object"/>
    <bpmn2:itemDefinition id="__5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_condition-
reportsOutputXItem" structureRef="Object"/>
    <bpmn2:itemDefinition id="__5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_consolidated1LNOutputXItem" structureRef="Object"/>
    <bpmn2:itemDefinition id="__5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_consolidated1PTOutputXItem" structureRef="Object"/>
    <bpmn2:itemDefinition id="__118418A4-F190-4074-862F-
CC06149554B2_TaskNameInputXItem" structureRef="String"/>
    <bpmn2:itemDefinition id="__118418A4-F190-4074-862F-
CC06149554B2_inputWFSUrlInputXItem" structureRef="String"/>
    <bpmn2:itemDefinition id="__118418A4-F190-4074-862F-
CC06149554B2_processDescriptionInputXItem" structureRef="String"/>
    <bpmn2:itemDefinition id="__118418A4-F190-4074-862F-CC06149554B2_wpsURLInputXItem"
```

```
structureRef="String"/>
    <bpmn2:itemDefinition id="__118418A4-F190-4074-862F-
CC06149554B2_vectorDataOutputXItem" structureRef="Object"/>
    <bpmn2:process id="Testbed_14_secured.Testbed_14_Gait_Process"
drools:version="1.0" name="Testbed_14_Gait_Process" isExecutable="true">
        <bpmn2:property id="AttError" itemSubjectRef="_AttErrorItem"/>
        <bpmn2:property id="ConReport" itemSubjectRef="_ConReportItem"/>
        <bpmn2:property id="QACR" itemSubjectRef="_QACRItem"/>
        <bpmn2:property id="ConsolLN" itemSubjectRef="_ConsolLNItem"/>
        <bpmn2:property id="ConsolPT" itemSubjectRef="_ConsolPTItem"/>
        <bpmn2:property id="Result" itemSubjectRef="_ResultItem"/>
        <bpmn2:startEvent id="_8102727C-CD99-4284-825E-E76E956AFFAA" name="">
            <bpmn2:extensionElements>
                <drools:metaData name="elementname">
                    <drools:metaValue>
                        <![CDATA[ ]]>
                    </drools:metaValue>
                </drools:metaData>
            </bpmn2:extensionElements>
            <bpmn2:outgoing>_794A0D3D-52B7-4F31-80AF-0471A38D2A3C</bpmn2:outgoing>
        </bpmn2:startEvent>
        <bpmn2:task id="_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0"
drools:taskName="TBGaitProcess" name="Secured Gait Process">
            <bpmn2:extensionElements>
                <drools:metaData name="elementname">
                    <drools:metaValue>
                        <![CDATA[ Secured Gait Process ]]>
                    </drools:metaValue>
                </drools:metaData>
            </bpmn2:extensionElements>
            <bpmn2:incoming>_794A0D3D-52B7-4F31-80AF-0471A38D2A3C</bpmn2:incoming>
            <bpmn2:outgoing>_8CC002A6-4270-4A8D-9C3B-7E9CC2737380</bpmn2:outgoing>
            <bpmn2:ioSpecification id="_gY3O0adlEeiaoq0Pfrgi7g">
                <bpmn2:dataInput id="_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_TaskNameInputX" drools:dtype="String" itemSubjectRef="__5BBD6BDD-0A50-
464F-9421-BDB1616DE6C0_TaskNameInputXItem" name="TaskName"/>
                <bpmn2:dataInput id="_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_bearerTokenInputX" drools:dtype="String" itemSubjectRef="__5BBD6BDD-0A50-
464F-9421-BDB1616DE6C0_bearerTokenInputXItem" name="bearerToken"/>
                <bpmn2:dataInput id="_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_input-
featuresInputX" drools:dtype="String" itemSubjectRef="__5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_input-featuresInputXItem" name="input-features"/>
                <bpmn2:dataInput id="_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_processDescriptionInputX" drools:dtype="String"
itemSubjectRef="__5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_processDescriptionInputXItem"
name="processDescription"/>
                <bpmn2:dataInput id="_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_wpsURLInputX" drools:dtype="String" itemSubjectRef="__5BBD6BDD-0A50-464F-
9421-BDB1616DE6C0_wpsURLInputXItem" name="wpsURL"/>
                <bpmn2:dataOutput id="_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_QACR_fileOutputX" drools:dtype="Object" itemSubjectRef="__5BBD6BDD-0A50-
```

```
464F-9421-BDB1616DE6C0_QACR_fileOutputXItem" name="QACR_file"/>
                <bpmn2:dataOutput id="_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_attribution-errorsOutputX" drools:dtype="Object"
itemSubjectRef="__5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_attribution-errorsOutputXItem"
name="attribution-errors"/>
                <bpmn2:dataOutput id="_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_condition-
reportsOutputX" drools:dtype="Object" itemSubjectRef="__5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_condition-reportsOutputXItem" name="condition-reports"/>
                <bpmn2:dataOutput id="_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_consolidated1LNOutputX" drools:dtype="Object" itemSubjectRef="__5BBD6BDD-
0A50-464F-9421-BDB1616DE6C0_consolidated1LNOutputXItem" name="consolidated1LN"/>
                <bpmn2:dataOutput id="_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_consolidated1PTOutputX" drools:dtype="Object" itemSubjectRef="__5BBD6BDD-
0A50-464F-9421-BDB1616DE6C0_consolidated1PTOutputXItem" name="consolidated1PT"/>
                <bpmn2:inputSet id="_gY314KdlEeiaoq0Pfrgi7g">
                    <bpmn2:dataInputRefs>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_bearerTokenInputX
</bpmn2:dataInputRefs>
                    <bpmn2:dataInputRefs>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_input-featuresInputX
</bpmn2:dataInputRefs>
                    <bpmn2:dataInputRefs>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_processDescriptionInputX
</bpmn2:dataInputRefs>
                    <bpmn2:dataInputRefs>_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_wpsURLInputX</bpmn2:dataInputRefs>
                    <bpmn2:dataInputRefs>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_TaskNameInputX
</bpmn2:dataInputRefs>
                </bpmn2:inputSet>
                <bpmn2:outputSet id="_gY314adlEeiaoq0Pfrgi7g">
                    <bpmn2:dataOutputRefs>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_QACR_fileOutputX
</bpmn2:dataOutputRefs>
                    <bpmn2:dataOutputRefs>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_attribution-errorsOutputX
</bpmn2:dataOutputRefs>
                    <bpmn2:dataOutputRefs>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_condition-reportsOutputX
</bpmn2:dataOutputRefs>
                    <bpmn2:dataOutputRefs>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_consolidated1LNOutputX
</bpmn2:dataOutputRefs>
                    <bpmn2:dataOutputRefs>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_consolidated1PTOutputX
</bpmn2:dataOutputRefs>
                </bpmn2:outputSet>
            </bpmn2:ioSpecification>
            <bpmn2:dataInputAssociation id="_gY314qdlEeiaoq0Pfrgi7g">
                <bpmn2:targetRef>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_TaskNameInputX
```

```
</bpmn2:targetRef>
                <bpmn2:assignment id="_gY3146dlEeiaoq0Pfrgi7g">
                    <bpmn2:from xsi:type="bpmn2:tFormalExpression"
id="_gY315KdlEeiaoq0Pfrgi7g">TBGaitProcess</bpmn2:from>
                    <bpmn2:to xsi:type="bpmn2:tFormalExpression"
id="_gY315adlEeiaoq0Pfrgi7g">
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_TaskNameInputX
</bpmn2:to>
                </bpmn2:assignment>
            </bpmn2:dataInputAssociation>
            <bpmn2:dataInputAssociation id="_gY315qdlEeiaoq0Pfrgi7g">
                <bpmn2:targetRef>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_bearerTokenInputX
</bpmn2:targetRef>
                <bpmn2:assignment id="_gY3156dlEeiaoq0Pfrgi7g">
                    <bpmn2:from xsi:type="bpmn2:tFormalExpression"
id="_gY316KdlEeiaoq0Pfrgi7g">
                        <![CDATA[ ae4c4f88-9edd-49f3-a765-e00be886da48 ]]>
                    </bpmn2:from>
                    <bpmn2:to xsi:type="bpmn2:tFormalExpression"
id="_gY316adlEeiaoq0Pfrgi7g">
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_bearerTokenInputX
</bpmn2:to>
                </bpmn2:assignment>
            </bpmn2:dataInputAssociation>
            <bpmn2:dataInputAssociation id="_gY316qdlEeiaoq0Pfrgi7g">
                <bpmn2:targetRef>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_input-featuresInputX
</bpmn2:targetRef>
                <bpmn2:assignment id="_gY3166dlEeiaoq0Pfrgi7g">
                    <bpmn2:from xsi:type="bpmn2:tFormalExpression"
id="_gY317KdlEeiaoq0Pfrgi7g">
                        <![CDATA[
http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&request=Ge
tFeature&typeName=tb14:AgricultureSrf&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:CultureSrf&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:FacilitySrf&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:HydrographySrf&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:InformationPnt&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:RecreationSrf&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:SettlementSrf&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:StructurePnt&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:StructureSrf&outputFormat=SHAPE-
```

```
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:TransportationGroundCrv&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:TransportationGroundSrf&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:UtilityInfrastructureSrf&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:o2s_A&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:o2s_L&outputFormat=SHAPE-
ZIP;http://testbed.dev.52north.org/geoserver/tb14/ows?service=WFS&version=1.0.0&reques
t=GetFeature&typeName=tb14:o2s_P&outputFormat=SHAPE-ZIP
]]>
                    </bpmn2:from>
                    <bpmn2:to xsi:type="bpmn2:tFormalExpression"
id="_gY317adlEeiaoq0Pfrgi7g">
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_input-featuresInputX
</bpmn2:to>
                </bpmn2:assignment>
            </bpmn2:dataInputAssociation>
            <bpmn2:dataInputAssociation id="_gY317qdlEeiaoq0Pfrgi7g">
                <bpmn2:targetRef>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_processDescriptionInputX
</bpmn2:targetRef>
                <bpmn2:assignment id="_gY3176dlEeiaoq0Pfrgi7g">
                    <bpmn2:from xsi:type="bpmn2:tFormalExpression"
id="_gY318KdlEeiaoq0Pfrgi7g">
                        <![CDATA[
org.n52.geoprocessing.project.testbed14.gait.GenericGaitToolAlgorithm
]]>
                    </bpmn2:from>
                    <bpmn2:to xsi:type="bpmn2:tFormalExpression"
id="_gY318adlEeiaoq0Pfrgi7g">
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_processDescriptionInputX
</bpmn2:to>
                </bpmn2:assignment>
            </bpmn2:dataInputAssociation>
            <bpmn2:dataInputAssociation id="_gY318qdlEeiaoq0Pfrgi7g">
                <bpmn2:targetRef>_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_wpsURLInputX</bpmn2:targetRef>
                <bpmn2:assignment id="_gY3186dlEeiaoq0Pfrgi7g">
                    <bpmn2:from xsi:type="bpmn2:tFormalExpression"
id="_gY319KdlEeiaoq0Pfrgi7g">
                        <![CDATA[
http://testbed.dev.52north.org/tb14-d146-secured/service/wps
]]>
                    </bpmn2:from>
                    <bpmn2:to xsi:type="bpmn2:tFormalExpression"
id="_gY319adlEeiaoq0Pfrgi7g">_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0_wpsURLInputX</bpmn2:to>
                </bpmn2:assignment>
```

```
            </bpmn2:dataInputAssociation>
            <bpmn2:dataOutputAssociation id="_gY319qdlEeiaoq0Pfrgi7g">
                <bpmn2:sourceRef>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_QACR_fileOutputX
</bpmn2:sourceRef>
                <bpmn2:targetRef>QACR</bpmn2:targetRef>
            </bpmn2:dataOutputAssociation>
            <bpmn2:dataOutputAssociation id="_gY3196dlEeiaoq0Pfrgi7g">
                <bpmn2:sourceRef>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_attribution-errorsOutputX
</bpmn2:sourceRef>
                <bpmn2:targetRef>AttError</bpmn2:targetRef>
            </bpmn2:dataOutputAssociation>
            <bpmn2:dataOutputAssociation id="_gY31-KdlEeiaoq0Pfrgi7g">
                <bpmn2:sourceRef>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_condition-reportsOutputX
</bpmn2:sourceRef>
                <bpmn2:targetRef>ConReport</bpmn2:targetRef>
            </bpmn2:dataOutputAssociation>
            <bpmn2:dataOutputAssociation id="_gY31-adlEeiaoq0Pfrgi7g">
                <bpmn2:sourceRef>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_consolidated1LNOutputX
</bpmn2:sourceRef>
                <bpmn2:targetRef>ConsolLN</bpmn2:targetRef>
            </bpmn2:dataOutputAssociation>
            <bpmn2:dataOutputAssociation id="_gY31-qdlEeiaoq0Pfrgi7g">
                <bpmn2:sourceRef>
_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0_consolidated1PTOutputX
</bpmn2:sourceRef>
                <bpmn2:targetRef>ConsolPT</bpmn2:targetRef>
            </bpmn2:dataOutputAssociation>
        </bpmn2:task>
        <bpmn2:endEvent id="_4389A7BA-201F-4637-B755-FD541CB38E58" name="">
            <bpmn2:extensionElements>
                <drools:metaData name="elementname">
                    <drools:metaValue>
                        <![CDATA[ ]]>
                    </drools:metaValue>
                </drools:metaData>
            </bpmn2:extensionElements>
            <bpmn2:incoming>_1C93B068-0C20-49DC-A41A-00F0698CA134</bpmn2:incoming>
        </bpmn2:endEvent>
        <bpmn2:task id="_118418A4-F190-4074-862F-CC06149554B2"
drools:taskName="GetDataFromWFS" name="Get data from WFS">
            <bpmn2:extensionElements>
                <drools:metaData name="elementname">
                    <drools:metaValue>
                        <![CDATA[ Get data from WFS ]]>
                    </drools:metaValue>
                </drools:metaData>
            </bpmn2:extensionElements>
```

```
            <bpmn2:incoming>_8CC002A6-4270-4A8D-9C3B-7E9CC2737380</bpmn2:incoming>
            <bpmn2:outgoing>_1C93B068-0C20-49DC-A41A-00F0698CA134</bpmn2:outgoing>
            <bpmn2:ioSpecification id="_gY31-6dlEeiaoq0Pfrgi7g">
                <bpmn2:dataInput id="_118418A4-F190-4074-862F-
CC06149554B2_TaskNameInputX" drools:dtype="String" itemSubjectRef="__118418A4-F190-
4074-862F-CC06149554B2_TaskNameInputXItem" name="TaskName"/>
                <bpmn2:dataInput id="_118418A4-F190-4074-862F-
CC06149554B2_inputWFSUrlInputX" drools:dtype="String" itemSubjectRef="__118418A4-F190-
4074-862F-CC06149554B2_inputWFSUrlInputXItem" name="inputWFSUrl"/>
                <bpmn2:dataInput id="_118418A4-F190-4074-862F-
CC06149554B2_processDescriptionInputX" drools:dtype="String"
itemSubjectRef="__118418A4-F190-4074-862F-CC06149554B2_processDescriptionInputXItem"
name="processDescription"/>
                <bpmn2:dataInput id="_118418A4-F190-4074-862F-
CC06149554B2_wpsURLInputX" drools:dtype="String" itemSubjectRef="__118418A4-F190-4074-
862F-CC06149554B2_wpsURLInputXItem" name="wpsURL"/>
                <bpmn2:dataOutput id="_118418A4-F190-4074-862F-
CC06149554B2_vectorDataOutputX" drools:dtype="Object" itemSubjectRef="__118418A4-F190-
4074-862F-CC06149554B2_vectorDataOutputXItem" name="vectorData"/>
                <bpmn2:inputSet id="_gY31_KdlEeiaoq0Pfrgi7g">
                    <bpmn2:dataInputRefs>
_118418A4-F190-4074-862F-CC06149554B2_inputWFSUrlInputX
</bpmn2:dataInputRefs>
                    <bpmn2:dataInputRefs>
_118418A4-F190-4074-862F-CC06149554B2_processDescriptionInputX
</bpmn2:dataInputRefs>
                    <bpmn2:dataInputRefs>_118418A4-F190-4074-862F-
CC06149554B2_wpsURLInputX</bpmn2:dataInputRefs>
                    <bpmn2:dataInputRefs>
_118418A4-F190-4074-862F-CC06149554B2_TaskNameInputX
</bpmn2:dataInputRefs>
                </bpmn2:inputSet>
                <bpmn2:outputSet id="_gY31_adlEeiaoq0Pfrgi7g">
                    <bpmn2:dataOutputRefs>
_118418A4-F190-4074-862F-CC06149554B2_vectorDataOutputX
</bpmn2:dataOutputRefs>
                </bpmn2:outputSet>
            </bpmn2:ioSpecification>
            <bpmn2:dataInputAssociation id="_gY31_qdlEeiaoq0Pfrgi7g">
                <bpmn2:targetRef>
_118418A4-F190-4074-862F-CC06149554B2_TaskNameInputX
</bpmn2:targetRef>
                <bpmn2:assignment id="_gY31_6dlEeiaoq0Pfrgi7g">
                    <bpmn2:from xsi:type="bpmn2:tFormalExpression"
id="_gY32AKdlEeiaoq0Pfrgi7g">GetDataFromWFS</bpmn2:from>
                    <bpmn2:to xsi:type="bpmn2:tFormalExpression"
id="_gY32AadlEeiaoq0Pfrgi7g">
_118418A4-F190-4074-862F-CC06149554B2_TaskNameInputX
</bpmn2:to>
                </bpmn2:assignment>
            </bpmn2:dataInputAssociation>
```

```
            <bpmn2:dataInputAssociation id="_gY32AqdlEeiaoq0Pfrgi7g">
                <bpmn2:sourceRef>ConsolPT</bpmn2:sourceRef>
                <bpmn2:targetRef>
_118418A4-F190-4074-862F-CC06149554B2_inputWFSUrlInputX
</bpmn2:targetRef>
            </bpmn2:dataInputAssociation>
            <bpmn2:dataInputAssociation id="_gY32A6dlEeiaoq0Pfrgi7g">
                <bpmn2:targetRef>
_118418A4-F190-4074-862F-CC06149554B2_processDescriptionInputX
</bpmn2:targetRef>
                <bpmn2:assignment id="_gY32BKdlEeiaoq0Pfrgi7g">
                    <bpmn2:from xsi:type="bpmn2:tFormalExpression"
id="_gY32BadlEeiaoq0Pfrgi7g">
                        <![CDATA[ storage.geoserver.GetWFSData ]]>
                    </bpmn2:from>
                    <bpmn2:to xsi:type="bpmn2:tFormalExpression"
id="_gY32BqdlEeiaoq0Pfrgi7g">
_118418A4-F190-4074-862F-CC06149554B2_processDescriptionInputX
</bpmn2:to>
                </bpmn2:assignment>
            </bpmn2:dataInputAssociation>
            <bpmn2:dataInputAssociation id="_gY32B6dlEeiaoq0Pfrgi7g">
                <bpmn2:targetRef>_118418A4-F190-4074-862F-
CC06149554B2_wpsURLInputX</bpmn2:targetRef>
                <bpmn2:assignment id="_gY32CKdlEeiaoq0Pfrgi7g">
                    <bpmn2:from xsi:type="bpmn2:tFormalExpression"
id="_gY32CadlEeiaoq0Pfrgi7g">
                        <![CDATA[ http://localhost:8000/wps/WebProcessingService ]]>
                    </bpmn2:from>
                    <bpmn2:to xsi:type="bpmn2:tFormalExpression"
id="_gY32CqdlEeiaoq0Pfrgi7g">_118418A4-F190-4074-862F-
CC06149554B2_wpsURLInputX</bpmn2:to>
                </bpmn2:assignment>
            </bpmn2:dataInputAssociation>
            <bpmn2:dataOutputAssociation id="_gY32C6dlEeiaoq0Pfrgi7g">
                <bpmn2:sourceRef>
_118418A4-F190-4074-862F-CC06149554B2_vectorDataOutputX
</bpmn2:sourceRef>
                <bpmn2:targetRef>Result</bpmn2:targetRef>
            </bpmn2:dataOutputAssociation>
        </bpmn2:task>
        <bpmn2:sequenceFlow id="_8CC002A6-4270-4A8D-9C3B-7E9CC2737380"
sourceRef="_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0" targetRef="_118418A4-F190-4074-862F-
CC06149554B2"/>
        <bpmn2:sequenceFlow id="_794A0D3D-52B7-4F31-80AF-0471A38D2A3C"
sourceRef="_8102727C-CD99-4284-825E-E76E956AFFAA" targetRef="_5BBD6BDD-0A50-464F-9421-
BDB1616DE6C0"/>
        <bpmn2:sequenceFlow id="_1C93B068-0C20-49DC-A41A-00F0698CA134"
sourceRef="_118418A4-F190-4074-862F-CC06149554B2" targetRef="_4389A7BA-201F-4637-B755-
FD541CB38E58"/>
    </bpmn2:process>
```

```
    <bpmndi:BPMNDiagram id="_gY4c8KdlEeiaoq0Pfrgi7g">
        <bpmndi:BPMNPlane id="_gY4c8adlEeiaoq0Pfrgi7g"
bpmnElement="Testbed_14_secured.Testbed_14_Gait_Process">
            <bpmndi:BPMNShape id="_gY4c8qdlEeiaoq0Pfrgi7g" bpmnElement="_8102727C-
CD99-4284-825E-E76E956AFFAA">
                <dc:Bounds height="56.0" width="56.0" x="143.0" y="198.0"/>
            </bpmndi:BPMNShape>
            <bpmndi:BPMNShape id="_gY4c86dlEeiaoq0Pfrgi7g" bpmnElement="_5BBD6BDD-
0A50-464F-9421-BDB1616DE6C0">
                <dc:Bounds height="102.0" width="154.0" x="342.0" y="175.0"/>
            </bpmndi:BPMNShape>
            <bpmndi:BPMNShape id="_gY4c9KdlEeiaoq0Pfrgi7g" bpmnElement="_4389A7BA-
201F-4637-B755-FD541CB38E58">
                <dc:Bounds height="56.0" width="56.0" x="834.0" y="198.0"/>
            </bpmndi:BPMNShape>
            <bpmndi:BPMNShape id="_gY4c9adlEeiaoq0Pfrgi7g" bpmnElement="_118418A4-
F190-4074-862F-CC06149554B2">
                <dc:Bounds height="102.0" width="154.0" x="565.0" y="175.0"/>
            </bpmndi:BPMNShape>
            <bpmndi:BPMNEdge id="_gY4c9qdlEeiaoq0Pfrgi7g" bpmnElement="_8CC002A6-4270-
4A8D-9C3B-7E9CC2737380" sourceElement="_gY4c86dlEeiaoq0Pfrgi7g"
targetElement="_gY4c9adlEeiaoq0Pfrgi7g">
                <di:waypoint xsi:type="dc:Point" x="419.0" y="226.0"/>
                <di:waypoint xsi:type="dc:Point" x="642.0" y="226.0"/>
            </bpmndi:BPMNEdge>
            <bpmndi:BPMNEdge id="_gY4c96dlEeiaoq0Pfrgi7g" bpmnElement="_794A0D3D-52B7-
4F31-80AF-0471A38D2A3C" sourceElement="_gY4c8qdlEeiaoq0Pfrgi7g"
targetElement="_gY4c86dlEeiaoq0Pfrgi7g">
                <di:waypoint xsi:type="dc:Point" x="171.0" y="226.0"/>
                <di:waypoint xsi:type="dc:Point" x="419.0" y="226.0"/>
            </bpmndi:BPMNEdge>
            <bpmndi:BPMNEdge id="_gY4c-KdlEeiaoq0Pfrgi7g" bpmnElement="_1C93B068-0C20-
49DC-A41A-00F0698CA134" sourceElement="_gY4c9adlEeiaoq0Pfrgi7g"
targetElement="_gY4c9KdlEeiaoq0Pfrgi7g">
                <di:waypoint xsi:type="dc:Point" x="642.0" y="226.0"/>
                <di:waypoint xsi:type="dc:Point" x="862.0" y="226.0"/>
            </bpmndi:BPMNEdge>
        </bpmndi:BPMNPlane>
    </bpmndi:BPMNDiagram>
    <bpmn2:relationship id="_gY4c-adlEeiaoq0Pfrgi7g" type="BPSimData">
        <bpmn2:extensionElements>
            <bpsim:BPSimData>
                <bpsim:Scenario xsi:type="bpsim:Scenario" id="default"
name="Simulationscenario">
                    <bpsim:ScenarioParameters xsi:type="bpsim:ScenarioParameters"/>
                    <bpsim:ElementParameters xsi:type="bpsim:ElementParameters"
elementRef="_5BBD6BDD-0A50-464F-9421-BDB1616DE6C0" id="_gY4c-qdlEeiaoq0Pfrgi7g">
                        <bpsim:TimeParameters xsi:type="bpsim:TimeParameters">
                            <bpsim:ProcessingTime xsi:type="bpsim:Parameter">
                                <bpsim:NormalDistribution mean="0.0"
standardDeviation="0.0"/>
```

```
                            </bpsim:ProcessingTime>
                        </bpsim:TimeParameters>
                        <bpsim:CostParameters xsi:type="bpsim:CostParameters">
                            <bpsim:UnitCost xsi:type="bpsim:Parameter">
                                <bpsim:FloatingParameter value="0.0"/>
                            </bpsim:UnitCost>
                        </bpsim:CostParameters>
                    </bpsim:ElementParameters>
                    <bpsim:ElementParameters xsi:type="bpsim:ElementParameters"
elementRef="_118418A4-F190-4074-862F-CC06149554B2" id="_gY4c-6dlEeiaoq0Pfrgi7g">
                        <bpsim:TimeParameters xsi:type="bpsim:TimeParameters">
                            <bpsim:ProcessingTime xsi:type="bpsim:Parameter">
                                <bpsim:NormalDistribution mean="0.0"
standardDeviation="0.0"/>
                            </bpsim:ProcessingTime>
                        </bpsim:TimeParameters>
                        <bpsim:CostParameters xsi:type="bpsim:CostParameters">
                            <bpsim:UnitCost xsi:type="bpsim:Parameter">
                                <bpsim:FloatingParameter value="0.0"/>
                            </bpsim:UnitCost>
                        </bpsim:CostParameters>
                    </bpsim:ElementParameters>
                    <bpsim:ElementParameters xsi:type="bpsim:ElementParameters"
elementRef="_8102727C-CD99-4284-825E-E76E956AFFAA" id="_gY4c_KdlEeiaoq0Pfrgi7g">
                        <bpsim:TimeParameters xsi:type="bpsim:TimeParameters">
                            <bpsim:ProcessingTime xsi:type="bpsim:Parameter">
                                <bpsim:NormalDistribution mean="0.0"
standardDeviation="0.0"/>
                            </bpsim:ProcessingTime>
                        </bpsim:TimeParameters>
                    </bpsim:ElementParameters>
                </bpsim:Scenario>
            </bpsim:BPSimData>
        </bpmn2:extensionElements>
        <bpmn2:source>_gY3O0KdlEeiaoq0Pfrgi7g</bpmn2:source>
        <bpmn2:target>_gY3O0KdlEeiaoq0Pfrgi7g</bpmn2:target>
    </bpmn2:relationship>
</bpmn2:definitions>
```

Appendix B: Revision History

Table 1. Revision History

Date	Editor	Release	Primary clauses modified	Descriptions
June 15, 2016	I. Simonis	.1	all	initial version
July 22, 2016	I. Simonis	.9	all	comments integrate
September 7, 2016	S. Simmons	1.0	various	preparation for publication
March 23, 2017	I. Simonis	2.0	all	template simplified
January 18, 2018	S. Serich	2.1	all	additional guidance to Editors; clean up headings in appendices
September 30, 2018	S. Meek	2.2	all	First draft submitted
October 02, 2018	G. Hobona	2.2	all	Review
October 31, 2018	S. Meek	2.2	all	Post to pending

Appendix C: Bibliography

1. Meek, S.: BPMN 2.0 for Orchestrating OGC Services Discussion Paper(Draft).
2. Full meta objects for flexible geoprocessing workflows: profiling WPS or BPMN? 16–30 (2016).
3. Matheus, A.: OGC Testbed-13: Security Engineering Report. (2018).
4. Pross, B., Stasch, C.: OGC Testbed-13: Workflows Engineering Report. (2018).
5. S. Meek, D.L., M. Jackson: A BPMN solution for chaining OGC services to quality assure location-based crowdsourced data. 87, 76–83 (2016).
6. Lee, C.A.: OGC Testbed-14 Federated Clouds Engineering Report. (2018).