Hyper-**N**etwork for **e**lectro**Mo**bility

NeMo Service Creation Tools - Manual

Nils Masuch, Johannes Fähndrich, Tobias Küster (TU Berlin)

May 2018

Table of contents

1 Introduction

1.1 Motivation

The Service Creation Domain will provide IT service developers with an environment that enables them to create new service solutions and to integrate existing ones into the NeMo Hyper-Network. More specific, the Service Creation Domain offers the following two modes for defining NeMo services:

Service Integration: The service that shall be part of the NeMo Hyper-Network is either already existing or atomic, which means that the implementation will be developed by the provider without additional NeMo Tool support. In both cases the Service Creation Domain will mainly be responsible to provide a service description interface based on a semantic model in order to define a semantic description and upload it to a Service Artefacts Server (see Figure 1), where it can be referenced from the service registry within the Service Delivery Domain in order to make it detectable for other business partners.

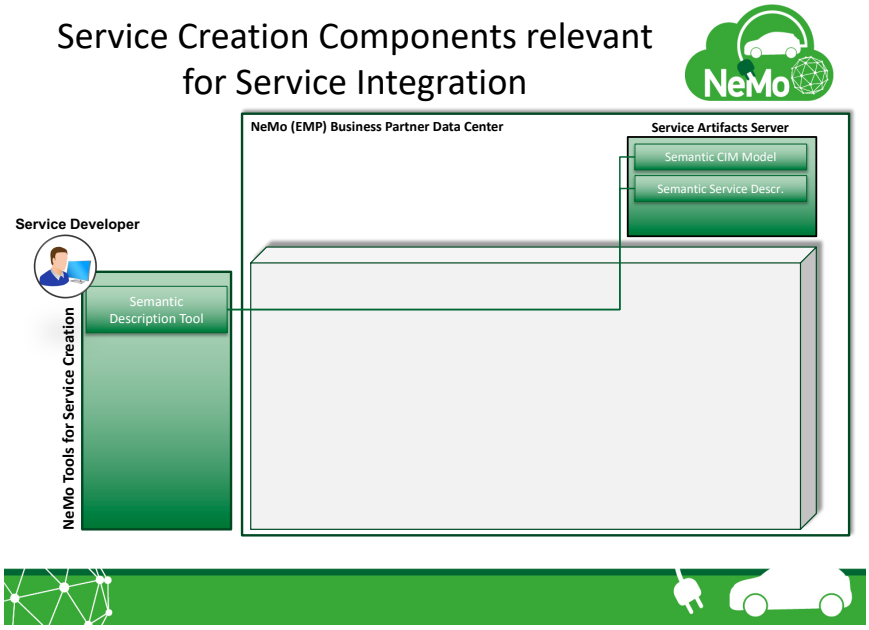

Figure 1: Components involved for integration of existing services into the NeMo Hyper-Network from the Service Creation Domain perspective

Service Creation: In this case, the IT Service Developer wants to develop a new service that is making use of existing services provided via the NeMo Hyper-Network. For this purpose, the IT Service Developer is being provided a set of tools that enable the search for existing services, the implementation of service composition processes, the management of these processes as well as testing features (see Figure 2). The tools are again coupled to the Service Artefacts Server that hosts the semantic service descriptions as well as the complex services for execution purposes. Further, the Low-Level Service Finder and Low-Level Service Optimizer will also be used at creation time for service selection features.

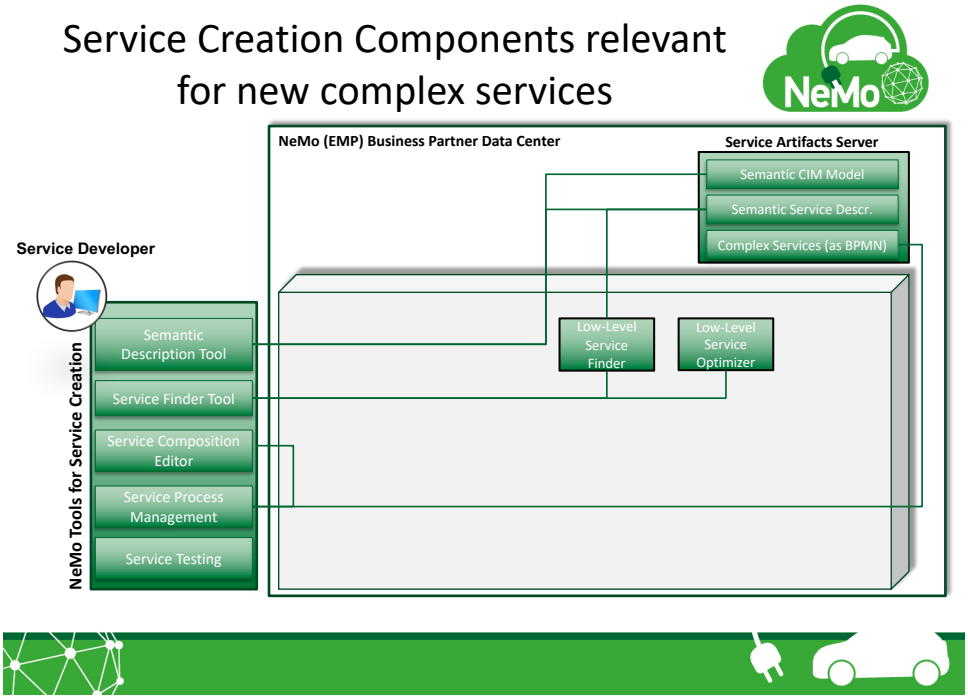

Figure 2: Components involved for the development and integration of new complex services based on existing NeMo services

In the following we will describe how to install the Service Creation Domain, explain shortly the provided components and give hand-on examples how to make use of the tolls in practice.

1.2 Installation

The NeMo Service Creation Domain is embedded in the Eclipse IDE. In order to make installation as simple as possible the whole environment is being provided within a Docker Image (see https://www.docker.com/). In the following, we guide you through the process of running the Docker Image on your machine.

While installing and updating the images is simple, setting up the Docker environment can be a bit more involved, as it requires forwarding of the UI, and an integration with the host system in order to access the projects created within the Eclipse Docker Image. The commands for starting the image are basically the same for each operating system, but there are some peculiarities that have to be taken into account. In all cases, it is recommended to create short `.bat` or `.sh` scripts for starting the containers.

1.2.1 Windows

1. Install and run Docker
2. You need to forward X11 connections, for example by installing *Xming X Server for Windows*[1] and checking *Disable access control* in the settings.
3. Replace `<YOUR IP HERE>` in the following console command with your actual IP, then execute it in a command window or as a `.bat` file:

[1] http://www.straightrunning.com/XmingNotes/

```
docker run -ti –rm ^
      -e DISPLAY=<YOUR IP HERE>:0 ^
      -v C:/Temp ^
      -v %USERPROFILE%/.eclipse-docker:/home/developer ^
      dailab/ssdt:nemo-demo
```

1.2.2 Mac OS

1. Install and run Docker
2. Make sure the following requirements are installed on your system:
   - an X11-Server implementation, for example XQuartz[2]
   - `socat` terminal command[3]
3. Execute this command and leave it running in the background: `socat TCP-LISTEN:6000,reuseaddr,fork UNIX-CLIENT:\"$DISPLAY\"`
4. Replace `<YOUR IP HERE>` in the following console command with your actual IP, then execute it in a new terminal:

```
docker run -ti --rm \
      -e DISPLAY=<YOUR IP HERE>:0 \
      -v /tmp/.X11-unix:/tmp/.X11-unix \
      -v `pwd`/.eclipse-docker:/home/developer \
      dailab/ssdt:nemo-demo
```

1.2.3 Linux

1. Install and run Docker
2. Make sure to create the `eclipse-docker` directory first, otherwise the Docker container will create it in read-only mode: `mkdir eclipse-docker`
3. Execute the following command to run the Docker image (here, `$DISPLAY` is a system variable and does *not* have to be replaced):

```
docker run -ti --rm \
      -e DISPLAY=$DISPLAY \
      -v /tmp/.X11-unix:/tmp/.X11-unix \
      -v `pwd`/eclipse-docker:/home/developer \
      dailab/ssdt:nemo-demo
```

1.2.4 Additional information

The following briefly explains the different options for starting the container. You do not need to know those, but it can help in finding potential mistakes.

- The basic command for running the Docker image is `docker run <imagename>`
- The `-e` option is used to set an environment variable, e.g. `-e DISPLAY=…`

---

[2] https://www.xquartz.org/

[3] http://www.dest-unreach.org/socat/

- The `-v` option is used for "mounting volumes", i.e. for synchronizing directories within the Docker container with your own computer, e.g. `-v \`pwd\`/.eclipse-docker:/home/developer`; this will synchronize the home directory within the Docker container with the `eclipse-docker` directory on your computer
- Finally, the last line in the startup scripts is the name of the image, usually in the format `supplier/imagename:versiontag`

As we update the docker image from time to time for bugfixing reasons or due to new requirements coming up in the project it makes sense to update the local docker image on your machine periodically. For this, you would have to enter the following command:

```
docker pull dailab/ssdt:nemo-demo
```

1.3 Getting Started

After the Eclipse IDE has started successfully, you can close the Welcome Screen and make yourself familiar with the environment. Afterwards, there's still a bit of setup to do.

- Setting up the JIAC node within the Eclipse IDE: Go to *Window → Preferences → JIAC Node Plugin* and make sure that the *Use Gateway* checkbox is *not* checked
- Setting up connections to the Nemo Service Repository and Gitlab server: Go to *Window → Preferences → SSM / OWL-S Editor* and fill out the settings
    - Service Repository URL: The URL of the NeMo Services Repository, for example: *http://nemokubcluster.uk-south.containers.mybluemix.net/api/v1/services*
    - Gitlab Base URL: The basic URL without any path of the Gitlab instance to use, for example: *https://isense-gitlab.iccs.gr*
    - Group Name: The group of the Gitlab project to use, for example: *fka*
    - Project Name: The name of the Gitlab project to use, for example: *artifacts*
    - Project ID: The ID of the project; you can find this in the general project settings in Gitlab, for example: *240*
    - Branch Name: The name of the Git branch to use, for example: *master*
    - Folder Path: Base path for files uploaded to Git (optional), for example: *CompositeService*
    - Commit Message: Default commit message, for example: *pushed from SSM*
    - Private Token: Your private token for authentication in Gitlab; you can find this in your account settings in Gitlab, it should have 20 digits consisting of numbers and letters
- The Docker Image comes with an example project, but for technical reasons, it can not be included in the workspace from the beginning. To import it, select *File → Import → General → Existing Projects into Workspace*, then browse to `/opt/nemo-demo-runtime`, check the *Copy Projects into Workspace* option and click on *Finish*.

Now you should be set up and ready to go with the tools of the NeMo Toolsuite, each of which will be described in detail in the following sections.

2 Components Overview

Each of the components – the Visual Service Design Tool (VSDT), the Semantic Service Manager (SSM) and the Low-Level Service Finder – are included in the customized Eclipse IDE that can be found in the NeMo Docker Image, as described in the last Section.

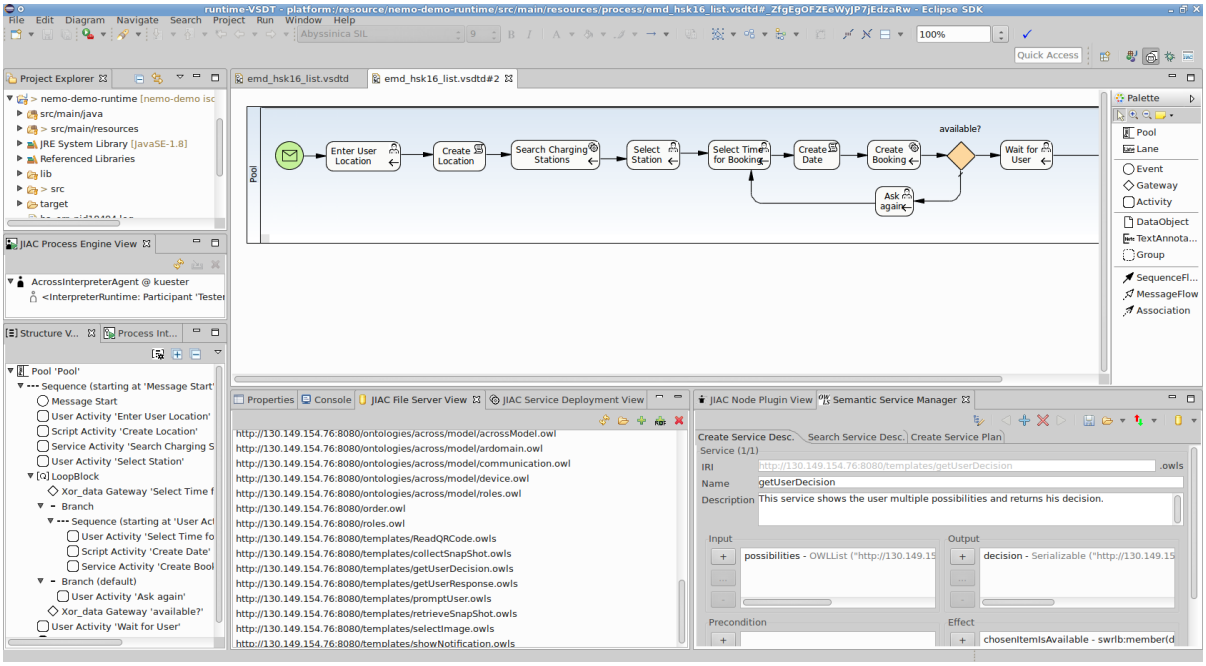

Figure 3: Visual Service Design Tool Eclipse Environment. Clockwise from the top: BPMN-Editor, SSM View, FileServer View, Structure View, Process Engine View, Project Explorer

2.1 Visual Service Design Tool (VSDT)

The Visual Service Design Tool (VSDT) is a BPMN editor developed by DAI-Labor, TU Berlin. While its primary use is as a modelling tool for the JIAC agent framework[4], it can also be used for modelling BPMN diagrams independently of the target language or execution platform.

Using the Business Process Model and Notation[5] (BPMN), the VSDT can be used by a wide variety of users with little training for orchestration services to complex processes. Those processes can then be translated to executable code of various kinds, or interpreted directly in JIAC's BPMN Interpreter Agent.

The VSDT consists of two editors: A use-case editor for managing the different business processes constituting a business process system, with each process representing one use case, and a BPMN editor for the actual business processes. When opening a diagram, at first the use case editor is shown. To view the BPMN editor, double-click on any of the use cases.

Besides the basic diagram editors, the VSDT also provides some useful features for managing the several non-visual attributes and elements that make up the BPMN diagram, such as services, properties, or assignments. The attributes can be accessed in the Properties View of the individual elements. Some of those views also show additional buttons

[4] http://www.jiac.de/development-tools/jiac-toolipse/

[5] http://www.bpmn.org/

for opening dialogues for creating and managing those elements that do not have a representation in the diagrams, such as variable assignments, or service descriptions.

Further, the VSDT provides tools for validating, structuring, and simulating the process. The validation can be triggered using the small check-mark symbol in the diagram's menu bar. Structuring and Simulation are each provided by a special view, and can be used for testing whether the diagram can be translated to a block-structured language (mapping the diagrams flow to corresponding if/else blocks and loops), and whether the different assignments and branching conditions are working correctly.

This is important in case the BPMN diagram is supposed to be transformed to executable code, such as a BPEL process or a JIAC multi-agent system, or for interpreting the process in JIAC's BPMN Interpreter Agent. The transformations are triggered by selecting "Export" from the menu, whereas the deployment to the JIAC Runtime is done via in Process Engine View.

The VSDT also integrates with the Semantic Service Manager (SSM) for importing service descriptions from the SSM into the VSDT. The SSM is described in more detail in the next Section.

Those functions of the VSDT that are specifically important in the context of the NeMo project are described in detail in Section 5 (Creating a new composite service with VSDT). For an in-depth explanation of the VSDT, how it is set up and used, its different features and capabilities, please refer to the manual at http://jiac.de/Downloads/toolipse/vsdt-manual.pdf.

2.2 Semantic Service Manager (SSM)

The SSM is a service description tool developed by DAI-Labor, TU Berlin. It is integrated into the VSDT Eclipse IDE, as visualized in Figure 3. Its user-friendly GUI allows for easy handling of *Semantic Service Descriptions* based on OWL-S[6], even without extensive knowledge of the underlying specifications.

With the SSM, *Semantic Service Descriptions* can be created, modified, saved and loaded, both locally and remotely. These descriptions contain information about the service they describe in a format that can be read and understood by machines. This includes details about the in- and output of the service, as well as any preconditions and effects it might have. The data types of these parameters are described in shared ontologies, which can also be managed through a special SSM interface.

Additionally, the SSM can be used to search for any desired functionality among existing services, for example a specific combination of input and output parameters. This feature utilizes the Low-Level Service Finder, which is described in the next section.

Any services created or found by the SSM can be transferred to the BPMN editor and used as functional building blocks within processes, or even to describe processes as a whole.

Detailed instructions on how the use the features mentioned here are provided in the later chapters of this manual.

Note that the SSM user interface consists of three independent tabs used for creating, searching and planning, respectively. Only the first two are relevant to the NeMo project use-case and the third tab may be hidden in a future NeMo-specific version of the SSM.

[6] https://www.w3.org/Submission/OWL-S/

2.3 Low-Level Service Finder

The service finder is a tool for finding (“matching”) services by measuring the similarity of various aspects of service descriptions. The SSM’s “Search Service Description” view offers functionalities for describing a search template and configuring the service finder’s different parameters.

A search template technically is also a service description and the service finder retrieves services from a repository and sorts them by their similarity with respect to the template. Thus a user can “filter” services in the repository which fulfil the functionalities the user described in the template the most.

The service finder matches input- and output parameters as well as precondition and effect rules. Note that the matcher does not evaluate those rules but rather compares or matches the rule structures. A textual search in the description field as well as in the service name field is also possible.

The SSM offers checkboxes which enable the user to configure the service finder’s matching process i.e. control which aspects like preconditions or effect should be taken into account by the service finder. In addition, the “importance” of the different similarities, for example input parameter similarities, can be controlled. For example, a user can express that high or low similarity in input parameters should be favoured over the occurrences of textual content in the name or description fields, that is, the user can weight the importance of partial results that finally get accumulated to an overall ranking.

Please note that the service finder always works on OWL-S descriptions and in the context of NeMo the finder pulls all known descriptions from the service registries service models. Only those models which declare a valid `semanticDescription` are taken into account i.e. those referenced OWL-S descriptions can be considered for matching.

2.4 Other Tools

Besides the VSDT, the SSM, and its integrated Low-Level Service Finder, the Eclipse IDE in the Nemo Docker Image also contains the JIAC File Server View, which can be used as a very simple client to the Git repositories used for hosting the semantic service descriptions in the NeMo project. This makes interacting with the Git repository particularly easy, as it is not necessary to add, commit, and push the files, and in particular, it is not necessary to have a local copy of the Git repository, as the View uses the Gitlab API.

It allows to add files to the Git repository, either from the workspace or directly from the active editor window, and likewise allows to access and download files from the Git repository. Most importantly, the Git FileServer View will automatically re-write the URI of the OWL-S files, so they are consistent with their new location on the Git server.

Of course, the Git repositories can also be used with any other Git client or using the Gitlab Web interface, but then the URIs have to be adapted manually.

3 Integrating an existing service into NeMo Network

3.1 Involved components

- SSM – Ontology Manager
- SSM – Service Description View

3.2 Workflow

3.2.1 Importing models

The available parameters for all services are described by shared common models, which need to be imported into the SSM. This needs to be done only once and will persist between restarts. To check if your model is already imported, navigate to the *Semantic Service Manager* view and open the *Manage Ontologies* window by clicking the corresponding button.

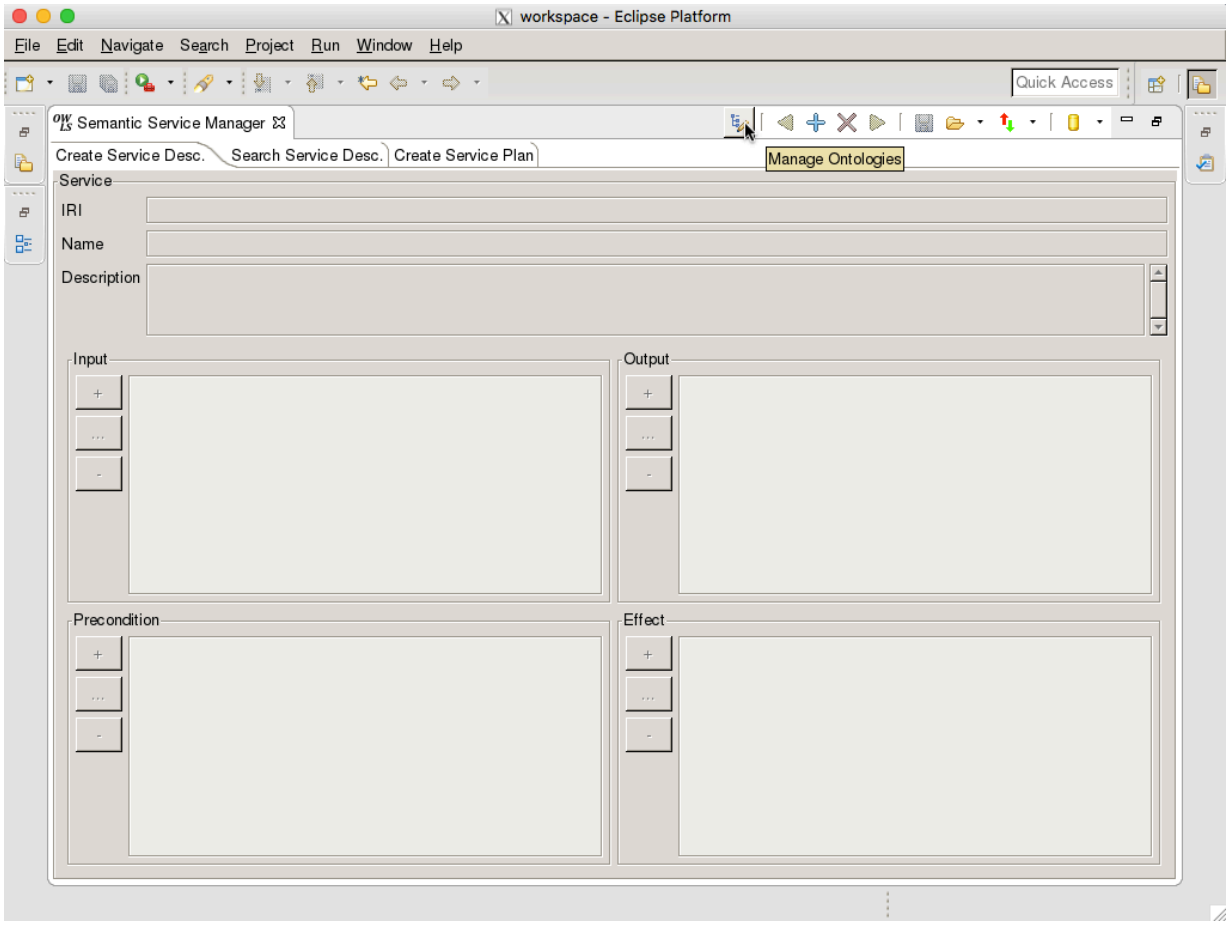

Figure 4: *Semantic Service Manager* with *Manage Ontologies* button highlighted

The next window lists all currently imported models and gives you the ability to remove models or add new remote or local models. For this example we have already added our common information model by adding *http://130.149.154.111:8080/ontologies/cim.owl* as a remote ontology.

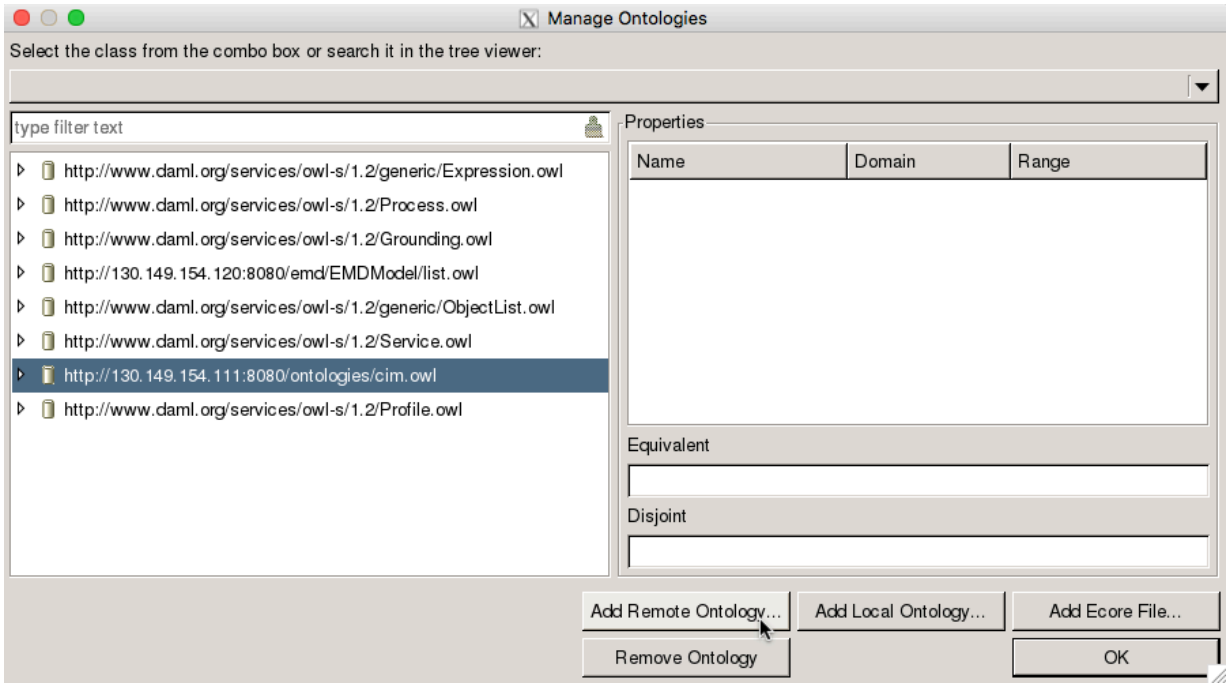

Figure 5: *Manage Ontologies* window with common information model ontology highlighted

3.2.2 Creating a new service

To create a new description of your service, navigate to the *Semantic Service Manager* view and press the *create new service description* button:

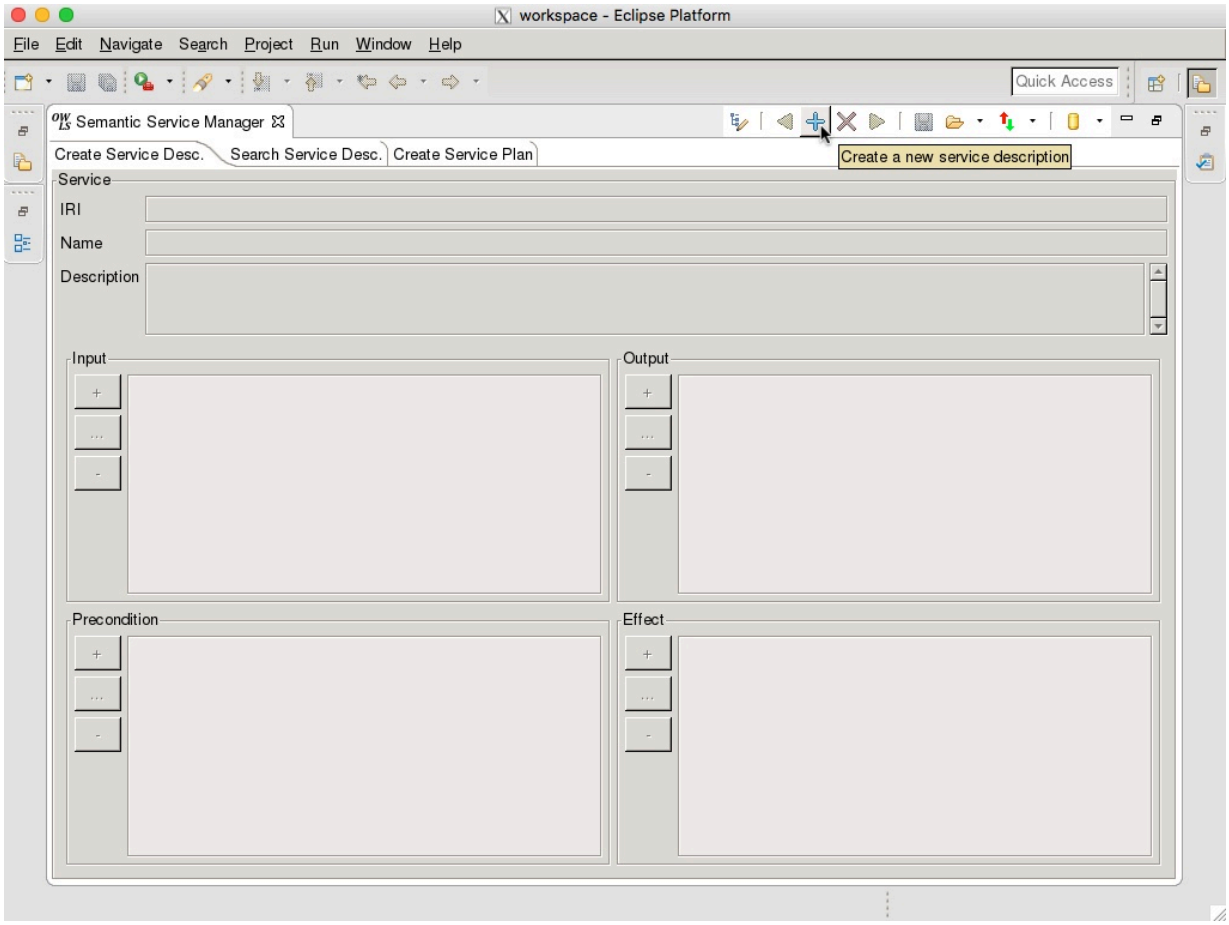

Figure 6: *Semantic Service Manager* with *create service description* button highlighted

Now specify the name of your service and give it a description. Note that the IRI is automatically generated by the SSM and may contain a placeholder server address until the service is actually deployed.

3.2.3 Adding inputs and outputs

Use the buttons next to the *Input* and *Output* fields to add, modify or remove I/O parameters. The *Add Parameter* window requires you to:

1. specify an identifier as a name for this parameter
2. (optional) filter the list for your desired parameter type
3. select the desired parameter type from the list
4. confirm with OK

Repeat this process for every I/O parameter that you want to add to your service.

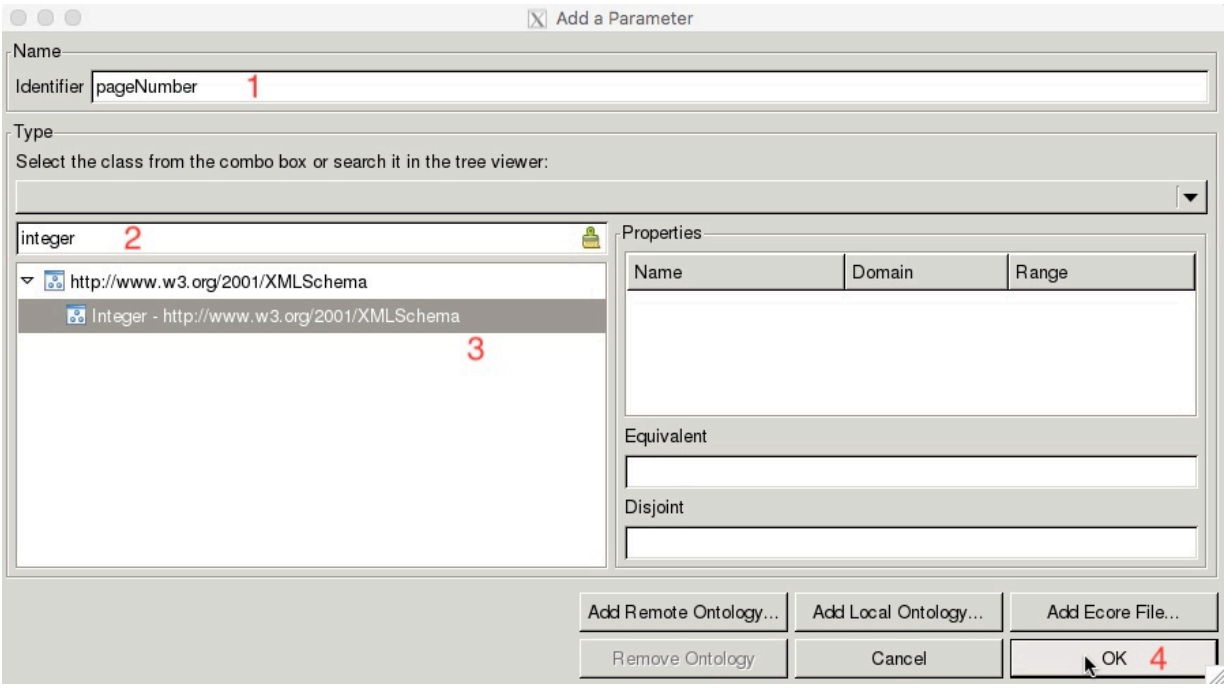

Figure 7: *Add Parameter* window with example parameter

When adding Lists as Parameters, choose *OWLList* as parameter type. After confirming the selection a new dialogue window opens. If you decline the list will not have any type. If you accept you can then select the type of list in yet another window.

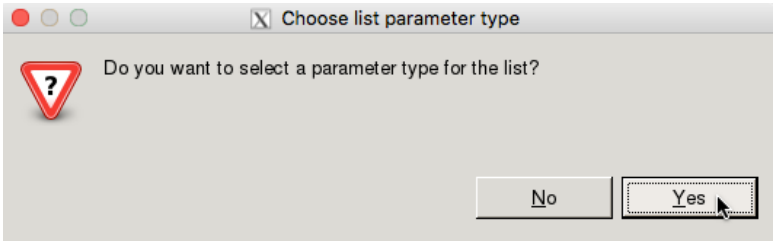

Figure 8: Dialog window when adding *OWLList* as I/O parameter type

3.2.4 Adding preconditions and effects

Preconditions and Effects are added, edited and deleted in a similar fashion to I/O parameters. The necessary buttons are found next to the corresponding fields in the *Semantic Service Manager*.

Preconditions and Effects are expressed in *SWRL Human Readable Syntax*[7]. The *new SWRL rule* window features a simple auto-completion to help you write correct rules. You can trigger the auto-completion with *CTRL-Space*.

[7] https://www.w3.org/Submission/SWRL/#2.2

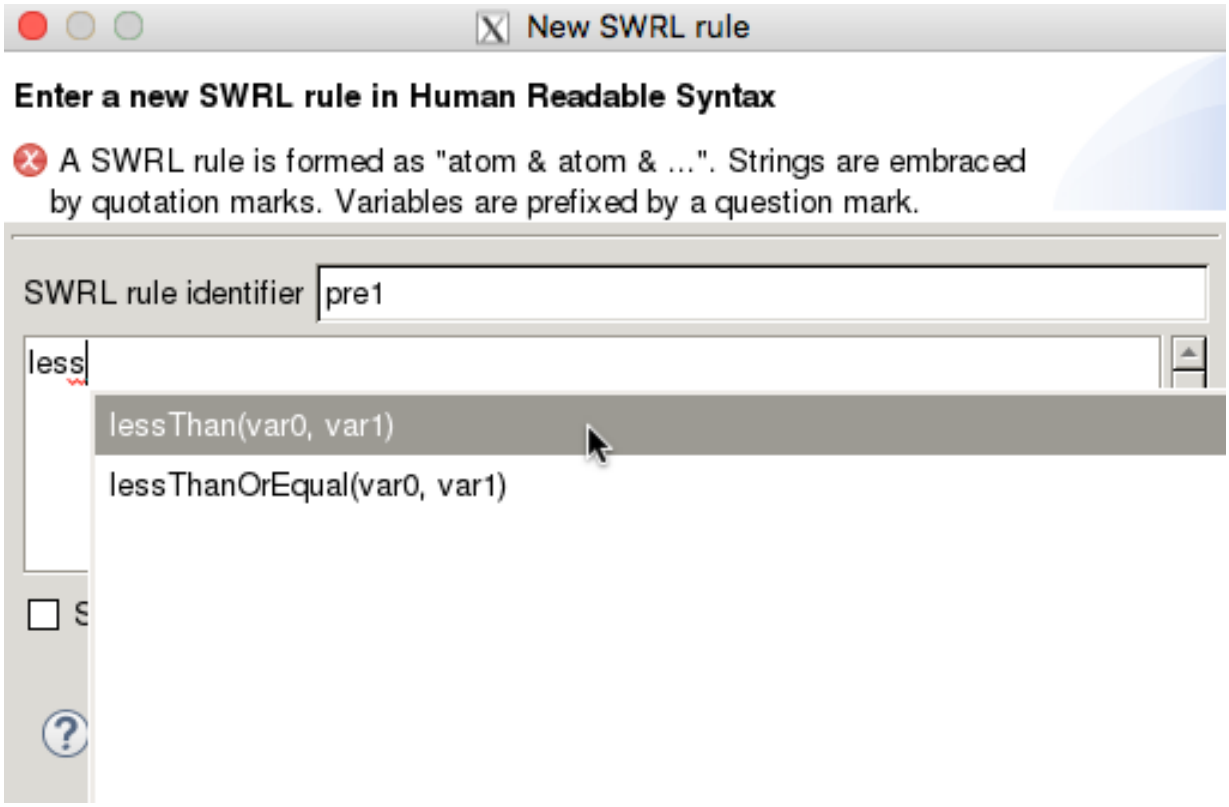

Figure 9: *New SWRL rule* window with autocompletion

I/O parameters can be referenced in any rule by their given identifier, prefixed by a question mark. In this example we have an input parameter of integer type called *pageNumber. We* define a rule that limits the number of pages to less than 3, using *SWRL Built-Ins*[8]:

[8] https://www.w3.org/Submission/SWRL/#8

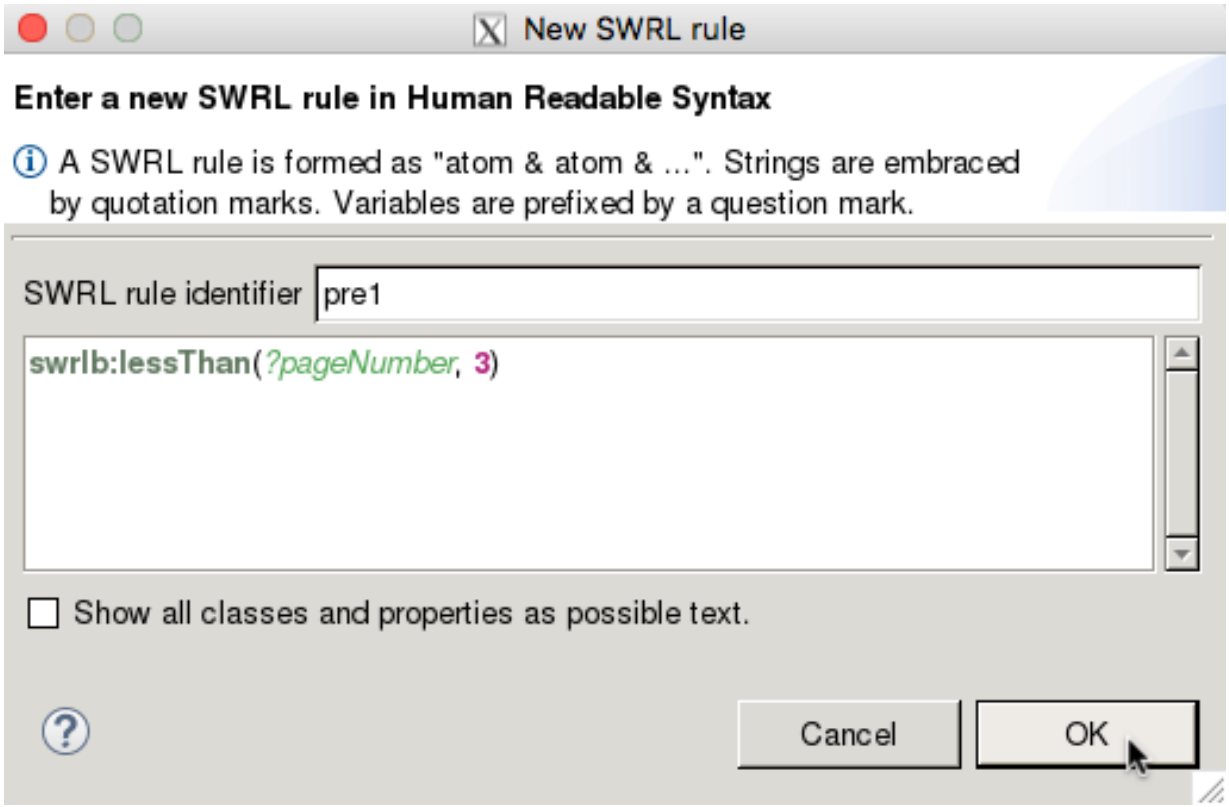

Figure 10: *New SWRL rule* window with complete example rule

Confirm with OK and repeat this process for every Precondition and Effect you want to add to your service.

3.2.5 Deploying to server

Make sure that you have set up the Gitlab server connection information as described in chapter 1.3 before you proceed.

Your completed service should look somewhat similar to our example in the following Figure. To deploy your finished service to the service repository, find the *Up- and Download to Service Repository Server* dropdown menu in the top right of the *Semantic Service Manager*.

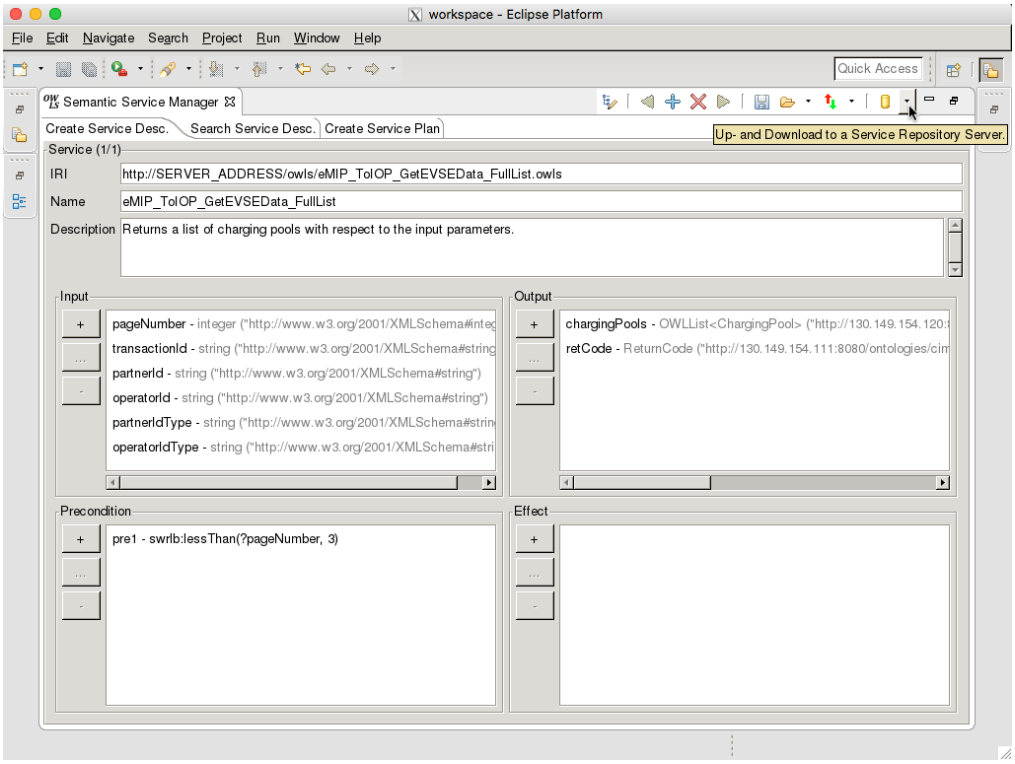

Figure 11: *Semantic Service Manager* with example service and dropdown menu location highlighted

In the dropdown menu, select *Deploy*. The *Semantic Service Manager* now pushes your service to the repository that you have set up beforehand. Depending on your connection this process may take a few seconds. Once it is done a dialogue window will inform you about the status of the operation.

3.2.6 Registering on the registry

After uploading the semantic description to the artefact server, the service itself has to be registered at the NeMo Service Registry. Therefore, NeMo provides an UI for Service Delivery, where the registration process can be performed (see Figure 12). In order to create a link between the registered service and the semantic service description it is important to specify the field *Semantic Description URL* with the URL where the semantic description has been uploaded. The URL can be retrieved either via the *File Server View* (see Section 2.4) in the Creation Environment, where all uploaded files to the artefact server of the respective Service Provider and its URLs can be seen. Alternatively, after uploading the Service Description via the SSM, the field IRI in Figure 9 will change to the concrete IRI where it can be copied from. For more details on the Service Registry process, please have look at the respective manual (https://redmine.iccs.gr/projects/nemo/dmsf?folder_id=3667).

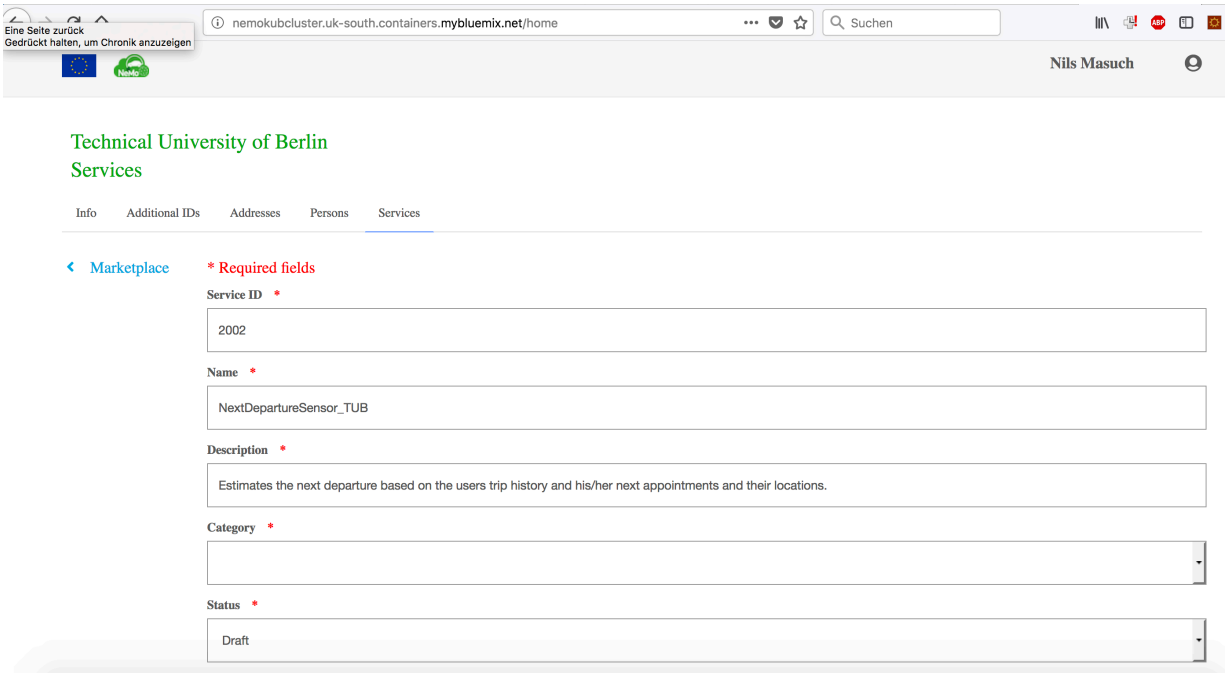

Figure 12: View of the Service Delivery UI

4 Searching for existing NeMo services

4.1 Involved components

- SSM – Ontology Manager
- SSM – Service Search View
- Low-Level Service Finder

4.2 Workflow

4.2.1 Importing models

As in the use case of integrating an existing service to the NeMo environment, in order to search for a service, it is necessary to load the Common Information Model (CIM) of NeMo into the Ontology Manager. Detailed instructions on how to achieve this are given in chapter 3.2.1.

4.2.2 Defining a search template

In this step it is necessary to define the parameters you want to search for within the NeMo Service Registry. To do this navigate to the *Search Service Description* tab within the SSM. This tab lets you define a service in much the same way as described in chapters 3.2.3 and 3.2.4.

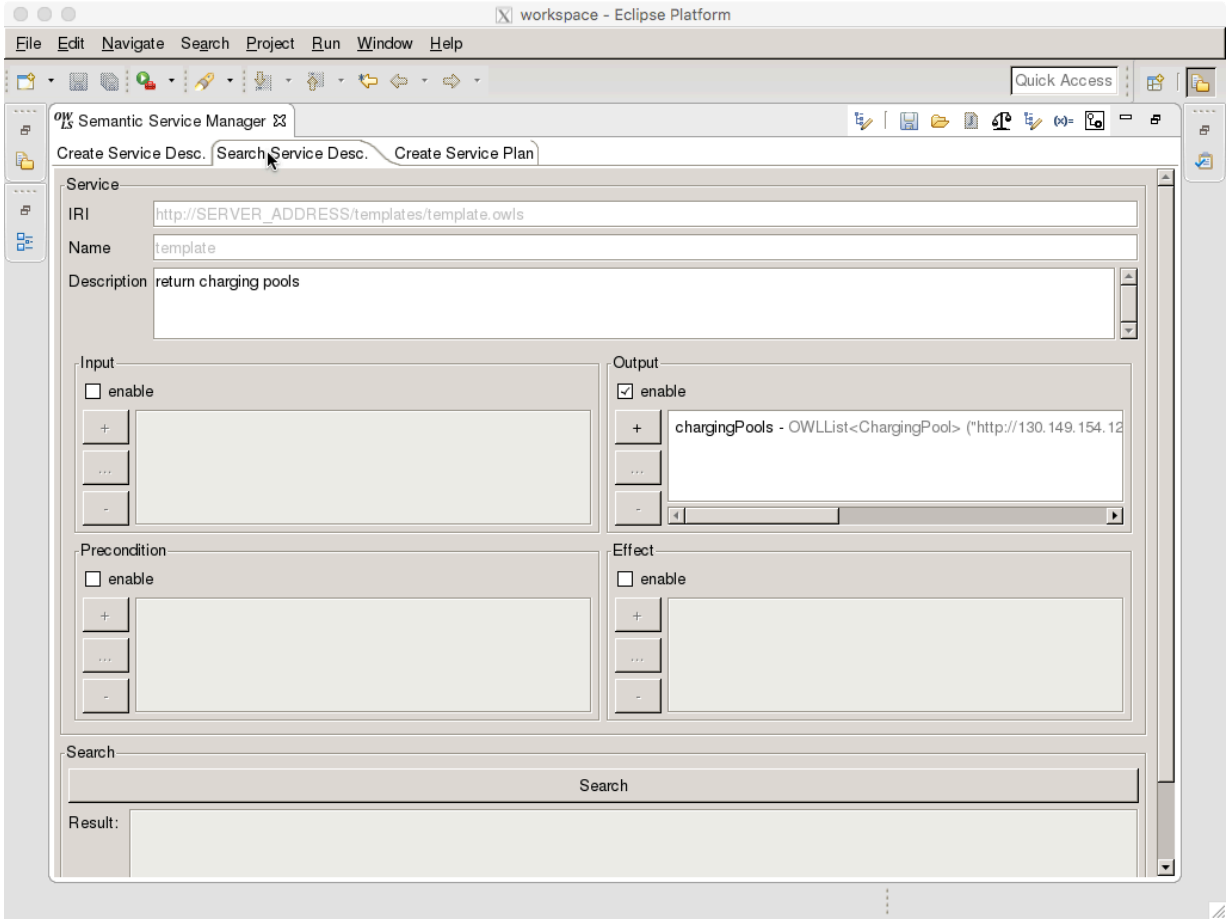

Figure 13: partially filled *Search Service Description* tab

You can specify a name and description as well as inputs, outputs, preconditions and effects you want to search for – or any combination of these. The *enable* checkboxes allow you to exclude a group of parameters from the search entirely. Note that searching with a group empty but enabled will return services without parameters in that group, whereas disabling a group entirely will return all services whether they have parameters in that group or not.

4.2.3 Searching and inspecting results

Once you have specified all desired parameters click on the *Search* button below the *Precondition* and *Effect* fields. The SSM now compares your specified template against all reachable services from the registry and returns a ranked list in the *Result* field below the *Search* button. Selecting a result from this list displays the parameters of that service in the same fields that you used to specify the search template before.

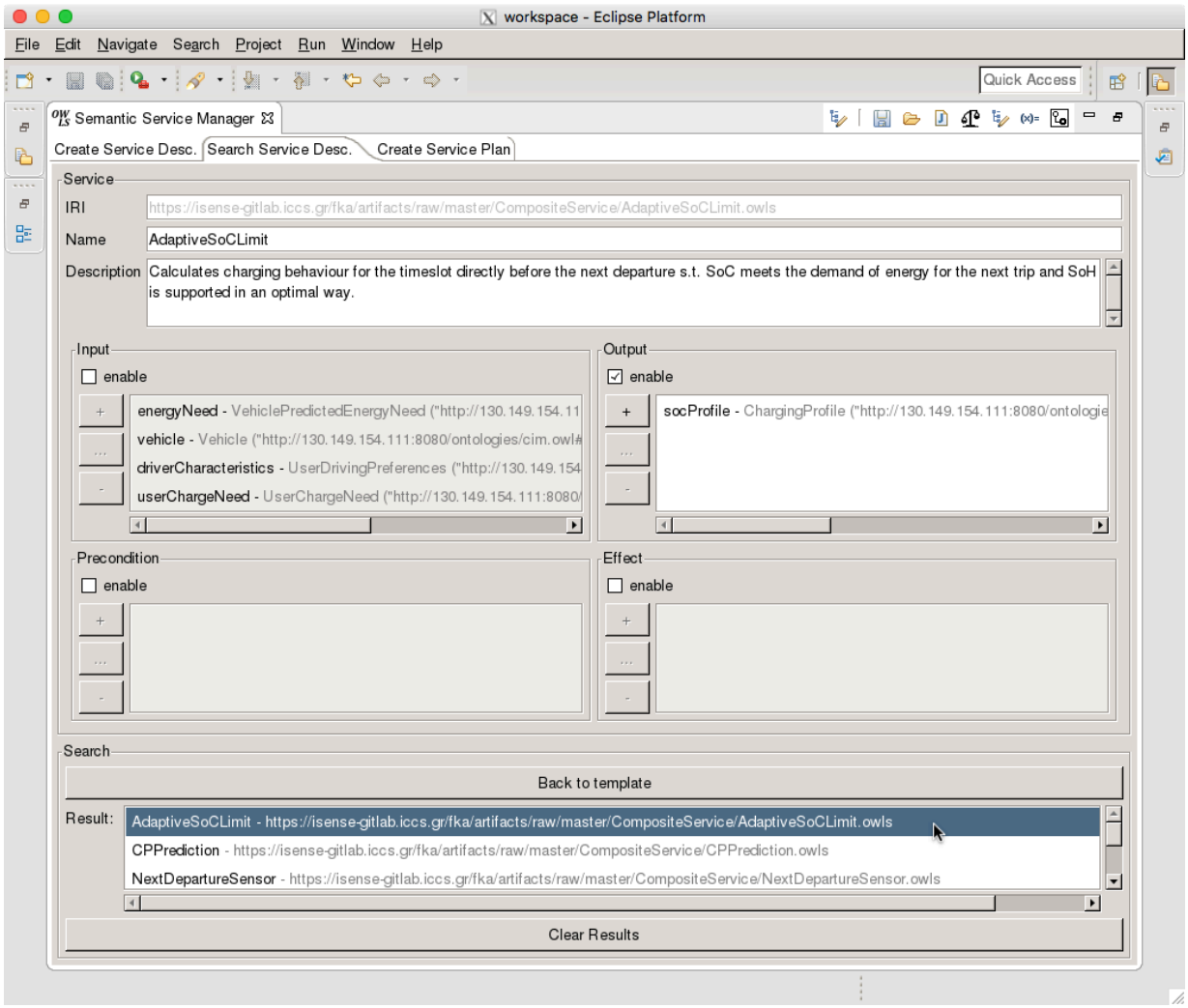

Figure 14: *Search Service Description* tab with first search result selected

If you want to review or edit the search parameters you specified before, a click on the *Back to template* button will restore them for you. The result list will only be updated if you run a new search or click the *Clear Results* button.

4.2.4 Service Integration to Composite Process

Once you have found and selected a suitable service with the previous steps, you can then import this service into your active VSDT by clicking the *Push selected Service Description to* VSDT button.

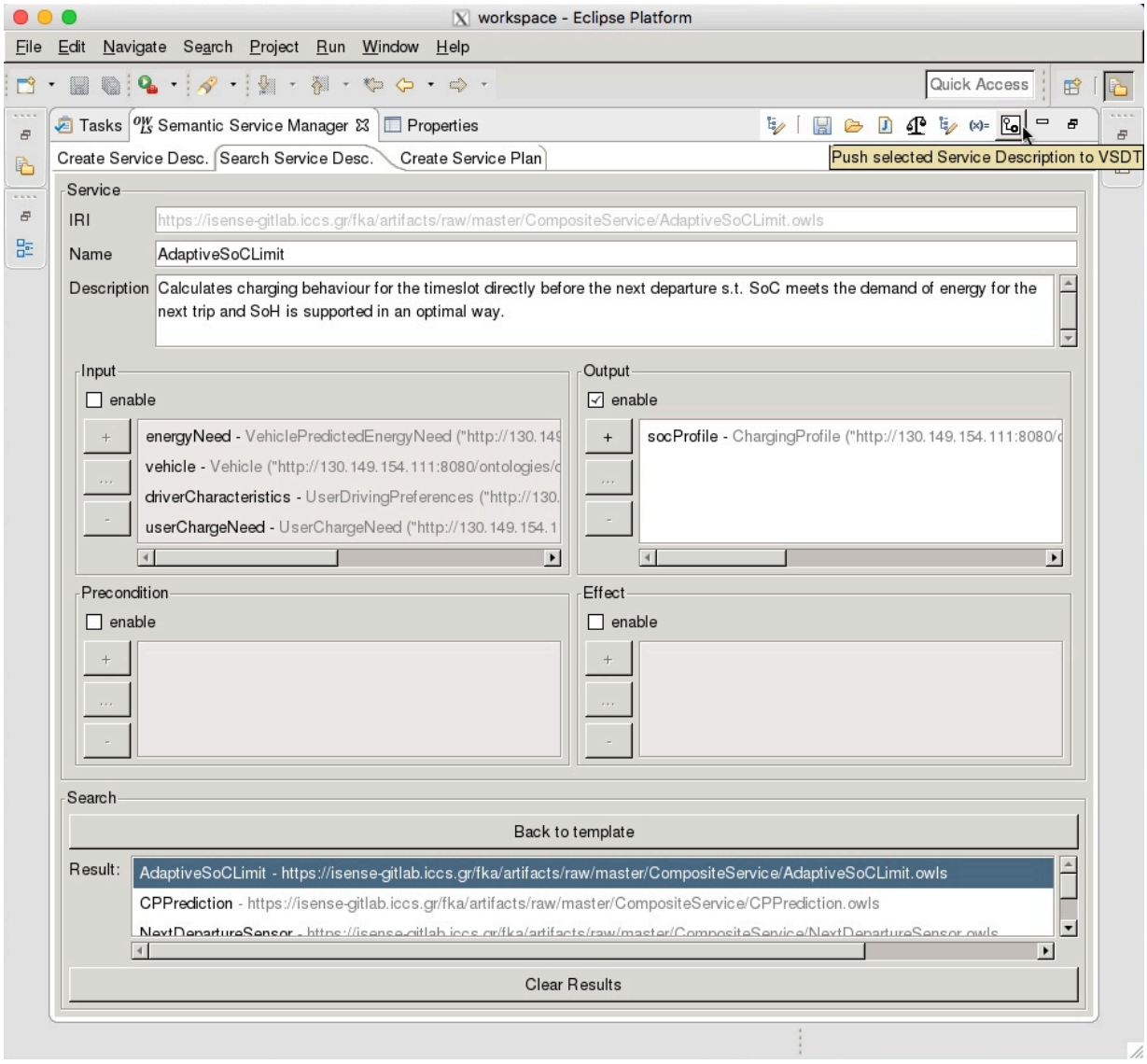

Figure 15: *Search Service Description* tab with *Push to* VSDT button highlighted

Once it is complete a dialogue window will inform you about the status of the operation. If it completed successfully your selected service should now be available within the VSDT. Chapter 5.2.2 includes details on how to include services into a BPMN process.

5 Creating a new composite service with VSDT

5.1 Involved components

- VSDT – BPMN Editor
- SSM – Search view
- SSM – Push selected service to VSDT
- FileServer View to push BPMN process to Artifacts Server

5.2 Workflow

5.2.1 Creating a new VSDT Diagram

First, create a new VSDT diagram. Open a project and select *New → Other → DAI-Labor → VSDT Meta Diagram*, enter a file name and meta information like description, author, etc. If you want the process to be executed (i.e. not purely for documentation) also check the accordant check box on the last wizard page.

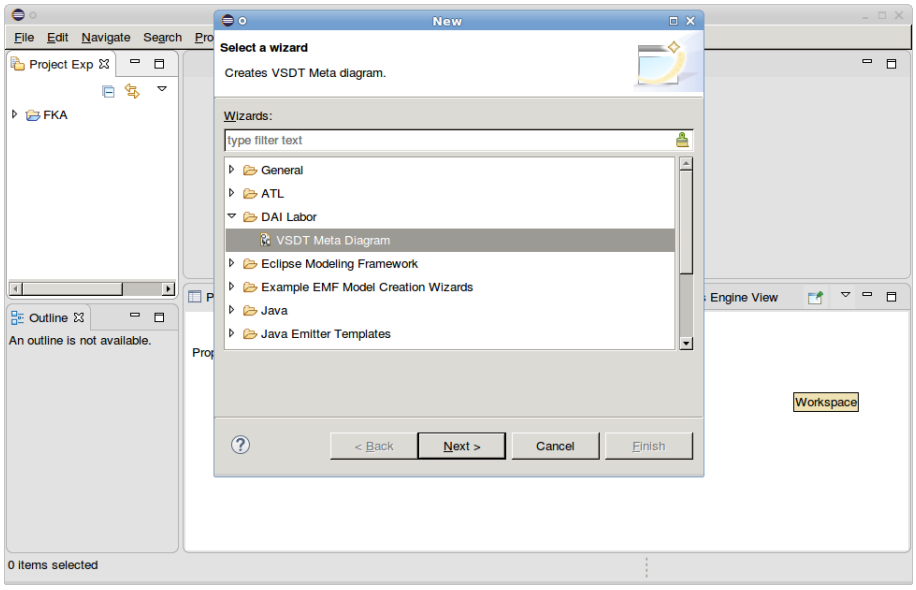

Figure 16: Dialogue for creating a new VSDT diagram

After opening the file, you will see the use-case view (sometimes also referred to as the “meta diagram”), where you can create diagrams similar to UML use case diagrams, where each use case represents a BPMN diagram, and each actor involved in those use cases represents a participant of one of the pools in that BPMN diagram. Double-clicking on the use-case will take you to the actual BPMN editor (make sure to target the actual node, not the label).

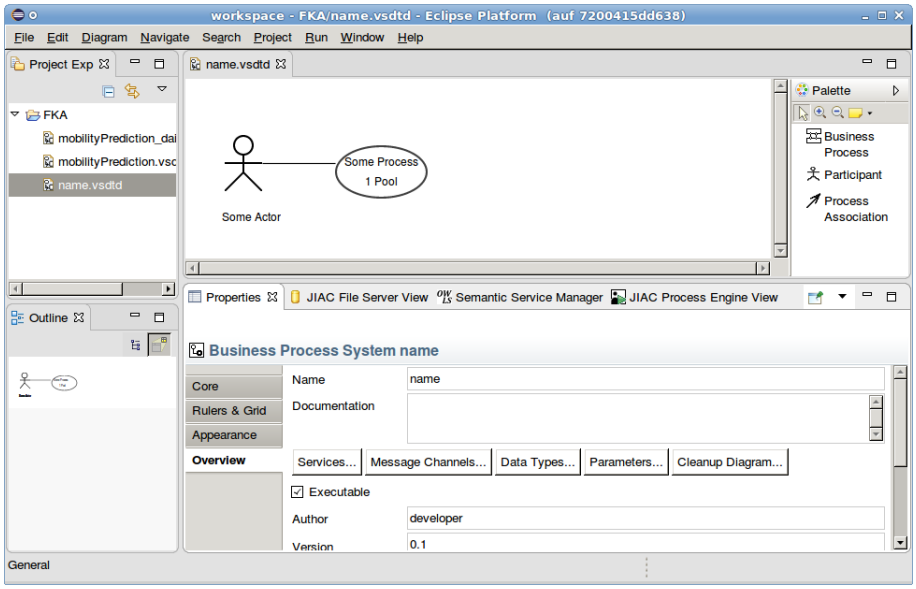

Figure 17: Use-case view

Make sure to save the use case diagram before opening and editing the diagrams for the individual processes, and also to save each of those diagrams before continuing editing the process for a different use case, or the use case diagram itself, otherwise the changes will overwrite each other. (You do not have to close and re-open the diagrams after editing saving a different diagram, though.)

We will not go into detail on how to use the VSDT or how to create the individual BPMN diagrams here. For general information on how to use the VSDT for modelling BPMN processes, please refer to the VSDT manual[9], and for general information about BPMN, refer to the specification[10] or other guides and resources on the standard.

5.2.2 Importing Services from the SSM into the VSDT

In this step, we describe how to use service descriptions from the Semantic Service Manager in the VSDT. This step does not have to be done strictly after the process has been finished; it can be done right after creating the process diagram, or at various times in between. Here, service descriptions from the SSM are used in two places: For services to be orchestrated in the process, but also for describing the process as a whole.

5.2.2.1 Searching and Importing Services to be Orchestrated

First, search for the service you want to import into the VSDT as described in Section 4.

Then, with the VSDT diagram opened in the active editor window, click the button “Push selected service description to VSDT” in the top-right corner of the SSM. You can ignore any warnings about unresolved ontologies. You should see a confirmation dialogue and the diagram should now be “dirty”, i.e. have unsaved changes.

[9] http://jiac.de/Downloads/toolipse/vsdt-manual.pdf

[10] https://www.omg.org/spec/BPMN/2.0/PDF

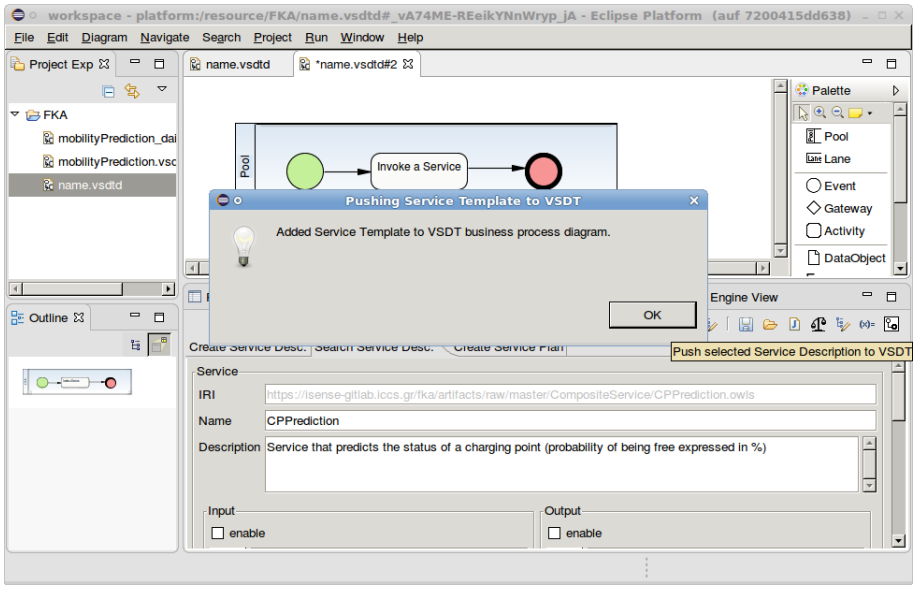

Figure 18: Confirmation window after pushing a *Service Description* to the VSDT

In the Properties View of the diagram (available when selecting the diagram background), click the *Services* button to find the newly imported service. Check the name of the service and particularly the inputs and outputs to see if everything is in order.

Now, you can use this service for an activity. Select the activity that should call that service and set its Activity Type to “Service” in the Properties View and select the newly added service as the “Implementation” in the “Task Attributes” group.

Finally, assign the parameters to the service invocation using the *Parameter Assignments* dialogue. Note that there can be problems if parameters of the service have exactly the same name as properties of the process. In this case, you can safely change the parameters names in the imported service, e.g. by adding a suffix like “`input_`” to each of them, without affecting the service being called.

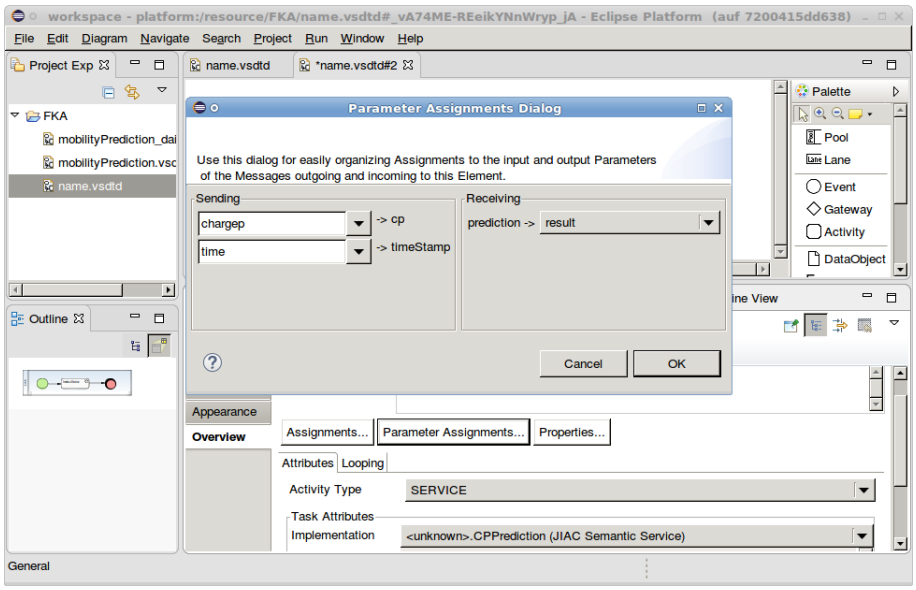

Figure 19: Parameter Assignments Dialogue of a service

5.2.2.2 Creating and Importing a Service for the Process as a Whole

The process as a whole is also a service, and should thus also be described in the SSM so it can be invoked or re-used in other processes.

For this, create a new service description as explained in Section 3. In particular, the input and output of that service should match the input and output that the process does need or provide respectively. When the service description is ready, again press the button labelled "Push selected service description to VSDT", which can be found in the menu next to the red/green arrows symbol. Also remember to deploy the service to the Artifact Server.

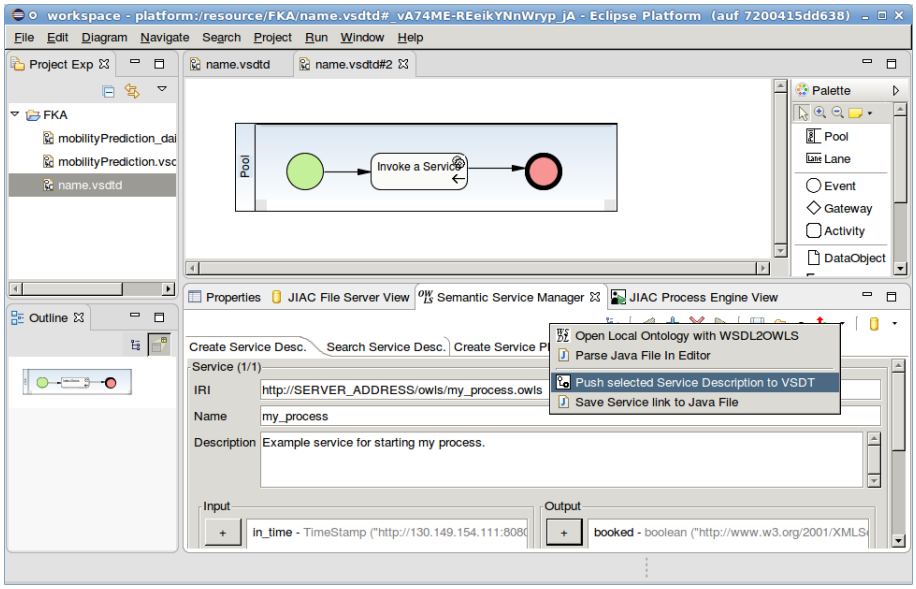

Figure 20: How to push a newly created *Service Description* to the VSDT

In the VSDT, open the Services dialogue and assign the newly created service to the Participant of the Pool representing that process. In the Pool, select the Start Event and set its Trigger Type to "Message" and its Implementation to the newly imported Service. (The process needs to have a Message Start Event in order to be callable as a service, but the same process might have additional Start Events, e.g. with a Timer, to be triggered in multiple ways.) If the service has any output parameters, the Pool will also need an accordant Message End Event with the same service as its Implementation.

Finally, use the Parameter Assignments dialogue on the Message Start- and End-Events to assign the input parameters to the properties of the process, and analogously for the output parameters. Note that other than when invoking services, here the input parameters will be *incoming* to the process, and vice versa.

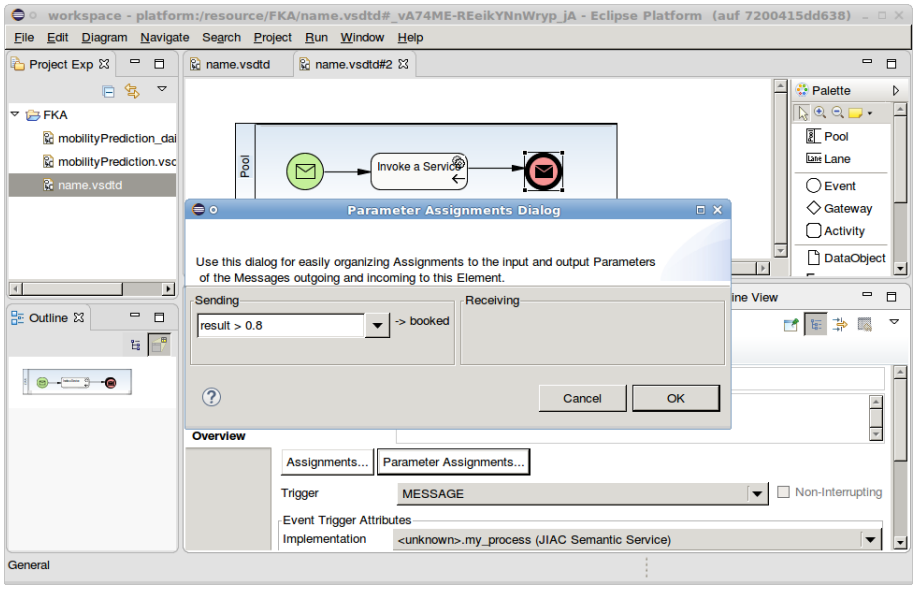

Figure 21: Parameter Assignments Dialogue of a message

At the moment, it is not possible to export a service described in the VSDT to the SSM for further refinement and for uploading it to the Artifact Server. So if you already defined the service for the process as a whole directly within the VSDT, you will have to replace that with the version imported from the SSM. In this case, also remember to fix any parameter assignments involving the properties of the old service. Similarly, if at a later time you need to update the service description in the SSM, you will have to re-import it into the VSDT, delete the old service, and adapt the parameter assignments accordingly.

5.2.3 Uploading the Process onto the Artifacts Server

When the process is ready, you can upload it to the Artifact Server. There are two ways of doing this: Either, you can use the “Store file…” button in the JIAC Process Engine View, or the “Add file…” button in the JIAC File Server View. The first option is a bit more comfortable, as it requires only a single click, but the latter option is more versatile and also allows specifying the exact path where to store the file on the server. Thus, we recommend using the button in the File Server View.

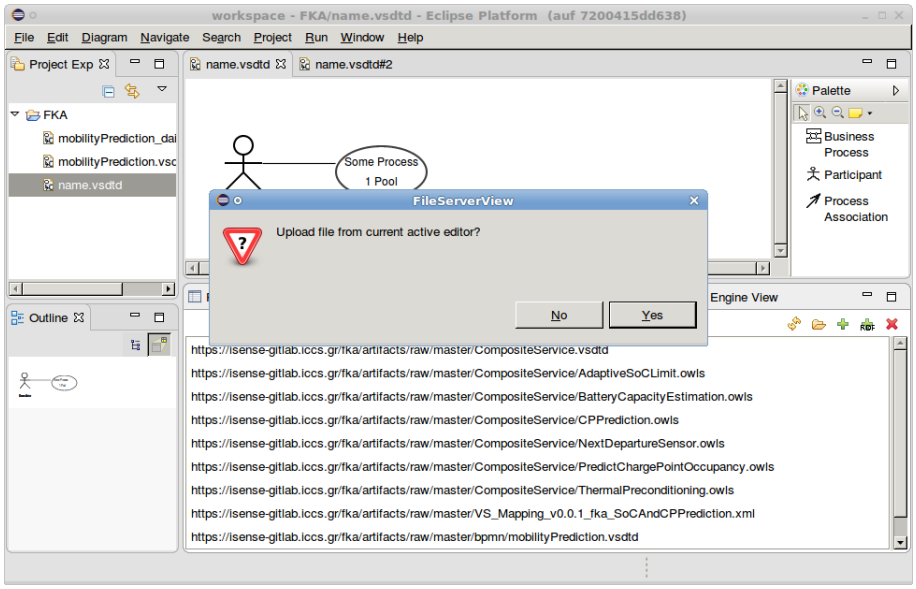

Figure 22: Confirmation window when uploading through the *File Server View*

Click the “Add file…” button (but not the “Add RDF file…” button next to it). If the VSDT diagram file is currently opened in your editor, you will be asked whether you want to upload it directly (note that this only works with the use case view, not with the BPMN view of the editor), otherwise you can select the file to upload. Next, enter the path (if any) and file name of the diagram to be uploaded; do not specify the full Gitlab URL, that will be added automatically. After a short delay, the new file should appear in the File Server View and you can check that everything went well by copying the URL (right click) and opening the file in a browser, or opening it directly in Eclipse, you will only see the file’s XML source code, though. To see the BPMN diagrams, download the file and put it into your workspace (possibly under a different name) and open it with the VSDT editor.