



SWWS – Semantic Web Enabled Web Services

Title: D12.3 BT Case Study: Demonstration Description and Architecture

Version: 1.1
Date: 16/09/2004
Pages: 15

Responsible Authors:
Alistair Duke, Marc Richardson

Co-Author(s):

Status:

- Draft
- To be reviewed
- Proposal
- Final / Released to CEC

Confidentiality:

- Public - for public use
- INT - for SWWS consortium (and Project Officer if requested)
- Restricted - for SWWS consortium and Project Officer only


Project ID: IST-2002-37134

Deliverable ID: None

Title: D12.3 BT Case Study: Demonstration Description and Architecture

Summary / Contents:

This Document describes the BT Case Study demonstration and outlines the architecture used in its implementation.

	BT Case Study Architecture Document Deliverable ID:	Page : 2 of 15
		Version: 1.1 Date: 16/09/2004
		Status: Proposal Confid.: Public

SWWS Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2002-37134. The partners in this project are: Leopold-Franzens Universität Innsbruck (IFI, Austria)); National University of Ireland, Galway (NUI, Galway, Ireland); Forschungszentrum Informatik (FZI, Germany); Intelligent Software Components S.A. (iSOCO, Spain); OntoText Lab. - Sirma AI Ltd. (SAI, Bulgaria); Hewlett Packard (HP, UK), British Telecom (BT, UK)

Leopold-Franzens Universität Innsbruck (IFI)

Institut für Informatik
Technikerstrasse 13
A-6020 Innsbruck Austria

Tel: +43 512 507 6489
Fax: +43 512 507 9872

Contact person: Juan Miguel Gomez
E-mail: juan.miguel@uibk.ac.at

National University of Ireland, Galway (NUI)

National University of Ireland,
University Road
Galway, Ireland

Tel: +353 91 750414
Fax: +353 91 562894

Contact person: Liam Caffrey
E-mail: Liam.Caffrey@nuigalway.ie

FZI – Forschungszentrum Informatik

Haid-und-Neu-Str. 10-14
76131 Karlsruhe, Germany

Tel: +49 721 9654816
Fax: +49 721 9654817

Contact person: Adreas Abecker
E-mail: abecker@fzi.de

Intelligent Software Components S.A. (iSOCO)

Francisco Delgado 11, 2nd Flor
28108 Alcobendas, Madrid, Spain

Tel: +34 913 349797
Fax: +34 913 349799

Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

OntoText Lab.- Sirma AI Ltd. (SAI)

OntoText Lab.
38A Chr. Botev Blvd.
Sofia 1000, Bulgaria

Tel: +35 92 9810018,
Fax: +35 92 9819058

Contact person: Atanas Kiryakov
E-mail: Atanas.Kiryakov@sirma.bg

Hewlett Packard (HP)

HP European Laboratories
Filton Road, Stoke Gifford
BS34 8QZ Bristol, UK

Tel: +44 117 3128631
Fax: +44 117 3129285

Contact person: Janet Bruten
E-mail: janet.bruten@hp.com


Associated Partner:

British Telecommunications plc. (BT)

Orion 5/12, Adastral Park
Ipswich ip5 3RE, UK

Tel: +44 1473 609583
Fax: +44 1473 609832

Contact person: John Davies
E-mail: john.nj.davies@bt.com


	BT Case Study Architecture Document Deliverable ID:	Page : 3 of 15
		Version: 1.1 Date: 16/09/2004
		Status: Proposal Confid.: Public

Change Log

Vers.	Date	Author	Description
1.0	06/07/04	Alistair Duke	First version
1.1	06/08/04	Alistair Duke, Marc Richardson	UML Diagrams added

Table of Contents

1	INTRODUCTION.....	4
2	ARCHITECTURE.....	4
2.1	UML Class diagrams of the components.....	5
2.1.1	Browser.....	5
2.1.2	Composition.....	7
2.2	Relationship to SWWS Technical Architecture	8
3	DEMONSTRATOR DESCRIPTION	10
4	NEXT STEPS.....	14
	REFERENCES.....	15

	BT Case Study Architecture Document Deliverable ID:	Page : 4 of 15
		Version: 1.1 Date: 16/09/2004
		Status: Proposal Confid.: Public

1 Introduction

The BT Case Study is focusing upon a scenario where a designer must find and compose Service components that will satisfy a particular process which in this case is problem handling. In the scenario, a number of Service components exist. These are based around OSS/J (Operational Support System through Java) initiative which attempts to standardize interfaces for OSS. OSS/J has been used in the Case Study to provide a set of realistic services at the correct level of granularity. The OSS/J interfaces have been wrapped as WSDL Web Services and further described semantically using OWL-S.

OWL-S enables the Services to be described in three ways. Firstly, the Services are categorized according to the eTOM Process Framework. eTom (enhanced Telecom Operations Map) is an attempt by the TeleManagement Forum to enable OSS processes to be described in a common way. Secondly, the Services are described according to their data requirements. A simple data ontology has been created which describes the entities and associated data elements. This ontology will be linked to the SID (Shared Information / Data Model) which is the TMF's data model. Thirdly, a process model for the scenario has been created. This allows preconditions and postconditions to be attributed to the Services which relate to the process model.

2 Architecture

The architecture for the case study is based upon BT's Semantic Web Services Browser which is a tool to aid in the discovery and composition of Web Services. The main components of the Browser and their interactions are shown in the Figure 1.

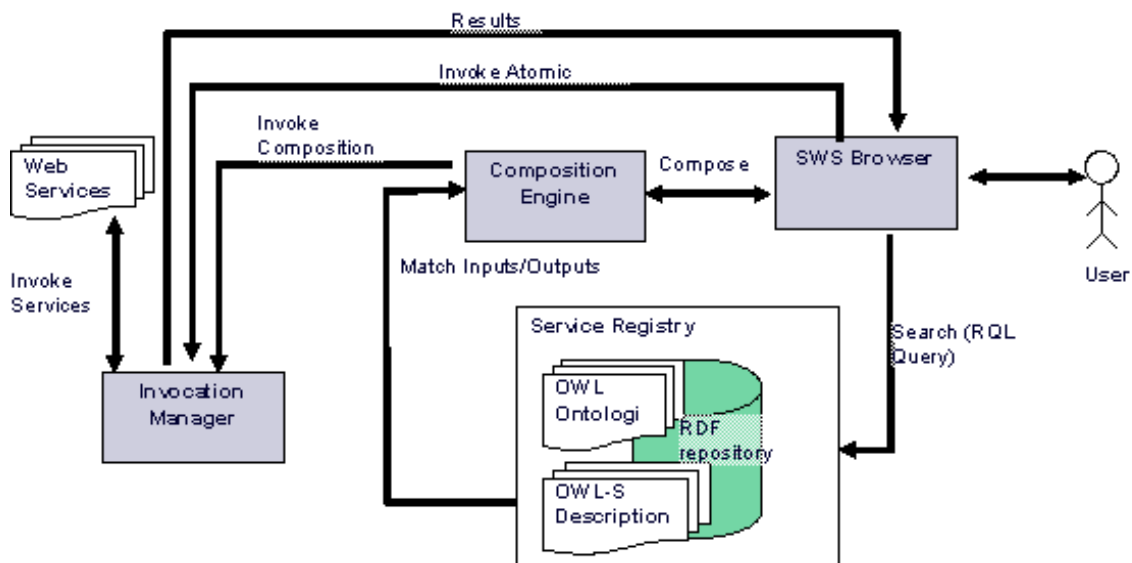



Figure 1. SWS Browser Components

The SWS browser works with Web Services described in OWL-S and stored in a Sesame-based [1] RDF repository. The browser uses a specific domain ontology of Web Services

	BT Case Study Architecture Document Deliverable ID:	Page : 5 of 15
		Version: 1.1 Date: 16/09/2004
		Status: Proposal Confid.: Public

Categories. The classes in the domain ontology define high-level categories for Web Services. Each category allows a number of keywords to be associated with it. This enables matching of Web Services based on keywords in their description. The browser also enables a client to browse the ontology hierarchy and select a category. It will then search for and display all services that are related to this. The client can then choose to execute this service at which point the browser will prompt for any required input before invocation. If an appropriate atomic service cannot be found, the user is able to combine several services in order to perform the desired action.

Having selected one Service, the user can choose to base a composition around this. The Browser offers the facility to combine Web Services so that the data output of one service can be fed into the input of another, thus creating a new composite Web Service. Currently, the Browser assumes that the data types of these inputs and outputs are the same. More realistically, a mediation function would be required to convert between differing data types. The Browser provides a graphical view of Web Service composition. The user is able to select the input of a particular service and search for a related service that can provide the required data. The Browser will search for services that have outputs that have been described using the same ontological concept as that attributed to the input.

2.1 UML Class diagrams of the components

This Section outlines the classes associated with the Browser's two main functions. Firstly the initial browsing of the category tree and selection of the desired service, and secondly the composition of this with further services, followed by the final invocation of the composed service.


2.1.1 Browser

The Browser Component is the interface provided to the user that allows them to browse a hierarchical tree of concepts or "categories" that can be associated with web services. In our case study we use the eTOM process framework as a basis for this, but in other scenarios this would be based on an ontology/taxonomy associated with the problem domain. The category tree is stored as an OWL ontology in a sesame repository, and uses the subclass relationship to derive a hierarchical tree.

The CategoryTreePane class (see Figure 2), accesses the sesame repository via the sesameQueryManager to obtain the category tree which it then displays in a pane in the Browser. Its main function then is to inform the BrowserPane class when a user selects a specific category. The BrowserPane class is responsible for fetching and displaying information on the Web Services stored in the repository. When a user selects a specific category it will query the sesame repository via the sesameQueryManager to obtain all services associated with that category. The Service class is used to store an internal representation of the Web Service, from information obtained from the repository. Once these services are displayed in the Browser, the use can select one and perform one of two main functions that the user. These are to

- directly invoke the service. This will present a form to allow inputs to be entered, which are then passed to the InvocationManger (shown in Figure 4)
- create a composition using the selected service as a starting point. This will launch the composition component.

As well as Browsing the CategoryTree to find services, there are also two types of searches than can be performed.

	BT Case Study Architecture Document Deliverable ID:	Page : 6 of 15
		Version: 1.1 Date: 16/09/2004
		Status: Proposal Confid.: Public

- searchCategory will search the category tree for matching keywords and display associated services. As each category is an OWL class you can also associate a text description with it, which will also be searched with this function.
- searchFreeText will search the Web Service OWL-S profiles and return any services with matching keywords

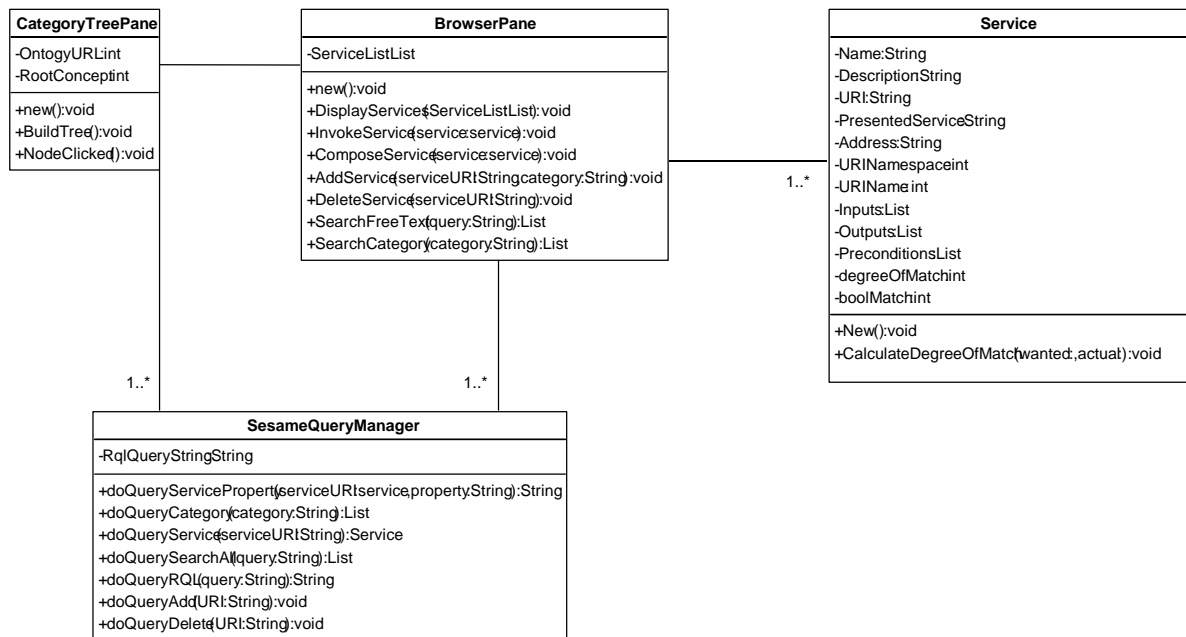


Figure 2. UML Class Diagram of Browser Component

The UML Sequence diagram in Figure 3 shows the sequence of operations involved with initialising the browser and using it to select a service to compose.

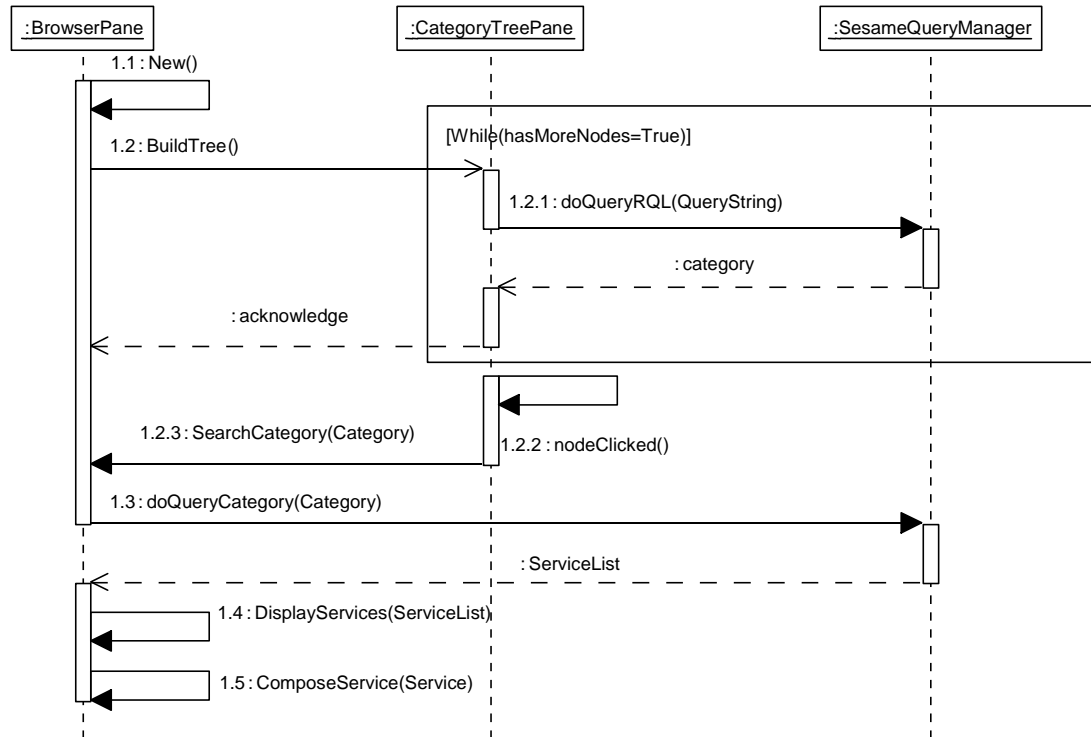


Figure 3. UML Sequence diagram of Simple Browser Operation


2.1.2 Composition

The composition component provides the facility to combine Web Services by identifying semantically equivalent inputs and outputs of different services and allowing them to be linked. This composition can then be invoked.

Figure 4 shows the classes involved with composition. The compositionGraph class is responsible for storing the internal representation of the composition as a directed graph. The model used for composition in the browser at present is simplistic allowing inputs and outputs to be connected, with no specific process flow modelling.

Given a specific starting service the user then selects an input from the service. The ServiceTable class will then use SesameQueryManager to find all services that have any semantically equivalent outputs. This assumes that all inputs and outputs of services have been linked to concepts in a common ontology. In our case study we used the SID data model as our common data ontology. It then displays information about these services in a table, allowing the user to select a service and add it to the composition. The matching input/output will then be linked in the compositionGraph. If no services are found with a matching output then there is the option to add input manually.

A composition can be invoked when all inputs are either fed from the output of another service, or the user has selected the enter input manually. When this is the case the option is presented to invoke the service. The CompositionManager is responsible for orchestrating the invocation of the composition. The CompositionManager maintains a table of inputs and outputs for each service, and handles the passing of outputs from one service to the inputs of another. It follows a simple iterative process of analysing the list of services to see which is

	BT Case Study Architecture Document Deliverable ID:	Page : 8 of 15
		Version: 1.1 Date: 16/09/2004
		Status: Proposal Confid.: Public

ready to be invoked (i.e. has all the required inputs). It then passes this information to the InvocationManger which actually invokes the concrete WSDL service, with the supplied inputs. The InvocationManager then passes the outputs back to the CompostionManager, which then updates its table of inputs and outputs and looks for the next ready service. Eventually all the atomic services will be invoked and the invocation of the composed service will be complete.

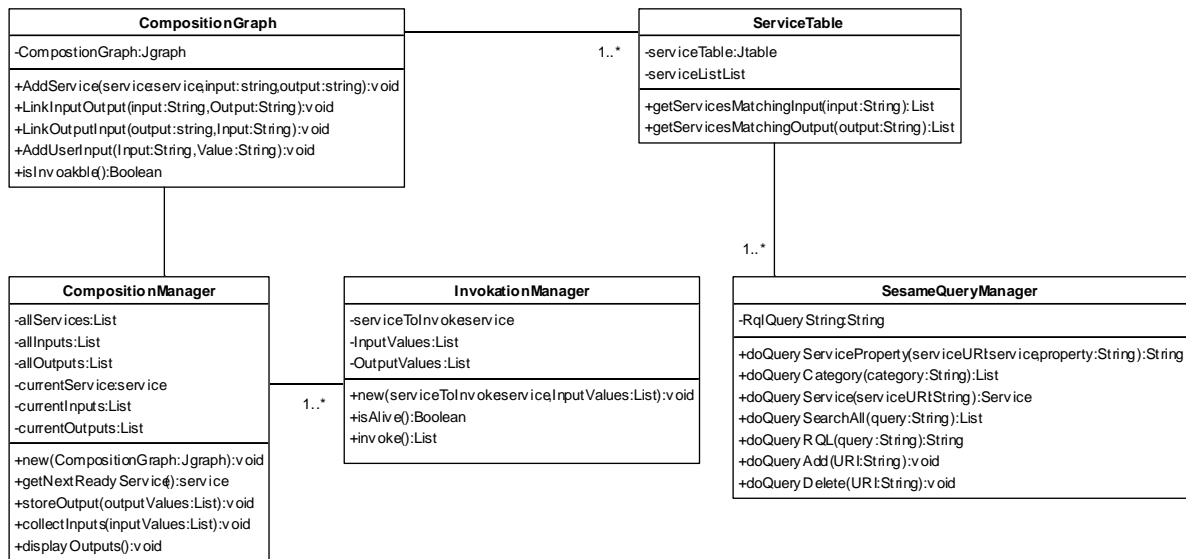


Figure 4. UML Class diagram of composition Component

The UML Sequence diagram in Figure 5 shows the sequence diagram of the operations involved with creating a composition and invoking it.

2.2 Relationship to SWWS Technical Architecture

This section relates the Case Study demonstrator to the SWWS Technical Architecture [3]. Two forms of discovery are carried out in the demonstrator. The first is a browsing activity where the user (in this case the designer) retains the discovery goal in their head rather than formally stating it as is the case in the SWWS TA. The designer is able to make use of the ontology of processes (i.e., the eTOM) in order to assist them in discovery. The second form of discovery is during composition when the user makes a request to find services that are related to a particular data input of a selected service. This form is more closely related to the SWWS TA since a simple ontological query is constructed. Following this the TA considers additional aspects such as matching based upon choreography and mediation that are not addressed in the demonstrator.

The composition part of the demonstrator aligns with the Composer/Orchestration tool of the TA in that it allows a simple workflow to be defined and then invoked.

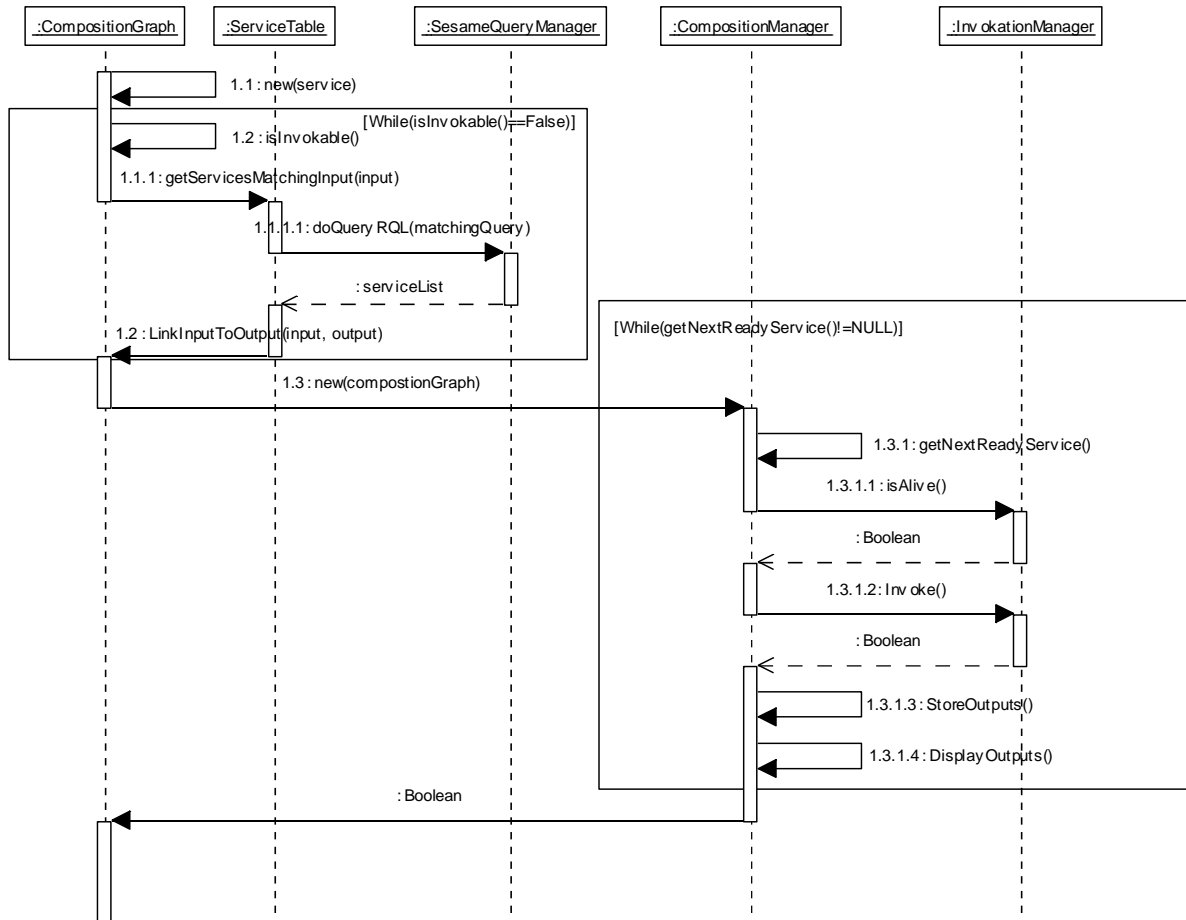



Figure 5. UML Sequence diagram of Composition

	BT Case Study Architecture Document Deliverable ID:	Page : 10 of 15
		Version: 1.1 Date: 16/09/2004
		Status: Proposal Confid.: Public

3 Demonstrator Description

The case study scenario has been described in Deliverable 12.2 [3]. This section will take a part of the scenario and describe how this will be illustrated using the SWS Browser. Figure 6 shows a portion of the storyboard. In this portion, a network alarm is triggered by some network hardware. This is detected by a message being sent to the Process Manager. The Process Manager can be considered to be the element that maintains the state of the process. It is aware of the state model (see Figure 24, D12.2) and conditions the must be met in order to effect state transitions. These transitions result in messages being sent to other appropriate entities. The Process Manager extracts the resourceID from the message which identifies the problem hardware. It then contacts the Inventory Manager to determine the (Telecoms) service that is affected and the customer(s) who are using the service. Having gained this information, the Process Manager can request that a Trouble Ticket is created on the Trouble Ticket system. Once this is done, the Process Manager is informed and this part of scenario is complete (the subsequent step is that the Trouble Ticket is picked up by a network administrator).

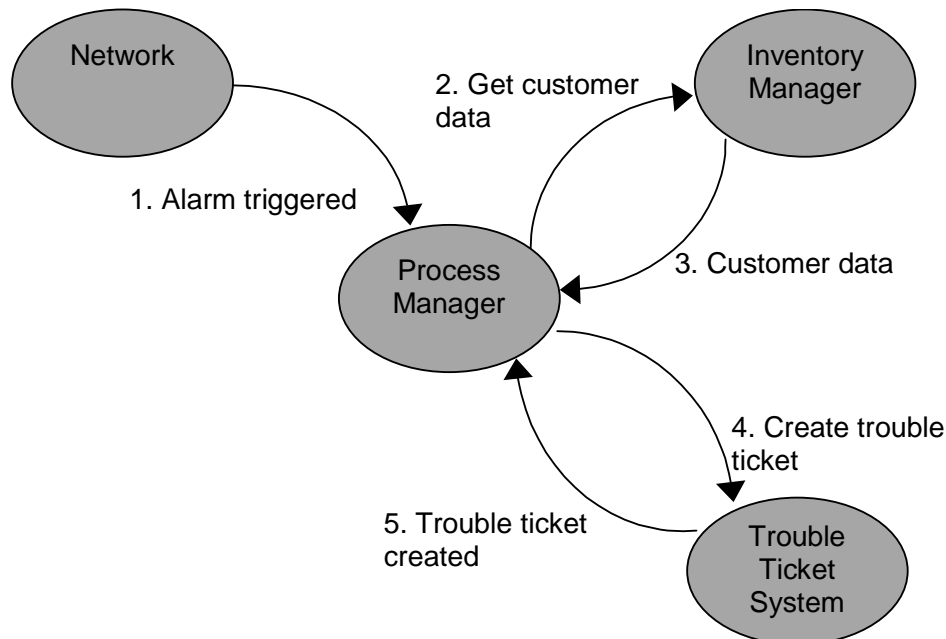


Figure 6. Alarm is triggered and a Trouble Ticket is created

The following are Services that are used to create the composition:


Service: handleAlarm

Description: Receives alarm and provides the affected resourceID

Input(s): alarm

Output(s): resourceID

Service: getServiceFromResource

	BT Case Study Architecture Document Deliverable ID:	Page : 11 of 15
		Version: 1.1 Date: 16/09/2004
		Status: Proposal Confid.: Public

Description: Searches the Inventory for Services using a particular Resource

Input(s): resourceID

Output(s): serviceID

Service: getCustomerFromResource

Description: Searches the Inventory for Customers using a particular Resource

Input(s): resourceID

Output(s): customerID

Service: createTT

Description: Creates a blank Trouble Ticket

Input(s): none

Output(s): troubleTicketID

Service: populateTT

Description: Add details to a Trouble Ticket

Input(s): troubleTicketID, serviceID, resourceID, Description

Output(s): troubleTicketID, serviceID, resourceID, Description

The Inventory Manager Services are dummy components (based upon simple WSDL wrapped Java Services) that return fixed customer and resource IDs regardless of the input they receive. The Trouble Ticket services are backed up by an OSS/J reference implementation, which is a system that creates unique IDs for trouble tickets and stores them in a database together with the assigned details.

The OWL-S descriptions stored in the repository are of Atomic Web Services which attribute the services to processes in the eTOM. These are then shown in the Browser when the appropriate process is selected (see Figure 7).

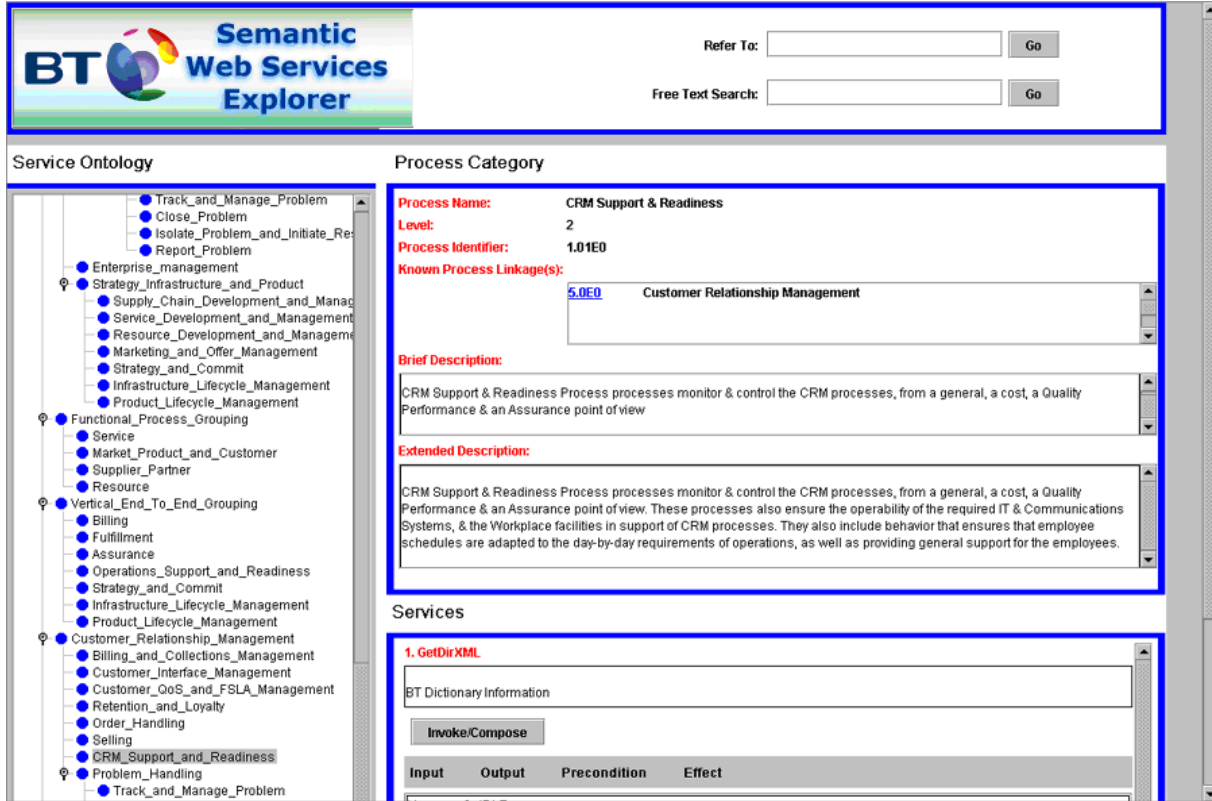



Figure 7. Browsing the eTOM to find available Services.

The user can invoke the Services by selecting it from the Browser. They are then presented with a web form where inputs can be entered. Following invocation with the inputs specified, the user can view the output.

In the scenario, the designer chooses a service as a basis for their composition (currently this must be the final service although it is hoped that a more flexible approach can be achieved). The composer window then opens as shown in Figure 8. The selected Service is shown in the window in the blue box. Its inputs are shown as red boxes while its output is a grey box.

	BT Case Study Architecture Document Deliverable ID:	Page : 13 of 15
		Version: 1.1 Date: 16/09/2004
		Status: Proposal Confid.: Public

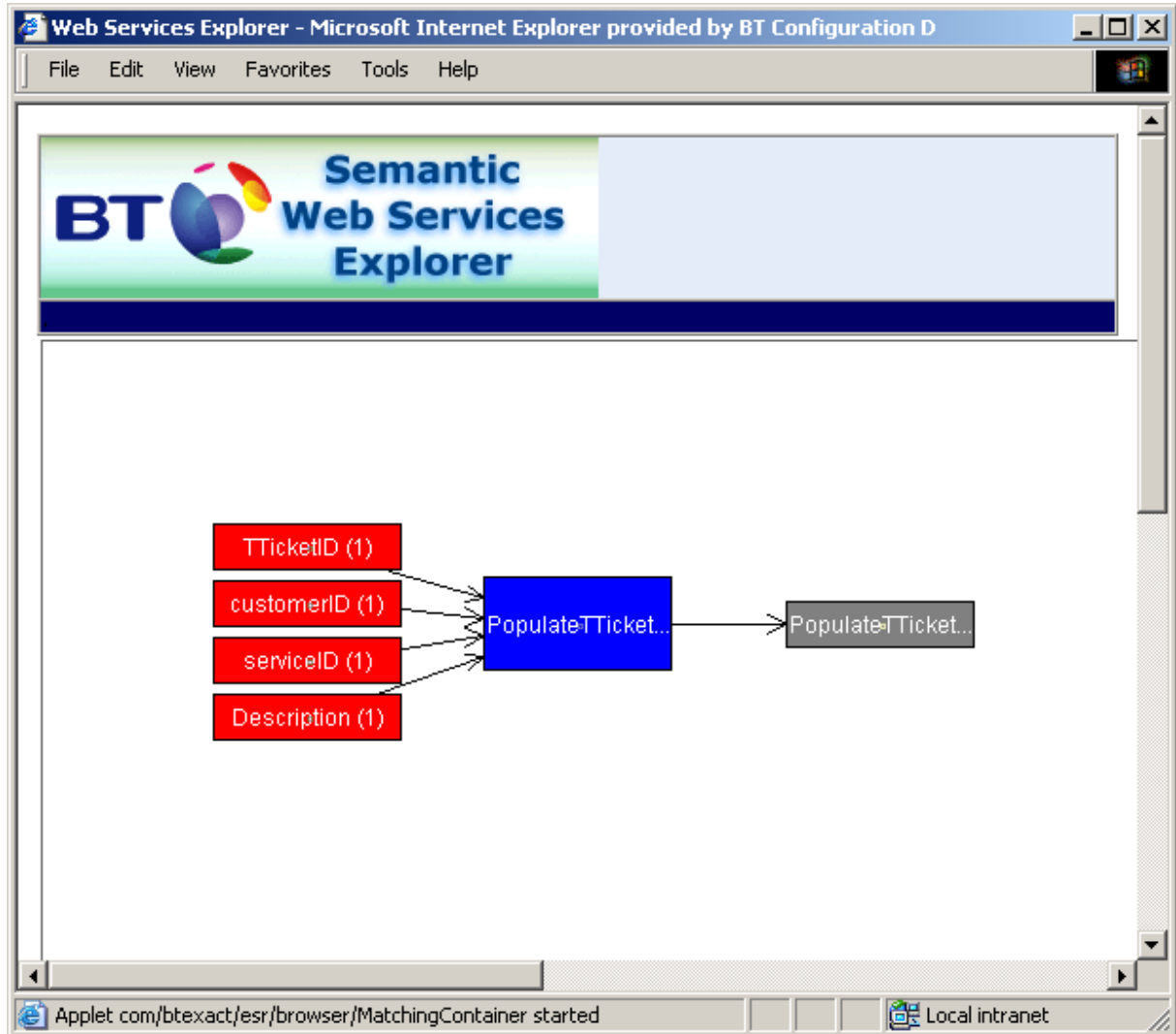


Figure 8. Service in Composer window.

Further Services can then be added. Upon selecting an input, the repository is searched for services that provide outputs that have been described, in OWL-S, with the same ontological concept from the data ontology. The user can choose one of the matching Services which is then added to the composition.

The completed composition for the scenario is shown in Figure 9.

Preconditions and postconditions have been applied to the service descriptions as described in Deliverable 12.2 [3]. The browser does not currently interpret these. The aim is that that during composition, the browser can assist the designer by highlighting appropriate conditions which the designer can then use to improve their decision making. Ideally, these conditions will be used during the discovery process to select Services that can, e.g., satisfy a precondition over a selected input.

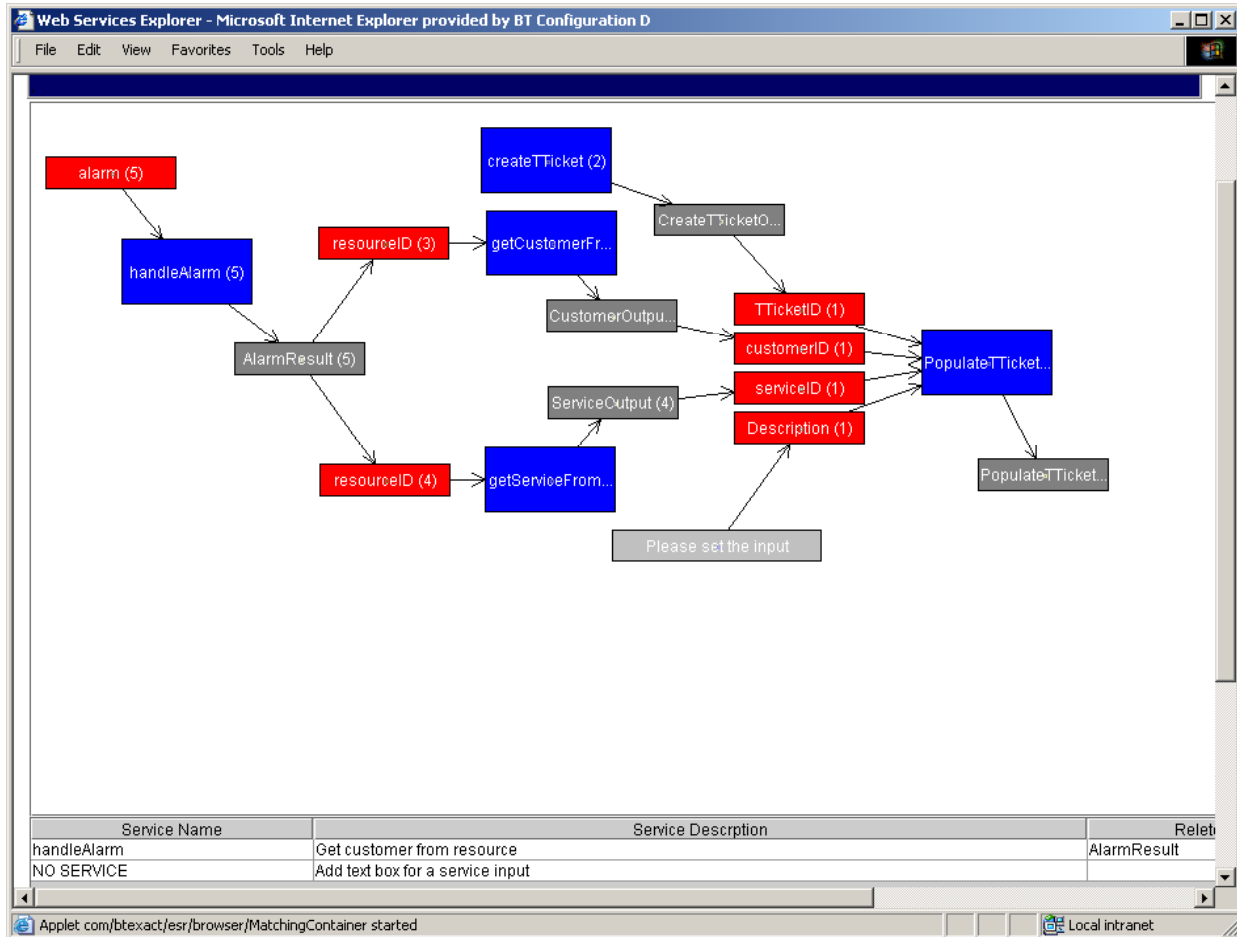



Figure 9. Result of Composition

4 Next Steps

There are a number of limitations to the Browser and some of these will be addressed in ongoing work on the Case Study. Firstly, it only supports very simple control structure during composition i.e. Services are invoked when the required inputs are present to invoke them. There is no guarantee one service will be invoked before another, if the required input is present for both services. This is in fact adequate for the Case Study scenario as it stands. A more complex scenario would require further support. Secondly, there is no support for a choice construct in the control flow. This would be required to allow error handling to be added to the scenario. Thirdly, a very simple matching algorithm is used. Candidate services are selected where they have data outputs described by the same concept as the input of the following service and no mediation function can be added if required. Finally, it does not support the evaluation of preconditions and postconditions when constructing a composed service. At the very least, such conditions should be presented to the designer so they are aware of them at design time.

The case study scenario is being used in a separate demonstrator which uses the SWWS Studio developed by Ontotext. The OWL-S services described in this document are being converted to WSMO services. The SWWS Studio will be used to further describe these

	BT Case Study Architecture Document Deliverable ID:	Page : 15 of 15
		Version: 1.1 Date: 16/09/2004
		Status: Proposal Confid.: Public

services allowing them to be used by the Studio's composer tool. This tool allows a more robust orchestration to be carried out in that it allows further control constructs and will support alignment of the data requirement of the services.

References

- [1] Broekstra, J.; Kampman, A.; van Harmelen, F.(2002), *Sesame: an architecture for storing and querying RDF and RDF schema*, Proc of the first International Semantic Web Conference (ISWC2002), pp54-68, Sardinia, Italy, June 2002.
- [2] Priest, C. (2004), SWWS Technical Architecture, SWWS Draft Deliverable, August 2004.
- [3] Duke, A.; Richardson, M. (2004), *VISP Case Study: Ontologies and Services*, SWWS Deliverable, March 2004