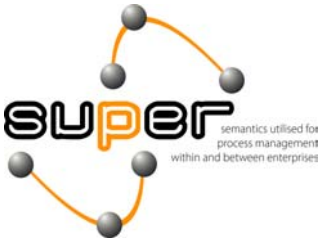

Project IST 026850 SUPER

Semantics Utilized for Process management within and between Enterprises

Deliverable 6.10

Reasoner Integration with the SUPER Infrastructure

Leading Partner: LFUI

Contributing Partner: OU, PUE, IBIS, CEFRIEL, TUE, NUIG

Security Classification: Public (PU)

September, 2008

Version 1.0

Project Details

IST Project Number	026850
Acronym	SUPER
Project Title	Semantics Utilised for Process management within and between EnteRprises
Project URL	http://www.ip-super.org
EU Project Officer	Werner Janusch

Authors (Partner)	Graham Hench (UIBK), Barry Norton (OU), Agata Filipowska (PUE), Stefan Zoeller (IBIS), Alessio Carenini (CEFRIEL), Ana Karla Alves de Medeiros (TUE), Maciej Zaremba (NUIG), Marwane ElKharbili (IDS)		
Deliverable Owner (Partner)	Graham Hench (UIBK)	**E-mail**	graham.hench@sti2.at
		Phone	+43-512-507-6461

Versioning and Contribution History

Version	Description	Comments
0.1	Initial draft with partner contributions	Graham Hench (UIBK)
0.2	Impl Peer Review comments from Mihail Konstantinov (ONTO)	Graham Hench (UIBK)
0.3	Impl EPMB Review comments from John Domingue (OU)	Graham Hench (UIBK)
1.0	Final Version	Graham Hench (UIBK)

Table of Contents

Table of Figures

List of Tables

Executive Summary

This deliverable focuses on the integration of the SBP reasoner developed in D6.3 (“Process Ontology Reasoner”) with the various SUPER components dependent upon its functionality. The work of this deliverable provides reasoning functionalities which are communicated between the dependent components and the SBP Reasoner over the SUPER Semantic Service Bus. The work of this deliverable focuses on successfully integrating the SBP Reasoner with several tools and components (translators, BP mediators, modelling suites, RBE, policy checking & process mining tools, and execution environments) whereby particular reasoning task must be resolved.

1 Introduction

The first section of this deliverable covers the alignment and integration of the SBP Reasoner, via the WSML2Reasoner interface, into the overall SUPER architecture; the second section then highlights the particular querying and reasoning requests from other SUPER components, concluded with a list of the description and dependencies of these components.

2 Alignment

This section explains how the SBP reasoner is aligned with each of the cornerstones of the SUPER project: architecture, methodology, and modeling/ontology stack.

2.1 Overall Architecture Alignment

The SBP Reasoner is one of the SUPER Platform Services which relies upon the Semantic Service Bus (SSB) in order to communicate with the other SUPER tools and components as depicted in Figure 1.

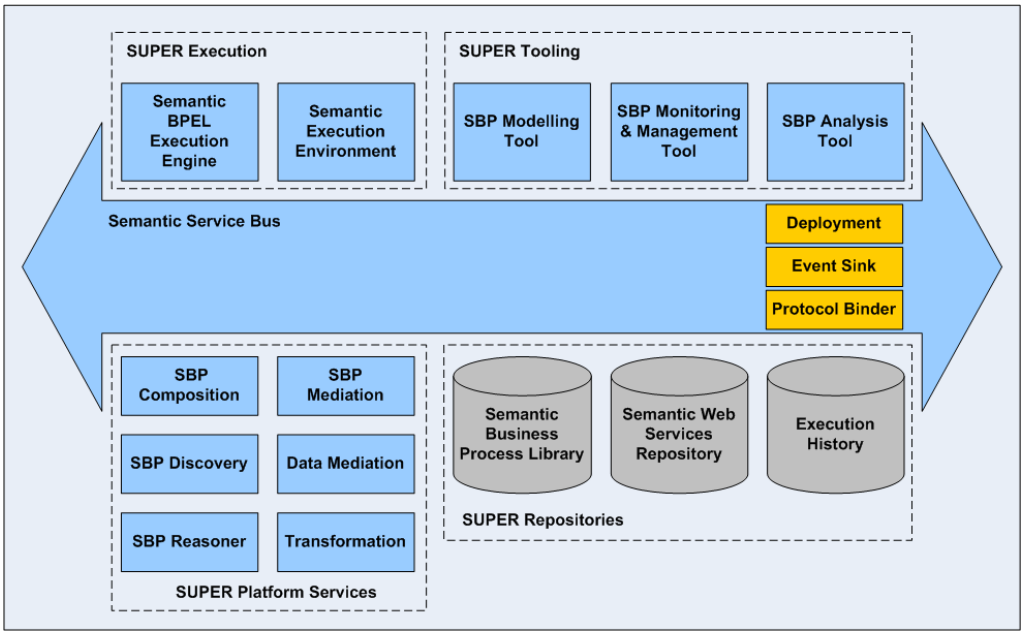

Figure 1 SUPER Architecture

The SBP Reasoner is integrated with several SUPER components (further discussed throughout this deliverable): translators, BP Mediators, modelling suites, RBE, policy checking & process mining tools, and execution environments.

2.2 Methodology Alignment

The SUPER methodology, as shown in Figure 2, describes the SBP lifecycle. Based on the ontological foundation, different roles involved with a company's BPM efforts, such as business analysts, domain experts and IT specialists work with and on the process models. During the lifecycle of such process models, they iteratively go through four different steps, or lifecycle phases, which enable the Strategic SBPM. During each phase of this lifecycle different functionalities are required to handle the lifecycle phase's objectives. The different use cases each highlight some of these phases, as described in the relevant use case sections in this deliverable. The SBP Reasoner should thus be able to support the components at all phases, however its design requirements are mostly based upon components from the Process Modelling, Execution, and Analysis phases.

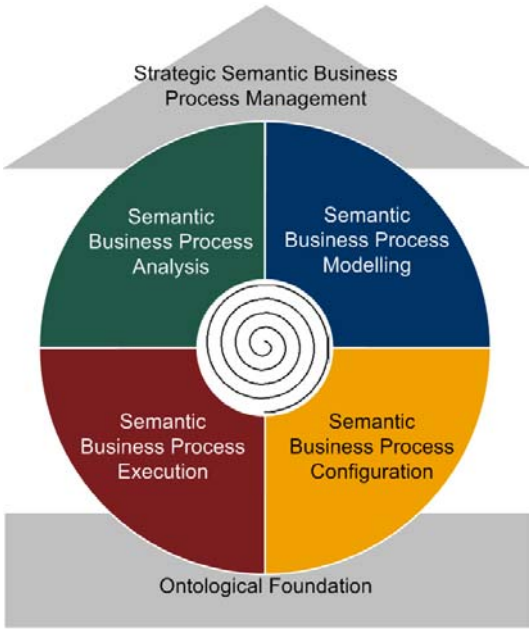

Figure 2 SUPER Methodology

2.3 Modelling Stack Alignment

The SUPER Modelling Stack, illustrated in Figure 3, describes the different levels of abstraction for the process models, from the high-level business view based on solution maps and ad-hoc modelling to the specific implementation of the functionalities needed by processes and enabled by the SUPER components. Since the SUPER prototypes aim to support all lifecycle phases from the SUPER methodology, it will also need to work with all artefacts created for each of the different modelling levels from the modelling stack. As a core component of the SUPER Platform Services, the SBP Reasoner is a Java implementation which sits at the lowest level, however the inferred results are used throughout all layers of the Modeling Stack (thus closing the business-IT gap).

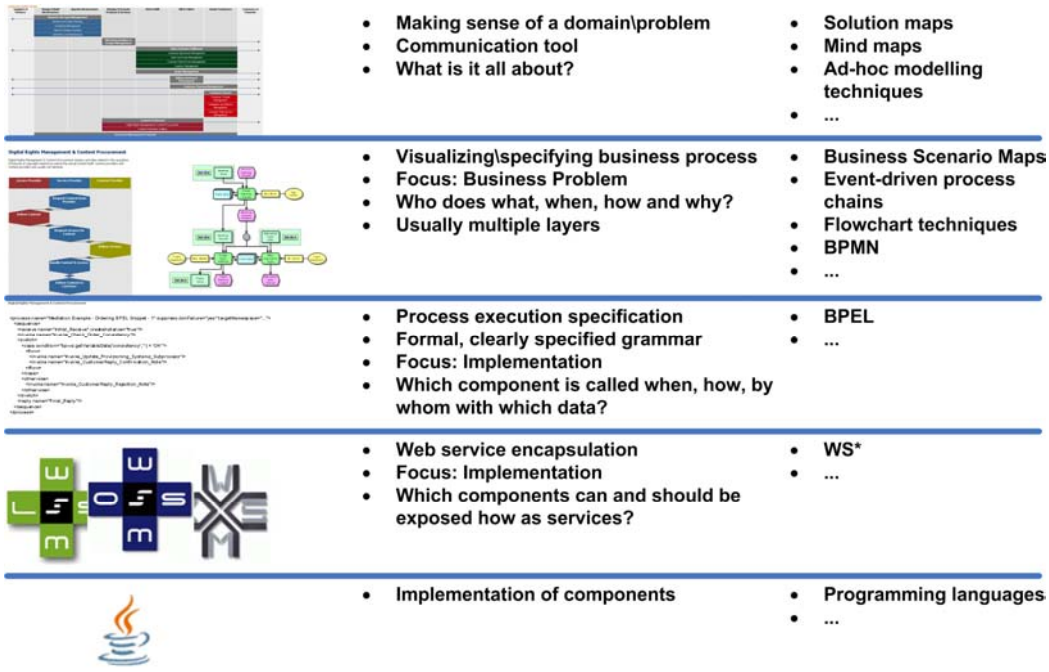

Figure 3 SUPER Modeling Stack

The SUPER Modeling Stack is supported and formalized by the SUPER Ontology stack shown in Figure 4.

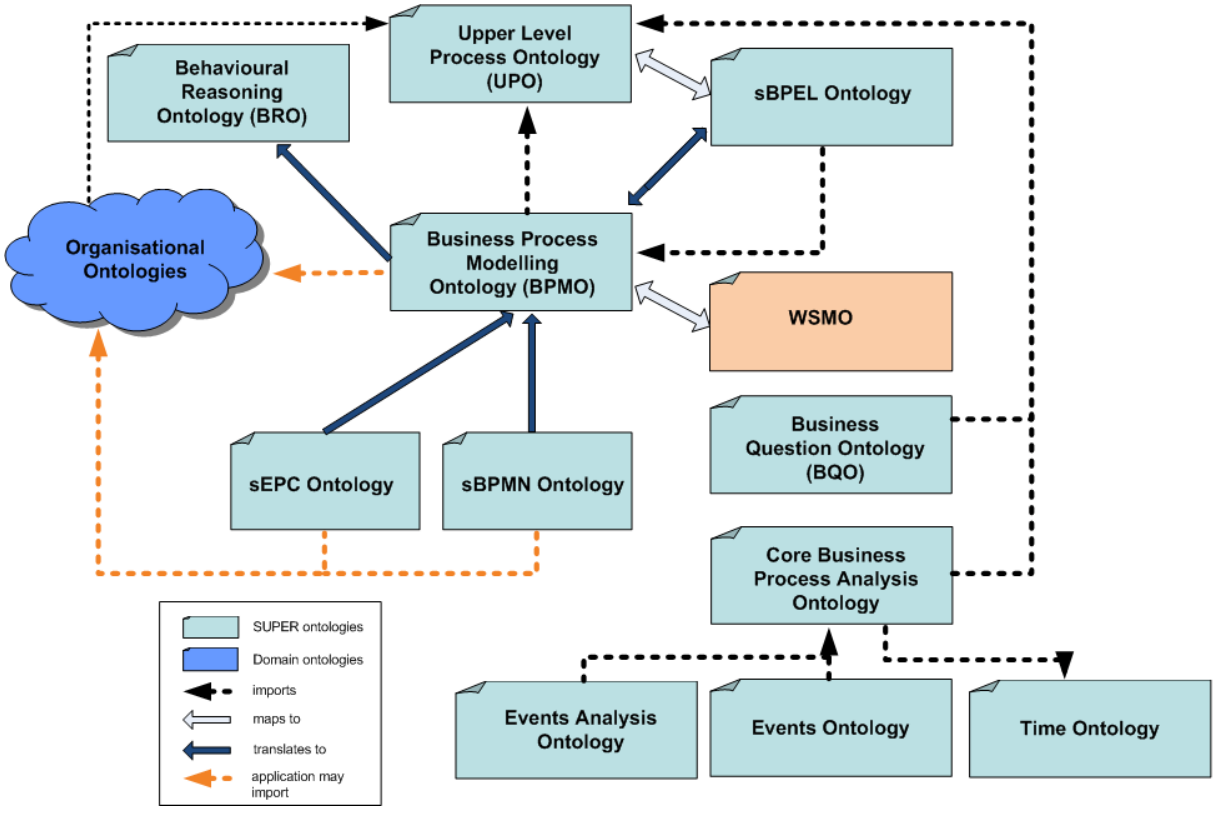

Figure 4 SUPER Ontology Stack

The top layer of the SBP Reasoner is the WSML2Reasoner Framework which provides the adapter interface and translation algorithms to transform the ontological syntax (WSML - the standard syntax of the SUPER ontology stack) into appropriate syntax of the underlying reasoning engine (in this case, the extended Datalog syntax to be handled by IRIS, the embedded reasoning engine of the SBP Reasoner developed in D6.3). Through this framework, multiple ontologies can be imported in order to reason over the various resources and repositories available in the SUPER suite.

3 Integration with SUPER Infrastructure

The SBP Reasoner has been integrated into several components of the SUPER infrastructure and tools suite. This section provides a brief description of the utility of the reasoner by the various SUPER components: translators, RBE tools, BP Mediator, policy checking, process mining, and execution environments.

3.1 Dependent SUPER Components

SUPER Component	sBPMN-to-BPMO translator

Description	Dependent upon the WSMLReasoner framework using the IRIS reasoner for instance retrieval.
	`WSMLReasonerFactory (DefaultWSMLReasonerFactory.getFactory() .createWSMLFlightReasoner(params);)`
	As a default reasoner we use built-in IRIS Reasoner (i.e.
	`params.put(WSMLReasonerFactory.PARAM_BUILT_IN_REASONER, WSMLReasonerFactory.BuiltInReasoner.IRIS);).`
	The instances are retrieved via getInstances() method.
	e.g.
	`Concept concept = ns.createConcept("sbpmn", conceptName); Set<Instance> instances = reasoner.getInstances((IRI)sbpmnontology.getIdentifier(), concept);`
	`//or instances = reasoner.getInstances((IRI)ontology.getIdentifier(),cBPMOBlockPattern);`
	Other queries:
	`String queryString = "?x[sbpmn#hasTarget hasValue Task1] memberOf sbpmn#SequenceFlow";`
	`queryString = "?x[bpmo#hasConditionalBranch hasValue ConditionalBranch1] memberOf bpmo#BlockPattern";`

Versions	0.6.2 wsml2reasoner_gpl-v0_6_2.jar and IRIS iris-0.57.jar.
Additional comments	Contact person: Krzysztof Węcel k.wecel@kie.ae.poznan.pl

Table 1 Integration with sBPMN-to-BPMO Translator

SUPER Component	Reverse Business Process Discovery Engine (sRBE Tool), described D5.4, current delivarable is D5.7 -
Description	Business Questions with predefined, but parametrizable reasoner queries are used in order to reason event logs from EVO / COBRA Used to load process logs and ontologies from files and to perform RBE analysis on semantic logs. Two kinds of queries are used: classical logical expression with the use of builtin predicates and WSML-A queries We only query for relation instances, the real complexity is in axioms that actually produce such relation instances. You can find examples on the SUPER SVN: https://svn.sti-innsbruck.at/super/WP5/srbe-lib/resources/BQR_20071108.wsml Example query: `WhichRouterOrdersWereProcessedWithDelay(?ActivityInstance, ?completionEvent, ?actor, ?period, ?startdate, ?enddate,?dur)`
Version	v0.6.2 → Any which is working properly with the other necessary components wsmo4j, IRIS and the general WSML specs - & modified version of trunk rev. 903
Additional comments	An extension to trunk WSML2Reasoner has been developed to integrate WSML-A query language with usual WSML2Reasoner objects (to be able to invoke “reasoner.executeWSMLAQuery(queryString);” from the main WSMLReasoner object. Code for integration can be found in WP5/srbe-lib/src/org/wsml/reasoner directory tree. As for the original WSML2Reasoner code, the “private” methods of DefaultWSMLReasonerFactory have been turned into “protected” to allow subclassing (the change is so small that I didn't include it in my “shadow” branch in srbe-lib). For detailed questions on sRBE please contact: Gernot Zeissler zeissler@ibis-thome.de Stefan Zoeller zoeller@ibis-thome.de Alessio Carenini Alessio.Carenini@cefriel.it

Table 2 Integration with sRBE Tool

SUPER Component	Business Process Mediator Reasoning Component (D4.3), dependent upon: - Behavioural Reasoning Ontology (D1.7) - BPMO2BRO (D4.8)
Description	Provides translation into BRO and behavioural reasoning tasks over the Behavioural Reasoning Ontology
Version	v0.6.2
Additional comments	The implementation of function symbols was very useful to the completion of this work. Contact: Barry Norton, b.j.norton@open.ac.uk

Table 3 Integration with BP Mediator Reasoning Component

SUPER Component	Compliance Checking Engine
Description	Uses IRIS on top of WSML2reasoner to reason on Business Policy and Rules Ontology (D1.6) instances for policy checking. Initially developed to be used in D5.6 Potentially assist in the Telefonica use case for business policies
Version	v0.6.2
Additional comments	Contact: Marwane ElKharbili, Marwane.ElKharbili@ids-scheer.com

Table 4 Integration with Compliance Checking Engine

SUPER Component	Semantic Process Mining Tool
Description	Dependent upon the WSML2Reasoner framework to perform various subsumption reasoning over the concepts and instances
Version	v0.5 (used by Semantic ProM plug-ins to perform semantic process mining analysis) (file: sml2reasoner-noncom-v0_5.jar) v0.6.2 (used by ProMimport plug-in to create SA-MXML files from History Repository) (file: wsml2reasoner_lgpl-v0_6_2.jar)
Additional comments	Contact: Ana Karla Alves de Medeiros, A.K.Medeiros@tue.nl

Table 5 Integration with Semantic Process Mining Tool

SUPER Component	Semantic Execution Environment

Description	The WSML2Reasoner framework with the built-in IRIS reasoner is extensively used both during Discovery and WSMO-Choreography (Abstract State Machine) based service execution. SEE is used mainly by SBPELEE for Goal resolution and SWS invocations. SEE is also used as an implementation of SWS-Repository utilized by for example SBP Composition, Data Mediation, etc.
	Example Discovery query:
	`postcondition instdisc#postcondition`
	`definedBy`
	`?x memberOf etel#ADSLRouterBidResponse [etel#estimate`
	`hasValue ?estimate] and`
	`?y memberOf etel#ADSLRouterBidRequest and`
	`?z[etel#reliability hasValue ?reliability] memberOf etel#QoS and`
	`?estimate[etel#dueBy hasValue ?dueDate] memberOf`
	`etel#TimedFullfillmentWindow and etel#isAvailable(?y).`
	Example Choreography query:
	`forall {?bidresponse, ?controlstate, ?transID, ?estimate} with`
	`(?bidresponse[etel#id hasValue ?transID, etel#estimate`
	`hasValue ?estimate] memberOf tel#ADSLRouterBidResponse`
	`and ?controlstate[oasm#value hasValue oasm#InitialState]`
	`memberOf oasm#ControlState)`
	`do`
	`add(_#1[etel#id hasValue ?transID] memberOf`
	`etel#ADSLRouterRequest)`
	`delete(?controlstate[oasm#value hasValue`

	oasm#InitialState]) add(?controlstate[oasm#value hasValue oasm#EndState]) add(_#[etel#estimate hasValue ?estimate] memberOf etel#ADSLRouterRequestResponse) //response modifier endForall
Version	v0.6.2
Additional comments	Contact: Sami Bhiri, sami.bhiri@deri.org

Table 6 Integration with Semantic Execution Environment

4 Conclusions

The work of this deliverable has successfully integrated the SBP Reasoner with several tools and components (translators, BP mediators, modelling suites, RBE, policy checking & process mining tools, and execution environments) whereby particular reasoning task were required to be resolved. In addition to the integration of the SBP Reasoner with the SUPER tools and components addressed throughout this deliverable, the interface with the semantic service bus allows the SBP Reasoner to be accessed by all SUPER components should the provided reasoning functionalities become necessary in other semantic BPM activities.