



Data, Information and Process integration
with Semantic Web Services

DIP

Data, Information and Process Integration with Semantic Web Services

FP6 – 507483

Document

Goal-oriented SWS composition prototype DIP Deliverable D4.15

Patrick Albert
Christian de Sainte Marie
Laurent Henocque
Mathias Kleiner

July 17, 2006



SUMMARY

1. Summary

This deliverable and its annexed documents and archives detail the principles, assumptions, organization of a prototype for a tool achieving automated goal directed composition of Semantic Web Services. The document and its annexes are intended to allow project partners to run the prototypes as a means of demoing or testing.

2. Contribution

This work is an original attempt to use constraint based finite model search to achieve SWS composition, in the objective of both:

- being usable as a helper application within the DIP tools that allow for manually or automatically editing Semantical Web Services,
- reaching the highest possible level of functionality .

3. Interactions within DIP

This deliverable is impacted by the WSMO specification for choreography and orchestration. The programming interface of the prototype is an input to DIP WP4 applications, as required for their integration. The prototype depends upon several DIP/WSMO APIs required to read/parse the specification of SWS.

4. Target audience

The deliverable is of particular interest to DIP participants involved in Choreography, Orchestration, Discovery, Process Mediation, Data Mediation, Use Cases.

5. : Disclaimer

The DIP Consortium is proprietary. There is no warranty for the accuracy or completeness of the information, text, graphics, links or other items contained within this material. This document represents the common view of the consortium and does not necessarily reflect the view of the individual partners.

DOCUMENT INFORMATION

IST Project Number	FP6 – 507483	Acronym	DIP
Full Title	Data, Information, and Process Integration with Semantic Web Services		
Project URL	http://dip.semanticweb.org/		
Document URL			
EU Project Officer	Kai Tullius		

Deliverable DIP Deliverable D4.15	Number	4.15	Title	Goal-oriented SWS composition prototype
Work Package	Number	4	Title	Service Usage

Date of Delivery	Contractual	M30	Actual	30-Jun-06
Status	version 0.4		final	<input checked="" type="checkbox"/>
Nature	prototype <input checked="" type="checkbox"/> report <input type="checkbox"/> dissemination <input type="checkbox"/> ontology <input type="checkbox"/>			
Dissemination Level	public <input checked="" type="checkbox"/> consortium <input type="checkbox"/>			








Authors (Partner)	see title page			
Resp. Author	Patrick Albert		E-mail	palbert@ilog.fr
	Partner	ILOG	Phone	+33 (1) 49-08-35-00

Abstract (for dissemination)	This deliverable implements the specification detailed in deliverable D4.12
Keywords	composition, configuration, constraint, policy, finite model, search, semantic web service, web service, workflow, modeling

Version Log			
Issue Date	Rev No.	Author	Change
17-May-06	1	Laurent Henocque	Created
7-June-06	2	Laurent Henocque	Initial content completed
12-June-06	3	Mathias Kleiner	Refined licencing and third party tools info
13-July-06	4	Laurent Henocque	Account for reviews, first pass

Reviewers				
	Barry Norton		E-mail	b.j.norton@open.ac.uk
	Partner	Open University	Phone	+44 (0) 1908 659399
	Vassil Momtchev		E-mail	vassil.momtchev@ontotext.com
	Partner	Ontotext	Phone	(+359 2) 9768 310

PROJECT CONSORTIUM INFORMATION

Partner	Acronym	Contact
National University of Galway	NUIG 	Dr. Sigurd Harand Digital Enterprise Research Institute (DERI) National University of Ireland, Galway Galway Ireland E-mail: sigurd.harand@deri.org Tel: +353 91 495112
Fundacion De La Innovacion.Bankinter	Bankinter 	Monica Martinez Montes Fundacion de la Innovation. BankInter, Paseo Castellana, 29 28046 Madrid, Spain Email: mmtnez@bankinter.es Tel: 916234238
British Telecommunications Plc.	BT 	Dr. John Davies BT Exact (Orion Floor 5 pp12) Adastral Park Martlesham Ipswich IP5 3RE, United Kingdom Email: john.nj.davies@bt.com Tel: +44 1473 609583
Swiss Federal Institute of Technology, Lausanne	EPFL 	Prof. Karl Aberer Distributed Information Systems Laboratory École Polytechnique Fédérale de Lausanne Bât. PSE-A 1015 Lausanne, Switzerland E-mail : Karl.Aberer@epfl.ch Tel: +41 21 693 4679
Essex County Council	Essex 	Mary Rowlatt, Essex County Council, PO Box 11, County Hall, Duke Street, Chelmsford, Essex, CM1 1LX, United Kingdom. E-mail: maryr@essexcc.gov.uk Tel: +44 (0)1245 436524
Forschungszentrum Informatik	FZI 	Andreas Abecker Forschungszentrum Informatik Haid-und-Neu Strasse 10-14 76131 Karlsruhe Germany E-mail: abecker@fzi.de Tel: +49 721 96540
Institut für Informatik, Leopold-Franzens Universität Innsbruck	UIBK 	Prof. Dieter Fensel Institute of computer science University of Innsbruck Technikerstr. 25 A-6020 Innsbruck, Austria Email: dieter.fensel@deri.org Tel: +43 512 5076485

ILOG SA		Christian de Sainte Marie 9 Rue de Verdun, 94253, Gentilly, France E-mail: csma@ilog.fr Tel: +33 1 49082981
inubit AG		Torsten Schmale, inubit AG, Lützowstraße 105-106 D-10785 Berlin, Germany E-mail: ts@inubit.com Tel: +49 30726112 0
Intelligent Software Components, S.A.		Dr. V. Richard Benjamins, Director R&D Intelligent Software Components, S.A. Pedro de Valdivia 10 28006 Madrid, Spain E-mail: rbenjamins@isoco.com Tel. +34 913 349 797
MDR Partners		Rob Davies MDR Partners 8 St. Andrew Street, Hertford, Herts. United Kingdom, SG14 1JA E-mail: rob.davies@mdrpartners.com Tel.: +44 (0)208 8763121
Hanival Internet Services GmbH		Alexander Wahler NIWA WEB Solutions Hanival Internet Services GmbH Kirchengasse 13/1a A-1070 Wien E-mail: wahler@niwa.at Tel.: +43 131 95843 11
The Open University		Dr. John Domingue Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, UK E-mail: j.b.domingue@open.ac.uk Tel.: +44 1908 655014
SAP AG		Dr. Elmar Dörner SAP Research, CEC Karlsruhe SAP AG Vincenz-Priessnitz-Str. 1 76131 Karlsruhe, Germany E-mail: elmar.dorner@sap.com Tel: +49 721 6902 31
Sirma AI Ltd.		Atanas Kiryakov, Ontotext Lab, - Sirma AI EAD, Office Express IT Centre, 3rd Floor 135 Tzarigradsko Chausse, Sofia 1784, Bulgaria E-mail: atanas.kiryakov@sirma.bg Tel.: +359 2 9768 303



Unicorn Solution Ltd.	<p>Unicorn</p> 	<p>Jeff Eisenberg Unicorn Solutions Ltd, Malcha Technology Park 1 Jerusalem 96951, Israel E-mail: Jeff.Eisenberg@unicorn.com Tel.: +972 2 6491111</p>
Vrije Universiteit Brussel	<p>VUB</p>  <p>Vrije Universiteit Brussel</p>	<p>Pieter De Leenheer, Starlab- VUB Vrije Universiteit Brussel Pleinlaan 2, G-10 1050 Brussel, Belgium E-mail: Pieter.De.Leenheer@vub.ac.be Tel.: +32 (0) 2 629 3749</p>

TABLE OF CONTENTS

1	INTRODUCTION	1
1.1	Scope	1
1.2	Context for goal oriented SWS composition	1
1.3	Composing as a Finite Model Search problem	2
1.4	A workflow language for choreographies and orchestrations	3
1.5	A language for abstract composition goals	3
1.6	Brief introduction to configuration	4
1.7	Related work	4
2	FACT SHEET	5
2.1	Deliverable name	5
2.2	Contact person with contact details	5
2.3	Brief description of purpose	5
2.4	Brief description of scope	5
2.5	Brief description of functionality	5
2.6	Type of API (e.g. JAVA, WSDL)	6
2.7	Format of program input	6
2.7.1	Abstract model for composition goals	7
2.7.2	Graphical representation	10
2.8	Format of program results	10
2.9	Files attached to the current document	10
3	DETAILED LICENSE INFORMATION	12
3.1	License	12
3.2	How to obtain a license	12
4	DETAILED THIRD PARTY TOOL INFORMATION	13
4.1	JAVA SDK 1.5.0+	13
4.2	ILOG JCONFIGURATOR	13
4.3	WSMO4J, WSMO, WSMXI APIs	14
4.4	Visual Paradigm for UML	14
5	TECHNICAL REQUIREMENTS FOR USING THE PROTOTYPE	15
5.1	JAVA SDK 1.5.0+	15
5.2	ILOG JCONFIGURATOR	15
5.3	WSMO4J, WSMO and WSMX libraries	15
6	DETAILED INFORMATION ON HOW TO USE/EVALUATE THE PROTOTYPE	16
6.1	Download	16
6.2	Installation	16
7	ROADMAP ON FUTURE PLANS FOR FURTHER DEVELOPING THE PROTOTYPE	17
7.1	Important Notice	17
8	USER MANUAL	18

9	REFERENCE MANUAL	19
10	DEMONSTRATION INFORMATION	20
10.1	How to obtain demo files	20

1 INTRODUCTION

This deliverable presents a prototype workflow composer for DIP based upon constraint programming technologies. The present document is expected to provide enough information to operate the prototype without any help. As a project partner, ILOG is ready and willing to help other partners in adapting the program to their needs. The rest of this section briefly presents the scope and context of configuration based workflow composition, and can be skipped by a reader already aware of these issues.

1.1 Scope

We place ourselves in the scope of automatic or computer aided SWS composition. The basic assumptions for composing SWS is that there exists a repository of SWS that are potential candidates for composition and can be accessed via discovery, as well as a directory listing of mediators needed to adapt messages between SWS having incompatible message ontologies.

We also assume that the composition process is goal oriented: the user must formulate a request to the composer in the form of a “composition goal”: a loose or precise abstract statement of the constraints posted on elementary goals that must participate to a composition. The requirement for an abstract composition language for the composition of semantic web services¹ has been acknowledged by the DIP Specification Deliverable D4.12[8], the main input for the current work. The D4.12 also precisely defines the scope of the current prototype and the reader is kindly directed to this source for further details. In order to be sufficiently self contained however, the present introduction borrows some material from the deliverable D4.12 and from the articles [1] and [2]. A reader aware of configuration techniques and of configuration applied to web service composition may skip this part or read it quickly. The DIP deliverable D4.10 proposes a formal presentation of composite goals that bears some relations with the language defined in D4.12.

The current deliverable also follows from an earlier DIP prototype delivered in June 2005 under the id D4a.9 as an internal (consortium only) deliverable.

1.2 Context for goal oriented SWS composition

Our approach to goal oriented SWS composition can be sketched by saying that the (workflows that implement the) *choreographies* of the SWS matching atomic goals are combined to form a valid orchestration for a resulting composite SWS under the constraints set by a request to the composer called a *composition goal*. The process at a glance is the following:

- the user defines his request to the composer in the form of a composition goal involving atomic goals plus constraints²,

¹There are many justifications for the statement of complex composition requests. An example: a composite SWS needs to sum up cost data output by several participants: there is no way for an automated process to determine the need for such a summation from the choreographies alone.

²This step is currently performed using a Java API - a fully integrated and interfaced prototype being planned for M36.

- the composition goal is optionally improved using a specific configuration phase³,
- atomic goals present in the request are used to discover matching semantic web services,
- the choreographies of matching SWS are input to the composer as elementary bricks of the solution,
- the composition goal is input to the composer as constraints that restrict the possible constructions,
- the composer calls a configurator to produce a valid orchestration from the available choreographies under the specified constraints,
- the result is then exported as a WS description involving a valid orchestration description, in the WSML-AD extension

1.3 Composing as a Finite Model Search problem

A large body of research addresses the SWS/Workflow composition problem as a reasoning task in a first order logic variant. We chose to consider this problem using an approach that falls within the field of finite model search. The underlying logic has the expressive power of description logics, and allows the statement of constrained object models.

The specification of the constrained object models is achieved via a combination of UML2 Class diagrams for the visual part, and the Z specification language for the formal specification of model constraints. We chose not to use the OCL2 language for this purpose. Arguments supporting this choice as well as complete examples can be found in [5, 8].

Finite model search for such theories is performed using a constraint based configurator call ILOG JConfigurator. JConfigurator accepts as input a constraint based variant of description logics[16], in which the problem specification is translated (this step is still manual but could be automatized in the future).

Configuration emerges as an AI technique with applications in many different areas, where the problem can be formulated as the production of a finite instance of an object model subject to constraints. Reasoning about workflows falls into this category, because a workflow description is an instance of a given metamodel (as is the UML metamodel for activity diagrams [15]). Composing workflows is a configuration problem in that in so doing, one must introduce an arbitrary number of previously non existent transitions (fork, join, split, merge, transformations, pre-defined user-interactions sequences), and interconnect input and output message pins provided they have compatible types.

The user of a workflow composition system expects in return for his input a complete “*composite*” workflow, that interleaves the execution of several of the elementary argument workflows, while ensuring that all possible integrity constraints as well as the constraints formulated in the composition goal request remain valid. Among *integrity* constraints are those that stem from the metamodel itself: for instance some

³This is not currently available and is planned for M36.

constraints state that two or more workflows should not enter deadlock situations, all waiting for some other to send a message. Other constraints are more problem specific and mentioned in the request, like those stating for instance that an item being shipped is indeed the one that was produced.

1.4 A workflow language for choreographies and orchestrations

We consider SWS having choreographies defined using a close variant to a subset of UML2 activity diagrams [15]. Such a definition is not mandatory for a SWS to be correctly published within DIP, but is required for composition. The rationale is that since the composer operates at the structure level of choreographies and orchestrations, this structure must be made explicit. The execution model of WSMX based on ASMs does not warrant the possibility of retrieving an implicit structure from the ASM specification.

The visual workflow language chosen for choreographies and orchestrations was formally specified as part of the DIP D3.4 and D3.5 deliverables in their common DIO annex [6, 7, 5]. The underlying operational semantics is that of colored Petri nets, where messages (tokens) have types. The composition process however doesn't need to consider the operational semantics of the workflow language, but rather focus on the properties of the corresponding metamodel. In other words, the composer deals with the static structure of workflows alone and the message ontologies, but does not need to emulate the message flows.

More precisely, we treat SWS composition as the process of connecting input and output message flows to preexisting or newly introduced workflow elements, as are for instance *fork*, *join*, *decision*, *merge* nodes, *mediators* or auxiliary user *input/output handling actions*. Hence the only elements retained for composition are the structural properties of argument workflows, messages, and transformations. We do not need to emulate workflows in any sense, but can however formulate constraints that to some extent guarantee the viability of the result⁴.

This general approach to workflow composition originates from several research communications [29, 11, 21] and was experimented using configuration techniques in [1]. We have illustrated in the D4.12[8] our central assumptions by considering the rather complex Producer/Shipper composition problem introduced in [21]. The problem is to compose a valid workflow from a producer workflow and a shipper workflow. One difficulty of this use case is that the execution of both workflows must be interleaved. The producer outputs results that must be fed into the shipper so that both “offers” can be aggregated and presented to the user. This inter-connection remains unknown to the user. The reader is kindly directed to [8, 6, 7, 5] for details.

1.5 A language for abstract composition goals

Composition requests abstract the entire set of valid compositions that can be produced from elementary SWS matching their atomic goals and cannot be expressed using a workflow language as UML-AD for instance. In the general case, more com-

⁴For instance some constraints prevent simple deadlock situations that arise when two threads wait for a message from the other

plex interrelation may occur, for instance to state the some data must represent the sum of several other messages, or that some messages must be aggregated to build a message from a more complex ontology (for instance aggregating three integers to build a date. The composition goal language proposed in this deliverable supports such definitions.

1.6 Brief introduction to configuration

A configuration task consists in building (a simulation of) a *complex product* from *components* picked from a catalog of *types*. Neither the number nor the actual types of the required components are known beforehand. Components are subject to *relations*, and their types are subject to *inheritance*. *Constraints* (also called well-formedness rules) generically define all the valid products. A configurator expects as input a fragment of a target object structure, and expands it to a solution of the configuration problem, if any. This problem is semi-decidable in the general case.

A configuration program is well described using a *constrained object model* in the form of a standard class diagram, together with well-formedness rules or constraints. Technically solving the associated enumeration problem can be made using various formalisms or technical approaches: extensions of the CSP paradigm [19, 13], knowledge based approaches [26], terminological logics [20], logic programming (using forward or backward chaining, and non standard semantics) [25], object-oriented approaches [17, 26]. Our experiments were conducted using the object-oriented configurator Ilog JConfigurator [17].

1.7 Related work

Automated workflow composition is a field of intense activity, with applications to at least two wide areas: Business Process Modeling and (Semantic) Web Services. Tentative techniques to address this problem are experimented using many formalisms and techniques, among which Situation calculus [18], Logic programming [23], Type matching: [10], Coloured Petri nets: [29, 11], Linear logic: [22], Process solving methods [3, 14, 27], AI Planning [4], Hierarchical Task Network (HTN) planning [24, 28], Markov decision processes [12].

2 FACT SHEET

2.1 Deliverable name

The prototype SWS composer is called GOBAC (for GOal BAsed Composer).

2.2 Contact person with contact details

For technical information about GOBAC or questions, please contact Mathias Kleiner, by email at mkleiner@ilog.fr, or by post at

ILOG
9 rue de Verdun BP85
94253 Gentilly France

2.3 Brief description of purpose

GOBAC is provided as a stand-alone application. Developed as an Eclipse plugin, it can be integrated within WSMO Studio where it provides a helper application for design time composition of SWS.

2.4 Brief description of scope

GOBAC addresses the problem of Semantic Web Composition for DIP. GOBAC exploits as a metamodel for workflows a large subset of UML2 activity diagram concepts, hence addresses the problem in a true generality.

GOBAC does not currently address scalability issues. Computation times are impacted by many factors, including the presence/absence of redundant constraints that help filter out unwanted configurations. Work in this direction is planned for the final prototype due at M36.

2.5 Brief description of functionality

The GOBAC prototype is packaged as an Eclipse plugin. As such, it gets smoothly integrated within the Eclipse plugin based WSMO Studio IDE. It may evolve as a WSMO Studio plugin, to offer extended interaction with other WSMO Studio tools, in the M36 improved version of the prototype.

The program also provides an API in order to integrate it in a Java program, this API is further documented in Section 9.

GOBAC allows one to generate a valid orchestration corresponding to a composition goal request, that can be emulated using one of the DIP implementation platforms, i.e. WSMX or IRSIII. The current state of the prototypes invokes the DIP compliant IRSIII tool using the standard DIP API.

The program expects as input:

- a composition goal: the main composer request,
- candidate participant SWS's, obtained by discovery as a specific interaction with the tool in the WSMO Studio
- all the required ontology descriptions, read from WSMO descriptors

Given such an input, GOBAC produces as its output an orchestration obeying the constraints mentioned in the composition goal. The result can be interpreted as an UML2 activity diagram, naturally, or as the orchestration of the resulting composite service, and is naturally exported as a WSML-AD description (see [9] for a specification of this extension).

2.6 Type of API (e.g. JAVA, WSDL)

GOBAC is distributed as an RCP stand-alone application, which requires the following third party libraries :

- JConfigurator 2.1
- WSM04J : wsmo4j-0.5.2
- WSM0 API : wsmo-api-0.5.2
- WSMX : wsmx-integration-api-0.3

We also provide a JAVA API for tools integration.

GOBAC is built using the JAVA SDK 1.5.0+.

2.7 Format of program input

Composition goals are currently described using a specific format, not yet considered for a WSML extension. The prototype in its final state will accept the specification of its input in the form of a specific language or of an XMI file produced by an UML2 compliant visual editor.

The specific composition goal language may become an extension to WSML as currently are the Activity Diagram and Cashew based choreography/orchestrations to WSML. Obviously, composition goal language might as well be implemented in the form of a WSMO ontology, an option that can be considered for month 36. Using such an ontology could allow to store and retrieve composition goals from a repository. Our recommendation is to let the ontology sketched in deliverable D3.10 to fully account for the D4.12 composition goal language.

The state of the prototype here documented requires the composition goal to be stated in a XMI file, exported by a compliant visual editor. The visual editing uses a specific adaptation of activity diagrams based on mandatory stereotypes and tagged values. We provide a full description of how to describe a composition goal, as well as

a template file for those stereotypes usable in a visual editor called Visual Paradigm for UML.

Choreography related input is read from DIP SWS descriptions using the WSML parser, to obtain machine processable data structures. Our activity diagram language is available as a WSML extension, described in the D3.8 Deliverable [9] on choreography and orchestrations.

2.7.1 Abstract model for composition goals

We present in Figures 2.1 and 2.2 the composition goal language used to formulate requests to the composer, as an UML abstract model together with Z constraints specifying well-formed composition goals.

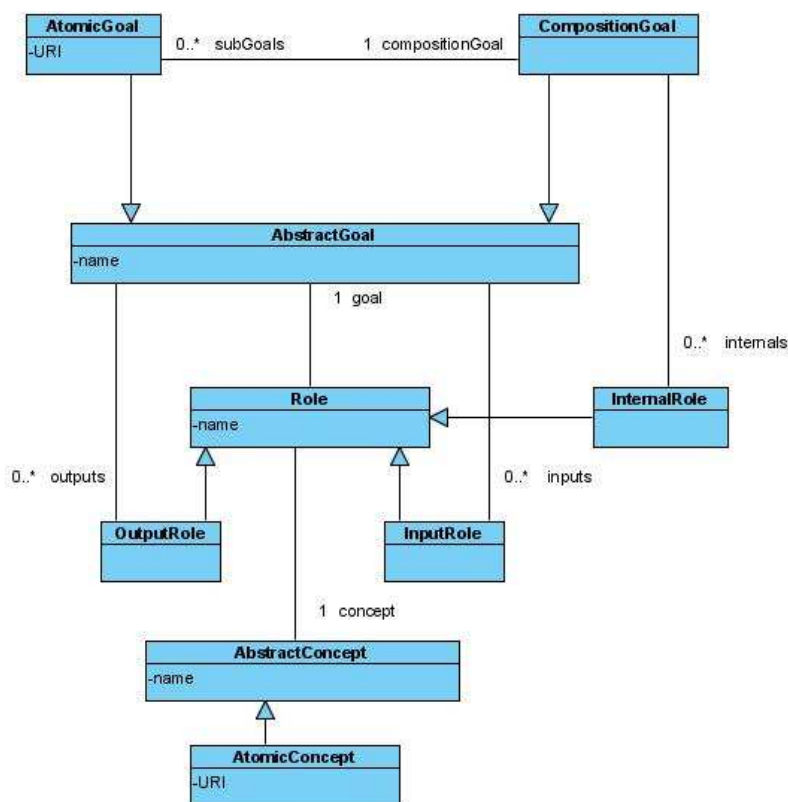


Figure 2.1: Goals, roles and concepts

Semantics

- Atomic goals: abstractions of SWS's. Used to perform discovery thus making the matching SWS's available for composition in the solution workflow.
- Roles: Inputs and outputs of matching SWS's. Internal roles can be used to denote intermediate objects in the orchestration. Each role has a concept taken from an ontology.

Z Constraints

- Goals to Roles relations reciprocity:

$$\begin{aligned} \forall n : AbstractGoal \bullet \\ & \quad inputs(n) = \{e : Role \mid e.goal = n\} \\ \forall n : AbstractGoal \bullet \\ & \quad outputs(n) = \{e : Role \mid e.goal = n\} \\ \forall n : AbstractGoal \bullet \\ & \quad internals(n) = \{e : Role \mid e.goal = n\} \end{aligned}$$

- Sources of dataflows must be of class OutputRole or InternalRole:

$$\begin{aligned} \forall n : Role \mid \#(n.isSourceOf) > 1 \bullet \\ & \quad n \subset OutputRole \vee \\ & \quad n \subset InternalRole \end{aligned}$$

- Targets of dataflows must be of class InputRole or InternalRole:

$$\begin{aligned} \forall n : Role \mid \#(n.isTargetOf) > 1 \bullet \\ & \quad n \subset InputRole \vee \\ & \quad n \subset InternalRole \end{aligned}$$

- Sources and targets of IdentityFlow, DecisionFlow and MergeFlow must share same concept:

$$\begin{aligned} \forall n : DataflowConstraint \mid \\ & \quad n \subset IdentityFlow \vee n \subset MergeFlow \vee n \subset DecisionFlow \bullet \\ & \quad sources(n).concept = targets(n).concept \end{aligned}$$

- DecisionFlow has only one source:

$$\begin{aligned} \forall n : DecisionFlow \bullet \\ & \quad \#(n.sources) = 1 \end{aligned}$$

- MergeFlow has only one target:

$$\begin{aligned} \forall n : MergeFlow \bullet \\ & \quad \#(n.targets) = 1 \end{aligned}$$

- ExtractionFlow has only one source, which concept is composite:

$$\begin{aligned} \forall n : ExtractionFlow \bullet \\ & \quad \#(n.sources) = 1 \end{aligned}$$

- AggregationFlow has only one target, which concept is composite:

$$\begin{aligned} \forall n : AggregationFlow \bullet \\ & \quad \#(n.targets) = 1 \end{aligned}$$

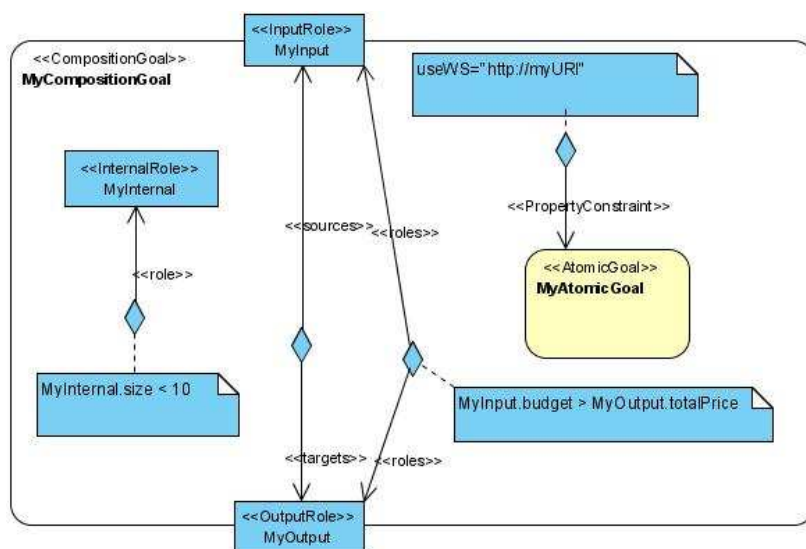


Figure 2.3: Graphical representation and stereotypes for Composition Goal constructs

2.7.2 Graphical representation

We provide a graphical representation based on UML2 Activity diagrams in order to draw composition goals in a user-friendly environment, using tools such as “Visual Paradigm for UML” or “Poseidon”. Figure 2.3 lists all constructs, stereotypes and their graphical notation. Concepts for roles are defined using text tagged value “concept”, and goals URI are defined using the tagged value “URI”.

It may be noted that the aforementioned elements are optional. It is not required in the general case for instance to attach ontological information to all of the roles, since this information is redundant with WSM, unless this is required for discovery.

2.8 Format of program results

GOBAC produces an orchestration that can be published to DIP using the WSM-AD extension. The orchestration can then be processed to generate a valid orchestration in the context of either DIP implementation platform (WSMX/IRSI).

The use of the WSMX platform for execution requires to generate the set of ASM rules that implement the choreography and orchestration. The specification of token flow semantics in the state of the art WSMX ASMs is not currently available (although under study), which postpones this possibility.

The use of the IRS-III platform as a service repository and execution platform requires that the choreographies be translated from Cashew to Activity Diagrams, which is detailed in D3.8, and that the orchestration produced be translated back to Cashew for execution, which we have prototypes of, and which will be detailed in D3.9.

2.9 Files attached to the current document

The DIP Deliverable D4.15 consists in the present document plus the following files:

1. *composition_GOBAC.zip*: a zip archive of all the sources needed to build the prototype
2. *VP_template.zip*: a zip archive of a Visual Paradigm for UML template file for building composition goals with appropriate stereotypes.

3 DETAILED LICENSE INFORMATION

This section describes the license and conditions granted by ILOG to test or use the prototype within the DIP project.

3.1 License

This deliverable has the dissemination level status.

ILOG grants a limited number of free licenses or the underlying ILOG products necessary to the testing of GOBAC by the relevant project's partners within the project timeframe.

3.2 How to obtain a license

For any licensing information regarding the ILOG JConfigurator commercial product embedded in the current prototype, please contact Christian de Sainte Marie at ILOG:

ILOG

9 rue de Verdun BP 85

94253 Gentilly France

4 DETAILED THIRD PARTY TOOL INFORMATION

4.1 JAVA SDK 1.5.0+

license

The Sun license for the Java SDK 1.5.0+ is available¹ at:

http://java.sun.com/j2se/1.5.0/jdk-1_5_0_04-license.txt

Third party licenses available² at:

http://java.sun.com/j2se/1.5.0/j2se-1_5_0-thirdpartyreadme.txt

obtain a license

The license is granted by accepting the proposed conditions at download time.

download

The Windows installer is available for download at the url³

<http://java.sun.com/j2se/1.5.0/download.jsp>

installation

Simply run the installer.

4.2 ILOG JCONFIGURATOR

license

This is a commercial product, sold or distributed under the terms of a commercial licence.

obtain a license

The product, and license, may be obtained from ILOG, via Christian de Sainte Marie, or Patrick Albert, who can be reached at the phone number +33149083500, or by mail at csma@ilog.fr, palbert@ilog.fr, or by post at

ILOG
9 rue de Verdun BP85
94253 Gentilly France

download

No download available.

¹Accessed in June 2006

²Accessed in June 2006

³Accessed in June 2006

installation

Simply execute the CD rom, that should start automatically then follow the instructions. For full compliance with the demoing instructions, the product should be installed in its default location: “C:/ILOG”. Commercial licenses entitle to full support.

4.3 WSMO4J, WSMO, WSMXI APIs

license and dowload

The license and download for WSMO4J is available at <http://wsmo4j.sourceforge.net/>. (Version required : wsmo4j-0.5.2) The license and download for WSMO API is available at <http://wsmo4j.sourceforge.net/>. (Version required : wsmo-api-0.5.2) The license and download for WSMX is available at <http://www.wsmx.org/>. (Version required : wsmx-integration-api-0.3)

4.4 Visual Paradigm for UML

license and download

The tool and license information is available at <http://www.visual-paradigm.com>. Visual paradigm proposes a free community license. The license allowing for XMI generation however is commercial.

5 TECHNICAL REQUIREMENTS FOR USING THE PROTOTYPE

The prototype is available for platforms running Windows XP, and requires at least 512MB of RAM. The program performance is satisfactory on machines having a 2 Giga-Hertz processor or faster. The machine must have at least 1 GB of free disk space if all components are to be installed (Java SDK inclusive).

5.1 JAVA SDK 1.5.0+

Running the prototype or the demo requires having installed a release of Java 1.5.0++.

5.2 ILOG JCONFIGURATOR

Running the prototype or the demo requires having installed a release of ILOG JConfigurator 2.1, under the patch level 7. Please contact Christian de Sainte Marie for any question regarding this product.

5.3 WSMO4J, WSMO and WSMX librairies

The composition prototype is implemented as an Eclipse RCP stand-alone application provided that the WSMO4J, WSMO and WSMX librairies are available.

6 DETAILED INFORMATION ON HOW TO USE/EVALUATE THE PROTOTYPE

6.1 Download

The prototype archives can be downloaded from the BSCW server, in the section dedicated to deliverable D4.15

6.2 Installation

The steps for installing the prototype are:

1. Install the J2SE 1.5.0 SDK (or higher) on your computer in a path containing NO SPACES.
2. Set your JAVA_HOME environment variable to your java sdk directory. (Under Windows : My Computer ⇒ properties ⇒ advanced ⇒ environment variables ⇒ system variables ⇒ new ⇒ JAVA_HOME)
3. Install JConfigurator 2.1 in the default path (Ilog/JConfigurator21)
4. Extract the GOBAC_composition.zip archive in the desired location.

You can launch the prototypes with the “composer.exe” command.
Further details regarding installation issues are provided in the deliverable archives.

7 ROADMAP ON FUTURE PLANS FOR FURTHER DEVELOPING THE PROTOTYPE

The current prototype is an important milestone in the process of developing a final prototype due at M36. The D4a.9 prototype was used to assess technical choices, to evaluate risks, and to raise important issues relative to its integration into the project. The three layer composition architecture for the DIP project integrating composition and immediate execution within the WSMO Studio Integrated Development Environment has been demonstrated at the second project review, and describes the infrastructure where all evolution of the prototype will take place.

7.1 Important Notice

The set of sources and executable programs attached to the current deliverable can be tested against the current state of the DIP projects as soon as the following two functionalities are available:

- The WSML parser accounts for the updated WSML grammar allowing to attach Activity Diagram based choreography and orchestration details to SWS. A first release of the WSML parser with these features is expected a end of July.
- A solution is offered in WSML to symbolically identify SWS message roles, as well as stating (and retrieving) a mapping between roles in goals and messages fulfilling those roles in SWS. An ongoing discussion about this topic is expected to extract from the ASM signature part information that could be shared across all the chor/orch formalisms (i.e. Activity Diagrams and Cashew). As long as this issue does not get fixed, the proposed composition goal language cannot be used to state role constraints, which is a limitation. This point is planned to be fixed for month 36.

8 USER MANUAL

The stand-alone application only consists in 3 steps :

- 1) Create a project
- 2) Import the XMI file specifying the composition goal (the file is populated in the project and can be viewed and edited as a text file)
- 3) Click on “compose” to populate the orchestration as a WSML-AD extension

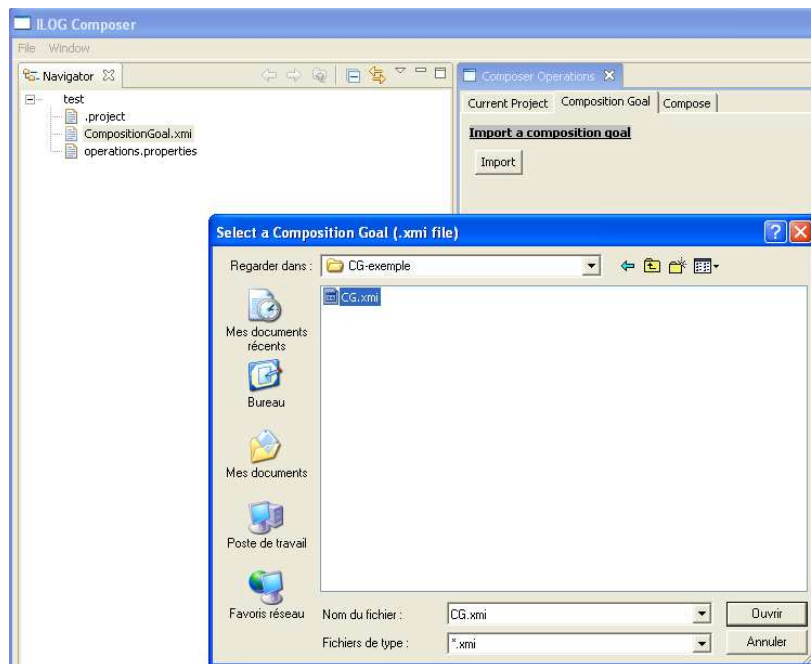


Figure 8.1: Importing the composition goal

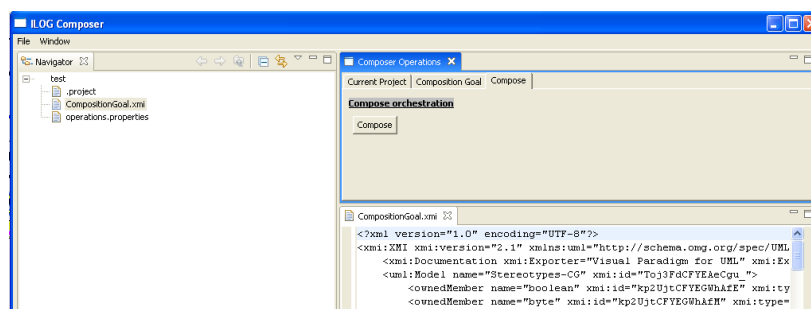


Figure 8.2: Compose the orchestration

9 REFERENCE MANUAL

The tool provides an entry point in order to integrate the composer into a java program.

```
public class Composer()  
public void C-Compose(String CG-path,String O-Path)
```

The Composer class provides methods for tweaking the composer solving preferences. Those will be further documented in the deliverable archives, and ILOG is willing to provide its help to adapt the product to specific partner requirements.

CG-path is an absolute path to a local file containing the composition goal in a XMI format, O-Path is the absolute path where the orchestration should be written.

10 DEMONSTRATION INFORMATION

10.1 How to obtain demo files

Prototype archives, as well as the powerpoint demo files are available from the DIP BSCW server, attached to the deliverable.

REFERENCES

- [1] Patrick Albert, Laurent Henocque, and Mathias Kleiner. Configuration based workflow composition. In *proceedings of International Conference on Web Services ICWS'05*, Orlando, Florida, USA, 2005.
- [2] Patrick Albert, Laurent Henocque, and Mathias Kleiner. A constrained object model for configuration based workflow composition. In *proceedings of the Third International Conference on Business Process Management*, page to appear, Nancy, France, september 2005.
- [3] V.R. Benjamins and D. Fensel. *Special Issue on Problem-Solving Methods. International Journal of Human-Computer Studies (IJHCS)*, 49(4):305–313, 1998.
- [4] M. Carman, L. Serafini, and P. Traverso. Web service composition as planning. In *proceedings of ICAPS03 International Conference on Automated Planning and Scheduling*, Trento, Italy, June 9-13 2003.
- [5] DIP Consortium. DIP Deliverable Annex DIO - Annex to the Choreography and Orchestration deliverables. Technical report, Data Integration Processes EU Integrated Project, 2005.
- [6] DIP Consortium. DIP Deliverable D3.4 - Choreography. Technical report, Data Integration Processes EU Integrated Project, 2005.
- [7] DIP Consortium. DIP Deliverable D3.4 - Orchestration. Technical report, Data Integration Processes EU Integrated Project, 2005.
- [8] DIP Consortium. DIP Deliverable D4.12 - Composition Module Specification. Technical report, Data Integration Processes EU Integrated Project, 2005.
- [9] DIP Consortium. DIP deliverable D3.8 - Choreography and Orchestration. Technical report, Data Integration Processes EU Integrated Project, 2006.
- [10] I. Constantinescu, B. Faltings, and W. Binder. Large scale, type-compatible service composition. In *proceedings of IEEE International Conference on Web Services (ICWS 2004)*, San Diego, USA, 2004.
- [11] R. Dijkman and M. Dumas. Service-oriented design: A multi-viewpoint approach, ctit technical report series no. 04-09. Technical report, Centre for Telematics and Information Technology, University of Twente, The Netherlands, February 2004.
- [12] Prashant Doshi, Richard Goodwin, Rama Akkiraju, and Kunal Verma. Dynamic workflow composition using markov decision processes. *International Journal of Web Services Research*, 2(1):1–17, Jan-March 2005.
- [13] G. Fleischanderl, G. Friedrich, A. Haselböck, H. Schreiner, and M. Stumptner. Configuring large-scale systems with generative constraint satisfaction. *IEEE Intelligent Systems - Special issue on Configuration*, 13(7), 1998.
- [14] Asunción Gómez-Pérez, Rafael González-Cabero, and Manuel Lamaa. A framework for design and composition of semantic web services. In *Semantic Web Services, 2004 AAAI Spring Symposium Series*, 22nd-24th March 2004.

- [15] Object Management Group. *UML v. 2.0 specification*. OMG, 2003.
- [16] Ulrich Junker and Daniel Mailharro. The logic of ilog (j)configurator: Combining constraint programming with a description logic. In *proceedings of Workshop on Configuration, IJCAI'03*, page to appear, 2003.
- [17] D. Mailharro. A classification and constraint based framework for configuration. *AI-EDAM : Special issue on Configuration*, 12(4):383 – 397, 1998.
- [18] S. McIlraith and T. Son. Adapting golog for composition of semantic web services. In *proceedings of Conference on Knowledge Representation and Reasoning*, April 2002.
- [19] S. Mittal and B. Falkenhainer. Dynamic constraint satisfaction problems. In *Proceedings of AAAI-90*, pages 25–32, 1990.
- [20] B. Nebel. Reasoning and revision in hybrid representation systems. *Lecture Notes in Artificial Intelligence*, 422, 1990.
- [21] Marco Pistore, F. Barbon, Piergiorgio Bertoli, D. Shaparau, and Paolo Traverso. Planning and monitoring web service composition. In *proceedings of the Workshop on Planning and Scheduling for Web and Grid Services held in conjunction with ICAPS 2004*, Whistler, British Columbia, Canada, June 3-7 2004.
- [22] J. Rao, P. Kungas, and M. Matskin. Logic-based web service composition: from service description to process model. In *proceedings of the 2004 IEEE International Conference on Web Services, ICWS 2004*, San Diego, California, USA, July 6-9 2004.
- [23] E. Sirin, J. Hendler, and B. Parsia. Semi automatic composition of web services using semantic descriptions. In *proceedings of the ICEIS-2003 Workshop on Web Services: Modeling, Architecture and Infrastructure*, Angers, France, April 2003.
- [24] Evren Sirin, Bijan Parsia, Dan Wu, James Hendler, and Dana Nau. HTN planning for web service composition using SHOP2. *Journal of Web Semantics*, 1(4):377–396, 2004.
- [25] T. Soinen, I. Niemelö, J. Tiihonen, and R. Sulonen. Unified configuration knowledge representation using weight constraint rules. In *ECAI 2000 Configuration Workshop*, 2000.
- [26] Markus Stumptner. An overview of knowledge-based configuration. *AI Communications*, 10(2):111–125, June 1997.
- [27] S. Thakkar, C.A. Knoblock, J.L. Ambite, and C. Shahabi. Dynamically composing web services from on-line sources. In *proceedings of AAAI-02 Workshop on Intelligent Service Integration*, Edmondson, Canada, July 2002.
- [28] M. Vukovic and P. Robinson. Adaptive, planning based, web service composition for context awareness. In *proceedings of the Second International Conference on Pervasive Computing*, page to appear, Vienna, Austria, 2004.

- [29] X. Yi and K. Kochut. A cp-nets-based design and verification framework for web services composition. In *proceedings of 2004 IEEE International Conference on Web Services, July 2004*, San Diego, California, USA, July 6-9 2004.