NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 – "Semantic-based knowledge and content systems"

D5.3.3. gOntt plug-in for Scheduling Ontology Projects

Deliverable Co-ordinator: **Mari Carmen Suárez-Figueroa**

Deliverable Co-ordinating Institution: **UPM**

Other Authors: **Asunción Gómez-Pérez (UPM), Oscar Muñoz (UPM).**

This deliverable presents the main functionalities of the gOntt plug-in, as well as the main technical aspects of such a plug-in.

Document Identifier:	NEON/2010/D5.3.3/v1.0	Date due:	January 31st , 2010
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	January 31st , 2010
Project start date:	March 1, 2006	Version:	v1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

This document is a part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Enrico Motta E-mail address: e.motta@open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 28 D-76128 Karlsruhe, Germany Contact person: Andreas Harth E-mail address: aha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.upm.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SI-1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 655 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Koblenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Martino della Battaglia, 44 - 00185 Roma-Lazio, Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 1 00100 Rome Italy Contact person: Caterina Caracciolo E-mail address: Caterina.Caracciolo@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to the writing of this document or its parts:

UPM

Change Log

Version	Date	Amended by	Changes
0.1	3-02-2010	Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez	First version of introduction
0.2	3-02-2010	Mari Carmen Suárez-Figueroa	First version of chapter 2
0.3	4-02-2010	Mari Carmen Suárez-Figueroa Oscar Muñoz	Update of chapter 2 First version of chapter 3
0.4	5-02-2010	Mari Carmen Suárez-Figueroa	General revision and update
0.5	5-02-2010	Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez	First version of conclusion
0.6	8-02-2010	Mari Carmen Suárez-Figueroa, Oscar Muñoz	Update of chapter 2
0.7	8-02-2010	Mari Carmen Suárez-Figueroa	General revision and update

Executive Summary

The scope and main contributions of this deliverable are:

1. The main functionalities of the gOntt plug-in, which is the technological support for the scheduling activity in the NeOn Toolkit.
2. The summary of the main technical aspects of the gOntt plug-in.

Table of Contents

List of Tables

List of Figures

1. Introduction

Planning and scheduling are related activities that are applied in different contexts such as civil engineering, software development, etc. While planning[1] is the act of drawing up plans, which are a series of steps to be carried out to achieve an objective, scheduling[2] is defined as the activity to set order and time to planned events. Scheduling should be performed after planning; and both are crucial activities in any development project.

In Software Engineering, every development project has a life cycle [1], which is produce by instantiating a particular life cycle model. Life cycle models can be seen as abstractions of the phases or stages through which a product passes along its life. Examples of life cycle models are [2, 3]: waterfall, incremental, iterative, evolutionary prototyping, and rapid throwaway prototyping.

To properly manage software development projects, it is crucial to have knowledge of the entire software development life cycle [4]. Software engineers always plan and schedule every development project before starting it. The project plan defines the tasks to be done and the actors to perform them. To estimate the effort required to perform each task, techniques such as [4] Wideband Delphi, PROBE and COCOMO II can be used.

The project schedule is a calendar that links the tasks to be done with the resources to support their performance. The most common form of schedules is a Gantt chart [4]; and the most popular tool for creating a project schedule is the Microsoft Project [4].

However, unlike what happens in Software Engineering, in the Ontology Engineering field planning and scheduling ontology developments are still in their early stages. Only METHONTOLOGY [5] defines the scheduling activity, but it does not provide guidelines for helping ontology developers to plan and schedule their project. Other methodologies, such as On-To-Knowledge [6] and DILIGENT [7], do not include such activities in their developments. Regarding the calculation of cost estimation of ontology projects, the only existing technique is ONTOCOM [8, 9], a cost estimation model whose goal is to predict the costs based on the total number of person months needed for building the ontology. In that sense, the ONTOCOM model does not provides details of the cost associated with a particular task in an ontology development project. So, *the Ontology Engineering field lacks methods to guide ontology developers in planning and scheduling their ontology projects*. Additionally, *there is no support tool for providing ontology developers with project schedules in the form of a Gantt chart*.

During the last years, the ontology building has evolved from the development of single ontologies towards the development of ontology networks. Ontology networks are built collaboratively by geographically distributed teams by reusing and re-engineering as much as possible knowledge-aware resources (thesauri, lexicons, databases, UML diagrams, etc.). In this scenario, planning and scheduling activities are extremely important in order to set order and time to the activities involved in the ontology development with the aim of easing their execution.

Thus, within the NeOn project we propose a NeOn Toolkit plug-in, called **gOntt**, for supporting end-users in carrying out the scheduling activity for their ontology development projects.

This deliverable is structured as follows:

- Chapter 2 presents the main functionalities of gOntt plug-in.
- Chapter 3 summarizes the main technical aspects of gOntt plug-in.
- Chapter 4 includes the conclusions and the future work.

[1] wordnet.princeton.edu/perl/webwn

[2] wordnet.princeton.edu/perl/webwn

2. gOntt Functionalities

gOntt is the NeOn Toolkit plug-in to be used to schedule ontology development projects in the form of a Gantt chart following the guidelines presented D5.3.2 [10]. gOntt **main objectives** are

1. To support ontology practitioners to decide which ontology network life cycle model is the most appropriate for building their ontologies.
2. To help ontology practitioners to decide which concrete process and activities should be carried out in the ontology network development and in which order.
3. To instantiate the life cycle model selected and to create a particular life cycle for the ontology development with the processes and activities needed, including time restrictions between processes and activities.
4. To inform ontology practitioners about how to carry out a particular process or activity, including the NeOn methodological guidelines and a reference to the concrete NeOn plug-in to be used.

Based on the aforementioned objectives, gOntt functionalities can be divided in three groups: (1) functionalities for scheduling an ontology network development (Section 2.1), (2) functionalities for informing ontology developers about methodological guidelines (Section 2.2), and (3) functionalities for launching other NeOn Toolkit plug-ins during the ontology network development (Section 2.3).

2.1. Functionalities for scheduling an ontology network development

Here we present the list of functionalities provided by gOntt for helping ontology developers in the scheduling activity.

- To *create particular schedules from scratch*, by allowing the ontology practitioner the inclusion of processes, activities, phases, and relationships and restrictions between them. Such processes and activities could either come from the NeOn Glossary of Processes and Activities or be new ones proposed by the ontology developer.
- To *create particular schedules in a guided way*. The ontology practitioner uses wizard menus to select the ontology life cycle model and to select processes and activities. Having selected the model and the set of processes and activities needed, gOntt automatically provides the ontology practitioner with an initial plan.

  Figure 1 shows these two possibilities for creating a particular schedule (from the scratch or in a guided fashion).

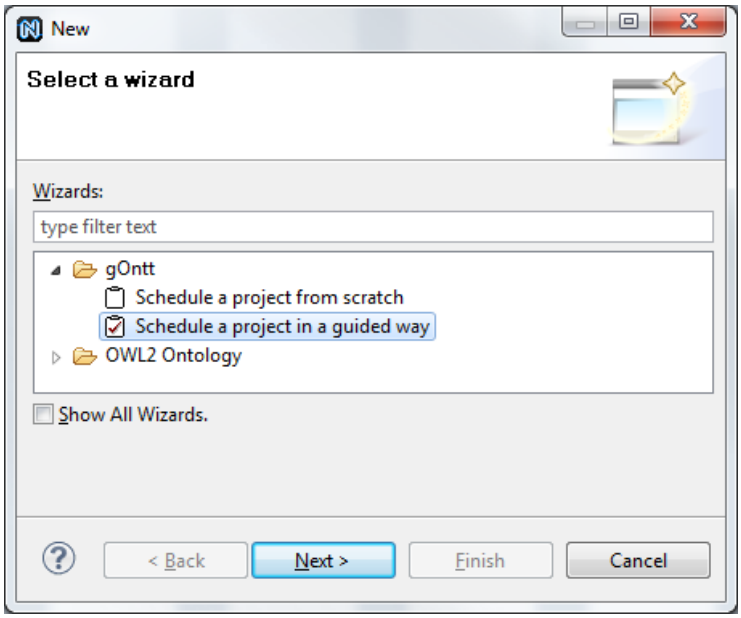

Figure 1. The two scheduling options: (1) to schedule a project from scratch and (2) to schedule a project in a guided way

In the second option (that is, *to schedule a project in a guided way*), to help ontology practitioners to decide which the most appropriate life cycle model is among those presented in D5.3.2 [10], the natural language question shown in Figure 2 is presented by gOntt. Based on the ontology developer answer, gOntt selects the waterfall model or the iterative-incremental model.

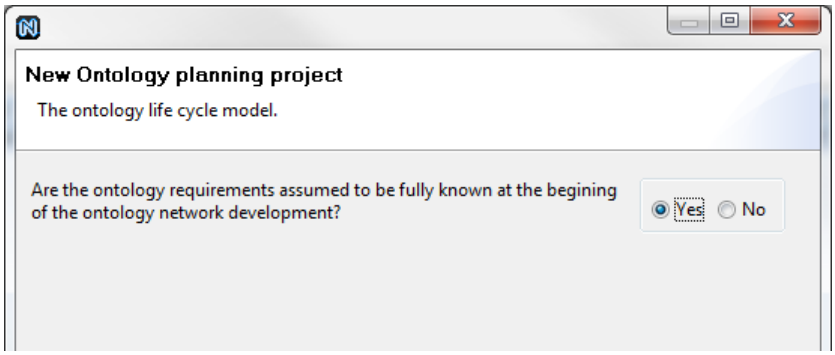

Figure 2. Natural language question to select the life cycle model

To help the ontology practitioner to select a particular model version[3] from those included in D5.3.2 [10], the set of natural language questions displayed Figure 3 are presented. These questions are related to the different scenarios identified in the NeOn Methodology [11]. To answer such questions the ontology developer takes into account the ontology requirements and available knowledge resources, selected as candidate to be reused. For this reason, both activities (requirements specification and quick search of resources) should be carried out before the scheduling one as already mentioned in the NeOn Methodology. Based on ontology developer answers to questions presented in Figure 3, gOntt decides which one of the waterfall model versions is the most appropriate. If ontology developers answer affirmatively one or more questions of those proposed in Figure 3, this

[3] It is worth mentioning that if the iterative-incremental model is selected, it is necessary to decide among the different versions of waterfall model by means of answering these natural language questions in each iteration.

means that several candidate models could be used. In that case, the model version selected should be the most specific one based on the pyramid[4] shown in Figure 4. Otherwise, if all answers are negative, then the four-phase waterfall model is selected by default.

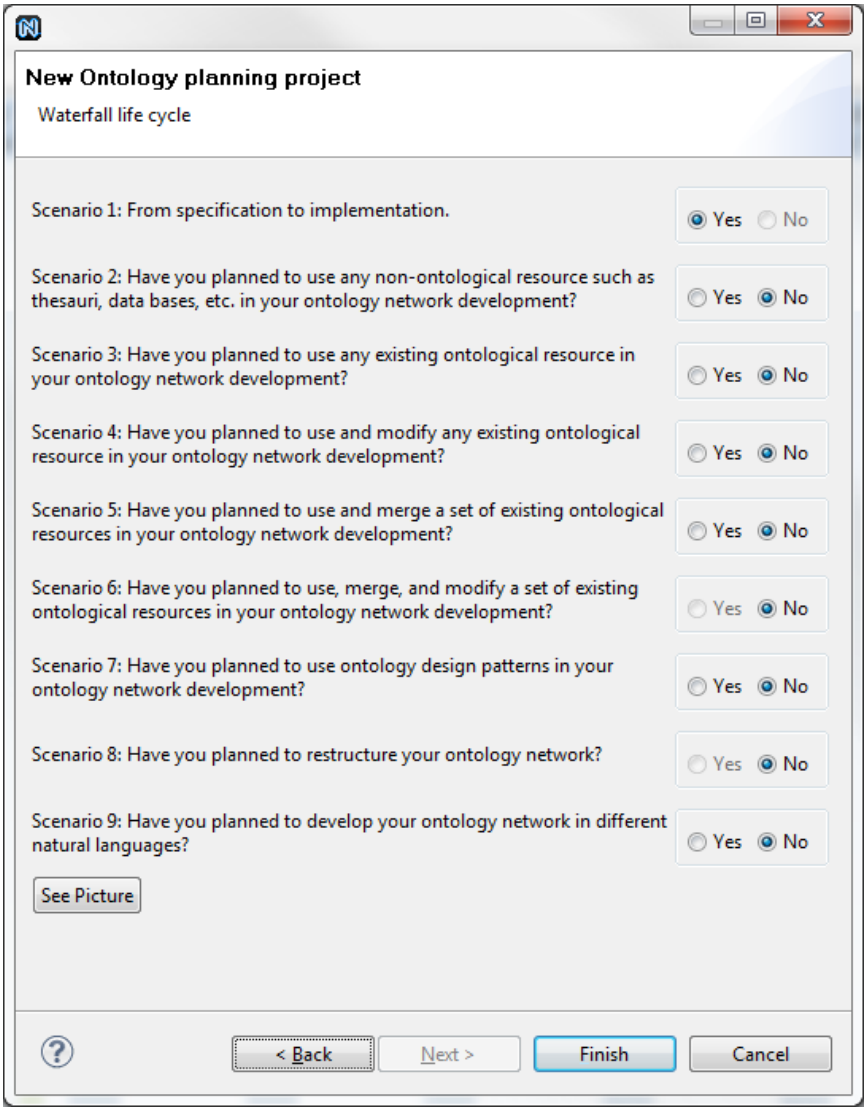

Figure 3. Schedule a project in a guided way: questions related with scenarios

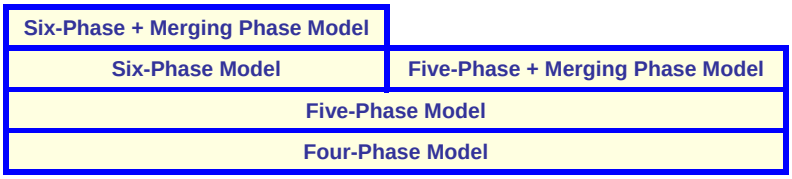

Figure 4. Pyramid of the versions of the Waterfall ontology network life cycle model

Based on ontology developer answers to questions presented in Figure 3, gOntt also identifies the scenarios involved in the ontology network development. The relation between processes

[4] The different versions of the waterfall model have been created incrementally, that is, four-phase is the basis for five-phase, five-phase is the basis for six-phase, etc.

and activities involved in the ontology development and the scenarios proposed by the NeOn Methodology has been identified in D5.3.2 [10]. Using such a relation between processes and activities and scenarios, gOntt is able to select the set of processes and activities to be performed during the ontology network development.

After having selected both the ontology life cycle model and the processes and activities need for the ontology network development, gOntt automatically generates the initial plan for the development. To achieve this functionality, gOntt uses a set of templates. Templates show a default plan based on the different possible combinations between life cycle models and processes and activities. Figure 5 shows one of the gOntt templates for the case in which the model is the six-phase waterfall and the scenarios involved are scenarios 2, 3, 7, and 9. This functionality of generating default plans provides a great advantage with respect to existing tools for scheduling software development projects (e.g. MS Project). It is worth mentioning that initial plans provided by gOntt can be modified by the user.

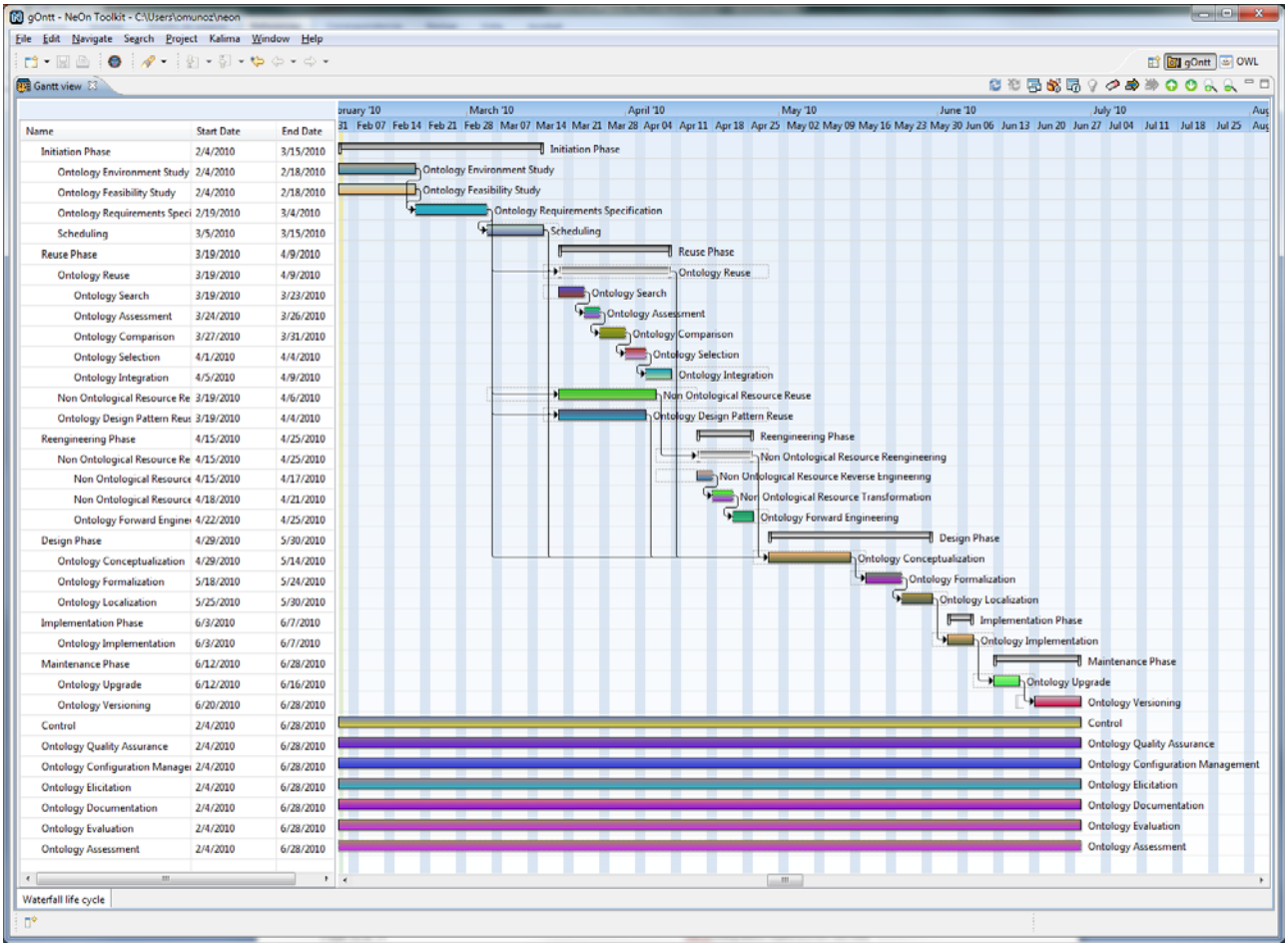

Figure 5. gOntt template that involves scenarios 2, 3, 7 and 9

- To create, modify, and delete gOntt projects.
- To save and open gOntt projects in an extension (.got) based on an xml standard.
- To provide graphical and textual visualizations of gOntt projects.
- To delete processes, activities and phases from a gOntt project.
- To modify the order of processes, activities and phases in a gOntt project.

- To create, modify and delete connections between activities, between processes, and between activities and processes. If a connection exists between two elements, the second one cannot start until the first one finishes.

  Figure 6 shows an example of this behaviour. In such a figure, the *Non Ontological Resource Reverse Engineering activity* is connected to the *Non Ontological Resource Transformation activity*; this means that the later activity needs as an input the output of the former one. In these cases, gOntt does not allow the first activity to finish after the second activity starts or the second activity starts before the first activity ends. A similar situation occurs between the *Non Ontological Resource Reengineering* process and the *Ontology Conceptualization* activity, as also shown in Figure 6.

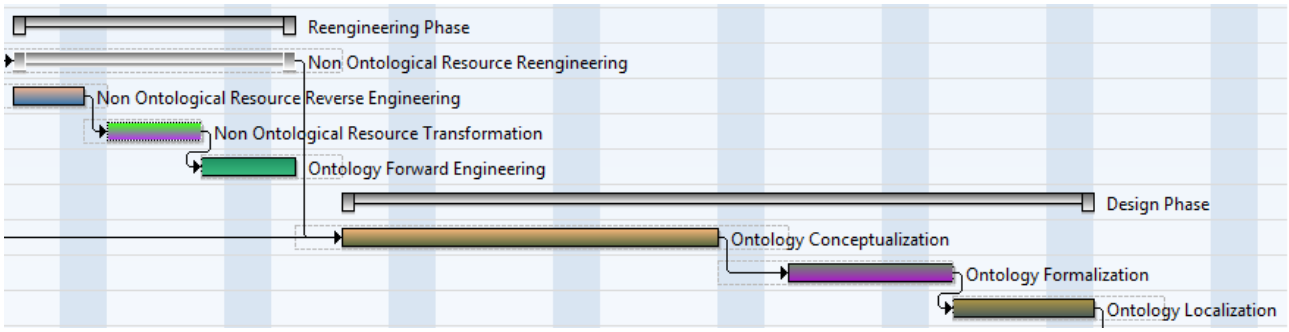

Figure 6. Example of activity and process limits

- To include and modify duration and starting date of processes, activities and phases.

2.2. Functionalities for informing about methodological guidelines

In this section, we explain the functionalities provided by gOntt to inform ontology developers about methodological guidelines useful for the ontology network development.

- gOntt displays a **filling card** including the process or activity definition, its goal, inputs and outputs, who carries it out, and when it should be done. As an example, Figure 7 shows the filling card for the ontology requirements specification activity.

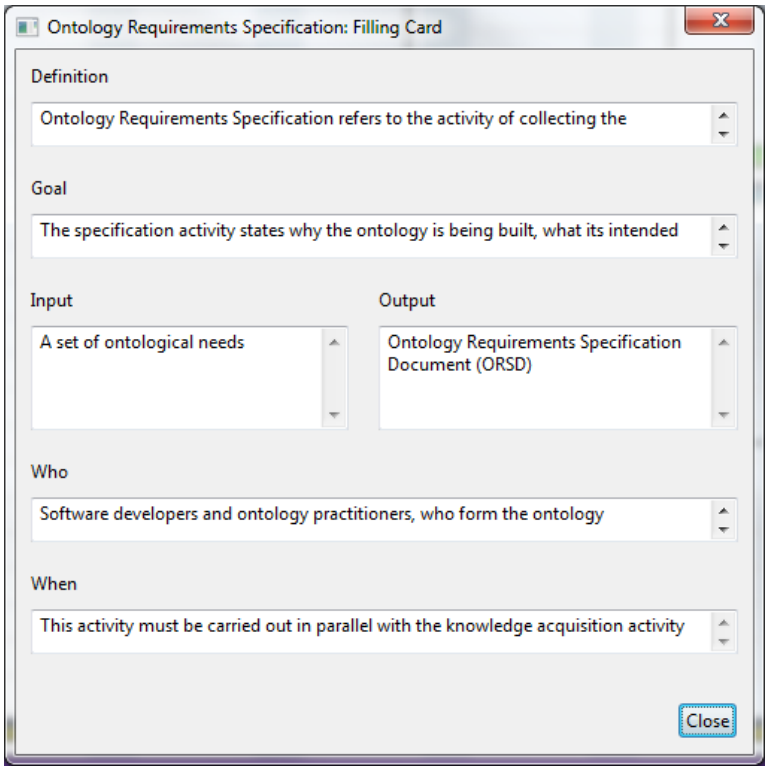

Figure 7. Filling Card of the Ontology Requirements Specification activity

- **Methodological guidelines** on how to perform the process or activity. At this moment, the following processes and activities of the NeOn Methodology have a workflow that explains how to carry them out in a prescriptive way: ontology requirements specification, scheduling, reusing and re-engineering non-ontological resources, reusing ontological resources, reusing ontology design patterns, ontology localization, ontology modularization, ontology evaluation and ontology evolution. Workflows are implemented with Eclipse Cheat Sheets[5]. Figure 8 presents the methodological guidelines for the ontology requirements specification activity.

[5] http://www.ibm.com/developerworks/opensource/library/os-ecl-cheatsheets

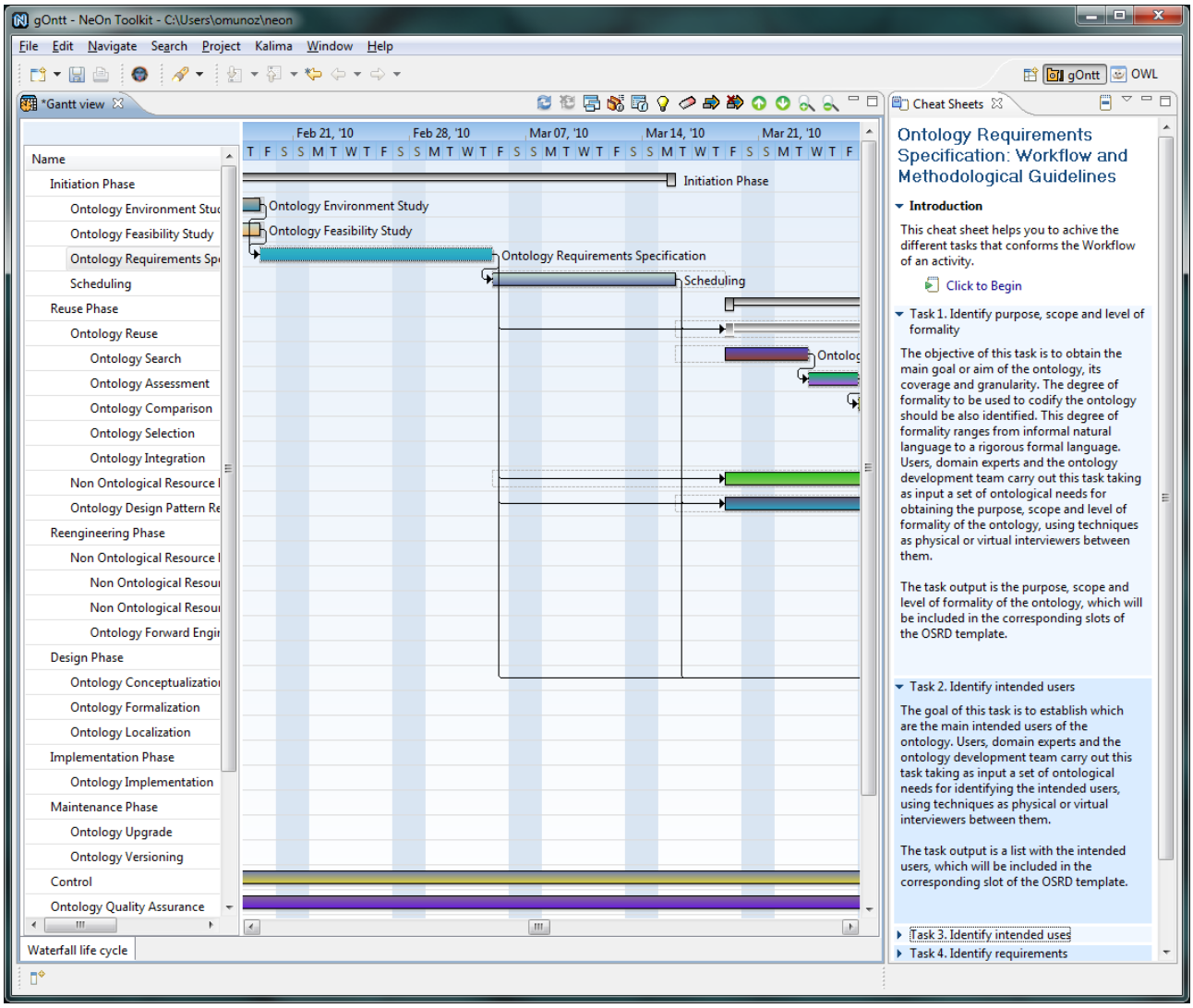

Figure 8. Methodological Guidelines of the Ontology Requirements Specification activity

2.3. Functionality for providing a direct access to NeOn Toolkit plug-ins

In this case, the idea is that gOntt provides a direct access to the NeOn Toolkit plug-ins associated to each process and activity planned. This means that gOntt triggers the different NeOn Toolkit plug-ins associated to each process or activity included in the plan made using gOntt.

To make this possible, the gOntt plug-in has to have some kind of communication with all other NeOn Toolkit plug-ins and the best way to do it is using *extension points*. gOntt has its own extension point that the other NeOn Toolkit plug-ins should implement (See Section 3.2 for more detail).

Using this extension point other NeOn Toolkit plug-ins can easily 'advertise' (that is, expose) their supported activities to gOntt. As an example, Figure 9 shows the plug-ins available for the *Ontology Implementation activity*.

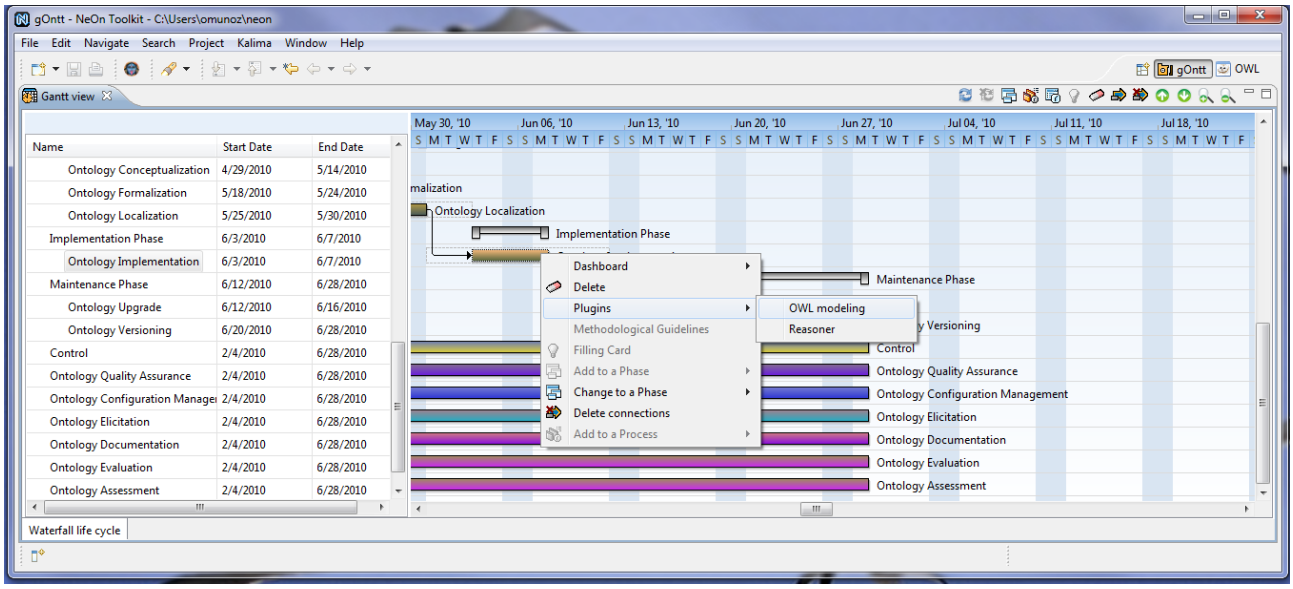

Figure 9. Plug-ins available for the Ontology Implementation activity

In addition, gOntt displays a quick-start guide for using the plug-in launched whenever the plug-in developers have specified a cheat-sheet describing this guide.

Thus, gOntt provides a direct and automatic association among processes and activities and the NeOn Toolkit plug-ins that could be used for their execution.

3. gOntt plug-in Technical Description

This chapter summarizes the gOntt plug-in at technical level from the perspective of its dependencies mainly with the NeOn Toolkit and Eclipse frameworks (Section 3.1) as well as the extension point mechanism used for other NeOn plug-ins to be integrated with gOntt (Section 3.2).

3.1. Plug-in dependencies

gOntt requires the following plug-ins from the ***Eclipse framework***:

- *org.eclipse.ui* for extending views, perspectives, wizards, actions and menus.
- *org.eclipse.ui.cheatsheets* for the methodological guidelines.
- *org.eclipse.core.runtime*
- *org.eclipse.ui.ide*
- *org.eclipse.core.resource*

gOntt is integrated with the NeOn Toolkit by storing scheduling projects into the NeOn Toolkit workspace. That is, for every ontology network development project there will be always an associated scheduling. gOntt requires the following plug-ins from the ***NeOn Toolkit framework***:

- *com.ontoprise.ontostudio.owl.gui* is used to retrieve the selected project name from the ontology navigator.
- *org.neontoolkit.core* is used to retrieve the complete path of the project inside the file system.
- *org.neontoolkit.gui.navigator* is used to retrieve the selected element from the ontology navigator.
- *org.neontoolkit.gui.navigator.elements* is used to know the type of the element selected in the ontology navigator.
- *org.neontoolkit.io* is used to retrieve the texts shown in the import and export wizards.
- *org.neontoolkit.filter* is used to retrieve the filters used in the import and export wizards.

In addition, the plug-in internally includes the following ***libraries***:

- **Gantt Chart Widget**[6]. The SWT GANTT Chart Widget is a customizable GANTT chart widget written in Java for SWT/JFace applications. It is used by gOntt to display the Gantt diagrams.
- **SWTCalendar**[7]. It is a GUI date picker for Java using SWT as the GUI toolkit. It is embedded in the gOntt dialogs for selecting dates (e.g. when creating a new activity).

3.2. gOntt extension point definition

gOntt has its own extension point that should be extended by NeOn Toolkit plug-ins to allow the activation of such plug-ins when the user is working in gOntt. In order to make the functionality of a

[6] http://hexapixel.com/software/ganttwidget

[7] http://swtcalendar.sourceforge.net/

given plug-in available from gOntt, no additional code needs to be created. Simply a few lines to the *plug-in.xml* providing only a small amount of information should be added. gOntt just needs to know:

- The plug-in name.
- The id of the perspective the plug-in works with. The perspective to show when you launch the plug-in.
- The id of the view the plug-in works in.
- The id of the activity that is associated to the plug-in.
- A help showing how to start the plug-in. This help should be in the form of Eclipse Cheat Sheet.

Next, two examples of gOntt extension point implementations are shown.

```
<extension point="org.neontoolkit.upm.gontt">
  <plug-in
    name= "Watson Plug-in"
    perspectiveId="com.ontoprise.ontostudio.owl.perspectives.OWLPerspective"
    viewId="uk.ac.open.kmi.watson.neontoolkitplug-in.WatsonResultsView"
    activityId="ontologyReuse">
  </plug-in>
</extension>
<extension point="org.neontoolkit.upm.gontt">
  <plug-in
    name= "Label Translator"
    perspectiveId="com.ontoprise.ontostudio.perspectives.Schema"
    viewId="com.ontoprise.ontostudio.views.navigator"
    activityId="ontologyLocalization"
    plug-inHelpId= "org.neontoolkit.xxx.cheatsheet.ontologyLocalizationIntro" >
  </plug-in>
</extension>
```

Table 1 shows the extension points specified for gOntt. For each plug-in in the table the activities or processes that it covers are shown. The table also shows if the quick-start guide is implemented by a given plug-in.

Table 1. Extension points for gOntt

NeOn Toolkit Plug-in	Partner	Cheatsheet for quick-star guide	Process or Activity
OWLDoc	UPM	Yes	Ontology Documentation
ODEMapster	UPM	Yes	Ontology Population
LabelTranslator	UPM	Yes	Ontology Localization
Oyster	UPM	Yes	Ontology Reuse
Customization	UKO-LD	No	Ontology Customization
SAIQL	UKO-LD	No	Ontology Verification
Cicero	UKO-LD	Yes	Ontology Specification / Ontology Customization / Ontology Formalization / Ontology Implementation
OntoAtlas	JSI	No	Ontology Population / Non Ontological

NeOn Toolkit Plug-in	Partner	Cheatsheet for quick-star guide	Process or Activity
			Resource Reuse / Ontology Verification
OntoConto	JSI	No	Ontology Aligning
SearchPoint/Watson	JSI	No	Ontology Search
Cyc Question Answering	JSI	No	Ontology Environment Study
Alignment	INRIA	No	Ontology Aligning
Module Partition	OU	No	Ontology Partitioning
Module Extraction	OU	No	Ontology Module Extraction
Cupboard	OU	No	Ontology Reuse
Watson for Knowledge Reuse	OU	Yes	Ontology Reuse
Evolva	OU	No	Ontology Evolution
Key Concept	OU	No	Ontology Summarizarion / Ontology Assesment / Ontology Evaluation
Gate Webservice	USFD	No	Ontology Elicitation
COAT	USFD	No	Ontology Enrichment
OWL Ontology Visualiser	ISOCO	No	Ontology Documentation
I2Ont	ISOCO	No	Non Ontological Resource Reuse
Module Composition	UKARL	No	Ontology Merging
RaDON	UKARL	No	Ontology Diagnosis / Ontology Repair
SPARQL	UKARL	Yes	Ontology Verification
Reasoner	UKARL	Yes	Ontology Implementation
EII XML Import	SAG	Yes	Non Ontological Resource Reengineering
Datasource	SAG	No	Non Ontological Resource Reengineering
EII Ontology Versioning	SAG	Yes	OntologyVersioning
Semantic EII	SAG	No	Ontology Integration
XDTools	CNR	No	Ontology Annotation / Ontology Reuse / Ontology Selection
Core OWL Plug-in	ONTO	No	Ontology Implementation / Ontology Formalization / Ontology Population

4. Conclusions and Future Work

To properly manage ontology development projects in complex settings either in academia or industry, it is crucial to have knowledge of the entire ontology development life cycle before starting the developments. The ontology project plan defines the tasks to be done, the time when they will be executed and the dependencies between tasks. The project plan is the only way, as done in other disciplines (e.g., civil engineering, software engineering, etc.), to commit people to the project and to show how the work will be performed. It also helps the ontology engineer to monitor its execution and assess the impact of a particular delay in planned tasks.

Thus, in this deliverable, we explain the technological infrastructure that supports the automatic generation of the initial ontology development plan in the form of a Gantt chart which is integrated within the NeOn Toolkit. Such a technological infrastructure is a NeOn Toolkit plug-in called **gOntt**[8] for supporting the scheduling activity, based on the methodological guidelines presented in D5.3.2 [10]. Additionally, gOntt provides methodological and technical help to ontology practitioners during ontology development project executions.

As future work we have plan to include in the gOntt plug-in the possibility of (1) establishing human resources restrictions and establishments and (2) including the history of the development, i.e., percentage of process or activity that has been done. We also plan to integrate the proposed guidelines and gOntt with the works done by the ONTOCOM team for predicting the total costs of the ontology development project. We also plan to extend such works by providing details of the cost associated to carry out a particular task in an ontology development project.

[8] gOntt version 1.4 is available in the NeOn Toolkit update site for been installed in NeOn Toolkit v2.3 (or later versions).

References

1. Taylor, J.C.: Project Scheduling and Cost Control: Planning, Monitoring and Controlling the Baseline. J. Ross Publishing, 2008, ISBN: 9781932159110

2. Pfleeger, S. L.: Software Engineering: Theory and Practice. 2nd edition. Prentice Hall 2001. ISBN: 0-13-029049-1

3. Sommerville, I.: Software Engineering. Eighth Edition 2007. Addison-Wesley. ISBN: 0-321-31379-8

4. Stellman, A., Greene, J.: Applied Software Project Management. ISBN: 0-596-00948-8

5. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering. Springer Verlag. Advanced Information and Knowledge Processing series. ISBN 1-85233-551-3. November 2003

6. Staab, S., Hans, P., Studer, R., Sure, Y.: Knowledge Processes and Ontologies. IEEE Intelligent Systems 16(1):26–34. (2001)

7. Pinto, H.S., Tempich, C., Staab, S.: DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. 16th European Conference on Artificial Intelligence (ECAI 2004), pp. 393--397

8. Simperl, E., Popov, I., Bürger, T.: ONTOCOM Revisited: Towards Accurate Cost Predictions for Ontology Development Projects. In: Proceedings of the European Semantic Web Conference 2009 (ESWC '09), Heraklion, Greece, May 20 - Jun, 04, 2009

9. Paslaru-Bontas, E., Tempich, C., Sure, Y.: ONTOCOM: A Cost Estimation Model for Ontology Engineering. Proceedings of the 5th International Semantic Web Conference (ISWC 2006). Volume 4273. Lecture Notes in Computer Science (LNCS), pp. 625--639. Springer-Verlag Berlin Heidelberg, 2006.

10. Suárez-Figueroa, M. C., Fernández-López, M., Gómez-Pérez, A., Dellschaft, K., Lewen, H., Dzbor, M.: NeOn D5.3.2. Revision and Extension of the NeOn Development Process and Ontology Life Cycle. NeOn Project (http://www.neon-project.org). November 2008.

11. Suárez-Figueroa, M.C., Gómez-Pérez, A.: NeOn Methodology for Building Ontology Networks: a Scenario-based Methodology. International Conference on Software, Services and Semantic Technologies (S3T 2009). Proceedings of the International Conference on Software, Services and Semantic Technologies (S3T 2009). ISBN: 978-954-9526-62-2. Pages: 160-167. Sofía, Bulgaria. 28-29 October 2009.