Chapter 14
Scheduling Ontology Engineering Projects Using gOntt

Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Oscar Muñoz-García

Abstract In order to manage properly ontology development projects in complex settings and to apply correctly the NeOn Methodology, it is crucial to have knowledge of the entire ontology development life cycle before starting the development projects. The ontology project plan and scheduling helps the ontology development team to have this knowledge and to monitor the project execution. To facilitate the planning and scheduling of ontology development projects, the NeOn Toolkit plugin called gOntt has been developed. gOntt is a tool that supports the scheduling of ontology network development projects and helps to execute them. In addition, prescriptive methodological guidelines for scheduling ontology development projects using gOntt are provided.

14.1 Introduction

One of the crucial aspects within engineering processes is the issue of planning and scheduling development projects. These two terms are often thought of as synonymous; however, they are not. While planning[1] is the act of drawing up a plan, that is, a series of steps to be carried out to achieve an objective, scheduling[2] is defined as the activity of placing planned events along a timeline. Scheduling clearly depends on planning, and both are crucial in any project.

Bearing in mind that ontologies are part of software products and that sometimes ontologies are considered a kind of software, experiences and practices in software

[1] http://www.wordnet-online.com/planning.shtml

[2] http://www.wordnet-online.com/scheduling.shtml

M.C. Suárez-Figueroa (✉) • A. Gómez-Pérez • O. Muñoz-García
Ontology Engineering Group, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo sn., 28660 Boadilla del Monte, Madrid, Spain
e-mail: mcsuarez@fi.upm.es; asun@fi.upm.es; omunozgarcia@gmail.com

M.C. Suárez-Figueroa et al. (eds.), *Ontology Engineering in a Networked World*,
DOI 10.1007/978-3-642-24794-1_14,

engineering can be adopted and adjusted in the ontology engineering community. For this reason and with the aim of achieving in ontology engineering a similar degree of maturity to that of the software engineering field, we take as basis software engineering works to provide ontology engineers with help in planning and scheduling ontology development projects.

In software engineering, every development project has a life cycle (Taylor 2008), which is produced 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 waterfall (Royce 1970), incremental (Sommerville 2007), iterative (Pfleeger 2001), evolutionary prototyping (Davis et al. 1988), and rapid throwaway prototyping (Davis et al. 1988).

To properly manage software development projects, it is crucial to have knowledge of the entire software development life cycle (Stellman and Greene 2005). In this regard, software engineers always plan and schedule every development project before starting it. The project plan defines the tasks to be carried out and establishes the human resources to perform the project work. To estimate the effort required to perform each task, techniques such as (Stellman and Greene 2005) Wideband Delphi, PROBE, and COCOMO II can be used.

The project schedule is a calendar that links the tasks to be performed with the resources to support their performance. One of the most common forms of representing schedules is to use a Gantt chart (Gantt 1974); and the most popular tool for creating a project schedule is Microsoft Project, according to (Stellman and Greene 2005). However, according to our knowledge, any tool for managing project schedules provides guidelines on how to execute the project.

Ontologies are used for making knowledge explicit and allowing it to be shared. One of the keys when building ontologies as in the case of software products is to plan and schedule the ontology development. However, in ontology engineering, planning and scheduling are still in their early stages. Only METHONTOLOGY (Fernández-López et al. 1997; Blázquez et al. 1998) defines the scheduling activity, but it does not provide guidelines for helping ontology developers to plan and schedule their projects. Other methodologies, such as On-To-Knowledge (Staab et al. 2001) and DILIGENT (Pinto et al. 2004), do not include these activities in their developments. Regarding the calculation of cost estimation of projects, the only technique available is ONTOCOM (Simperl et al. 2009), a model that predicts the costs of ontology development projects.

To cover this lack of methods and tools for planning and scheduling, this chapter describes (a) the gOntt plugin, a tool that supports the scheduling of ontology network development projects and helps to execute them, and (b) prescriptive methodological guidelines for scheduling ontology development projects using gOntt.

14.2 Scheduling Ontology Development Projects

Scheduling, as defined in the NeOn Glossary of Processes and Activities mentioned in Chap. 2, refers to the activity of identifying the different processes and activities to be performed during ontology development, their arrangement, and the time and resources needed for their completion. Thus, this activity includes as an important task the establishment of the *ontology network life cycle*, that is, the specific ordered sequence of processes and activities that ontology developers carry out during the life of the ontology network.

The goal of scheduling is to organize the different processes and activities in time, that is, to state a concrete programming or scheduling that guides the ontology network development, including processes and activities, their order and time, as well as human resource restrictions.

To establish the concrete schedule for the ontology network development, four important questions have to be answered:

1. How to organize an ontology development project into phases, or in other words, which ontology network life cycle model is the most appropriate for the ontology network development?
2. Which particular processes and activities should be carried out in the ontology network development?
3. Which order and dependencies exist among processes and activities?
4. What amount of resources (human and time) is needed and available for the development of the ontology network?

The first three questions are related to the establishment of the ontology network life cycle, and their responses would result in a general plan for the ontology network development. The fourth question is related to the inclusion of time and human resources restrictions for each process and activity included in the plan, and its response would result in the concrete schedule for the ontology network development. An estimate on how many people should be involved in the ontology network development can be obtained using the ONTOCOM model (Simperl et al. 2009). This is a cost estimation model, whose goal is to predict the costs (expressed in person per month) arising in typical ontology engineering processes.

As in the case of software engineering projects, an ontology engineer cannot start the ontology development project scheduling without having first identified the ontology requirements. In addition, the scheduling activity needs as input the types of potential knowledge resources (ontological resources, non-ontological resources, and/or ontology design patterns) to be reused during the development.

The *filling card* for the scheduling activity, presented in Fig. 14.1, includes the definition, goal, inputs and outputs, performer of the activity, and time of the activity.

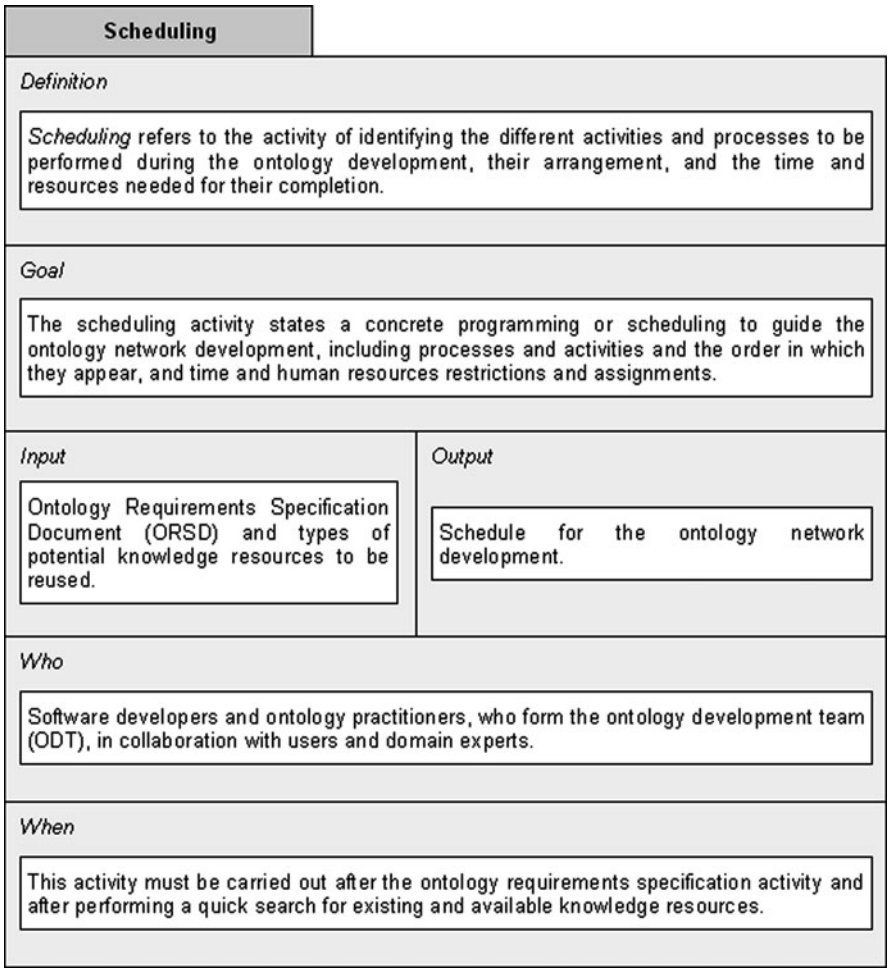

Fig. 14.1 Scheduling filling card

14.3 gOntt: NeOn Toolkit Plugin for the Scheduling Activity

To support the scheduling of ontology development projects, the NeOn Toolkit provides a plugin called *gOntt*. This plugin is conceptually based on the following ingredients that are explained in Chap. 2: (a) the scenarios of the NeOn Methodology, (b) the set of ontology network life cycle models, and (c) the NeOn Glossary of Processes and Activities.

The gOntt plugin has the following main objectives:

1. To support ontology developers in the decision of which ontology network life cycle model is the most appropriate for building their ontologies
2. To help ontology developers in the decision of which concrete processes 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 on them
4. To inform ontology developers about how to carry out a particular process or activity through the NeOn methodological guidelines and a reference to the concrete NeOn plugins to be used – that is, to help ontology developers in the ontology project execution.

According to the aforementioned objectives, gOntt functionalities can be divided into two main groups: *functionalities for scheduling ontology development projects* and *functionalities for helping in the execution of ontology development projects*.

The functionalities for *scheduling an ontology network development* are:

- To create particular schedules from scratch, by allowing the ontology developer to include processes, activities, phases, and relationships, along with 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 developer.
- To create particular schedules in a guided way. gOntt creates preliminary plans for the ontology development with a simple two-step wizard. The ontology developer uses the wizard to answer a set of simple and intuitive questions that implicitly allow him to select the ontology life cycle model and the processes and activities to be carried out.
  - gOntt internally uses a set of heuristics based on methodological foundations and scheduling templates[3] (Suárez-Figueroa 2010) to automatically generate the initial plan.
  - gOntt provides the user with an initial plan in the form of a Gantt chart that the user can modify in the following fashion: (a) by including or deleting processes and activities, (b) by changing order and dependencies among processes and activities, and (c) by including resource assignments and restrictions to the planned processes and activities (this possibility is out of the scope of this chapter).
- To create, modify, and delete gOntt projects.
- To export and import gOntt projects in an interchange format based on XML (files with .got extension).

[3] Such scheduling templates show ontology project default plans based on the different and possible combinations among life cycle models, scenarios, and processes and activities.

- To provide graphical and textual visualizations of gOntt projects.
- To rename, reorder, and delete processes, activities, and phases from a gOntt project.
- To change the scope of a given process or activity (e.g., to change a given activity to a different phase).
- To create, modify, and delete connections between activities, between processes, and between activities and processes. If a connection exists between two elements, the latter cannot start until the former is completed. These connections can have two different meanings:
  - Logical dependencies: when it is required that one activity is carried out before another because of the nature of the activities (e.g., diagnosis before repair in ontology validation)
  - Temporal dependencies: when an activity should be performed after another because of project requirements (e.g., ontology reuse and non-ontological reuse can be carried out in parallel because they have no restrictions between them but, in some cases, there are not enough human resources to perform the activities in parallel, and so they should be set to perform in sequence)
- To include and modify the duration and the starting date of the processes, activities, and phases.
- To check gOntt projects with respect to logical and temporal constraints.
- To provide usual editing capabilities such as copy, paste, undo, and redo.

The functionalities for *helping in the execution of ontology development projects* are:

- To provide the developer with some methodological guidelines for the processes and activities identified in the NeOn Methodology, and thus
  - To display a *filling card*, which includes the process or activity definition, its goal, inputs and outputs, performer of the process or activity, and time of the performance. Figure 14.2 shows an example of the filling card for the ontology localization activity.
  - To display a *workflow* and some *methodological guidelines* explaining how the process or the activity should be carried out, including its inputs, outputs, and actors involved. Figure 14.3 shows an example of the methodological guidelines for the ontology localization activity. Workflows are implemented with Eclipse cheat sheets[4].

[4] Cheat sheets is a new emerging technology within Eclipse V3.0 that is meant to guide a developer through a series of complex tasks to achieve some overall goal. Some tasks can be performed automatically, such as launching the required tools for the user. Other tasks need to be completed manually by the user.

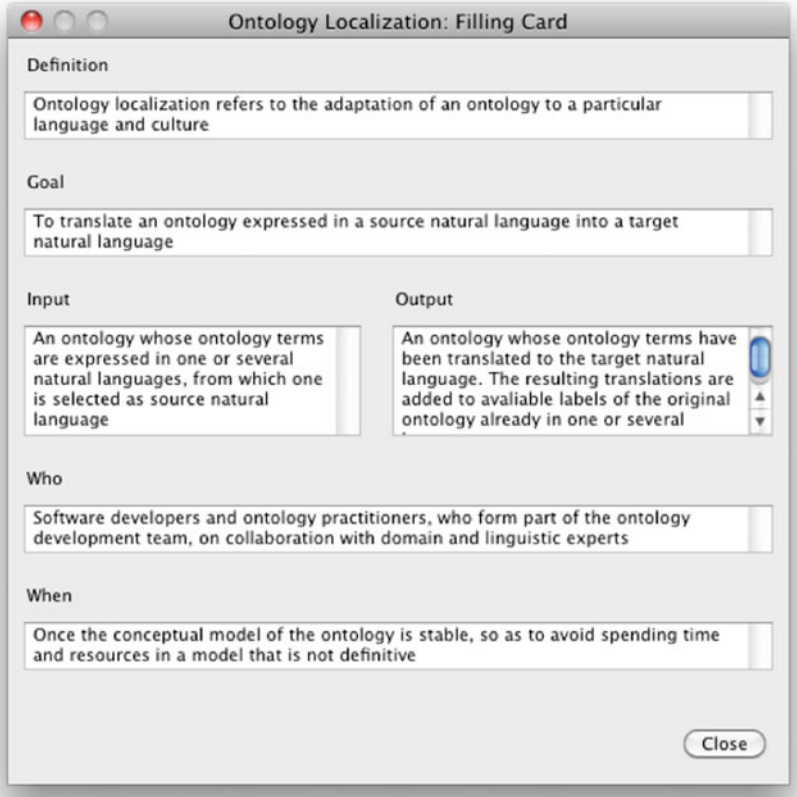

Fig. 14.2 Example of the ontology localization filling card in gOntt

- To provide a direct access to the NeOn plugins associated to each process and activity planned. This means that gOntt triggers the different NeOn Toolkit plugins associated to each process or activity included in the plan. To make this possible, gOntt uses the so-called *extension points*[5].
- To launch the Kali-ma dashboard (described in Chap. 15) with the plugins that are related to a given activity, process, phase, or whole scheduling projects.

Figure 14.4 shows the general appearance of the gOntt plugin, whereas Fig. 14.5 shows a specific plan; as for the latter, it is worth mentioning that in the reuse phase, ontological and non-ontological resource reuses can be replicated for

[5] gOntt has its own extension point that the rest of NeOn Toolkit plugins should implement (see Suárez-Figueroa et al. 2010 for more detail).

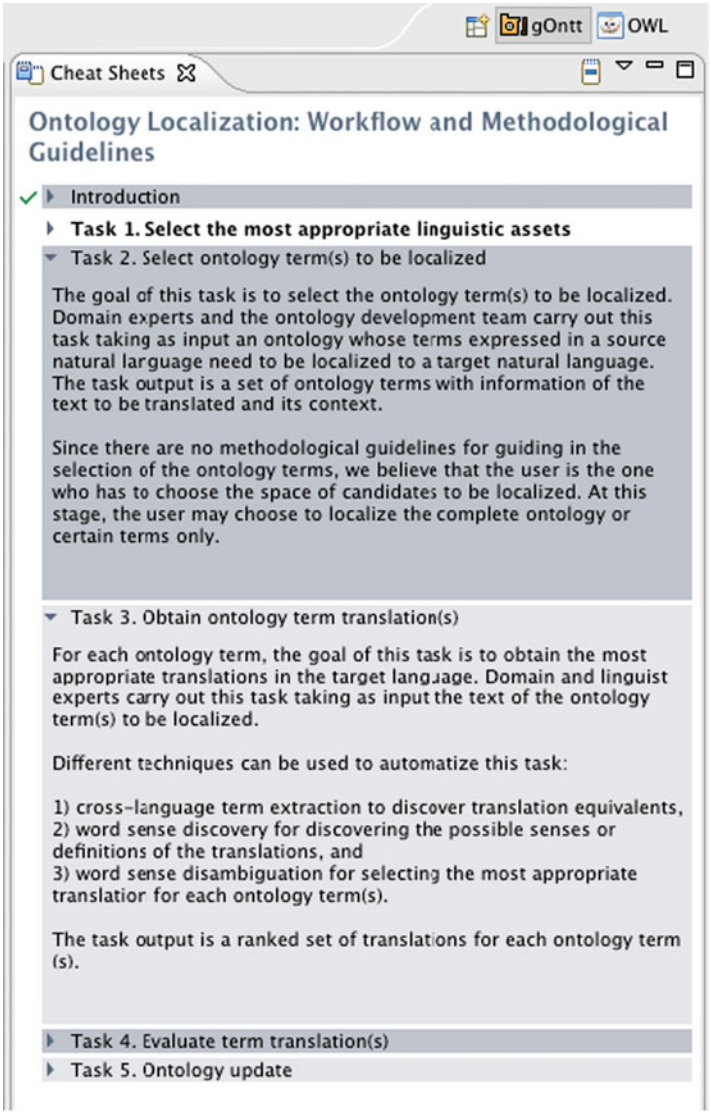

Fig. 14.3 Example of the workflow and of some methodological guidelines for the ontology localization activity in gOntt

every single resource used in the ontology development; the same holds for the reengineering phase in the non-ontological resource reengineering process.

For each ontology network development project within the NeOn Toolkit, up to one gOntt scheduling can be created.

To create a new gOntt project, one of the following NeOn Toolkit new wizards must be launched (see Fig. 14.6): “Schedule a project from scratch”

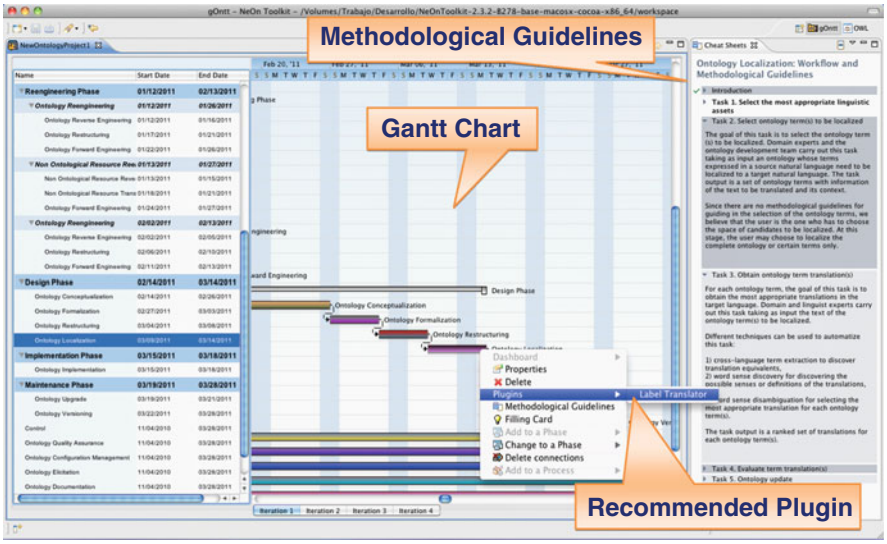

Fig. 14.4 Screenshot of the gOntt plugin

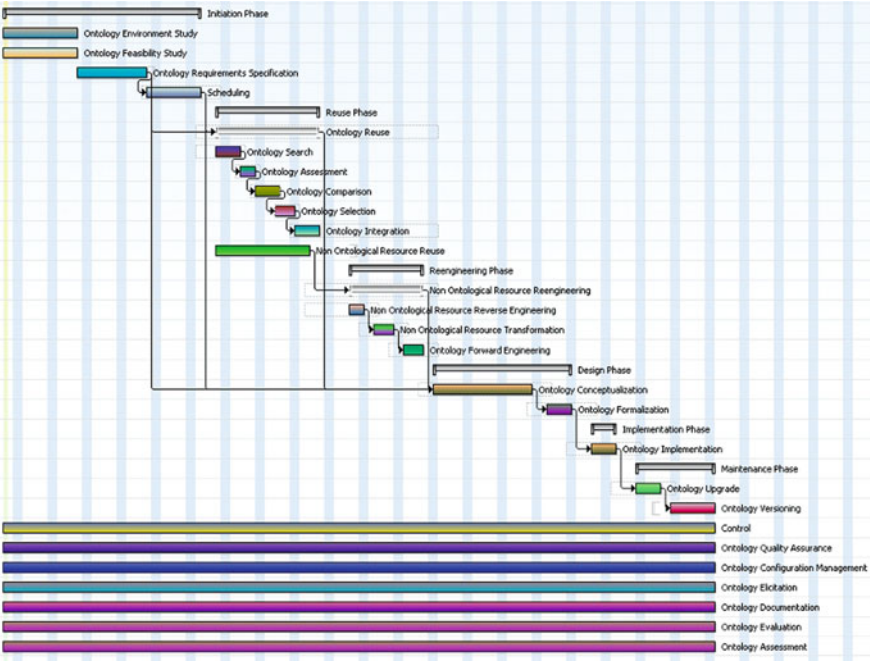

Fig. 14.5 A specific plan generated by gOntt

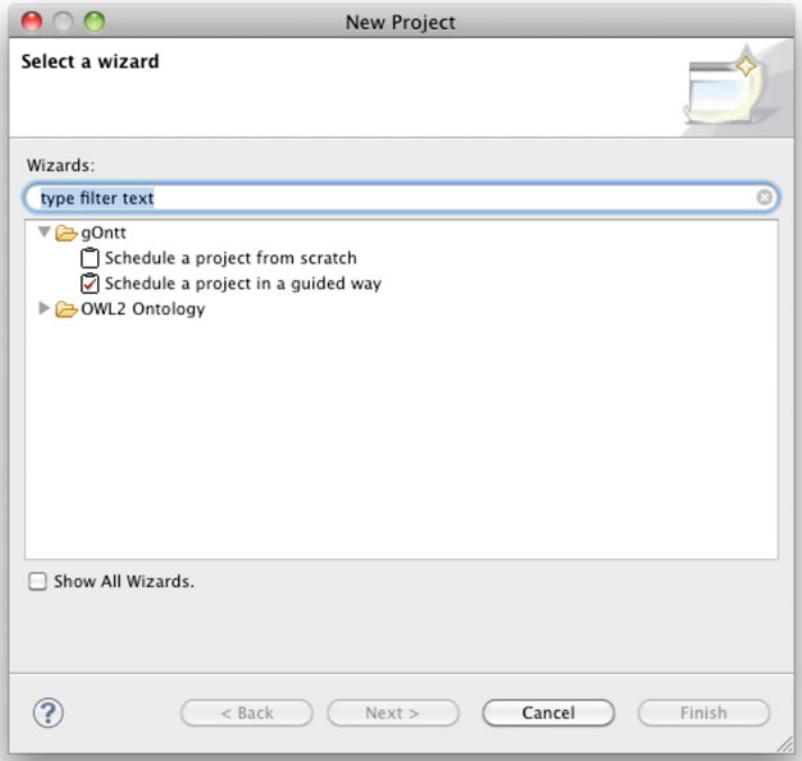

Fig. 14.6 gOntt wizards for scheduling new projects

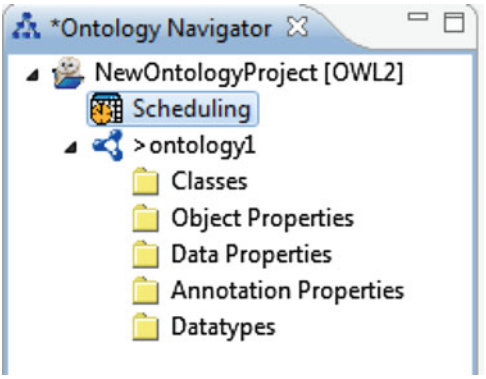

Fig. 14.7 Accessing a scheduling from the Ontology Navigator

or "Schedule a project in a guided way." Then the wizard selected will ask about the ontology network development project that ontology developers want to schedule.

Once the scheduling has been created for a given project, ontology developers can access to the gOntt project by clicking on the "scheduling" item in the Ontology Navigator as Fig. 14.7 shows.

14.4 Guidelines for Scheduling Ontology Development Project Using gOntt

The *workflow* for carrying out the scheduling of ontology development projects using gOntt is presented in this section. Each of the tasks in the workflow proposed includes prescriptive methodological guidelines. The tasks for carrying out the scheduling activity are shown in Fig. 14.8 and are explained in detail below.

Task 1. Selecting the ontology network life cycle model. The goal of this task is to obtain the most appropriate ontology network life cycle model for the ontology network to be developed. Users, domain experts, and the ontology development team carry out this task, taking as input both the ontology requirement specification

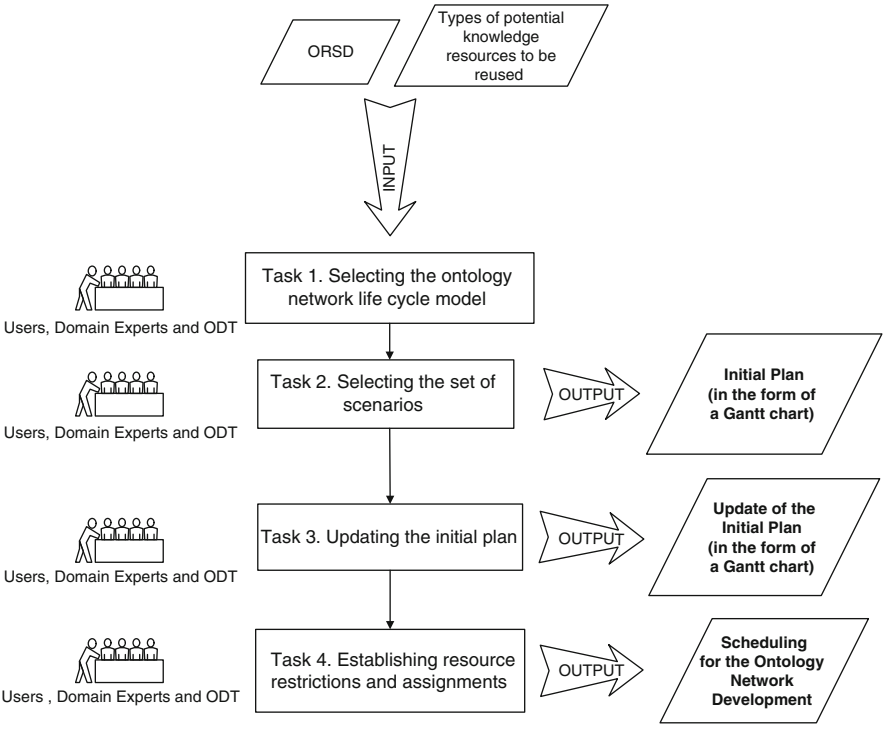

Fig. 14.8 Tasks for scheduling ontology development projects with gOntt

Fig. 14.9 Choosing the ontology network life cycle model

document (ORSD) (see Chap. 5) and the types of potential knowledge resources to be reused during the development. To help ontology developers in the decision of which is the most appropriate life cycle model among those presented in Chap. 2, gOntt presents a simple natural language question, displayed in Fig. 14.9. Based on the response given, either the waterfall model is selected, or the iterative-incremental model. In the latter case, ontology developers should also provide the expected number of iterations in the ontology development.

Task 2. Selecting the set of scenarios. The goal of this task is to select the set of scenarios to be followed during the ontology network development. Users, domain experts, and the ontology development team carry out this task, taking as input both the ontology requirement specification document (ORSD) and the set of potential knowledge resources to be used during the development.

To help ontology developers in this task, gOntt presents the set of natural language questions displayed in Fig. 14.10. If the model selected in Task 1 is the waterfall one, then questions should be answered once; on the other hand, if the model selected is the iterative-incremental one, then the set of questions should be answered once for each iteration expected.

With the responses to these questions, the set of heuristics based on methodological foundations and the set of scheduling templates (Suárez-Figueroa 2010),

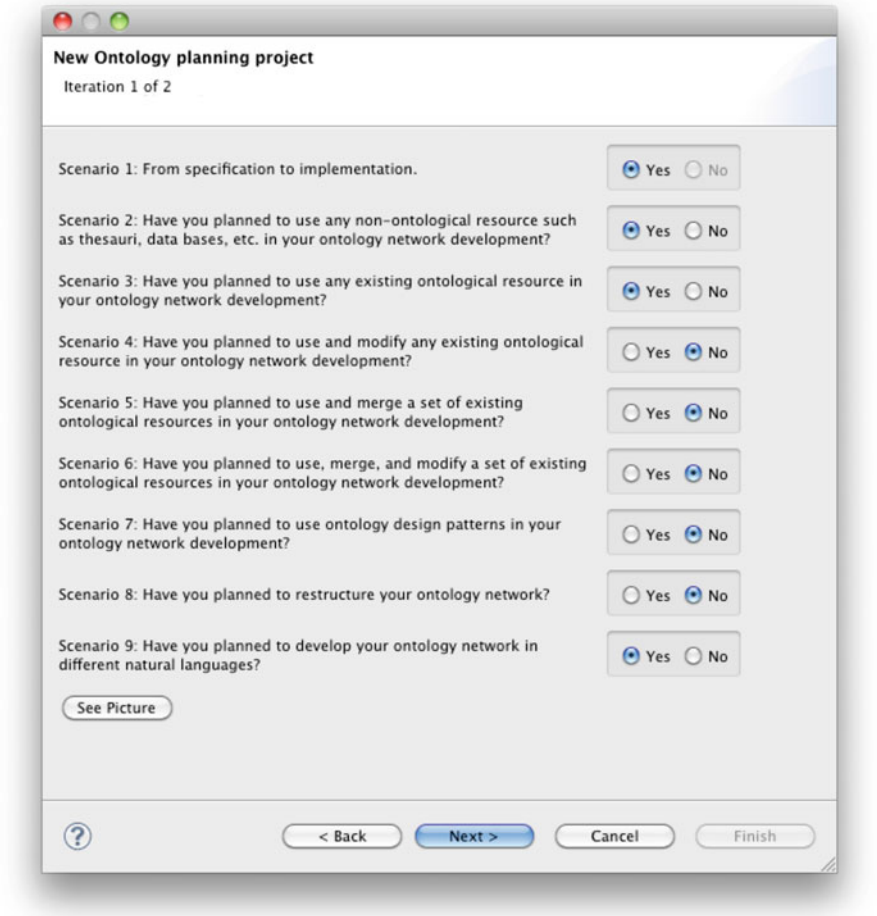

Fig. 14.10 Selecting the scenarios

gOntt is able to obtain the initial ontology network life cycle, that is, an initial plan for the ontology network development. The task output is represented as a Gantt chart, which is *de facto standard* in software project management.

Task 3. Updating the initial plan. The goal of this task is to modify (if necessary) the initial plan presented by gOntt. Users, domain experts, and the ontology development team carry out this task, taking as input both the ontology requirement specification document (ORSD) and the set of potential knowledge resources to be used during the development.

Ontology developers can modify the initial plan in the following ways: (a) by including or deleting processes, activities, and model phases and (b) by changing order and dependencies among processes and activities.

Task 4. Establishing resource restrictions and assignments. The goal of this task is to include information about temporal scheduling and human resource assignments in the life cycle obtained in Task 3. Users, domain experts, and the ontology development team carry out this task, taking as input both the ontology requirement specification document (ORSD) and the set of potential knowledge resources to be used during the development.

14.5 Conclusions

In order to manage properly ontology development projects in complex settings and to apply the NeOn Methodology correctly, it is crucial to have knowledge of the entire ontology development life cycle before starting development. The ontology project plan defines the tasks to be executed, the time when the tasks will be executed, and the dependencies between tasks. The project plan is the only way, as can be shown in other disciplines, to commit people to the project and to show how the work will be performed. It also aids ontology engineers in monitoring project execution and assessing the impact of a particular delay in the planned tasks.

With these notions in mind, this chapter presented (a) the gOntt plugin, a tool that supports the scheduling of ontology networks developments as well as their execution, and (b) prescriptive methodological guidelines for scheduling ontology development projects using gOntt.

In relation to the work presented in this chapter, an integration of gOntt and the guidelines proposed with the ONTOCOM model (Simperl et al. 2009) is planned. Additionally, such works will be extended by providing details of the cost associated to carrying out a particular task in an ontology development project.

References

Blázquez M, Fernández-López M, García-Pinar JM, Gómez-Pérez A (1998) Building ontologies at the knowledge level using the ontology design environment. In: Gaines BR, Musen MA (eds) 11th international workshop on Knowledge Acquisition, Modeling and Management (KAW 1998), Banff, Canada, SHARE4:1–15

Davis AM, Bersoff EH, Comer ER (1988) A strategy for comparing alternative software development life cycle models. IEEE Trans Softw Eng 14–10:1453–1461

Fernández-López M, Gómez-Pérez A, Juristo N (1997) METHONTOLOGY: from ontological art towards ontological engineering. In: Spring symposium on ontological engineering of AAAI, Stanford University, Standord, CA, pp 33–40

Gantt HL (1974) Work, wages and profit, published by The Engineering Magazine. New York, 1910; republished as Work, wages and profits, Hive Publishing Company, Easton, PA, 1974, ISBN 0879600489

Pfleeger S (2001) Software engineering: theory and practice, 2nd edn. Prentice Hall, Upper Saddle River. ISBN 0-13-029049-1

Pinto HS, Tempich C, Staab S (2004) DILIGENT: towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies. In: López de

Mantaras R, Saitta L (eds) Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004). IOS Press, Valencia, Spain, pp 393–397, 22–27 Aug 2004. ISBN: 1-58603-452-9. ISSN: 0922-6389

Royce WW (1970) Managing the development of large software systems: concepts and techniques. In: Proceedings Western Electronic Show and Convention (WESCON), Los Angeles, 25–28 Aug 1970

Simperl E, Popov I, Bürger T (2009) ONTOCOM revisited: towards accurate cost predictions for ontology development projects. In: Proceedings of the European Semantic Web Conference 2009 (ESWC 2009), Heraklion, Greece, 20 May–04 June 2009

Sommerville I (2007) Software engineering, 8th edn. Addison-Wesley, Harlow/New York. ISBN 0-321-31379-8

Staab S, Schnurr HP, Studer R, Sure Y (2001) Knowledge processes and ontologies. IEEE Intell Syst 16(1):26–34

Stellman A, Greene J (2005) Applied software project management. ISBN: 0-596-00948-8. O'Reilly

Suárez-Figueroa MC (2010) NeOn Methodology for building ontology networks: specification, scheduling and reuse. PhD thesis, Universidad Politécnica de Madrid, España, June 2010. Available at http://oa.upm.es/3879/

Suárez-Figueroa MC, Gómez-Pérez A, Muñoz O (2010). NeOn deliverable. D5.3.3. gOntt plugin for scheduling ontology projects. NeOn project. Available at http://www.neon-project.org/nw/images/c/c1/NeOn_2010_D533.pdf

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