Chapter 2
The NeOn Methodology for Ontology Engineering

Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López

Abstract In contrast to other approaches that provide methodological guidance for ontology engineering, the NeOn Methodology does not prescribe a rigid workflow, but instead it suggests a variety of pathways for developing ontologies. The nine scenarios proposed in the methodology cover commonly occurring situations, for example, when available ontologies need to be re-engineered, aligned, modularized, localized to support different languages and cultures, and integrated with ontology design patterns and non-ontological resources, such as folksonomies or thesauri. In addition, the NeOn Methodology framework provides (a) a glossary of processes and activities involved in the development of ontologies, (b) two ontology life cycle models, and (c) a set of methodological guidelines for different processes and activities, which are described (a) functionally, in terms of goals, inputs, outputs, and relevant constraints; (b) procedurally, by means of workflow specifications; and (c) empirically, through a set of illustrative examples.

2.1 Introduction

Given the large increase in the number of ontologies, which are available online, ontology development is more and more becoming a reuse-centric process (Simperl 2009). In particular, the level of reuse may vary significantly, depending on whether

M.C. Suárez-Figueroa (✉) • A. Gómez-Pérez
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

M. Fernández-López
Escuela Politécnica Superior, Universidad San Pablo CEU, Urbanización Montepríncipe sn., 28668 Boadilla del Monte, Madrid, Spain
e-mail: mfernandez.eps@ceu.es

M.C. Suárez-Figueroa et al. (eds.), *Ontology Engineering in a Networked World*,
DOI 10.1007/978-3-642-24794-1_2, © Springer-Verlag Berlin Heidelberg 2012

it concerns (a) other ontologies, such as DOLCE[1], SUMO (Pease et al. 2002), and Kowien[2]; (b) ontology modules (Cuenca-Grau et al. 2007); (c) ontology statements and ontology design patterns (Gangemi 2007; Presutti and Gangemi 2008); and (d) non-ontological resources (Jimeno-Yepes et al. 2009), such as thesauri, lexicons, DBs, UML diagrams, and classification schemas (e.g., NAICS[3] and SOC[4]).

Thus, in this context ontology development can be then characterized as the construction of a network of ontologies, where the different resources may be managed by different people, possibly in different organizations.

Given this new vision of ontology engineering by reuse, it then becomes important to provide strong methodological support for the collaborative development of ontology networks.

Methodological frameworks are widely accepted in different mature fields (Fernández-López 1999), like Software Engineering and Knowledge Engineering. Such methodological frameworks cover aspects, such as development process, life cycle models, as well as the methods, techniques, and tools that can be used to support the development process. Accordingly, a mature methodology for developing ontologies should also cover these aspects.

This chapter describes the *NeOn Methodology* for building ontologies and ontology networks, a scenario-based methodology that supports different aspects of the ontology development process, as well as the reuse and dynamic evolution of networked ontologies in distributed environments, where knowledge is introduced by different people (domain experts, ontology practitioners) at different stages of the ontology development process.

This methodology includes the following components:

- The *NeOn Glossary* (Sect. 2.2), which identifies and defines the processes and activities potentially involved in the ontology network construction.
- A *set of nine scenarios for building ontologies and ontology networks,* which are described in Sect. 2.3. Each scenario is decomposed in different processes and activities taken from those included in the *NeOn Glossary*.
- *Two ontology network life cycle models* (Sect. 2.4) that specify how to organize the processes and activities of the NeOn Glossary into *phases*[5].
- A *set of prescriptive methodological guidelines for processes and activities* (Sect. 2.5).

[1] http://www.loa-cnr.it/DOLCE.html

[2] Skill Ontology from the University of Essen, which defines concepts representing the competencies required to describe job position requirements and job applicant skills. Available at http://www.kowien.uni-essen.de/publikationen/konstruktion.pdf

[3] North American Industry Classification System, which provides industry-sector definitions for Canada, Mexico, and the United States to facilitate uniform economic studies across the boundaries of these countries. Available at http://www.census.gov/epcd/www/naics.html

[4] Standard Occupational Classification, which classifies workers into occupational categories (23 major groups, 96 minor groups, and 449 occupations). Available at http://www.bls.gov/soc/

[5] A phase is a distinct period or stage in a process of development.

In addition to applying the NeOn Methodology to the development of the ontology networks associated with use cases of the NeOn project as shown in Chaps. 18, 19, and 20. This methodology has been used to build ontology networks in different domains and areas and by people with diverse background, for example, and just to name a few, in e-employment (Villazón-Terrazas et al. 2011), in education (Clemente et al. 2011), in tourism (Lamsfus et al. 2009), and in mobile environments (Poveda-Villalón et al. 2010).

Finally, it is worth mentioning that the NeOn Methodology can also be used within the Linked Data initiative (Bizer et al. 2009) since this is based on knowledge resource reused and re-engineering as well as on mapping resources. Publishing Linked Data is a process that involves a high number of activities, design decisions as well as a wide range of technologies. The main activities are (1) identification of the data sources, (2) vocabulary modeling, (3) generation of the RDF data, (4) publication of the RDF data, and (5) linking the RDF data with other datasets in the cloud. In the vocabulary modeling activity, ontologies to model the data contained in the selected sources should be developed. The most important recommendation here is to reuse as much as possible available knowledge resources that model the knowledge needed. In this regard, the NeOn Methodology provides precise guidelines to help practitioners to create the vocabularies needed. One example of the use of the NeOn Methodology in this initiative can be found in (Vilches-Blázquez et al. 2010).

2.2 The NeOn Glossary

The *NeOn Glossary* identifies and defines the processes and activities potentially involved in the ontology network construction. This glossary has been established by a consensus reaching process among ontology experts and is a first step in addressing the lack of a standard glossary in Ontology Engineering – in contrast with the Software Engineering field that can claim the IEEE Standard Glossary of Software Engineering Terminology (IEEE 1990). The NeOn Glossary of Processes and Activities (Suárez-Figueroa 2010)[6] includes 59 processes and activities listed in Table 2.1.

2.3 Nine Scenarios for Building Ontology Networks

In the NeOn Methodology framework, a set of nine flexible scenarios for collaboratively building ontologies and ontology networks, placing special emphasis on reusing and re-engineering knowledge resources (ontological and non-ontological), has been identified.

[6] http://mayor2.dia.fi.upm.es/oeg-upm/files/pdf/NeOnGlossary.pdf

Table 2.1 NeOn Glossary of processes and activities

Processes	
Ontology aligning	Non-ontological resource reuse
Ontology design pattern reuse	Ontological resource reuse
Ontology module reuse	Ontology reuse
Ontology re-engineering	Ontology statement reuse
Non-ontological resource re-engineering	Ontology validation
Activities	
Ontology annotation	Ontology merging
Ontology assessment	Ontology modification
Ontology comparison	Ontology modularization
Ontology conceptualization	Ontology module extraction
Ontology configuration management control	Ontology partitioning
Ontology customization	Ontology population
Ontology diagnosis	Ontology pruning
Ontology documentation	Ontology quality assurance
Ontology elicitation	Ontology repair
Ontology enrichment	Ontology requirements specification
Ontology environment study	Non-ontological resource reverse Engineering
Ontology evaluation	Non-ontological resource transformation
Ontology evolution	Ontology restructuring
Ontology extension	Ontology reverse engineering
Ontology feasibility study	Scheduling
Ontology formalization	Ontology search
Ontology forward engineering	Ontology selection
Ontology implementation	Ontology specialization
Ontology integration	Ontology summarization
Knowledge acquisition for ontologies	Ontology translation
Ontology learning	Ontology update
Ontology localization	Ontology upgrade
Ontology mapping	Ontology verification
Ontology matching	Ontology versioning

Figure 2.1 presents the set of the nine most plausible scenarios for building ontologies and ontology networks. Directed arrows with associated numbered circles represent the different scenarios. Each scenario is decomposed into different processes or activities. Processes and activities are represented with colored circles or with rounded boxes and are defined in the NeOn Glossary of Processes and Activities presented in Sect. 2.2. Figure 2.1 also shows (as dotted boxes) the existing knowledge resources to be reused, and the possible outputs that result from the execution of some of the presented scenarios.

This section includes, as independent subsections, the most common scenarios that may unfold during the ontology network development. However, the reader should keep in mind that this list is not meant to be exhaustive.

- *Scenario 1: From specification to implementation*. The ontology network is developed from scratch, that is, without reusing available knowledge resources.

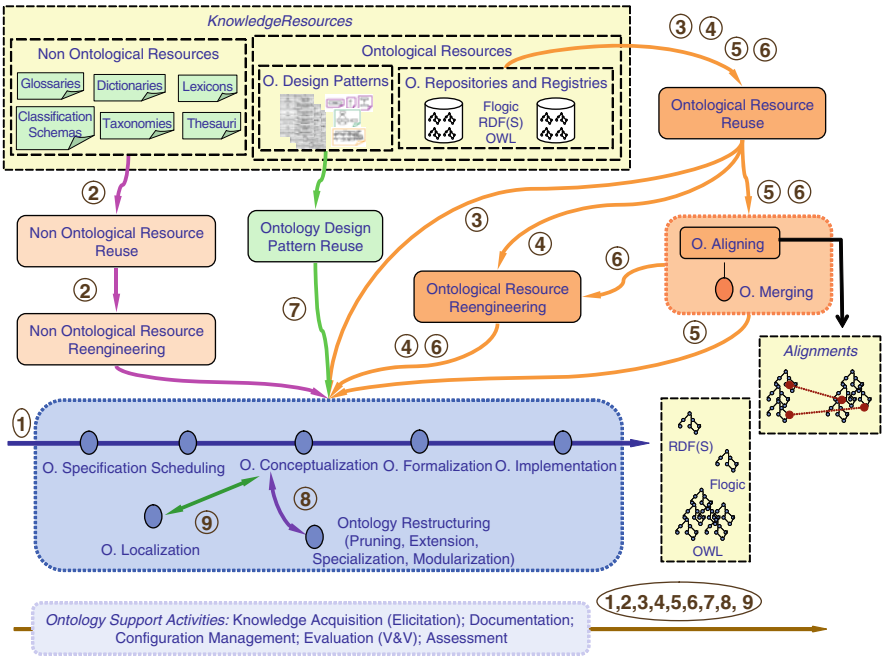

Fig. 2.1 Scenarios for building ontologies and ontology networks

- *Scenario 2: Reusing and re-engineering non-ontological resources*. This scenario covers the case where ontology developers need to analyze non-ontological resources and decide, according to the requirements the ontology should fulfill which non-ontological resources can be reused to build the ontology network. The scenario also covers the task of re-engineering the selected resources into ontologies.
- *Scenario 3: Reusing ontological resources*. Here, ontology developers reuse ontological resources (ontologies as a whole, ontology modules, and/or ontology statements).
- *Scenario 4: Reusing and re-engineering ontological resources*. Here, ontology developers both reuse and re-engineer ontological resources.
- *Scenario 5: Reusing and merging ontological resources*. This scenario unfolds only in those cases where several ontological resources in the same domain are selected for reuse and when ontology developers wish to create a new ontological resource from two or more ontological resources.
- *Scenario 6: Reusing, merging, and re-engineering ontological resources*. This scenario is similar to Scenario 5; however, here developers decide not to use the set of merged resources as it is, but to re-engineer it.
- *Scenario 7: Reusing ontology design patterns (ODPs)*. Ontology developers access ODPs repositories to reuse them.

- *Scenario 8: Restructuring ontological resources*. Ontology developers restructure (modularizing, pruning, extending, and/or specializing) ontological resources to be integrated in the ontology network being built.
- *Scenario 9: Localizing ontological resources*. Ontology developers adapt an ontology to other languages and culture communities, thus producing a multilingual ontology.

Knowledge acquisition, documentation, configuration management, evaluation, and assessment should be carried out during the whole ontology network development, that is, in any scenario used for developing the ontology network. The intensity of such support activities depends on the concrete phase of the development progress.

It is worth mentioning that these scenarios can be combined in different and flexible ways, and that any combination of scenarios should include Scenario 1 because this scenario is made up of the core activities that have to be performed in any ontology development. Indeed, as Fig. 2.1 shows, the results of any other scenario should be integrated in the corresponding activity of Scenario 1.

The following subsections present the various scenarios identified; each subsection includes (a) motivation for the scenario; (b) sequence of processes, activities, and tasks to be carried out, where the processes and activities included are taken from the NeOn Glossary of Processes and Activities (Sect. 2.2); and (c) outcomes for the scenario.

2.3.1 Scenario 1: From Specification to Implementation

This scenario refers to the development of ontologies from scratch. The scenario is made up of the core activities that have to be performed in any ontology development and should be combined with the rest of scenarios.

In this scenario, ontology developers[7] should specify first the requirements that the ontology should fulfill, by means of the *ontology requirements specification activity*. The objective of this activity is to output the ontology requirements specification document (ORSD) that includes the purpose, the scope, and the implementation language of the ontology network, the target group, and the intended uses of the ontology network, as well as the set of requirements that the ontology network should fulfill, mainly in the form of *competency questions* (CQs)[8] and a pre-glossary of terms. Prescriptive methodological guidelines for this activity are provided in Chap. 5.

[7] In this book, ontology developers refer to software developers and ontology practitioners involved in the development of ontologies.

[8] An example of CQ can be "where is located the device Z? The device Z is at coordinates X, Y".

After the ontology requirements specification activity, it is recommended to carry out a look for candidate knowledge resources (ontologies, non-ontological resources, and ontology design patterns) to be reused in the development, using as input terms included in the ORSD. These candidate resources provide clues for the identification of the scenarios to be followed during the ontology development. Then, the *scheduling activity* must be carried out, using the ORSD and the results of such a look for resources. During the scheduling activity, the team establishes the ontology network life cycle and the human resources needed for the ontology project. Chapter 14 presents guidelines and a tool for performing the scheduling of ontology development projects.

Then, the ontology developers assigned to the ontology project should carry out (1) the *ontology conceptualization activity*, in which knowledge is organized and structured into meaningful models at the knowledge level; (2) the *ontology formalization activity*, in which the conceptual model is transformed into a semi-computable model; and (3) the *ontology implementation activity*, in which a computable model (implemented in an ontology language) is generated.

The principal output is a network of ontologies that represents the expected domain implemented in an ontology language (OWL[9], F-Logic, etc.). In addition, a broad range of documents, such as the ontology requirements specification document, the ontology description document, and the ontology evaluation document, will be generated as output by the different activities.

2.3.2 Scenario 2: Reusing and Re-engineering Non-Ontological Resources

Currently, ontology developers are realizing the benefits of "not reinventing the wheel" at each ontology development. They are starting to reuse as much as possible non-ontological resources, such as classification schemes, thesauri, lexicons, and folksonomies, built by others that already have reached some degree of consensus, with the aim of speeding up the ontology development process (Villazón-Terrazas et al. 2010). The reuse of such resources involves necessarily their re-engineering into ontologies. Therefore, this scenario unfolds in those cases in which ontology developers wish to reuse the non-ontological resources at their disposal.

As Fig. 2.1 shows (by arrows with the number 2), ontology developers should accomplish first the non-ontological resource reuse process and then choose the most suitable non-ontological resources (thesauri, glossaries, databases, etc.) to be used for building the ontology network. Such non-ontological resources cover to some extent the domain of the ontology network being built. If ontology developers

[9] http://www.w3.org/TR/owl-ref/

decide that one or more resources are useful for the ontology network development, then the non-ontological resource re-engineering process should be carried out to transform the selected non-ontological resources into ontologies. After this process, ontology developers should use the resultant ontologies as input of some of the activities included in Scenario 1 (explained in Sect. 2.3.1), as shown in Fig. 2.1.

The activities for carrying out the non-ontological resource reuse process are briefly explained below; prescriptive methodological guidelines for this activity are described in Chap. 6:

1. *Activity 1. Search non-ontological resources*. The goal of the activity is to find non-ontological resources in highly reliable websites, domain-related sites, and resources within organizations. The input for this activity is the ontology requirements specification document (ORSD).
2. *Activity 2. Assess the set of candidate non-ontological resources*. The goal of this activity is to assess the set of candidate non-ontological resources obtained in Activity 1. To carry out this activity, the following criteria should be used: coverage, precision, and consensus about the knowledge and terminology used in the resource, which is a subjective criterion.
3. *Activity 3. Select the most appropriate non-ontological resources*. The goal of this activity is to select the most appropriate non-ontological resources from those candidates obtained in Activity 2.

As mentioned before, the goal of the non-ontological resource re-engineering process is to transform a non-ontological resource into an ontology. This process can be divided into the following activities, and prescriptive methodological guidelines for performing them are included in Chap. 6:

1. *Activity 1. Non-ontological resource reverse engineering*. The goal of this activity is to analyze a non-ontological resource in order to identify its underlying components and create representations of the resource at the different levels of abstraction (design, requirements, and conceptual).
2. *Activity 2. Non-ontological resource transformation*. The goal of this activity is to generate a conceptual model from the non-ontological resource.
3. *Activity 3. Ontology forward engineering*. The goal of this activity is to output a new implementation of the ontology on the basis of the new conceptual model identified in Activity 2.

The principal output is an ontology network that represents the expected domain implemented in an ontology language (OWL, F-Logic, etc.). Furthermore, a broad range of documents containing the requirements specification, the ontology documentation, the ontology evaluation, etc. will be generated as output of different activities. Additionally, the non-ontological resources selected to be reused have been "ontologized" by means of the non-ontological resource re-engineering activity.

2.3.3 Scenario 3: Reusing Ontological Resources

As more ontological resources are available in ontology repositories and on the Internet[10], ontology developers are starting to reuse them not only with the idea of "not reinventing the wheel", but also with the aim of taking advantage of them. Thus, this scenario unfolds in those cases in which ontology developers have at their disposal ontological resources useful for their problem and that can be reused in the ontology development.

As Fig. 2.1 shows (by arrows with the number 3), ontology developers should perform the *ontological resource reuse process*, which is composed of the following activities:

1. *Activity 1. Ontology search*. Ontology developers search for candidate ontological resources that satisfy the requirements in repositories and registries like Swoogle[11], Watson[12], and Sindice[13]. These ontological resources could be implemented in different languages or could be available in different ontology tools.
2. *Activity 2. Ontology assessment*. Ontology developers must inspect the content and granularity of the ontological resources obtained in Activity 1. The goal of this activity is to find out if such resources satisfy the needs identified in the ORSD.
3. *Activity 3. Ontology comparison*. Ontology developers should compare the ontological resources assessed in Activity 2, taking into account a set of criteria identified by developers (e.g., reuse economic cost, code clarity, and content quality).
4. *Activity 4. Ontology selection*. Ontology developers should select the set of ontological resources that are the most appropriate for their ontology network requirements, based on the comparisons obtained in Activity 3.

   After selecting the most appropriate ontological resources, ontology developers should define the reuse mode; that is, ontology developers need to decide how they will reuse the selected ontological resources. There are three possible modes:

   - The ontological resources selected will be reused as they are.
   - The ontology re-engineering activity should be carried out with the ontological resources selected.
   - Some ontological resources will be merged to obtain a new ontological resource.

[10] See, for example, a list of novel ontology search engines described at: http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/SemanticWebSearchEngines

[11] http://swoogle.umbc.edu/

[12] http://watson.kmi.open.ac.uk/WatsonWUI/

[13] http://sindice.com/

Before reusing the selected ontological resources by means of any reuse mode, it is also convenient to evaluate these resources through the *ontology evaluation activity*.

5. *Activity 5. Ontology integration*. Ontology developers should include, as they are, the ontological resources selected (the code) in Activity 4 into the ontology network being built following the activities of Scenario 1 (Sect. 2.3.1).

Prescriptive methodological guidelines to reuse general ontologies are provided in Chap. 7.

The principal output is an ontology network that represents the expected domain implemented in an ontology language (OWL, F-Logic, etc.). Additionally, a broad range of documents including the requirements specification, the ontology documentation, the ontology evaluation, etc. will be generated as output of different activities.

2.3.4 Scenario 4: Reusing and Re-engineering Ontological Resources

This scenario unfolds in those cases in which ontology developers have at their disposal ontological resources useful for their problem, which can be reused in the ontology network development. However, such resources are not exactly useful as they are, so they should be modified (i.e., re-engineered) to serve to the intended purpose or problem.

As Fig. 2.1 shows (by arrows with the number 4), ontology developers should perform first the *ontological resource reuse process* to select the most suitable ontological resources to be used for building the ontology network. Then, they should carry out the *ontological resource re-engineering process* to modify the selected ontological resources. Finally, they should use the resultant ontological resources as input to some of the activities included in Scenario 1 (explained in Sect. 2.3.1), as shown in Fig. 2.1.

Specifically, ontology developers should carry out some activities as part of the ontological resource reuse process; such activities are the following: *ontology search*, *ontology assessment*, *ontology comparison*, and *ontology selection* as already explained in Scenario 3 (Sect. 2.3.3).

After the ontology selection activity, ontology developers should decide how they will reuse the ontological resources. They should also decide whether to perform the ontological resource re-engineering process with the selected ontological resources because these resources may not absolutely correct for the concrete use case as they are and they need to be transformed in some way.

The ontological resource re-engineering process proposed here has been created taking as inspiration the software re-engineering process (Byrne 1992). It is composed of the following activities: *ontological resource reverse engineering*, *ontological resource restructuring*, and *ontological resource forward engineering*.

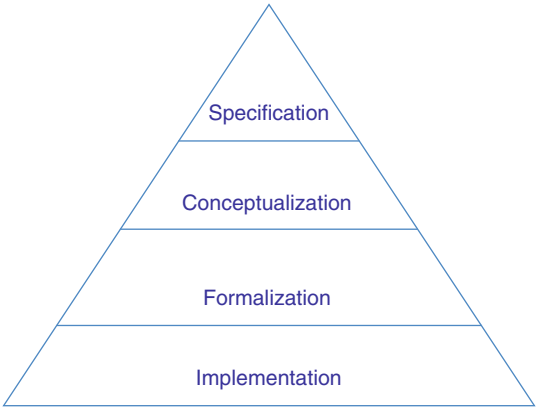

Fig. 2.2 Levels of abstraction for the ontological resource re-engineering process

Additionally, this process is related to the levels of abstraction shown in Fig. 2.2 that are based on (Byrne 1992) and are described below.

- Specification is the highest level of abstraction. In this level, requirements, purpose, and scope, among other components of the specification, are described.
- In the conceptualization level, ontology characteristics such as structure and components are described. The knowledge that the ontology represents is organized following a set of knowledge representation primitives (concepts, relations, etc.). In this level, the knowledge is structured in meaningful models at the knowledge level (Newell 1982). To organize the knowledge, intermediate representations based on tabular and graphical notations (Gómez-Pérez et al. 2003), which can be understood by ontology practitioners, can be used.
- In the formalization level, the formal or semi-computable model that was used to transform the conceptual model is described.
- The implementation level is the lowest abstraction level. Here, the ontology description focuses on implementation characteristics and is represented in an ontology language understandable by computers and usable by automatic reasoners.

Figure 2.3 presents the ontological resource re-engineering model. This model suggests different paths to re-engineer an ontological resource, taking into account the levels of abstraction presented in Fig. 2.2. Examples of these paths are:

- At implementation level: from ontological resource 1 code to ontological resource 2 code
- At formalization level: reverse engineering (from code 1 to formalization 1), restructuring formalization 1 to obtain formalization 2, and forward engineering to obtain code of resource 2
- At conceptualization level: reverse engineering (from code 1 to conceptualization 1), restructuring conceptualization 1 to obtain conceptualization 2, and forward engineering to obtain formalization or implementation 2

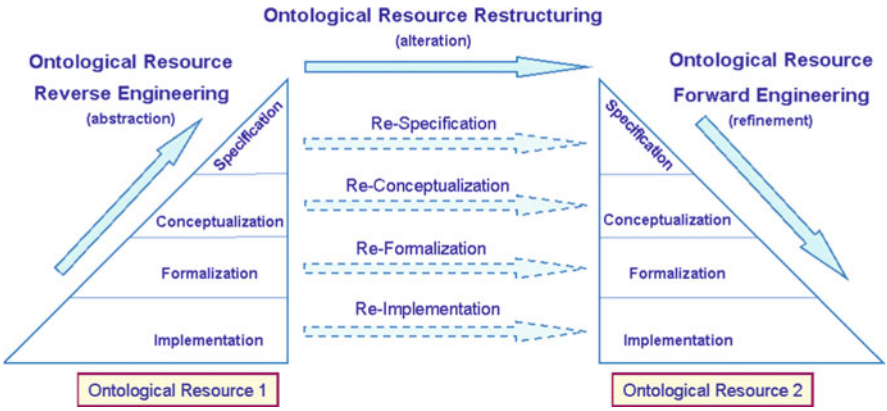

Fig. 2.3 Ontological resource re-engineering model

- At specification level: reverse engineering (from code 1 to specification 1), restructuring specification 1 to obtain specification 2, and forward engineering to obtain conceptualization, formalization, or implementation 2

The choice of a concrete path depends on the ontological resource characteristics that have to be changed. Thus, in Fig. 2.3 the following types of changes can be distinguished:

- Re-specification. If the ontology developer restructures the requirements specification, she changes requirements, purpose and scope, among other elements of the requirements specification. For example, changes in requirements, addition or deletion of requirements, etc.
- Re-conceptualization. If she restructures the conceptualization, changes might refer to modification of ontology structure, modification of granularity and richness of the knowledge, removal or addition of axioms, restructuration of ontology architecture (modularization), inclusion of new concepts, use of ontology design patterns, etc.
- Re-formalization. If she restructures the formalization level, the changes refer to formalization characteristics (such as changing the ontology paradigm from description logic to frames).
- Re-implementation. If she restructures the implementation level, the changes are focused on implementation characteristics that are tightly related to the ontology implementation language (e.g., translation from RDF(S) to OWL). Other changes could be conforming to coding standards, improving code readability, renaming code items, etc.

Ontology developers should decide at which level they need to carry out the ontological resource re-engineering process. Once ontology developers have decided the level, they should carry out the ontological resource re-engineering process, and then they should integrate the result of such a process (code,

formalization, conceptualization, or specification) into the corresponding activity of Scenario 1 (Sect. 2.3.1).

The principal outcome is an ontology network that represents the expected domain implemented in an ontology language (OWL, F-Logic, etc.). Additionally, a broad range of documents including requirements specification, ontology documentation, ontology evaluation, etc. will be generated as output of different activities. Furthermore, new ontological resources from those selected for their reuse are generated through the ontological resource re-engineering process. Such new resources can be considered as new versions of the ontological resources after the re-engineering process.

2.3.5 Scenario 5: Reusing and Merging Ontological Resources

This scenario unfolds in those cases where several ontological resources in the same domain can be selected for reuse and when the ontology developer wishes to create a new ontological resource from two or more, possibly overlapping, ontological resources. It could also occur that the ontology developer wishes only to establish alignments among the ontological resources selected in order to create the ontology network.

As Fig. 2.1 shows (by arrows with the number 5), ontology developers should perform first the *ontological resource reuse process* to select the most suitable ontological resources that will be used for building the ontology network. Concretely, ontology developers should carry out the activities presented in Scenario 3 (Sect. 2.3.3) as part of the ontological resource reuse process. After the ontology selection activity, ontology developers should decide how they will reuse the ontological resources selected. In this scenario, ontology developers decide to perform the following activities because the selected resources are valid as they are, but not in a complete way, if they were reused in a separate fashion. The activities to be performed are the following:

1. *Activity 1. Ontology aligning*. Ontology developers carry out this activity with the aim of obtaining a set of alignments among the selected ontological resources. Prescriptive methodological guidelines for this activity are described in Chap. 12.
2. *Activity 2. Ontology merging*. Ontology developers can merge the selected ontological resources using the alignments (output of Activity 1) to obtain a new ontological resource from the overlapping selected ones.

Ontology developers have here two different possibilities: (1) to establish the mappings among such selected resources and (2) to establish the mappings and also to merge the selected resources.

After this activity, ontology developers should use the resultant merged ontological resource as input of some of the activities included in Scenario 1 (explained in Sect. 2.3.1), as shown in Fig. 2.1.

The principal outputs are (a) a set of alignments among the selected ontological resources and (b) a set of new ontological resources to be integrated as they are in the ontology network.

2.3.6 Scenario 6: Reusing, Merging, and Re-engineering Ontological Resources

This scenario unfolds in those cases in which several ontological resources in the same domain can be selected to build the ontology network. Ontology developers decide to create a new ontological resource merging two or more, possibly overlapping, ontological resources. Such a merged ontological resource is not useful as it is, so it should be modified (i.e., re-engineered) to serve to the intended purpose.

As Fig. 2.1 shows (see arrows with number 6), ontology developers should perform first the *ontological resource reuse process* to select the most suitable ontological resources for building the ontology network (as explained in Scenario 3 (Sect. 2.3.3)). Then, they should decide how they will reuse the selected ontological resources. It is in this scenario where ontology developers decide to perform the *ontology aligning and ontology merging* activities because the selected resources are valid but not in a complete way for the concrete case if they are considered separately, as explained in Scenario 5 (Sect. 2.3.5). After merging the selected resources, they should carry out the *ontological resource re-engineering process* as described in Scenario 4 (Sect. 2.3.4). After that, they should use the resultant ontological resource as input of some of the activities included in Scenario 1 (explained in Sect. 2.3.1), as shown in Fig. 2.1.

The principal output is an ontology network that represents the expected domain implemented in an ontology language (OWL, F-Logic, etc.). Additionally, a broad range of documents including the requirements specification, the ontology documentation, the ontology evaluation, etc. will be generated as output of different activities.

Furthermore, a merged ontological resource, taken from those selected for reuse, and a re-engineered merged ontological resource are generated. Alignments between the ontological resources selected are also outputs of this scenario.

2.3.7 Scenario 7: Reusing Ontology Design Patterns

Recently, within the Ontology Engineering field, *ontology design patterns* (ODPs) have emerged as (1) a way of helping ontology developers to model OWL ontologies (Gangemi 2005; Pan et al. 2007) and (2) a new mode of encoding best practices, based on experiences and knowledge of "good" solutions. As any other

type of patterns, ODPs are perceived as having three kinds of benefits (Blomqvist et al. 2009): (1) reuse benefits, (2) guidance benefits, and (3) communication benefits. ODPs can be found in online libraries that include both the description and the OWL code associated to the patterns as, for example, "the Ontology Design Pattern Wiki"[14], or they can be obtained from the "Semantic Web Best Practices and Deployment"[15] working group. Thus, this scenario unfolds in those cases where best practices can be applied to the development of ontology networks.

Ontology developers work on the development of an ontology network and very often encounter problems regarding the way in which certain knowledge should be modeled. This may happen during the ontology conceptualization activity, the ontology formalization activity, or during the ontology implementation activity. In these situations, ontology developers can access on-line libraries in order to find modeling solutions.

Ontology developers should perform the *ontology design pattern reuse process* to select the most suitable ODPs for building the ontology network. The principal output of this reuse process is a set of ontology design patterns integrated into the ontology network being developed. Guidelines to perform this reuse are provided in Chap. 3.

2.3.8 Scenario 8: Restructuring Ontological Resources

This scenario unfolds in those cases where the knowledge contained in the conceptual model of the ontology network should be corrected and reorganized to obtain the network that covers the ontology requirements.

Ontology developers should perform the *ontology restructuring activity* to modify the ontology network being built, after the ontology conceptualization activity. The *ontology restructuring activity* can be performed by executing any of the following sub-activities, combining them in any manner and order:

- *Ontology modularization activity*. Ontology developers create different ontology modules in the ontology network, which facilitates the reuse of the knowledge included in the network. Prescriptive methodological guidelines to carry out this activity are presented in Chap. 10.
- *Ontology pruning activity*. Ontology developers prune those branches of the taxonomies included in the ontology network that are considered not necessary to cover the ontology requirements.
- *Ontology enrichment activity*. This activity can be carried out by performing any of the two sub-activities that follow:

[14] http://ontologydesignpatterns.org/

[15] http://www.w3.org/2001/sw/BestPractices/

- *Ontology extension activity*. Ontology developers extend the ontology network, including (in width) new concepts and relations.
- *Ontology specialization activity*. Ontology developers specialize those branches of the ontology network that require more granularity and include more specialized concepts and relations.

Note that this activity (ontology restructuring) can be performed (1) in an independent way as explained in this scenario or (2) as part of the ontological resource re-engineering process, as described in Scenario 4 in Sect. 2.3.4.

The principal output is a conceptual model of the ontology network that represents the expected domain.

2.3.9 Scenario 9: Localizing Ontological Resources

Although access to top-quality ontologies (e.g., Galen, CYC, or AKT) is, in many cases, free and unlimited for users all around the world, most of these ontologies are available only in English. Due to the language barrier, non-English users therefore often encounter problems when trying to access ontological knowledge in their own languages. Moreover, more and more ontology-based systems are being built for multilingual applications (e.g., multilingual machine translation or multilingual information retrieval). For these reasons, the need for multilingual ontologies has increased. Thus, this scenario unfolds in those cases in which the ontology network to be developed should be written in different natural languages.

Ontology developers should perform the *ontology localization activity* once the ontology has been conceptualized and restructured. This activity requires the translation of all the ontology terms into another natural language (Spanish, French, German, etc.) different from the language used in the conceptualization, using multilingual thesauri and electronic dictionaries (e.g., EuroWordNet[16]). This ontology localization activity is composed of the following tasks (Espinoza et al. 2009):

1. *Task 1. Selecting the most appropriate linguistic assets*. The goal of this task is to select the most appropriate linguistic assets that help to reduce the cost, to improve the quality of the localization, and to increase the consistency of the localization activity.
2. *Task 2. Selecting ontology label(s) to be localized*. The goal of this task is to select the ontology label(s) to be localized.
3. *Task 3. Obtaining ontology label translation(s)*. The goal of this task is to obtain the most appropriate translation in the target language for each ontology label.
4. *Task 4. Evaluating label translation(s)*. The goal of this task is to evaluate the label translations in the target language.

[16] http://www.illc.uva.nl/EuroWordNet/

5. *Task 5. Updating the ontology*. The goal of this task is to update the ontology with the label translations obtained for each localized label. The task output is an ontology enriched with labels in the target language associated to each localized term.

Prescriptive methodological guidelines for localizing ontologies are presented in Chap. 8.

After this localization activity, the resulting conceptual model should be integrated in the conceptualization activity of Scenario 1 (Sect. 2.3.1).

The principal outcome is a conceptual model of the ontology network in different natural languages (i.e., a multilingual conceptual model) that represents the expected domain.

2.4 Two Ontology Network Life Cycle Models

Ontologies are artifacts designed for the purpose of satisfying certain requirements and needs that are emerging in the real world.

Thus, the *ontology network development process* is defined as the process by which user's needs are translated into an ontology network. This means that the ontology network development process can be seen as a specific case of the software development process.

An *ontology network life cycle model* is defined as a model to describe how to develop (and maintain) an ontology network project; in other words, how to organize the processes and activities of the NeOn Glossary into phases or stages.

This section includes the two ontology network life cycle models, which include the *waterfall model* (Sect. 2.4.1) and the *iterative-incremental model* (Sect. 2.4.2). Additionally, it is worth mentioning that these two models are intrinsically related to the set of nine flexible scenarios for collaboratively building ontologies and ontology networks, presented in Sect. 2.3. Such a relation is due to the creation of both models and scenarios, taking into account the importance of reusing and re-engineering knowledge resources and merging resources.

2.4.1 Waterfall Ontology Network Life Cycle Models

The main characteristic of the waterfall life cycle model family proposed for the ontology network development is the representation of the stages of an ontology network as sequential phases. This model represents the stages as a waterfall. In this model, a concrete stage must be completed before the following stage begins, and no backtracking is permitted except in the case of the maintenance phase.

The main assumption for using the waterfall ontology network life cycle model proposed is that the requirements are completely known, without ambiguities, and unchangeable at the beginning of the ontology network development.

This model could be used in the following situations:

- In ontology projects with a short duration (e.g., 2 months)
- In ontology projects in which the goal is to develop an existing ontology in a different formalism or language
- In ontology projects in which the requirements are closed, for instance, to implement an ontology based on an ISO standard, or based on resources with previous consensus in the included knowledge
- In ontology projects when ontologies cover a small, well-understood domain

Taking into account the characteristics of the ontology development scenario, this model includes a set of support activities that should be performed in all of the phases. This set of support activities includes the acquisition of knowledge in the domain in which the ontology network is being developed, the evaluation (from a content-oriented perspective) and the assessment (from user and need perspectives) of the different phase outputs, project and configuration management, and documentation.

Because of the importance of reusing and re-engineering knowledge resources and merging ontological resources, the following five significantly different versions of the waterfall ontology network life cycle model have been defined. These versions have been created incrementally (i.e., the four-phase is the basis for the five-phase, the five-phase is the basis for the six-phase, etc.).

Before detailing the different versions, they can be summarized in the following way:

- The four-phase waterfall model. It represents the stages of an ontology network, starting with the initiation phase and going through the design phase and the implementation phase to the maintenance phase.
- The five-phase waterfall model. It extends the four-phase model with the reuse of ontological resources as they are.
- The five-phase + merging phase waterfall model. It is a special case of the five-phase model. It includes the merging phase to obtain a new ontological resource from two or more ontological resources previously selected in the reuse phase.
- The six-phase waterfall model. It extends the five-phase model with re-engineering phase. It allows the re-engineering of knowledge resources (ontological and non-ontological). It could happen that several knowledge resources are transformed into ontologies in the re-engineering phase.
- The six-phase + merging phase waterfall model. It extends the six-phase model by including the merging phase after the reuse phase.

2.4.1.1 The Four-Phase Waterfall Ontology Network Life Cycle Model

This model represents the stages of an ontology network, starting with the initiation phase and going through the design phase, the implementation phase to the maintenance phase.

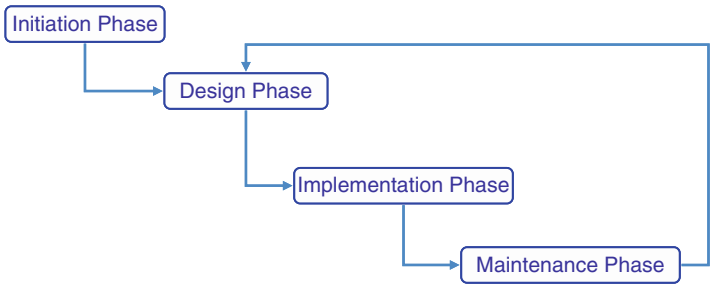

Fig. 2.4 The four-phase waterfall ontology network life cycle model

The model proposed is shown in Fig. 2.4, and the main purposes and outcomes for each phase in the model are the following:

- *Initiation phase*. In this phase, it is necessary to produce an ontology requirement specification document (ORSD) (explained in Chap. 5), including the requirements that the ontology network should satisfy and taking into account knowledge about the concrete domain. Also in this phase, the approval or rejection of the ontology network development should be obtained. This phase has also as requisite to identify the development team and to establish the resources, responsibilities, and timing (i.e., the scheduling for the ontology project).
- *Design phase*. The output of this phase should be both an informal model and a formal one that satisfy the requirements obtained in the previous phase. The formal model cannot be used by computers, but it can be reused in other ontology networks.
- *Implementation phase*. In this phase, the formal model is implemented in an ontology language. The output of this phase is an ontology implemented in RDF(S), OWL, or other language that can be used by semantic applications or by other ontology networks.

  It is worth mentioning that the last two phases (design and implementation ones) are normally performed together when ontology development tools (such as NeOn Toolkit, Protégé, etc.) are used.
- *Maintenance phase*. If, during the use of the ontology network, errors or missing knowledge are detected, then the ontology development team should go back to the design phase. Additionally, in this phase the generation of new versions for the ontology network should also be carried out.

2.4.1.2 The Five-Phase Waterfall Ontology Network Life Cycle Model

This model extends the four-phase model with a new phase in which the reuse of already implemented ontological resources is considered. The main purpose in the *reuse phase* is to obtain one or more ontological resources to be reused in the

ontology network being developed. The output of this reuse phase could be either an informal model or a formal one to be used in the design phase, or an implemented model (in an ontology language) to be used in the implementation phase.

For the other phases, the purposes and outcomes are the same as those presented in the four-phase model.

2.4.1.3 The Five-Phase + Merging Phase Waterfall Ontology Network Life Cycle Model

This model is a special case of the five-phase model. Now, a new phase (the *merging phase*) is added after the reuse one. This merging phase has as a main purpose to obtain a new ontological resource from two or more ontological resources selected in the reuse phase.

For the other phases, the purposes and outcomes are the same as those presented in the five-phase model.

2.4.1.4 The Six-Phase Waterfall Ontology Network Life Cycle Model

In this model, the five-phase model is taken as general basis, and a new phase (*re-engineering phase*) is included after the reuse one. This model allows the reuse of knowledge resources (ontological and non-ontological) and their later re-engineering. In this model, the reuse phase has as output one or more knowledge resources to be reused in the ontology network that is being developed. After this phase, the non-ontological resources are transformed into ontologies in the re-engineering phase; the ontological resources, on the other hand, can or cannot be re-engineered, a decision that should be taken by the ontology development team.

For the other phases, the purposes and outcomes are the same as those presented in the six-phase model.

2.4.1.5 The Six-Phase + Merging Phase Waterfall Ontology Network Life Cycle Model

This model, extended from the six-phase model, includes the *merging phase* after the reuse phase. For the other phases, the purposes and outcomes are the same as those presented in the six-phase model.

2.4.2 Iterative-Incremental Ontology Network Life Cycle Model

The main feature of this model is the development of ontology networks organized in a set of iterations (or short mini-projects with a fixed duration). Each individual iteration is similar to an ontology network project that uses any

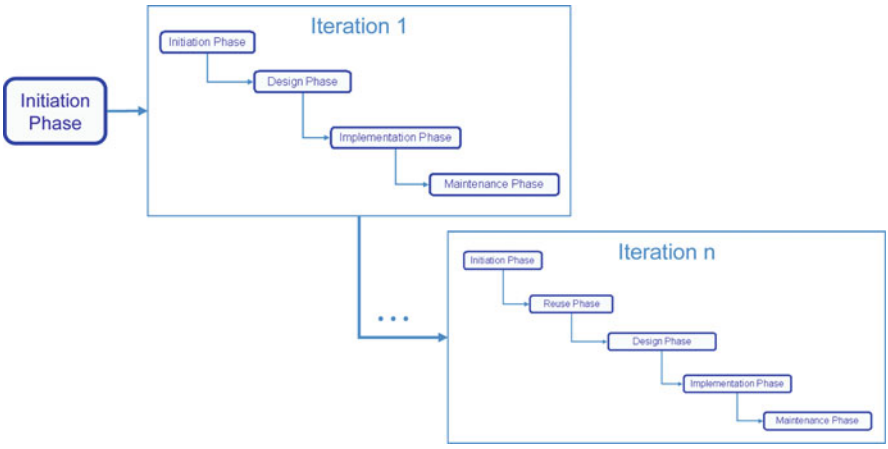

Fig. 2.5 Schematic vision of the iterative-incremental model

type of waterfall model from those presented in Sect. 2.4.1, as shown schematically in Fig. 2.5.

This model could be used in the following situations:

- In ontology projects with large groups of developers having different profiles and roles
- In ontology projects in which the development involves several different domains that are not well understood
- In ontology projects in which requirements are not completely known or can change during the ontology development

Ontology requirements specified in the ORSD can be divided in different subsets. The result of any iteration is a functional and partial ontology network that meets a subset of the ontology network requirements. Such a partial ontology network can be used, evaluated, and integrated in any other ontology network.

This model is based on the continuous improvement and extension of the ontology network resulted from performing multiple iterations with cyclic feedback and adaptation. In this way, the ontology network grows incrementally along the development. Generally, in each iteration new requirements are taken into account, but, occasionally, in a particular iteration, the partial ontology network could be only enhanced.

This model focuses on a set of basic requirements; from these requirements, a subset is chosen and considered in the development of the ontology network. The partial result is reviewed, the risk of continuation with the next iteration is analyzed and the initial set of requirements is increased and/or modified in the next iteration until the complete ontology network is developed.

The main benefit of this model is to identify and alleviate the possible risks as soon as possible. Other benefits are:

- The development team is motivated by rapidly producing an adequate ontology.
- Some priorities can be established in the set of requirements.
- The development can be possibly adapted to changes in the requirements.
- The scheduling of each iteration can be adapted based on the experience of previous iterations.

It is worth mentioning that at the beginning of the ontology network project, the number of iterations during the ontology project is influenced by:

- The decision of performing a more complete and detailed ontology requirements specification. In this case, the number of iterations will be lower.
- The decision of carrying out a simpler and less complete requirements specification, in which case more number of iterations and more revisions will be needed.

Figure 2.5 shows the schematic vision of the iterative-incremental model. The first initiation phase shown in the figure has as main outcomes the ontology network requirements and the general and global plan for the whole ontology network development. Regarding the different iterations, as mentioned before, each iteration in the iterative-incremental model can follow a different version of the waterfall model from those presented in Sect. 2.4.1. However, any version of the waterfall model to be used in the iterative-incremental model should be modified in the following way:

- No backtracking is allowed between phases in a particular iteration, because the refinement should be performed in the next iterations.
- Revising the ontology network requirements and the global plan should be carried out in the initiation phase of each iteration. Additionally, a detailed plan for the particular iteration should be performed.

2.4.3 Relation Between Scenarios and Life Cycle Models

The set of nine flexible scenarios for building ontologies and ontology networks presented in Sect. 2.3 and the two proposed ontology network life cycle models presented in this section are intrinsically related because both scenarios and life cycle models have been created (1) taking into account the importance of reusing and re-engineering knowledge resources (ontological and non-ontological) and merging ontological resources and (2) assuming a controlled setting for ontology engineering in which approaches such as mining ontologies from tags are not considered.

Table 2.2 summarizes the relationships between scenarios for building ontology networks and ontology network life cycle models. These relationships have been established based on the following:

Table 2.2 Relation between scenarios and life cycle models

	Four-phase model	Five-phase model	Five-phase + merging phase model	Six-phase model	Six-phase + merging phase model
Scenario 1	X				
Scenario 2				X	
Scenario 3		X			
Scenario 4				X	
Scenario 5			X		
Scenario 6					X
Scenario 7		X			
Scenario 8	X				
Scenario 9	X				

- Scenario 1 (as stated in Sect. 2.3.1) is for building ontology networks from scratch. The scenario mainly includes core activities such as specification, conceptualization, and implementation. This way of building ontologies fits with the stages represented in the four-phase waterfall model (initiation phase, design phase, implementation phase, and maintenance phase).
- Scenario 2 (as stated in Sect. 2.3.2) is for building ontology networks by reusing and re-engineering non-ontological resources, which is represented in the six-phase waterfall model.
- Scenario 3 (as stated in Sect. 2.3.3) is for building ontology networks by reusing ontological resources. This way of building ontologies is represented by the five-phase waterfall model.
- Scenario 4 (as stated in Sect. 2.3.4) refers to the development of ontology networks by reusing and re-engineering ontological resources. This way of building ontologies is represented by the six-phase waterfall model.
- Scenario 5 (as stated in Sect. 2.3.5) is for building ontology networks by reusing and merging ontological resources, which is represented by the five-phase + merging phase waterfall model.
- Scenario 6 (as stated in Sect. 2.3.6) refers to the development of ontology networks by reusing, merging, and re-engineering ontological resources. This way of building ontology networks is represented by the six-phase + merging phase waterfall model.
- Scenario 7 (as stated in Sect. 2.3.7) is for building ontology networks by reusing ontology design patterns, which is represented by the five-phase waterfall model.
- Scenario 8 (as stated in Sect. 2.3.8) is for building ontology networks by restructuring ontological resources. This is mainly related to the core activities already mentioned in Scenario 1. Thus, this Scenario 8 is also represented by the four-phase waterfall model.
- Scenario 9 (as stated in Sect. 2.3.9) refers to the development of ontology networks by localizing ontologies. This way of building ontologies is mainly related to Scenario 1 and thus represented by the four-phase waterfall model.

As explained in Sect. 2.4.2, the iterative-incremental model is basically formed by a set of iterations that can follow any version of waterfall ontology network life cycle model. Thus, the relation between scenarios and the iterative-incremental model depends on the different versions of waterfall model used in the iterative-incremental one, and for this reason, the relations presented in Table 2.2 are also valid for this model.

2.5 Methodological Guidelines for Processes and Activities

In the second part of this book (called Ontology Engineering Activities), methodological guidelines for a subset of the processes and activities included in the NeOn Glossary are provided. To describe each of the processes and activities included in the NeOn Methodology presented in this book, the following content is provided for most of the cases:

- A general introduction to the process or activity, where the value of the process or activity is discussed.
- The detailed guidelines proposed for carrying out the process or the activity, including the following fields: (a) *definition*, which is taken from the NeOn Glossary of Processes and Activities and included in Sect. 2.2; (b) *goal*, which explains the main objective intended to be achieved by the process or the activity; (c) *input*, which includes the resources needed for carrying out the process or the activity; (d) *output*, which includes the results obtained after carrying out the process or the activity; (e) *who*, which identifies the people or teams involved in the process or the activity; and (f) *when*, which explains in which stage of the development the process or the activity should be carried out.

  All the aforementioned information is provided in the so-called *filling cards*. These filling cards explain the information of each process and activity of the NeOn Methodology in a practical and easy way. Each card is filled according to the *filling card template* shown in Table 2.3.
- A graphical *workflow* that shows how the process or the activity should be carried out is also included. This workflow contains the inputs, outputs, actors involved, and details for carrying out a process or activity in a prescriptive manner. Additionally, methods, techniques, and tools supporting the process or activity are proposed.
- Examples explaining the guidelines proposed are also given.

It should be noted that in the framework of the NeOn Methodology, there are a wide range of prescriptive methodological guidelines for carrying out different processes and activities. Along this book, the reader can find guidelines for *Scenario 1*, particularly for ontology requirements specification (Chap. 5) and scheduling (Chap. 14), *Scenario 2* (Chap. 6), *Scenario 3* (Chap. 7), *Scenario 5* (Chap. 12), *Scenario 7* (Chap. 3), *Scenario 8*, for ontology modularization (Chap. 10), and *Scenario 9* (Chap. 8). In addition, there are also guidelines for ontology evaluation (Chap. 9) and for ontology evolution (Chap. 11).

Table 2.3 Template for the process and activity filling card

Process or Activity Name	
Definition	
Goal	
Input	Output
Who	
When	

References

Bizer C, Heath T, Berners-Lee T (2009) Linked data – the story so far. Int J Semant Web Inf Syst 5 (3):1–22

Blomqvist E, Gangemi A, Presutti V (2009) Experiments on pattern-based ontology design. In: Proceedings of the 5th international conference on Knowledge Capture (K-CAP 2009), Redondo Beach, CA, USA, 1–4 Sept 2009. ISBN: 978-1-60558-658-8

Byrne EJ (1992) A conceptual foundation for software re-engineering. In: Proceedings of the international conference on software maintenance and reengineering. IEEE Computer Society Press, Orlando, pp 226–235

Clemente J, Ramírez A, de Antonio A (2011) A proposal for student modeling based on ontologies and diagnosis rules. Expert Syst Appl 38(7):8066–8078

Cuenca-Grau B, Horrocks I, Kazakov Y, Sattler U (2007) Just the right amount: extracting modules from ontologies. In: Proceedings of the 16th international conference on world wide web, Banff, Alberta, Canada, pp 717–726. ISBN: 978-1-59593-654-7

Espinoza M, Montiel-Ponsoda E, Gómez-Pérez A (2009) Ontology Iocalization. In: Proceedings of the fifth international conference on Knowledge Capture (KCAP 2009), Redondo Beach, CA, USA, pp 33–40. ISBN: 978-1-60558-658-8

Fernández-López M (1999) Overview of methodologies for building ontologies. In: Proceedings of the IJCAI-99 workshop on ontologies and problem-solving methods: lessons learned and future trend, Stockholm, Sweden, August 1999. http://oa.upm.es/5480/

Gangemi A (2005) Ontology design patterns for semantic web content. In: Musen M et al (eds) Proceedings of the fourth international semantic web conference, Galway, Ireland. Springer, Berlin

Gangemi A (2007) Design patterns for legal ontology construction. In: Casanovas P, Noriega P, Bourcier D, Galindo F (eds) Trends in legal knowledge: the semantic web and the regulation of electronic social systems. European Press Academic Publishing, Florence

Gómez-Pérez A, Fernández-López M, Corcho O (2003) Ontological engineering. Advanced information and knowledge processing series. Springer, Heidelberg, ISBN 1-85233-551-3

IEEE Standard Glossary of Software Engineering Terminology (1990) IEEE Std. 610.12–1990 (Revision and redesignation of IEEE Std. 792–1983)

Jimeno-Yepes A, Jimenez-Ruiz E, Berlanga-Llavori R, Rebholz-Schuhmann D (2009) Reuse of terminological resources for efficient ontological engineering in life sciences. BMC Bioinformatics 10:S4. ISSN: 1471–2105

Lamsfus C, Alzua-Sorzabal A, Martin D, Salvador Z, Usandizaga A (2009) Human-centric ontology-based context modelling in tourism. In: Proceedings of KEOD 2009 Proceedings of the International Conference on Knowledge Engineering and Ontology Development, Funchal - Madeira, Portugal, October 6–8, 2009. INSTICC Press 2009, ISBN 978-989-674-012-2, pp 424–434

Newell A (1982) The knowledge level. Artif Intell 18(1):87–127

Pan JZ, Lancieri L, Maynard D, Gandon F, Cuel R, Leger A (2007) Knowledge web deliverable D1.4.2.v2. Success stories and best practices. Available at http://knowledgeweb.semanticweb.org/semanticportal/deliverables/D1.4.2v2.pdf

Pease RA, Niles I, Li J (2002) The suggested upper merged ontology: a large ontology for the semantic web and its applications. In: Workshop on ontologies and the semantic web at the AAAI 2002, Edmonton

Poveda-Villalón M, Suárez-Figueroa MC, García-Castro R, Gómez-Pérez A (2010) A context ontology for mobile environments. In: Workshop on Context, Information and Ontologies (CIAO 2010) co-located with EKAW 2010, Lisbon

Presutti V, Gangemi A (2008) Content ontology design patterns as practical building blocks for web ontologies. In: Proceedings of the 27th international conference on conceptual modeling (ER2008), Barcelona, Spain

Simperl E (2009) Reusing ontologies on the Semantic Web: A feasibility study. Data Knowledge Engineering 68(10):905–925

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

Vilches-Blázquez LM, Villazón-Terrazas B, Saquicela V, de Leon A, Corcho O, Gómez-Pérez A (2010) GeoLinked data and INSPIRE through an application case. In: 18th ACM SIGSPATIAL international conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2010), San Jose, CA, 2–5 Nov 2010

Villazón-Terrazas B, Suárez-Figueroa MC, Gómez-Pérez A (2010) A pattern-based method for re-engineering non-ontological resources into ontologies. Int J Semant Web Inf Syst 6(4):27–63

Villazón-Terrazas B, Ramírez J, Suárez-Figueroa MC, Gómez-Pérez A (2011) A network of ontology networks for building e-employment advanced systems. Expert Syst Appl 38(11):13612–13624

http://www.springer.com/978-3-642-24793-4

Ontology Engineering in a Networked World

(Eds.)M.C. Suárez-Figueroa; A. Gómez-Pérez; E. Motta; A. Gangemi

2012, XII, 444 p. 148 illus., Hardcover
ISBN: 978-3-642-24793-4