## Main Tasks in Ontology Development

Ontology development has similar tasks to the software development life cycle (SDLC) and can be grouped into three overlapping sections. Note these sections define types of activities and are not necessarily done in a linear fashion.

| 1. Requirements and Knowledge Acquisition | 2. Ontology Construction | 3. Testing and Maintenance |
|---|---|---|
| These tasks gather the requirements that the output ontology should fulfill, identify who will use it and why and document them for use in the construction and testing of the ontology. | These tasks design, formalise and construct the ontology into a formalised, shareable, computer readable format. | These tasks make sure the ontology satisfies the specifications outlined and adapt the ontology to any further changes. |
| **1.1. Identification**: List all accessible resources to obtain key concepts, relationships, terminologies and ontological requirements, why the ontology is needed and who will use it. These are sourced from from domain documentation, glossaries, online resources, existing databases and ontologies, and interviews with domain experts. | **2.1. Conceptualization**: Define the main concepts and their relationships within the domain. This task involves identifying classes, subclasses, properties, attributes, and their hierarchical structure. | **3.1. Verification:** Assess the quality of an ontology based on certain criteria and metrics, which includes CQ verification (a test against the specifications made) and metric-based evaluation (a quantitative evaluation of the ontology quality). |
| **Output**: Domain Description Document. | **Outputs**: Concept hierarchy or taxonomy | **Outputs**: Test reports |
| **1.2. Domain Analysis**: Analyse and turn the user requirements in the DDD into distinct and well-defined functions that illustrate diagrammatically and textually the key concepts and their relationships. | **2.2. Formalization**: Express the ontology using a formal language to ensure precise semantics and enable automated reasoning. This involves specifying logical axioms, constraints, and rules that govern the behavior and consistency of the ontology. | **3.2. Maintenance:** Make updates or corrections to the ontology, as necessary. After evaluation or publishing the online version of the ontology may have possible errors or missing domain knowledge. |
| **Output**: UML use case diagram. | **Outputs**: An OWL/RDF or other computer readable file representing an empty ontology. | **Outputs**: An updated OWL/RDF or other computer readable file with a populated ontology. |

## Main Tasks in Ontology Development (cont)

**1.3. Specification**: Transform the refined requirements list into a list of competency questions (CQs) which are used as ontology functional requirements. The final ontology can be tested against these CQs.

**Output**: Competency Questions (CQs).

**2.3. Implementation:** Populate the ontology with new data properties and individuals and assigns them to the appropriate classes, relationship mapping etc using an ontology editor (such as protege).

**Outputs**: An OWL/RDF or other computer readable file with a populated ontology.

**4. Documentation**

Documentation is done throughout each of the above tasks. Why decisions are made, all the artefacts created and other management documentation is done throughout the life cycle of the ontology.

## Overview of Methodological Approach

There are many methodological approaches to completing the tasks of ontology development. These can be broken down into **macro-level** methodologies, which outline the entire development process in a structured fashion, and **micro-level** methodologies, which focus on the formalization aspects of ontology authoring.

**1. Macro-level methodologies**

**1.1. Waterfall**: a sequential design process which follows a linear, step-by-step approach where each phase must be completed before moving on to the next. Specification, conceptualization, formalization, implementation, maintenance are all done one after the other (though NeON has many different routes and sub-steps).

**Examples:** Methontology, NeON

**2. Micro-level methodologies**

These methodologies focus on the guidelines to formalize the subject domain. Specifically moving from an informal representation to a formal, logic based representation. Though these share many of the steps of the macro-level methodologies, their focus/viewpoint is on implementation specific choices. i.e. distributed or or not, logic framework choice, design architecture, expressiveness, foundational ontology choice, ontology language choice etc. Their focus is on detail rather than overview which the macro-level focusses on.

**Examples:** OntoSpec, OD101, DiDOn

## Overview of Methodological Approach (cont)

**1.2. Lifecycle:** a framework for the development and management of ontologies throughout their entire lifecycle, from inception to retirement. The phases identified (requirements development, ontological analysis, ontology design, system design, ontology development & reuse, system development & integration, deployment and operation & maintenance) outline common abstracted activities and are either done in parallel or sequentially and are repeated or only done once, depending on the management of the development process.

**Examples:** Ontology Life Cycle Model

**1.2. Agile:** an approach that emphasizes flexibility, collaboration, and iterative progress. These ontologies are developed incrementally through short iterations where each iteration focusses on a specific aspect of the ontology. In this way it is easier to be adaptable to changes, to be able to respond to stake holder feedback, to collaborate in smaller bursts and to be able to have working prototypes earlier in development.

**Examples:** OntoMaven

## Main Features of Foundational Ontologies

Foundational ontologies are high-level abstractions of common ontological patterns.