

UNIT 5

Ontology Engineering

(6 Hours) — Knowledge Engineering (CACS458)

⚠ Please Note

These notes are generated to help with revision and **may contain errors or omissions**. Please verify important definitions, formulas and examples against your textbook, lecture notes and the official TU syllabus before relying on them in an exam.

5.1 Overview of Ontology

In philosophy, *ontology* is the study of what exists. In computer science, Tom Gruber defined an **ontology** as “**an explicit specification of a conceptualisation**”. In practice, an ontology is a formal, shared and machine-readable model of the concepts in a domain and the relationships between them.

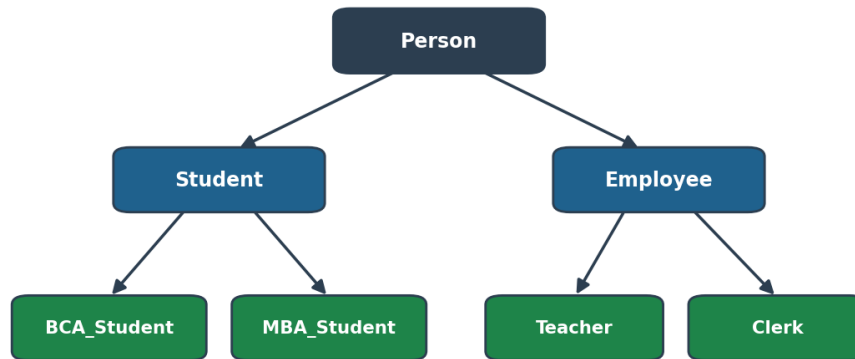
Key Characteristics

- **Formal** — expressed in a logic-based language so machines can reason with it.
- **Explicit** — concepts and constraints are clearly defined.
- **Shared** — represents consensus knowledge of a community.
- **Conceptualisation** — an abstract model of the domain’s concepts.

Components of an Ontology

Component	Meaning	Example
Classes / Concepts	Categories of things in the domain	Person, Student, Course
Instances / Individuals	Specific objects belonging to classes	Ram (a Student), CACS458 (a Course)
Properties / Relations	Links between objects, or object and value	enrolledIn, hasName, teaches
Attributes	Features of concepts with data values	age = 21
Axioms / Constraints	Rules that must always hold	A Student must be enrolled in at least one Course

Component	Meaning	Example
Taxonomy (is-a hierarchy)	Subclass–superclass structure	Student $\sqsubseteq$ Person



Arrows denote is-a (subclass-of) relationships

Figure 5.1: A small class hierarchy (taxonomy) in the people domain.

Why Ontologies?

- Share a common understanding of a domain between people and software agents.
- Enable reuse of domain knowledge and separate it from operational knowledge.
- Make domain assumptions explicit and support automatic reasoning.
- Enable interoperability and integration of heterogeneous data (the foundation of the Semantic Web).

Basis	Database schema	Ontology
Purpose	Store and query data efficiently	Capture meaning of a domain
Assumption	Closed world	Usually open world
Reasoning	Limited	Supports logical inference
Sharing	Application-specific	Designed to be shared and reused

5.2 Classifications of Ontology

(a) By Level of Generality (Guarino)

Type	Description	Example
Top-level / upper ontology	Very general concepts independent of any domain (object, event, time, space)	SUMO, DOLCE, Cyc upper level
Domain ontology	Concepts of one domain	Medical ontology (SNOMED CT), Gene Ontology
Task ontology	Concepts related to a generic task or activity	Diagnosis, scheduling
Application ontology	Specialised for a particular application; combines domain and task ontologies	A university course-registration system

(b) By Level of Formality

- **Highly informal** — natural-language glossary.
- **Semi-informal** — structured natural language.
- **Semi-formal** — expressed in an artificial, formally defined language (e.g. frames, UML).
- **Rigorously formal** — theorems and logic-based definitions with proofs (e.g. OWL DL).

(c) Lightweight vs Heavyweight

Lightweight ontology	Heavyweight ontology
Concepts, taxonomies, simple relations	Adds axioms, constraints and rich logical rules
Easier to build; limited reasoning	Powerful reasoning; costly to build
Examples: thesauri, taxonomies, Dublin Core	Examples: full OWL ontologies, Cyc

(d) By Purpose / Scope (other views)

- **Knowledge-representation ontologies** (e.g. Frame Ontology, OWL).
- **Linguistic ontologies** such as WordNet.
- **Meta-ontologies / generic ontologies** reused across domains.

5.3 Methodology Used in Ontology Engineering

Ontology engineering is the set of activities for building, maintaining and using ontologies. Several methodologies exist; all follow a life-cycle of *specification* → *conceptualisation* → *formalisation* → *implementation* → *evaluation* → *maintenance*.

Methodology	Main idea / steps
Uschold & King	Identify purpose and scope → build the ontology (capture, code, integrate existing ontologies) → evaluate → document.
Grüninger & Fox (TOVE)	Motivating scenarios → informal competency questions → terminology in FOL → formal competency questions → axioms → completeness theorems.
METHONTOLOGY	Specification, knowledge acquisition, conceptualisation, integration, implementation, evaluation, documentation; follows an evolving-prototype life cycle.
On-To-Knowledge (OTKM)	Feasibility study → kick-off → refinement → evaluation → application and evolution; includes knowledge-management aspects.
DILIGENT	Distributed, loosely controlled and evolving; supports collaborative construction by many stakeholders.
NeOn	Scenarios for building networks of ontologies, reuse and re-engineering.
“Ontology Development 101” (Noy & McGuinness)	A practical, iterative seven-step guide (below).

Ontology Development 101 — The Seven Steps

- **Step 1:** Determine the domain and scope (competency questions).
- **Step 2:** Consider reusing existing ontologies.
- **Step 3:** Enumerate important terms.
- **Step 4:** Define classes and the class hierarchy (top-down, bottom-up or combination).
- **Step 5:** Define properties of classes (slots).
- **Step 6:** Define facets (cardinality, value type, domain/range).
- **Step 7:** Create instances.

Example 5.1 — Building a small university ontology

Domain: a college/university. Competency questions: Which courses does a student take? Who teaches a course? Which department offers a programme?

Result of steps 3–7

Terms: University, College, Department, Programme, Course, Student, Teacher.

Hierarchy: Person → {Student, Teacher}; Organisation → {University, College, Department}.

Properties: enrolledIn (Student→Course), teaches (Teacher→Course), offeredBy (Programme→Department), hasName (xsd:string).

Facets: a Course must have exactly one offeredBy department (cardinality = 1).

Instances: Ram (Student), TU (University), BCA (Programme), CACS458 (Course).

Ontology Tools and Evaluation

- **Editors:** Protégé (most popular), TopBraid Composer, OntoEdit.
- **Reasoners:** HermiT, Pellet, FaCT++ — check consistency and classify.
- **Evaluation criteria:** clarity, coherence, extendibility, minimal encoding bias, minimal ontological commitment (Gruber).
- **Methods for building:** manual, semi-automatic ontology learning from text, and reuse/merging (ontology alignment).

5.4 Ontology versus Language

An ontology is a *model of content* (what a domain contains), while an **ontology language** is the *notation* used to write that model down. The same ontology can often be written in different languages, just as the same story can be written in different human languages.

Basis	Ontology	Ontology language
What it is	A conceptual model of a domain (classes, relations, axioms)	A formal syntax and semantics for writing ontologies
Example	University ontology; Gene Ontology	RDF(S), OWL, KIF, Ontolingua, F-Logic
Concerned with	Content — what we say	Expressiveness — how we can say it
Reusable by	Many applications in the domain	Many ontologies

Common Ontology Languages

Language	Notes
KIF / Ontolingua	Early, very expressive FOL-based languages for knowledge sharing.

Language	Notes
F-Logic / OIL	Frame-based logic; OIL combined frames with description logic.
RDF	Graph of subject–predicate–object triples; basic web data model.
RDFS	Adds classes, subclass, properties, domain and range to RDF.
OWL (Lite, DL, Full; OWL 2)	W3C standard based on description logic; richer constraints and reasoning.
SKOS	Lightweight vocabulary for thesauri and taxonomies.

Note: Trade-off: a more expressive language gives more modelling power but makes reasoning slower or even undecidable. OWL DL was designed to keep reasoning decidable.

5.5 Ontologies in Colleges and Universities

Note: The syllabus lists “Colleges & Universities” as the last point of this unit; this seems to be a listing artefact, so please confirm with your teacher or the official syllabus. The short section below covers it as an application area.

Educational institutions are a good example domain for ontology engineering because they contain many connected concepts (students, teachers, courses, programmes, departments, exams) that must be shared across several information systems.

- **Course and curriculum ontologies** — describe programmes, subjects, prerequisites and learning outcomes.
- **Student information integration** — merge admission, exam, library and finance data using one shared vocabulary.
- **Semantic search** over research papers, lecture notes and e-learning content.
- **Recommendation** of courses or resources to students based on their profile.
- **Interoperability** between a college and its affiliating university (e.g. results, enrolment data).

Key Takeaways — Unit 5

- ✓ An ontology is a formal, explicit, shared specification of a conceptualisation of a domain.
- ✓ Components: classes, instances, properties, attributes, axioms and a taxonomy.
- ✓ Classified by generality (top-level, domain, task, application) and by formality or richness (lightweight vs heavyweight).
- ✓ Methodologies: Uschold & King, TOVE, METHONTOLOGY, On-To-Knowledge, DILIGENT, and the 7-step Ontology Development 101.
- ✓ An ontology is the content model; an ontology language (RDF(S), OWL, KIF) is the notation used to express it.

— *End of Unit 5* —

bcalink.com