



Standard

EFS 2507:2025

**Terminology management —
Concept relationships**

First edition

© 2025 Enosema Foundation.

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from the Enosema Foundation.

Enosema Foundation
2885 Sanford Ave Sw Suite 32361
Grandville MI 49418-1342
USA

E-mail: info@enosema.org

Published in the USA

Contents

Page

Abstract	vi
Foreword	vii
Introduction	viii
1 Scope	1
2 Normative references	2
3 Terms and definitions	2
4 Abstract model for concept relationships	4
4.1 General	4
4.2 Core components	4
4.2.1 Relationship structure	4
4.2.2 Relationship assertion	5
4.2.3 Relationship context	5
4.3 Relationship properties	6
4.3.1 Logical properties	6
4.3.2 Cardinality	6
4.3.3 Directionality	6
4.3.4 Strength	6
4.4 Metarelationships	7
4.5 Relationship lifecycle	7
4.5.1 Creation and documentation	7
4.5.2 Validation	7
4.5.3 Evolution and maintenance	8
4.6 Abstract model implementation	8
5 Classification of relationship types	8
5.1 General	8
5.2 Hierarchical relationships	9
5.2.1 General characteristics	9
5.2.2 Generic relationships	9
5.2.3 Partitive relationships	9
5.2.4 Instance relationships	10
5.3 Associative relationships	10
5.3.1 General characteristics	10
5.3.2 Causal relationships	11
5.3.3 Temporal relationships	11
5.3.4 Spatial relationships	11
5.3.5 Functional relationships	11
5.3.6 Developmental relationships	12
5.3.7 Possessive relationships	12
5.4 Equivalence relationships	12
5.4.1 General characteristics	12
5.4.2 Exact equivalence	12
5.4.3 Inexact equivalence	13
5.4.4 Partial equivalence	13
5.4.5 Cross-language equivalence	13
5.5 Relationship type registry	14
5.5.1 Purpose and structure	14
5.5.2 Extension mechanisms	14
6 Domain contexts	15
6.1 General	15
6.2 Same-domain relationships	15
6.2.1 Characteristics of same-domain relationships	15
6.2.2 Domain-specific relationship patterns	15
6.2.3 Domain-specific relationship types	16

6.3	Cross-domain relationships.....	16
6.3.1	Challenges of cross-domain relationships.....	16
6.3.2	Approaches for establishing cross-domain relationships.....	16
6.3.3	Cross-domain relationship documentation.....	17
6.4	Context-sensitive relationship semantics.....	17
6.4.1	Contextual variation in relationship meaning.....	17
6.4.2	Context parameters.....	18
6.4.3	Context-based relationship constraints.....	18
6.5	Managing domain boundaries.....	18
6.5.1	Domain boundary identification.....	18
6.5.2	Boundary-spanning concepts.....	19
6.5.3	Domain relationship mapping.....	19
7	Implementation in terminology systems.....	19
7.1	General.....	19
7.2	Implementation in reference works.....	20
7.2.1	Dictionaries and glossaries.....	20
7.2.2	Thesauri.....	20
7.3	Implementation in knowledge organization systems.....	21
7.3.1	Taxonomies.....	21
7.3.2	Ontologies.....	21
7.3.3	Semantic networks.....	22
7.4	Implementation in terminology databases.....	22
7.4.1	TBX implementation.....	22
7.4.2	Relational database implementation.....	22
7.5	User interface considerations.....	23
7.5.1	Relationship visualization.....	23
7.5.2	Relationship editing.....	23
7.5.3	Search and retrieval.....	23
7.6	Implementation challenges and solutions.....	24
7.6.1	Scalability.....	24
7.6.2	Relationship maintenance.....	24
7.6.3	Cross-system integration.....	24
8	Enosema's approach.....	24
8.1	General.....	24
8.2	Core principles.....	25
8.2.1	Concept-centricity.....	25
8.2.2	Relationship explicitness.....	25
8.2.3	Contextual validity.....	25
8.2.4	Semantic precision.....	25
8.2.5	Practical usability.....	25
8.3	Enosema relationship registry.....	26
8.3.1	Registry structure.....	26
8.3.2	Core relationship types.....	26
8.3.3	Extension mechanisms.....	27
8.4	Documentation guidelines.....	27
8.4.1	Relationship assertion documentation.....	27
8.4.2	Context documentation.....	28
8.4.3	Relationship validation documentation.....	28
8.5	Quality assurance.....	28
8.5.1	Validation processes.....	28
8.5.2	Quality criteria.....	28
8.5.3	Governance framework.....	29
8.6	Implementation support.....	29
8.6.1	Implementation patterns.....	29
8.6.2	Conformance levels.....	29
8.6.3	Migration strategies.....	30
8.7	Enosema tools and resources.....	30
8.7.1	Relationship management tools.....	30

EFS 2507:2025

8.7.2	Training and guidance.....	30
8.7.3	Community resources.....	30
Annex A Common relationship patterns.....		32
Annex B Mappings to existing standards.....		41
Bibliography		47

Abstract

This document specifies an abstract model for defining, documenting, and managing concept relationships in terminology resources and knowledge organization systems. It provides a generic mechanism for concept relationships that can be implemented across different standards, frameworks, and applications.

Foreword

The Enosema Foundation (“ES”) is the premier non-profit standardization organization for terminology standardization and terminology related processes such as management. It facilitates the education, standardization, research, promotion, definition, and usage of terminology resources and management practices globally.

ES works with international partners and experts across the globe, reflecting the international nature of its mission. More information about ES is available on the official website (<https://www.enosema.org>).

The procedures used to develop this document and those intended for its further maintenance are described in the ES Directives.

In particular, the different approval criteria needed for the different types of ES documents should be noted. This document was drafted in accordance with the editorial rules of the ES Directives.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ES shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be provided in the Introduction.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

This document was prepared by Technical Committee *Terminology*.

Introduction

Concepts form the foundation of human knowledge and understanding. They represent units of thought that allow us to categorize, organize, and communicate about the world around us. However, concepts do not exist in isolation. They derive much of their meaning and utility from the relationships they form with other concepts. These concept relationships are fundamental to terminology management, knowledge organization, and semantic interoperability.

In our increasingly interconnected and information-rich world, the ability to precisely define, document, and exchange concept relationships has become essential. Organizations across domains face challenges in managing terminology consistently, sharing knowledge across boundaries, and ensuring that information systems can interoperate effectively. At the heart of these challenges lies the need for a standardized approach to concept relationships.

While numerous standards and frameworks exist for representing concept relationships—including UML, OWL, SKOS, and various ISO standards—they often implement relationships in ways specific to their particular paradigms and applications. This has led to fragmentation and challenges in cross-system interoperability. What has been missing is an abstract, generic mechanism for defining concept relationships that can serve as a foundation for these various implementations.

NOTE 1 Concept relationships exist independently of their representations. The same underlying relationship can be expressed through various notations, visualizations, or formalisms without changing its essential nature. This distinction between relationships and their representations is crucial for developing a truly abstract model.

The challenges of managing concept relationships include:

- Lack of standardization in relationship types and their semantics
- Difficulty in establishing relationships across domain boundaries
- Inconsistent implementation of relationships in different systems
- Ambiguity in relationship meaning and interpretation
- Limited mechanisms for documenting relationship properties and constraints
- Challenges in validating relationship assertions

This document addresses these challenges by providing a comprehensive framework for concept relationships that is abstract enough to be realized in various implementation contexts while being specific enough to ensure semantic precision. It establishes a foundation for:

- Defining concept relationships in a standardized, implementation-independent manner
- Classifying relationship types according to their semantic properties
- Documenting relationship characteristics and constraints
- Managing relationships within and across domain boundaries
- Implementing relationships in various terminology and knowledge systems

The framework presented in this document is designed to support a wide range of applications, including:

- Terminology databases and management systems
- Knowledge organization systems (taxonomies, thesauri, ontologies)
- Semantic networks and knowledge graphs
- Information retrieval systems
- Natural language processing applications

— Data integration and exchange platforms

By providing a generic mechanism for defining concept relationships, this document aims to enhance semantic interoperability, improve terminology management practices, and facilitate knowledge exchange across systems, domains, and languages.

NOTE 2 While concept relationships can be visualized through concept maps and similar representations, this document focuses on the underlying relationship structures rather than specific visualization techniques. The abstract model presented here can be realized through various visual and non-visual representations.

This standard builds upon the significant contributions of terminology and knowledge organization experts worldwide and the foundational work established in ISO standards. The Enosema Foundation recognizes the critical importance of concept relationships in facilitating mutual understanding across domains and disciplines.

Terminology management — Concept relationships

1 Scope

This document specifies an abstract model for defining, documenting, and managing concept relationships in terminology resources and knowledge organization systems. It provides a generic mechanism for concept relationships that can be implemented across different standards, frameworks, and applications.

The abstract model and guidelines specified in this document apply to concept relationships in all subject fields, domains, and disciplines. The standardized approach enhances semantic precision, improves interoperability, and facilitates knowledge exchange across systems and domains.

This document provides:

- An abstract model for concept relationships independent of specific implementation technologies
- A classification system for relationship types based on their semantic properties
- Guidelines for documenting relationship characteristics and constraints
- Approaches for managing relationships within and across domain boundaries
- Implementation guidance for various terminology and knowledge systems
- Examples illustrating the application of the abstract model in different contexts

This document is intended for:

- Terminology managers and terminologists responsible for developing and maintaining terminology resources
- Knowledge engineers and ontologists designing knowledge organization systems
- Standards developers working on terminology and knowledge representation standards
- Information architects designing information retrieval systems
- Software developers implementing terminology management and knowledge organization systems
- Subject matter experts contributing to terminology resources and knowledge bases
- Researchers in terminology science, knowledge organization, and semantic technologies

The abstract model specified here complements existing standards for terminology management, knowledge organization, and semantic technologies, providing a foundation that can be realized in various implementation contexts.

This document does not:

- Prescribe specific notations or formalisms for representing relationships
- Replace existing standards for terminology management or knowledge organization
- Specify implementation details for particular systems or applications
- Mandate specific visualization techniques for concept relationships
- Provide domain-specific relationship taxonomies

Instead, it offers a generic framework that can be adapted and extended to meet the needs of specific domains, applications, and implementation contexts.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 704:2022, *Terminology work — Principles and methods*

ISO 1087:2019, *Terminology work and terminology science — Vocabulary*

ISO 24156-1:2014, *Graphic notations for concept modelling in terminology work and its relationship with UML — Part 1: Guidelines for using UML notation in terminology work*

ISO 16642:2017¹⁾, *Computer applications in terminology — Terminological markup framework*

ISO 30042:2019, *Management of terminology resources — TermBase eXchange (TBX)*

ISO 19146:2018, *Geographic information — Cross-domain vocabularies*

ISO 25964-1:2011, *Information and documentation — Thesauri and interoperability with other vocabularies — Part 1: Thesauri for information retrieval*

ISO 25964-2:2013, *Information and documentation — Thesauri and interoperability with other vocabularies — Part 2: Interoperability with other vocabularies*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <https://www.electropedia.org>

3.1

concept

unit of knowledge created by a unique combination of characteristics

[SOURCE: ISO 1087:2019]

3.2

concept relationship

semantic connection between two or more concepts

Note 1 to entry: Concept relationships can exist within a single domain or across multiple domains.

3.3

relationship type

category of concept relationship characterized by specific semantic properties

Note 1 to entry: Examples include hierarchical, associative, and equivalence relationship types.

1) Withdrawn.

3.4

abstract relationship model

conceptual framework that defines the structure, properties, and behavior of relationships independent of specific implementation technologies

3.5

relationship assertion

statement that declares the existence of a specific relationship between two or more concepts

3.6

relationship property

characteristic or attribute of a relationship that defines its semantic behavior

Note 1 to entry: Examples include transitivity, symmetry, and reflexivity.

3.7

hierarchical relationship

concept relationship where one concept is in a superordinate position to the other concept

[SOURCE: ISO 1087:2019]

3.8

generic relationship

hierarchical relationship between two concepts where the intension of the subordinate concept includes the intension of the superordinate concept plus at least one additional delimiting characteristic

[SOURCE: ISO 1087:2019]

3.9

partitive relationship

hierarchical relationship between two concepts where the superordinate concept represents a whole and the subordinate concept represents a part of that whole

[SOURCE: ISO 1087:2019]

3.10

associative relationship

concept relationship that is neither hierarchical nor equivalence-based

Note 1 to entry: Associative relationships include spatial, temporal, causal, and functional connections between concepts.

3.11

equivalence relationship

concept relationship where two or more concepts are considered to have the same or nearly the same meaning

Note 1 to entry: Equivalence relationships can be exact, inexact, or partial, depending on the degree of conceptual overlap.

3.12

cross-domain relationship

concept relationship that connects concepts from different subject fields or domains

3.13

relationship cardinality

specification of the number of concept instances that can participate in a relationship

3.14

relationship direction

specification of whether a relationship is unidirectional, bidirectional, or non-directional

3.15

metarelationship

relationship between relationships

Note 1 to entry: Metarelationships can express dependencies, constraints, or other connections between different relationship assertions.

3.16

relationship registry

structured collection of relationship type definitions with their associated properties, constraints, and documentation

3.17

semantic interoperability

ability of different information systems to exchange and use information based on shared understanding of meaning

3.18

knowledge organization system

system for organizing knowledge that supports the description, management, and retrieval of information resources

Note 1 to entry: Examples include taxonomies, thesauri, ontologies, and classification schemes.

3.19

domain

field of special knowledge

[SOURCE: ISO 1087:2019]

3.20

relationship constraint

restriction or condition that must be satisfied by a relationship assertion

Note 1 to entry: Constraints can apply to the concepts involved, the relationship type, or the context in which the relationship is valid.

4 Abstract model for concept relationships

4.1 General

The abstract model for concept relationships provides a generic framework for defining, documenting, and managing relationships between concepts. This model is independent of specific implementation technologies and can be realized in various representation formats and systems.

The abstract model establishes a foundation for understanding concept relationships at a conceptual level, focusing on their essential structure, properties, and behavior rather than on particular notations or formalisms.

4.2 Core components

4.2.1 Relationship structure

The fundamental structure of a concept relationship consists of:

- a) Source concept(s): The concept(s) from which the relationship originates
- b) Target concept(s): The concept(s) to which the relationship points
- c) Relationship type: The semantic category of the relationship
- d) Relationship properties: Characteristics that define the relationship's behavior
- e) Relationship metadata: Information about the relationship itself

This structure can be expressed as a triple (source, relationship type, target) with associated properties and metadata, or in more complex forms for relationships involving multiple concepts.

NOTE While the basic structure resembles a triple, the abstract model does not mandate a specific representation format. The same relationship can be represented in various ways, including RDF triples, UML associations, or natural language statements.

EXAMPLE A generic relationship between “mammal” and “animal” could be expressed as:

- Source concept: mammal
- Relationship type: is-a (generic relationship)
- Target concept: animal
- Properties: transitive, asymmetric
- Metadata: asserted by taxonomic classification systems

This same relationship could be realized in different ways: * Natural language: “A mammal is a type of animal”
* UML: A generalization relationship from Mammal to Animal * RDF: <mammal> <rdfs:subClassOf> <animal> *
TBX: A hierarchical relationship with type=“generic” between the concepts

4.2.2 Relationship assertion

A relationship assertion is a statement that declares the existence of a specific relationship between two or more concepts. Each assertion includes:

- a) The concepts involved in the relationship
- b) The relationship type
- c) Any qualifiers or parameters specific to this assertion
- d) The context in which the assertion is valid
- e) The authority for the assertion

Relationship assertions can be:

- Stated (explicitly declared in a terminology resource)
- Inferred (derived from other assertions through logical rules)
- Inherited (acquired through hierarchical relationships)

4.2.3 Relationship context

Every relationship exists within a context that influences its interpretation and validity. Context elements include:

- a) Domain context: The subject field(s) in which the relationship is valid
- b) Temporal context: The time period during which the relationship holds
- c) Cultural context: Cultural factors that affect the relationship’s meaning
- d) Application context: The specific use case or application for which the relationship is relevant

NOTE The same concepts may have different relationships depending on the context. For example, “water” and “ice” might have a state-of-matter relationship in a physical science context, but a material-product relationship in a manufacturing context.

4.3 Relationship properties

4.3.1 Logical properties

Logical properties define the mathematical behavior of relationships:

- a) Transitivity: If A relates to B and B relates to C, then A relates to C
- b) Symmetry: If A relates to B, then B relates to A
- c) Reflexivity: A concept relates to itself
- d) Asymmetry: If A relates to B, then B does not relate to A
- e) Irreflexivity: A concept cannot relate to itself

These properties have important implications for relationship inference and validation.

EXAMPLE

- Transitive: If “sparrow” is-a “bird” and “bird” is-a “animal”, then “sparrow” is-a “animal”
- Symmetric: If “spouse A” is-married-to “spouse B”, then “spouse B” is-married-to “spouse A”
- Reflexive: Every concept is-identical-to itself
- Asymmetric: If “child” has-parent “parent”, then “parent” does not has-parent “child”
- Irreflexive: A concept cannot be-parent-of itself

4.3.2 Cardinality

Cardinality specifies the number of concept instances that can participate in a relationship:

- a) One-to-one: Each source concept relates to exactly one target concept
- b) One-to-many: Each source concept relates to multiple target concepts
- c) Many-to-one: Multiple source concepts relate to the same target concept
- d) Many-to-many: Multiple source concepts relate to multiple target concepts

Cardinality constraints can be further refined with minimum and maximum values.

4.3.3 Directionality

Relationships can have different directional characteristics:

- a) Unidirectional: The relationship applies in one direction only
- b) Bidirectional: The relationship applies in both directions
- c) Non-directional: The relationship has no inherent direction

NOTE Directionality is distinct from symmetry. A relationship can be bidirectional without being symmetric if the relationship type differs in each direction.

4.3.4 Strength

Relationship strength indicates the degree of connection between concepts:

- a) Mandatory: The relationship must exist for the concepts to be valid
- b) Optional: The relationship may or may not exist

- c) Weighted: The relationship has a numerical value indicating its strength or probability

4.4 Metarelationships

Metarelationships are relationships between relationships. They enable the expression of complex semantic structures and constraints:

- a) Dependency: One relationship depends on the existence of another
- b) Exclusivity: If one relationship exists, another cannot
- c) Composition: A relationship is composed of other relationships
- d) Derivation: A relationship is derived from other relationships

EXAMPLE

- Dependency: The “has-symptom” relationship between “disease” and “symptom” depends on the “affects” relationship between “disease” and “body-part”
- Exclusivity: If “concept A” is-synonym-of “concept B”, then “concept A” cannot be-antonym-of “concept B”
- Composition: The “is-grandparent-of” relationship is composed of two consecutive “is-parent-of” relationships
- Derivation: The “is-sibling-of” relationship is derived from shared “is-child-of” relationships to a common parent

4.5 Relationship lifecycle

4.5.1 Creation and documentation

Relationship creation involves:

- a) Identifying the concepts to be related
- b) Determining the appropriate relationship type
- c) Documenting the relationship properties and constraints
- d) Providing supporting evidence or rationale
- e) Recording metadata about the creation process

Documentation should include:

- a) Clear identification of the concepts involved
- b) Precise specification of the relationship type
- c) Explicit statement of properties and constraints
- d) Context information for proper interpretation
- e) Source or authority information
- f) Creation and modification timestamps

4.5.2 Validation

Relationship validation ensures that relationships are semantically correct and consistent. Validation checks include:

- a) Type compatibility: Verifying that the concepts are valid for the relationship type
- b) Property compliance: Ensuring the relationship satisfies its declared properties

- c) Constraint satisfaction: Confirming that all constraints are met
- d) Consistency: Checking for conflicts with existing relationships
- e) Context validity: Verifying the relationship is valid in its stated context

4.5.3 Evolution and maintenance

Relationships evolve over time due to:

- a) Changes in concept definitions
- b) Advances in domain knowledge
- c) Refinement of relationship types
- d) Expansion to new contexts
- e) Correction of errors or inconsistencies

Effective maintenance requires:

- a) Version control for relationship assertions
- b) Change tracking and history
- c) Impact analysis for relationship changes
- d) Propagation of changes to dependent relationships
- e) Regular review and validation

4.6 Abstract model implementation

The abstract model can be implemented in various ways, depending on the specific requirements and constraints of the application context. Implementation considerations include:

- a) Representation format: How relationships are encoded (e.g., RDF, UML, natural language)
- b) Storage mechanism: How relationships are stored and retrieved
- c) Inference capabilities: How implicit relationships are derived from explicit ones
- d) Validation mechanisms: How relationships are checked for correctness
- e) User interface: How relationships are presented to and manipulated by users

NOTE The abstract model intentionally separates the conceptual understanding of relationships from their implementation details. This separation allows for flexibility in how relationships are realized in specific systems while maintaining semantic consistency.

5 Classification of relationship types

5.1 General

Relationship types provide a semantic categorization of the connections between concepts. A well-defined classification of relationship types is essential for ensuring semantic precision, facilitating knowledge organization, and enabling effective information retrieval.

This section presents a classification system for relationship types based on their semantic properties. The classification is designed to be comprehensive enough to cover the major categories of relationships while remaining abstract enough to be applicable across domains and implementation contexts.

NOTE The classification presented here is not intended to be exhaustive but rather to provide a foundation that can be extended and refined for specific domains and applications. The focus is on establishing clear semantic distinctions between major relationship categories.

5.2 Hierarchical relationships

5.2.1 General characteristics

Hierarchical relationships establish a superordinate-subordinate structure between concepts. They are characterized by:

- a) Asymmetry: If A is hierarchically related to B, B is not hierarchically related to A in the same way
- b) Transitivity: If A is hierarchically related to B and B is hierarchically related to C, then A is hierarchically related to C
- c) Irreflexivity: A concept cannot be hierarchically related to itself

Hierarchical relationships are fundamental to organizing concepts into structured systems and supporting inheritance of characteristics.

5.2.2 Generic relationships

Generic relationships (also called “is-a” or “type-of” relationships) connect a more specific concept (species) to a more general concept (genus). The subordinate concept inherits all characteristics of the superordinate concept and adds at least one delimiting characteristic.

EXAMPLE

- “Oak” has a generic relationship to “tree”
- “Laptop” has a generic relationship to “computer”
- “Pneumonia” has a generic relationship to “respiratory disease”

In each case, the subordinate concept (oak, laptop, pneumonia) inherits all the characteristics of the superordinate concept (tree, computer, respiratory disease) and adds at least one delimiting characteristic.

Generic relationships support:

- a) Classification of concepts into categories
- b) Inheritance of characteristics from superordinate to subordinate concepts
- c) Reasoning from specific to general and general to specific
- d) Organization of concepts into taxonomies

5.2.3 Partitive relationships

Partitive relationships (also called “part-of” or “has-part” relationships) connect a part to its whole. The superordinate concept represents the whole, while the subordinate concept represents a component or constituent.

EXAMPLE

- “Wheel” has a partitive relationship to “car”
- “Chapter” has a partitive relationship to “book”
- “Neuron” has a partitive relationship to “nervous system”

In each case, the subordinate concept (wheel, chapter, neuron) is a component or constituent of the superordinate concept (car, book, nervous system).

Partitive relationships can be further classified based on the nature of the part-whole connection:

- a) Component-object: Physical parts of objects (e.g., wheel-car)
- b) Member-collection: Elements of collections (e.g., tree-forest)
- c) Portion-mass: Portions of masses (e.g., slice-pie)
- d) Material-object: Substances of objects (e.g., steel-car)
- e) Feature-activity: Phases of activities (e.g., payment-purchase)
- f) Place-area: Areas of larger areas (e.g., city-country)

5.2.4 Instance relationships

Instance relationships connect an individual instance to its class or category. Unlike generic relationships, which connect classes to more general classes, instance relationships connect specific individuals to their classes.

EXAMPLE

- “Eiffel Tower” has an instance relationship to “tower”
- “Albert Einstein” has an instance relationship to “physicist”
- “Amazon River” has an instance relationship to “river”

In each case, the subordinate concept represents a specific individual that is an instance of the superordinate concept.

Instance relationships are particularly important in:

- a) Knowledge representation systems that distinguish between classes and instances
- b) Proper name handling in terminology resources
- c) Reference to specific entities in domain models

5.3 Associative relationships

5.3.1 General characteristics

Associative relationships connect concepts that are semantically related but not in a hierarchical or equivalence manner. They are characterized by:

- a) Semantic relevance: The concepts are meaningfully related
- b) Non-hierarchical nature: Neither concept is superordinate to the other
- c) Domain-specificity: The relationship types often reflect domain-specific connections

Associative relationships capture a wide range of semantic connections and are essential for expressing the rich network of relationships between concepts in a domain.

NOTE Associative relationships are often less formally defined than hierarchical relationships and may have different properties depending on the specific relationship type. Some may be symmetric, others asymmetric; some may be transitive, others non-transitive.

5.3.2 Causal relationships

Causal relationships connect concepts where one represents a cause and the other an effect. Subtypes include:

- a) Cause-effect: Direct causal connection (e.g., virus-infection)
- b) Process-result: Outcome of a process (e.g., combustion-ash)
- c) Agent-action: Performer of an action (e.g., teacher-teaching)
- d) Action-recipient: Target of an action (e.g., teaching-student)

EXAMPLE In healthcare terminology:

- “Smoking” has a causal relationship to “lung cancer”
- “Virus” has a causal relationship to “infection”
- “Bacteria” has a causal relationship to “inflammation”

These relationships express that one concept (smoking, virus, bacteria) causes or contributes to the other (lung cancer, infection, inflammation).

5.3.3 Temporal relationships

Temporal relationships connect concepts based on their time sequence. Subtypes include:

- a) Precedence: One concept precedes another (e.g., pregnancy-childbirth)
- b) Succession: One concept follows another (e.g., treatment-recovery)
- c) Simultaneity: Concepts occur at the same time (e.g., fever-chills)
- d) Duration: One concept represents the timespan of another (e.g., childhood-development)

EXAMPLE In project management terminology:

- “Planning” has a temporal relationship (precedes) to “execution”
- “Testing” has a temporal relationship (precedes) to “deployment”
- “Maintenance” has a temporal relationship (follows) to “implementation”

These relationships express the sequence in which the activities occur.

5.3.4 Spatial relationships

Spatial relationships connect concepts based on their physical or conceptual location. Subtypes include:

- a) Containment: One concept contains another (e.g., building-room)
- b) Adjacency: Concepts are next to each other (e.g., neighboring countries)
- c) Orientation: Concepts have a directional relationship (e.g., north-south)
- d) Distance: Concepts are separated by space (e.g., near-far)

5.3.5 Functional relationships

Functional relationships connect concepts based on their purpose, role, or function. Subtypes include:

- a) Tool-function: Object and its purpose (e.g., hammer-pounding)
- b) Process-instrument: Activity and its tool (e.g., writing-pen)

- c) Object-application: Item and its use (e.g., medicine-treatment)
- d) Role-activity: Position and its responsibilities (e.g., doctor-diagnosis)

EXAMPLE In information technology terminology:

- “Firewall” has a functional relationship to “network protection”
- “Database” has a functional relationship to “data storage”
- “Algorithm” has a functional relationship to “problem solving”

These relationships express the purpose or function of each concept.

5.3.6 Developmental relationships

Developmental relationships connect concepts representing different stages or states of development. Subtypes include:

- a) Source-product: Raw material and result (e.g., flour-bread)
- b) Immature-mature: Developmental stages (e.g., larva-butterfly)
- c) Natural-processed: Natural and processed forms (e.g., cotton-fabric)

5.3.7 Possessive relationships

Possessive relationships connect concepts where one possesses or owns the other. Subtypes include:

- a) Owner-possession: Legal ownership (e.g., author-copyright)
- b) Container-contained: Physical containment (e.g., bottle-water)
- c) Attribute-entity: Characteristic and its bearer (e.g., color-object)

5.4 Equivalence relationships

5.4.1 General characteristics

Equivalence relationships connect concepts that have the same or similar meaning. They are characterized by:

- a) Semantic similarity: The concepts share significant meaning
- b) Substitutability: One concept can replace another in certain contexts
- c) Varying degrees of equivalence: From exact to partial

Equivalence relationships are crucial for terminology harmonization, cross-domain mapping, and information retrieval.

5.4.2 Exact equivalence

Exact equivalence connects concepts that are identical in meaning and can be substituted for each other in all contexts. Characteristics include:

- a) Complete conceptual overlap
- b) Same essential and delimiting characteristics
- c) Interchangeability in all contexts
- d) Symmetry: If A is exactly equivalent to B, B is exactly equivalent to A

EXAMPLE

- “Heart attack” and “myocardial infarction” in medical terminology
- “Automobile” and “car” in general language
- “H₂O” and “water” in chemistry

These pairs represent the same concept and can be used interchangeably in their respective domains.

NOTE True exact equivalence is rare outside of synonymy within a single language. Even apparent synonyms often have subtle differences in connotation, register, or usage context.

5.4.3 Inexact equivalence

Inexact equivalence connects concepts that are similar in meaning but not identical. Characteristics include:

- Significant conceptual overlap
- Minor differences in characteristics or scope
- Limited interchangeability (context-dependent)
- Symmetry: If A is inexactly equivalent to B, B is inexactly equivalent to A

EXAMPLE

- “Happiness” and “joy” (similar emotions with different intensities)
- “Forest” and “woods” (similar but with potential differences in size and density)
- “Efficiency” and “effectiveness” (related but distinct performance measures)

These pairs represent similar but not identical concepts, with subtle differences in meaning or application.

5.4.4 Partial equivalence

Partial equivalence connects concepts where one includes the other but also covers additional meaning. Characteristics include:

- Partial conceptual overlap
- One concept has broader or narrower scope than the other
- Limited interchangeability (direction-dependent)
- Asymmetry: If A is partially equivalent to B, the relationship from B to A is different

EXAMPLE

- “Furniture” and “chair” (chair is a type of furniture, but furniture includes more than chairs)
- “Disease” and “infection” (all infections are diseases, but not all diseases are infections)
- “Vehicle” and “car” (all cars are vehicles, but not all vehicles are cars)

In each case, one concept (furniture, disease, vehicle) is broader than the other (chair, infection, car).

5.4.5 Cross-language equivalence

Cross-language equivalence connects concepts expressed in different languages. This type of equivalence is particularly important for translation and multilingual terminology management. Characteristics include:

- Consideration of linguistic and cultural differences
- Varying degrees of equivalence (from exact to zero)

- c) Context-dependent validity
- d) Potential asymmetry due to linguistic and cultural factors

EXAMPLE

- English “privacy” and French “vie privée” (partial equivalence due to cultural and legal differences)
- English “computer” and German “Computer” (near-exact equivalence)
- Japanese “wabi-sabi” and English (no direct equivalent — zero equivalence)

These examples illustrate different degrees of cross-language equivalence, from near-exact to zero.

5.5 Relationship type registry

5.5.1 Purpose and structure

A relationship type registry provides a structured collection of relationship type definitions with their associated properties, constraints, and documentation. The registry serves as a reference for consistent application of relationship types across terminology resources.

A comprehensive relationship type registry should include:

- a) Unique identifier for each relationship type
- b) Name and definition of the relationship type
- c) Category within the classification system
- d) Logical properties (transitivity, symmetry, etc.)
- e) Domain and range constraints
- f) Inverse relationship (if applicable)
- g) Usage examples and notes
- h) References to standards or authorities

5.5.2 Extension mechanisms

The relationship type classification can be extended to meet the needs of specific domains and applications through:

- a) Specialization: Creating subtypes of existing relationship types
- b) Composition: Combining relationship types to create complex relationships
- c) Contextualization: Adapting relationship types for specific contexts
- d) Custom types: Defining domain-specific relationship types

Extension mechanisms should maintain compatibility with the core classification while allowing for the expression of domain-specific semantics.

NOTE When extending the relationship type classification, it is important to clearly document the relationship between custom types and the core classification to facilitate interoperability and understanding across systems.

6 Domain contexts

6.1 General

Concept relationships do not exist in isolation but are embedded within specific domain contexts that influence their meaning, validity, and application. This section addresses the management of concept relationships within and across domain boundaries, focusing on the challenges and approaches for establishing semantic connections in different contextual environments.

The domain context of a relationship affects its interpretation, properties, and constraints. Understanding and properly documenting these contextual factors is essential for ensuring semantic precision and facilitating knowledge exchange across domains.

6.2 Same-domain relationships

6.2.1 Characteristics of same-domain relationships

Same-domain relationships connect concepts within a single subject field or domain of knowledge. They are characterized by:

- a) Shared conceptual framework: Concepts exist within the same system of knowledge
- b) Common terminology: Concepts are typically expressed using domain-specific terms
- c) Established relationship patterns: The domain often has conventional ways of relating concepts
- d) Domain-specific relationship types: Specialized relationship types may exist for the domain
- e) Consistent interpretation: Relationships are understood similarly by domain experts

Same-domain relationships benefit from the shared understanding and conventions of the domain, making them generally easier to establish and validate than cross-domain relationships.

NOTE Even within a single domain, sub-domains or schools of thought may have different conceptual frameworks that affect how relationships are understood and applied. These differences should be documented as part of the relationship context.

6.2.2 Domain-specific relationship patterns

Each domain tends to develop characteristic patterns of relationships that reflect its conceptual structure and knowledge organization. Examples include:

- a) Healthcare: Emphasis on causal relationships (disease-symptom, treatment-outcome) and hierarchical classifications (taxonomies of diseases, procedures)
- b) Engineering: Focus on functional relationships (component-function, material-property) and structural relationships (system-component)
- c) Law: Prominence of authority relationships (statute-regulation), temporal relationships (precedent-subsequent ruling), and definitional relationships (term-legal definition)
- d) Chemistry: Structured around compositional relationships (compound-element) and transformational relationships (reactant-product)

EXAMPLE In the domain of medicine, the relationship between “bacteria” and “infection” is typically causal, while the relationship between “antibiotic” and “bacteria” is functional (treatment-target). These relationship patterns are characteristic of medical knowledge organization and would be understood by domain experts without extensive explanation.

6.2.3 Domain-specific relationship types

Domains often develop specialized relationship types that capture domain-specific connections between concepts. These relationship types extend the general classification presented in Section 5 with domain-specific semantics.

EXAMPLE In pharmacology, specialized relationship types might include:

- “has-mechanism-of-action”: Connects a drug to its biochemical mechanism
- “has-contraindication”: Links a medication to conditions where it should not be used
- “has-adverse-effect”: Associates a drug with potential negative outcomes

These relationship types have specific meaning within pharmacology and may not be directly applicable to other domains.

When documenting domain-specific relationship types, it is important to:

- a) Provide clear definitions that capture the domain-specific semantics
- b) Map to the general relationship classification where possible
- c) Document any special properties or constraints
- d) Include domain-specific usage examples
- e) Reference authoritative domain sources

6.3 Cross-domain relationships

6.3.1 Challenges of cross-domain relationships

Cross-domain relationships connect concepts from different subject fields or domains of knowledge. They present several challenges:

- a) Conceptual framework differences: Domains may have different ways of organizing knowledge
- b) Terminological variations: The same term may have different meanings across domains
- c) Relationship interpretation differences: The same relationship type may be understood differently
- d) Validation complexity: Expertise from multiple domains is needed for validation
- e) Documentation requirements: More extensive context information is needed

Despite these challenges, cross-domain relationships are essential for interdisciplinary work, knowledge integration, and comprehensive information retrieval.

NOTE Cross-domain relationships often reveal interesting connections and insights that might not be apparent within a single domain. They can stimulate innovation and new perspectives by bridging different knowledge areas.

6.3.2 Approaches for establishing cross-domain relationships

Several approaches can be used to establish meaningful cross-domain relationships:

- a) Interface concepts: Identifying concepts that naturally span domain boundaries
- b) Bridging vocabularies: Creating intermediate vocabularies that connect domain terminologies
- c) Upper ontologies: Using high-level conceptual frameworks that transcend specific domains
- d) Mapping patterns: Developing standard patterns for relating concepts across domains

e) Expert collaboration: Bringing together experts from different domains to establish relationships

EXAMPLE The concept “temperature” exists in multiple domains including meteorology, medicine, and physics. It can serve as an interface concept for establishing cross-domain relationships:

- In meteorology: “temperature” has-effect-on “weather patterns”
- In medicine: “body temperature” indicates “health status”
- In physics: “temperature” is-measure-of “molecular kinetic energy”

By using “temperature” as an interface concept, relationships can be established between these domains, such as how “weather patterns” affect “health status.”

6.3.3 Cross-domain relationship documentation

Cross-domain relationships require more extensive documentation than same-domain relationships to ensure proper interpretation. Documentation should include:

- a) Clear identification of the source and target domains
- b) Domain-specific definitions of the concepts involved
- c) Explanation of how the relationship bridges the domains
- d) Any constraints or conditions on the cross-domain application
- e) Validation method and authority
- f) Potential interpretation issues or ambiguities

EXAMPLE A cross-domain relationship between the medical concept “inflammation” and the economic concept “inflation” might be documented as:

- Source concept: “inflammation” (Medicine: A localized physical condition in which part of the body becomes reddened, swollen, hot, and often painful)
- Target concept: “inflation” (Economics: A general increase in prices and fall in the purchasing value of money)
- Relationship type: metaphorical-similarity
- Explanation: Both concepts involve expansion or increase (physical tissues vs. prices) and are often described as harmful when excessive
- Constraints: This is an analogical relationship for explanatory purposes, not a causal or definitional relationship
- Validation: Linguistic analysis of economic literature showing the metaphorical use of medical terminology

6.4 Context-sensitive relationship semantics

6.4.1 Contextual variation in relationship meaning

The same relationship type may have different semantic nuances depending on the context. These variations should be documented to ensure proper interpretation.

EXAMPLE The “part-of” relationship has different implications in different contexts:

- Physical objects: A wheel is part-of a car (physical component that can be separated)
- Organizations: A department is part-of a company (administrative division with functional independence)
- Processes: Payment is part-of a purchase transaction (temporal phase that cannot exist independently)

— Abstract entities: A chapter is part-of a book (content division that has meaning in relation to the whole)

While all these are partitive relationships, their specific semantics vary with the nature of the related concepts.

6.4.2 Context parameters

To fully specify the context of a relationship, several parameters should be documented:

- a) Domain context: The subject field(s) in which the relationship is valid
- b) Temporal context: Time period during which the relationship holds
- c) Cultural context: Cultural factors affecting interpretation
- d) Application context: Specific use case or purpose
- e) Authority context: Source of the relationship assertion
- f) Certainty context: Degree of confidence in the relationship

These parameters help users understand the conditions under which the relationship is valid and applicable.

NOTE Context parameters should be documented at an appropriate level of detail for the intended application. Not all parameters are equally relevant for all relationships, and the documentation should focus on those that significantly affect interpretation.

6.4.3 Context-based relationship constraints

Relationship constraints may vary based on context. These context-dependent constraints should be explicitly documented:

- a) Domain-specific cardinality constraints
- b) Context-dependent property variations
- c) Conditional validity rules
- d) Contextual relationship strength

EXAMPLE The relationship “treats” between “medication” and “disease” might have different constraints in different contexts:

- Clinical practice context: Many-to-many cardinality (multiple medications may treat multiple diseases)
- Regulatory approval context: One-to-many cardinality (a medication is approved for specific diseases only)
- Research context: Probabilistic strength (degree of evidence for efficacy)
- Historical context: Temporal validity (treatments considered effective change over time)

6.5 Managing domain boundaries

6.5.1 Domain boundary identification

Clearly identifying domain boundaries is essential for proper relationship management. Approaches include:

- a) Subject field classification: Using established classification systems to delineate domains
- b) Concept clustering: Identifying natural groupings of closely related concepts
- c) Terminology analysis: Examining term usage patterns to identify domain vocabularies
- d) Expert consensus: Relying on expert judgment to define domain boundaries

- e) Hybrid approaches: Combining multiple methods for more robust boundary identification

Domain boundaries should be documented in a way that supports relationship management while recognizing that boundaries may be fluid and overlapping.

6.5.2 Boundary-spanning concepts

Some concepts naturally span domain boundaries and can serve as anchors for cross-domain relationships. Characteristics of boundary-spanning concepts include:

- a) Relevance to multiple domains
- b) Consistent core meaning across domains
- c) Recognition by experts from different fields
- d) Frequent appearance in interdisciplinary contexts
- e) Potential for different specializations in each domain

EXAMPLE The concept “risk” spans multiple domains including finance, healthcare, engineering, and environmental science. While its core meaning of “potential for unwanted outcome” remains consistent, each domain has developed specialized aspects:

- Finance: Quantified as volatility or probability of loss
- Healthcare: Associated with factors that increase disease likelihood
- Engineering: Analyzed through failure modes and effects
- Environmental science: Assessed through impact and probability matrices

“Risk” can serve as a boundary-spanning concept to establish relationships between these domains, such as how financial risks relate to environmental risks.

6.5.3 Domain relationship mapping

Systematic mapping of relationships between domains can facilitate cross-domain knowledge integration. Approaches include:

- a) Domain interface analysis: Identifying areas where domains naturally overlap
- b) Relationship pattern mapping: Documenting how relationship types translate across domains
- c) Concept alignment: Establishing correspondences between domain-specific concepts
- d) Relationship transformation rules: Defining how relationships change when crossing domain boundaries

Domain relationship mapping provides a foundation for more sophisticated cross-domain knowledge integration and semantic interoperability.

NOTE Domain relationship mapping is not merely about connecting individual concepts but about understanding how entire conceptual structures relate across domains. This higher-level mapping can reveal structural similarities and differences that affect how relationships should be interpreted.

7 Implementation in terminology systems

7.1 General

The abstract model for concept relationships described in this document can be implemented in various terminology systems and knowledge organization frameworks. This section provides guidance on

implementing the model in different contexts, focusing on practical considerations while maintaining the abstract nature of the underlying relationships.

Implementation approaches should balance semantic precision with practical usability, ensuring that the relationship model serves the needs of its intended users while maintaining conceptual integrity.

NOTE Implementation details will necessarily vary based on the specific requirements, constraints, and technologies of each system. The guidance provided here focuses on general principles and approaches rather than prescribing specific technical solutions.

7.2 Implementation in reference works

7.2.1 Dictionaries and glossaries

Dictionaries and glossaries typically implement concept relationships implicitly through their definitional structure and cross-references. Implementation considerations include:

- a) Definitional relationships: Using definition patterns that consistently express hierarchical and associative relationships
- b) Cross-reference systems: Implementing explicit references to related entries
- c) Usage examples: Illustrating relationships through contextual examples
- d) Semantic indicators: Using typographical or symbolic markers to indicate relationship types
- e) Entry structure: Organizing entry components to reflect conceptual relationships

EXAMPLE Dictionary entry implementing concept relationships:

photosynthesis /ˌfɒtəʊˈsɪnθəˌsɪs/ *n.* The process by which green plants and some other organisms use sunlight to synthesize nutrients from carbon dioxide and water. → See also **chlorophyll**, **respiration**.

This entry implements: * Generic relationship through the definition pattern “The process by which...” (photosynthesis is-a process) * Functional relationship through “use sunlight to synthesize...” (sunlight has-function-in photosynthesis) * Associative relationships through cross-references to related terms

7.2.2 Thesauri

Thesauri implement concept relationships more explicitly, typically using standardized relationship indicators. Implementation considerations include:

- a) Standard relationship markers: Using consistent abbreviations or symbols for relationship types (BT, NT, RT, UF, etc.)
- b) Hierarchical organization: Implementing generic and partitive relationships through broader/narrower term structures
- c) Equivalence handling: Managing synonyms and near-synonyms through use/used-for relationships
- d) Associative relationship specificity: Determining the appropriate level of detail for associative relationships
- e) Scope notes: Providing context information to clarify relationship meaning

EXAMPLE Thesaurus entry implementing concept relationships:

Renewable energy BT: Energy sources NT: Solar energy Wind energy Hydroelectric energy Geothermal energy Biomass energy RT: Sustainability Energy conservation Carbon footprint UF: Green energy Clean energy SN: Energy from sources that are naturally replenishing and virtually inexhaustible

This entry implements: * Generic relationships through broader term (BT) and narrower term (NT) * Associative relationships through related term (RT) * Equivalence relationships through used for (UF) * Context information through scope note (SN)

7.3 Implementation in knowledge organization systems

7.3.1 Taxonomies

Taxonomies primarily implement hierarchical relationships in a structured classification system. Implementation considerations include:

- Hierarchical structure: Implementing consistent parent-child relationships
- Relationship type specification: Distinguishing between generic, partitive, and instance hierarchies
- Sibling relationships: Managing concepts at the same hierarchical level
- Polyhierarchy: Handling concepts that belong to multiple hierarchical paths
- Faceted classification: Implementing multiple classification dimensions

EXAMPLE Taxonomy implementation of hierarchical relationships:

```

` Living organisms |--- Animals | |--- Vertebrates | |--- Mammals | |--- Primates | |--- ...
Humans | |--- ... | |--- Birds | |--- ... |--- Invertebrates |--- Plants |--- Flowering plants |--- ...

```

This taxonomy implements generic relationships (is-a) between classes of organisms in a hierarchical structure.

7.3.2 Ontologies

Ontologies implement comprehensive relationship models with formal semantics. Implementation considerations include:

- Formal relationship definitions: Implementing relationships with precise logical properties
- Relationship constraints: Encoding domain, range, and cardinality constraints
- Inference rules: Implementing rules for deriving implicit relationships
- Relationship hierarchies: Organizing relationship types into their own hierarchical structure
- Metarelationships: Implementing relationships between relationships

EXAMPLE Ontology implementation of relationships in OWL (Web Ontology Language):

```

` <!-- Define a relationship type --> <owl:ObjectProperty rdf:about="#treats">
<rdfs:domain rdf:resource="#Medication"/> <rdfs:range rdf:resource="#Disease"/> <rdf:type
rdf:resource="#owl:AsymmetricProperty"/> <rdfs:subPropertyOf rdf:resource="#hasTherapeuticEffect"/>
</owl:ObjectProperty>

```

```

<!-- Use the relationship in concept definitions --> <owl:Class rdf:about="#Antibiotic"> <rdfs:subClassOf
rdf:resource="#Medication"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#treats"/>
> <owl:someValuesFrom rdf:resource="#BacterialInfection"/> </owl:Restriction> </rdfs:subClassOf>
</owl:Class> `

```

This example implements: * A functional relationship type "treats" with domain and range constraints * Logical property (asymmetric) * Relationship hierarchy (subPropertyOf) * Relationship use in concept definition (Antibiotic treats BacterialInfection)

7.3.3 Semantic networks

Semantic networks implement concept relationships as a graph of interconnected nodes. Implementation considerations include:

- a) Node-link structure: Implementing concepts as nodes and relationships as links
- b) Relationship labeling: Providing clear labels for relationship types
- c) Directionality: Implementing directional or bidirectional links as appropriate
- d) Weighting: Adding weight or strength indicators to relationships
- e) Clustering: Organizing the network to reveal conceptual structures

7.4 Implementation in terminology databases

7.4.1 TBX implementation

TermBase eXchange (TBX) as specified in ISO 30042:2019 provides a standard format for implementing concept relationships in terminology databases. Implementation considerations include:

- a) Concept-oriented structure: Implementing the concept-based approach through termEntry elements
- b) Relationship categories: Using appropriate data categories for relationship types
- c) Cross-references: Implementing concept relationships through cross-references
- d) Relationship properties: Documenting properties using appropriate data categories
- e) Data category selection: Choosing appropriate data categories from standard registries

EXAMPLE TBX implementation of concept relationships:

```
<?xml <termEntry id="ID67"> <descrip type="subjectField">Medicine</descrip> <langSet xml:lang="en">
<tig> <term>diabetes mellitus</term> </tig> <descrip type="definition">Metabolic disease characterized
by high blood glucose levels.</descrip> </langSet> <xref type="broaderConcept" target="ID42">metabolic
disease</xref> <xref type="relatedConcept" target="ID89">insulin</xref> <xref type="relatedConcept"
target="ID93">hyperglycemia</xref> </termEntry>
```

This example implements: * Generic relationship through broaderConcept cross-reference * Associative relationships through relatedConcept cross-references

7.4.2 Relational database implementation

Relational databases can implement concept relationships through various table structures. Implementation considerations include:

- a) Concept-relationship-concept structure: Implementing relationships as associations between concept records
- b) Relationship type tables: Creating tables of standardized relationship types
- c) Relationship properties: Storing properties as attributes of relationship records
- d) Context information: Including contextual parameters in relationship records
- e) Integrity constraints: Implementing validation rules as database constraints

EXAMPLE Relational database schema for concept relationships:

```
<?sql CREATE TABLE Concepts ( ConceptID INT PRIMARY KEY, DomainID INT, FOREIGN KEY (DomainID)
REFERENCES Domains(DomainID) );
```

```
CREATE TABLE RelationshipTypes ( RelationshipTypeID INT PRIMARY KEY, Name VARCHAR(100), Category VARCHAR(50), IsTransitive BOOLEAN, IsSymmetric BOOLEAN );
```

```
CREATE TABLE ConceptRelationships ( SourceConceptID INT, TargetConceptID INT, RelationshipTypeID INT, ContextID INT, Strength FLOAT, ValidFrom DATE, ValidTo DATE, PRIMARY KEY (SourceConceptID, TargetConceptID, RelationshipTypeID, ContextID), FOREIGN KEY (SourceConceptID) REFERENCES Concepts(ConceptID), FOREIGN KEY (TargetConceptID) REFERENCES Concepts(ConceptID), FOREIGN KEY (RelationshipTypeID) REFERENCES RelationshipTypes(RelationshipTypeID), FOREIGN KEY (ContextID) REFERENCES Contexts(ContextID) );`
```

This schema implements: * Concepts as entities in their own table * Relationship types with their properties * Relationships as associations between concepts with type, context, and temporal validity

7.5 User interface considerations

7.5.1 Relationship visualization

User interfaces for concept relationships should provide clear visual representations. Considerations include:

- a) Graphical representations: Using diagrams, graphs, or networks to visualize relationships
- b) Hierarchical displays: Implementing tree or indented list views for hierarchical relationships
- c) Relationship indicators: Using consistent visual cues for different relationship types
- d) Interactive exploration: Allowing users to navigate relationship networks
- e) Filtering and focus: Enabling users to focus on specific relationship types or concepts

7.5.2 Relationship editing

Interfaces for creating and editing relationships should support accurate and efficient relationship management. Considerations include:

- a) Guided relationship creation: Providing templates or wizards for common relationship patterns
- b) Validation feedback: Alerting users to potential issues or inconsistencies
- c) Relationship type selection: Offering organized lists of available relationship types
- d) Context specification: Enabling documentation of contextual parameters
- e) Batch operations: Supporting efficient editing of multiple relationships

7.5.3 Search and retrieval

Search interfaces should leverage relationship information to enhance information retrieval. Considerations include:

- a) Relationship-based queries: Allowing users to search based on relationship types and properties
- b) Semantic expansion: Using relationships to expand search results to related concepts
- c) Faceted search: Implementing relationship-based facets for result filtering
- d) Relevance ranking: Incorporating relationship strength in result ordering
- e) Query visualization: Showing how search results relate to query terms

7.6 Implementation challenges and solutions

7.6.1 Scalability

Large terminology resources may contain millions of concepts and relationships, presenting scalability challenges. Solutions include:

- a) Efficient data structures: Optimizing storage for relationship representation
- b) Indexing strategies: Creating indexes for common relationship queries
- c) Caching mechanisms: Caching frequently accessed relationship patterns
- d) Distributed processing: Implementing distributed approaches for relationship computation
- e) On-demand loading: Loading relationship data only when needed

7.6.2 Relationship maintenance

Maintaining relationship consistency over time presents challenges as concepts and domains evolve. Solutions include:

- a) Change impact analysis: Assessing how concept changes affect relationships
- b) Relationship versioning: Maintaining historical versions of relationships
- c) Automated validation: Implementing regular consistency checks
- d) Deprecation processes: Managing the lifecycle of outdated relationships
- e) User feedback mechanisms: Collecting and processing user input on relationships

7.6.3 Cross-system integration

Implementing relationships across different systems presents interoperability challenges. Solutions include:

- a) Standard exchange formats: Using TBX and other standards for relationship exchange
- b) Mapping services: Implementing services to translate between relationship models
- c) Shared identifiers: Using common concept and relationship identifiers
- d) API standardization: Developing consistent APIs for relationship access
- e) Federated queries: Implementing distributed query capabilities across systems

NOTE Cross-system integration often requires compromise between the ideal abstract model and the practical constraints of existing systems. The goal should be to preserve the essential semantics of relationships while accommodating system-specific limitations.

8 Enosema's approach

8.1 General

The Enosema Foundation has developed a comprehensive approach to concept relationships that builds upon the abstract model presented in this document. This approach provides practical guidance for implementing concept relationships in terminology resources while maintaining semantic precision and interoperability.

The Enosema approach is designed to be adaptable to various domains and implementation contexts while ensuring consistency in how relationships are defined, documented, and managed.

8.2 Core principles

8.2.1 Concept-centricity

The Enosema approach places concepts at the center of terminology work, recognizing that relationships exist between concepts rather than between terms. This principle ensures that:

- a) Relationships are established based on conceptual analysis, not merely linguistic connections
- b) Relationship assertions remain valid across different linguistic expressions of the same concepts
- c) Conceptual structures are maintained independently of specific terminological representations
- d) Relationships can be effectively translated across languages and cultures

8.2.2 Relationship explicitness

Relationships should be explicitly defined and documented rather than left implicit. This principle ensures that:

- a) Relationship semantics are clearly understood by all users
- b) Relationship assertions can be validated against defined criteria
- c) Relationships can be effectively exchanged between systems
- d) Ambiguity in relationship interpretation is minimized

8.2.3 Contextual validity

Relationships exist within specific contexts that affect their validity and interpretation. This principle ensures that:

- a) Relationship assertions include appropriate contextual parameters
- b) Domain-specific relationship semantics are properly documented
- c) Relationships are not inappropriately generalized beyond their valid contexts
- d) Users can assess the applicability of relationships to their specific needs

8.2.4 Semantic precision

Relationships should be defined with sufficient semantic precision to support their intended applications. This principle ensures that:

- a) Relationship types are clearly distinguished from one another
- b) Relationship properties are explicitly documented
- c) Relationship assertions can be validated against formal criteria
- d) Relationships support accurate inference and reasoning

8.2.5 Practical usability

Relationship models should balance theoretical rigor with practical usability. This principle ensures that:

- a) Relationship frameworks can be implemented in real-world systems
- b) Users can effectively work with relationships without excessive complexity
- c) Relationship documentation is accessible and understandable

- d) Implementation guidance is provided for different contexts

8.3 Enosema relationship registry

8.3.1 Registry structure

The Enosema Foundation maintains a relationship type registry that provides a standardized collection of relationship types with their associated properties, constraints, and documentation. The registry includes:

- a) Core relationship types applicable across domains
- b) Domain-specific relationship types for selected fields
- c) Mapping to standard relationship types in other systems
- d) Implementation guidance for each relationship type
- e) Examples illustrating proper usage

The registry serves as a reference for consistent application of relationship types across terminology resources and provides a foundation for semantic interoperability.

8.3.2 Core relationship types

The core relationship types in the Enosema registry include:

- a) Hierarchical relationships
 - generic (is-a, type-of)
 - partitive (part-of, has-part)
 - instance (instance-of, has-instance)
- b) Associative relationships
 - causal (causes, caused-by)
 - temporal (precedes, follows)
 - functional (has-function, function-of)
 - spatial (located-in, contains)
- c) Equivalence relationships
 - exact-equivalence (same-as)
 - inexact-equivalence (similar-to)
 - partial-equivalence (broader-than, narrower-than)

Each relationship type is defined with:

- a) Unique identifier
- b) Preferred name and synonyms
- c) Definition and scope
- d) Logical properties (transitivity, symmetry, etc.)
- e) Inverse relationship (if applicable)

- f) Usage notes and examples
- g) Implementation guidance

8.3.3 Extension mechanisms

The Enosema registry provides mechanisms for extending the core relationship types to meet specific domain needs:

- a) Specialization: Creating subtypes of core relationship types
- b) Composition: Combining relationship types to create complex relationships
- c) Contextualization: Adapting relationship types for specific contexts
- d) Custom types: Defining domain-specific relationship types

Extensions should maintain compatibility with the core registry while addressing domain-specific requirements.

EXAMPLE Extension of a core relationship type for a specific domain:

Core type: “causes” (causal relationship) Domain: Pharmacology Extension: “has-side-effect”

Definition: A specialized causal relationship between a medication and an unintended physiological response.

Properties: * Inherits asymmetry from “causes” * Domain-specific cardinality: one-to-many (one medication can have many side effects) * Domain-specific constraint: side effects must be unintended effects

This extension maintains compatibility with the core “causes” type while adding domain-specific semantics and constraints.

8.4 Documentation guidelines

8.4.1 Relationship assertion documentation

The Enosema approach includes guidelines for documenting relationship assertions:

- a) Source and target concepts should be clearly identified with unique identifiers
- b) Relationship type should be specified using registry identifiers
- c) Relationship properties should be explicitly stated
- d) Context parameters should be documented as appropriate
- e) Authority information should be included
- f) Creation and modification metadata should be recorded

EXAMPLE Documented relationship assertion:

- Source concept: C0018681 (Headache)
- Target concept: C0038454 (Stress, Psychological)
- Relationship type: RT-023 (may-be-caused-by)
- Properties: non-transitive, asymmetric
- Context: Medical domain, general clinical knowledge
- Strength: probable (not definitive)
- Authority: Medical Subject Headings (MeSH)

- Created: 2024-03-15
- Last validated: 2025-01-10

8.4.2 Context documentation

Context documentation should include:

- a) Domain context: Subject field(s) in which the relationship is valid
- b) Temporal context: Time period during which the relationship holds
- c) Cultural context: Cultural factors affecting interpretation
- d) Application context: Specific use case or purpose
- e) Authority context: Source of the relationship assertion
- f) Certainty context: Degree of confidence in the relationship

The level of detail for context documentation should be appropriate for the intended application of the relationship.

8.4.3 Relationship validation documentation

Validation documentation should include:

- a) Validation method: How the relationship was validated
- b) Validation criteria: What standards or rules were applied
- c) Validation authority: Who performed or approved the validation
- d) Validation date: When the validation was performed
- e) Validation outcome: Results of the validation process
- f) Validation limitations: Any constraints or caveats

8.5 Quality assurance

8.5.1 Validation processes

The Enosema approach includes processes for validating relationship assertions:

- a) Formal validation: Checking compliance with logical properties and constraints
- b) Expert validation: Review by domain experts
- c) Consistency validation: Checking for conflicts with existing relationships
- d) Contextual validation: Verifying validity within stated contexts
- e) User validation: Gathering feedback from terminology users

Validation processes should be applied at appropriate stages in the relationship lifecycle, including creation, modification, and periodic review.

8.5.2 Quality criteria

Quality criteria for relationship assertions include:

- a) Accuracy: Correctness of the relationship assertion

- b) Precision: Specificity of the relationship type
- c) Completeness: Inclusion of all required information
- d) Consistency: Compatibility with other relationship assertions
- e) Clarity: Understandability of the relationship documentation
- f) Traceability: Ability to track the source and history of the assertion

These criteria should be applied during validation processes and used to guide continuous improvement of relationship resources.

8.5.3 Governance framework

The Enosema governance framework for relationships includes:

- a) Roles and responsibilities for relationship management
- b) Decision-making processes for relationship issues
- c) Change management procedures
- d) Conflict resolution mechanisms
- e) Review and approval workflows
- f) Feedback processing procedures

The governance framework ensures that relationships are managed consistently and that changes are properly controlled and documented.

8.6 Implementation support

8.6.1 Implementation patterns

The Enosema approach includes implementation patterns for different contexts:

- a) Reference work patterns: How to implement relationships in dictionaries, glossaries, and thesauri
- b) Knowledge organization patterns: How to implement relationships in taxonomies, ontologies, and semantic networks
- c) Database patterns: How to implement relationships in terminology databases and other structured repositories
- d) Exchange patterns: How to represent relationships in exchange formats

These patterns provide practical guidance while maintaining compatibility with the abstract model.

8.6.2 Conformance levels

The Enosema approach defines conformance levels for relationship implementation:

- a) Level 1: Basic relationship support with minimal documentation
- b) Level 2: Structured relationship types with standard properties
- c) Level 3: Comprehensive relationship model with context documentation
- d) Level 4: Full implementation of the abstract model with all features

These levels allow systems to implement relationships at an appropriate level of sophistication for their needs while providing a path for future enhancement.

8.6.3 Migration strategies

For existing terminology resources, the Enosema approach provides migration strategies:

- a) **Assessment:** Evaluating current relationship implementations
- b) **Mapping:** Relating existing relationships to the abstract model
- c) **Enrichment:** Adding missing relationship information
- d) **Validation:** Checking migrated relationships against quality criteria
- e) **Integration:** Incorporating relationships into the overall terminology system

Migration strategies recognize that many systems have existing relationship implementations that need to be preserved while enhancing their semantic precision and interoperability.

8.7 Enosema tools and resources

8.7.1 Relationship management tools

The Enosema Foundation provides or recommends tools for relationship management:

- a) **Relationship registry browser:** For exploring standardized relationship types
- b) **Relationship validator:** For checking relationship assertions against rules
- c) **Relationship visualization tools:** For exploring relationship networks
- d) **Relationship exchange utilities:** For converting between formats
- e) **Relationship documentation templates:** For consistent documentation

These tools support the practical implementation of the abstract model in various contexts.

8.7.2 Training and guidance

The Enosema Foundation provides training and guidance materials:

- a) **Introductory guides to concept relationships**
- b) **Best practice documentation**
- c) **Implementation tutorials**
- d) **Case studies of successful relationship management**
- e) **Troubleshooting guides for common issues**

These materials help organizations develop the knowledge and skills needed for effective relationship management.

8.7.3 Community resources

The Enosema Foundation fosters a community of practice around concept relationships:

- a) **Forums for discussing relationship issues**
- b) **Repositories of shared relationship resources**

- c) Collaborative development of relationship standards
- d) Expert networks for relationship validation
- e) Events for knowledge exchange and professional development

Community resources leverage collective expertise to advance the state of the art in relationship management and provide mutual support for implementation challenges.

Annex A

(normative)

Common relationship patterns

A.1 General

This annex provides a collection of common relationship patterns that can be used as templates for implementing concept relationships in various domains and applications. These patterns represent recurring structures of relationships that have proven useful in terminology management and knowledge organization.

The patterns are presented at an abstract level and can be adapted to specific domains and implementation contexts. Each pattern includes a description, typical applications, implementation considerations, and examples.

A.2 Hierarchical classification pattern

A.2.1 Pattern description

The hierarchical classification pattern organizes concepts into a tree or directed acyclic graph structure using generic (is-a) relationships. Each concept (except the root) has one or more superordinate concepts, and concepts inherit characteristics from their superordinates.

A.2.2 Structure

- a) Root concept(s) at the top level
- b) Multiple levels of subordinate concepts
- c) Generic relationships connecting levels
- d) Inheritance of characteristics down the hierarchy
- e) Potential for multiple inheritance (polyhierarchy)

A.2.3 Applications

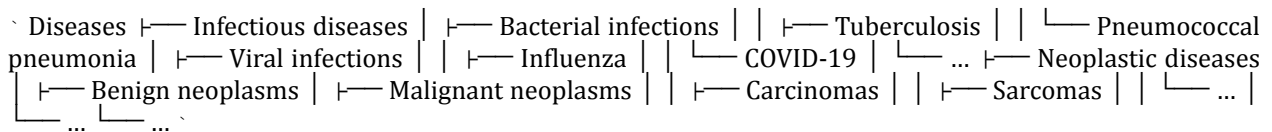
- a) Scientific taxonomies (biology, chemistry, etc.)
- b) Product and service classifications
- c) Organizational structures
- d) Document and content categorization
- e) Skill and competency frameworks

A.2.4 Implementation considerations

- a) Deciding between strict hierarchy (single inheritance) and polyhierarchy (multiple inheritance)
- b) Determining appropriate granularity at each level
- c) Handling cross-classification needs

- d) Managing inheritance exceptions
- e) Versioning and evolution of the classification

EXAMPLE Medical disease classification:



This pattern uses generic relationships to classify diseases by etiology and pathology, with each level inheriting characteristics from higher levels.

A.3 Compositional pattern

A.3.1 Pattern description

The compositional pattern represents how concepts combine to form more complex structures. It uses partitive relationships to connect components to wholes and can represent multiple levels of composition.

A.3.2 Structure

- a) Whole concept at the top level
- b) Component concepts at lower levels
- c) Partitive relationships connecting components to wholes
- d) Potential for multiple levels of decomposition
- e) Optional cardinality constraints on components

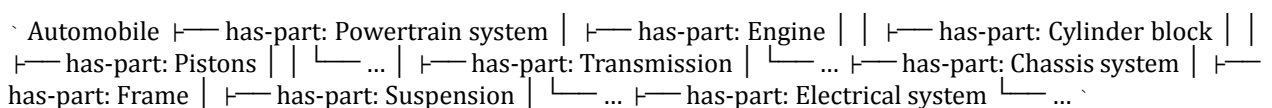
A.3.3 Applications

- a) Product and system architecture
- b) Document and content structure
- c) Organizational composition
- d) Geographical and administrative divisions
- e) Anatomical structures

A.3.4 Implementation considerations

- a) Distinguishing between mandatory and optional components
- b) Handling cardinality constraints
- c) Representing ordering of components (if relevant)
- d) Managing shared components
- e) Documenting component roles and functions

EXAMPLE Automotive compositional structure:



This pattern uses partitive relationships to represent the physical composition of an automobile, with multiple levels of decomposition.

A.4 Process flow pattern

A.4.1 Pattern description

The process flow pattern represents sequences of activities, events, or states using temporal relationships. It captures the order of steps in a process and can include branching, loops, and parallel paths.

A.4.2 Structure

- a) Process steps or states as concepts
- b) Temporal relationships connecting steps
- c) Potential branch and merge points
- d) Optional conditions on transitions
- e) Start and end points

A.4.3 Applications

- a) Business processes
- b) Clinical pathways
- c) Manufacturing procedures
- d) Project workflows
- e) System behavior models

A.4.4 Implementation considerations

- a) Representing sequential, parallel, and alternative paths
- b) Handling conditional transitions
- c) Documenting iteration and looping
- d) Managing exceptions and error paths
- e) Connecting to resources and actors

EXAMPLE Clinical diagnostic process:

Patient presentation $\vdash$ precedes: Initial assessment $\mid \vdash$ precedes: Physical examination $\mid \vdash$ precedes: Medical history taking $\vdash$ may-precede: Emergency intervention (if critical) $\vdash$...

Physical examination + Medical history taking $\vdash$ precedes: Differential diagnosis $\mid \vdash$ precedes: Diagnostic testing $\mid \mid \vdash$ precedes: Test result interpretation $\mid \mid \vdash$ precedes: Diagnosis confirmation (if positive) $\mid \mid \vdash$ precedes: Alternative testing (if negative) $\mid \mid \vdash$... $\mid \vdash$... $\vdash$...

This pattern uses temporal relationships to represent the clinical process from patient presentation to diagnosis, including alternative paths based on conditions.

A.5 Causal network pattern

A.5.1 Pattern description

The causal network pattern represents cause-effect relationships between concepts. It can capture complex causal chains, multiple causes and effects, and causal strength or probability.

A.5.2 Structure

- a) Cause and effect concepts
- b) Causal relationships connecting them
- c) Potential for causal chains (A causes B causes C)
- d) Possible multiple causes for a single effect
- e) Optional causal strength or probability indicators

A.5.3 Applications

- a) Risk analysis
- b) Fault diagnosis
- c) Scientific explanations
- d) Medical etiology
- e) Environmental impact assessment

A.5.4 Implementation considerations

- a) Distinguishing between necessary and sufficient causes
- b) Representing causal strength or probability
- c) Handling complex causal networks
- d) Documenting temporal aspects of causation
- e) Managing uncertainty and conflicting evidence

EXAMPLE Environmental causal network:

```

Industrial emissions ┤── causes: Air pollution ┤── causes: Respiratory diseases ┤── causes: Increased
healthcare costs ┤── causes: Reduced quality of life ┤── causes: Acid rain ┤── causes: Water
acidification ┤── causes: Aquatic ecosystem damage ┤── causes: Soil degradation ┤── causes: Reduced
agricultural productivity ┤── causes: Greenhouse gas increase ┤── causes: Climate change ┤── causes: Sea
level rise ┤── causes: Extreme weather events ┤── ...
  
```

This pattern uses causal relationships to represent the environmental impacts of industrial emissions, showing causal chains and multiple effects.

A.6 Functional pattern

A.6.1 Pattern description

The functional pattern represents the purposes, roles, and functions of concepts and how they relate to activities, goals, and requirements. It connects entities to their functions and functions to their contexts.

A.6.2 Structure

- a) Entity concepts (objects, systems, etc.)
- b) Function concepts (purposes, capabilities)
- c) Functional relationships connecting entities to functions
- d) Context concepts (situations, environments)
- e) Context relationships connecting functions to contexts

A.6.3 Applications

- a) Product and system design
- b) Requirements engineering
- c) Job and role descriptions
- d) Tool and technology classification
- e) Service descriptions

A.6.4 Implementation considerations

- a) Distinguishing between primary and secondary functions
- b) Representing functional hierarchies
- c) Connecting functions to performance criteria
- d) Documenting functional dependencies
- e) Handling multi-functional entities

EXAMPLE Medical device functional pattern:

$\backslash$ Ventilator $\vdash$ has-function: Oxygen delivery $\mid \vdash$ has-context: Respiratory failure $\mid \vdash$ has-context: Anesthesia $\vdash$ has-function: Airway pressure management $\mid \vdash$ has-context: Acute respiratory distress syndrome $\mid \vdash$ has-context: Chronic obstructive pulmonary disease $\vdash$ has-function: Respiratory rate control $\vdash$...

Oxygen delivery $\vdash$ has-requirement: Oxygen source $\vdash$ has-requirement: Delivery mechanism $\vdash$ has-performance-indicator: Oxygen concentration $\backslash$

This pattern uses functional relationships to connect a medical device (ventilator) to its functions, contexts, and requirements.

A.7 Equivalence mapping pattern

A.7.1 Pattern description

The equivalence mapping pattern represents relationships between equivalent or similar concepts across different systems, domains, languages, or versions. It captures different degrees of equivalence and provides context for the mappings.

A.7.2 Structure

- a) Source concepts from one system or domain

- b) Target concepts from another system or domain
- c) Equivalence relationships connecting source and target
- d) Equivalence type indicators (exact, inexact, partial)
- e) Mapping context and metadata

A.7.3 Applications

- a) Terminology mapping and alignment
- b) Standard harmonization
- c) Cross-language terminology
- d) System integration
- e) Version migration

A.7.4 Implementation considerations

- a) Documenting the degree of equivalence
- b) Handling one-to-many and many-to-many mappings
- c) Providing mapping rationale and evidence
- d) Managing mapping versioning
- e) Addressing gaps and untranslatable concepts

EXAMPLE Cross-terminology medical mapping:

SNOMED CT: Myocardial infarction (22298006) $\vdash$ exact-equivalent-to ICD-10: Acute myocardial infarction (I21) $\vdash$ exact-equivalent-to MeSH: Myocardial Infarction (D009203) $\vdash$ broader-than ICD-10: Acute transmural myocardial infarction of anterior wall (I21.0)

SNOMED CT: Pneumonia (233604007) $\vdash$ exact-equivalent-to ICD-10: Pneumonia, organism unspecified (J18) $\vdash$ exact-equivalent-to MeSH: Pneumonia (D011014) $\vdash$ narrower-than ICD-10: Pneumonia (J12-J18) $\vdash$ broader-than ICD-10: Viral pneumonia, not elsewhere classified (J12)

This pattern uses equivalence relationships to map concepts between different medical terminologies (SNOMED CT, ICD-10, MeSH), indicating the type of equivalence for each mapping.

A.8 Semantic field pattern

A.8.1 Pattern description

The semantic field pattern organizes concepts into interconnected networks based on their semantic relationships within a domain. It represents the rich web of associations that give concepts their full meaning in context.

A.8.2 Structure

- a) Central concept(s) as focal points
- b) Related concepts connected by various relationship types
- c) Multiple relationship layers (hierarchical, associative, etc.)
- d) Cross-connections between related concepts

- e) Domain context binding the field together

A.8.3 Applications

- a) Domain vocabulary development
- b) Semantic analysis
- c) Content organization
- d) Knowledge representation
- e) Educational materials

A.8.4 Implementation considerations

- a) Balancing comprehensiveness with usability
- b) Selecting appropriate relationship types
- c) Determining field boundaries
- d) Handling overlapping semantic fields
- e) Representing relationship strength or salience

EXAMPLE Semantic field for “education”:

```

` Education ⊢ is-a: Process ⊢ has-part: Teaching | ⊢ has-agent: Teacher | | ⊢ has-role: Instructor
| | ⊢ has-role: Mentor | | ⊢ has-role: Facilitator | ⊢ has-recipient: Student | ⊢ has-characteristic:
Learning style | ⊢ has-activity: Studying ⊢ has-part: Learning | ⊢ has-result: Knowledge acquisition
| ⊢ has-result: Skill development | ⊢ has-result: Attitude formation ⊢ has-setting: Educational
institution | ⊢ has-instance: School | ⊢ has-instance: University | ⊢ has-instance: Training center ⊢
has-instrument: Curriculum | ⊢ has-part: Course | ⊢ has-part: Lesson | ⊢ has-part: Assessment ⊢
... `

```

This pattern uses multiple relationship types to represent the semantic field of education, showing how concepts are interconnected in this domain.

A.9 Cross-domain bridge pattern

A.9.1 Pattern description

The cross-domain bridge pattern connects concepts from different domains through interface concepts and specialized cross-domain relationships. It facilitates knowledge integration and interdisciplinary communication.

A.9.2 Structure

- a) Source domain concepts
- b) Target domain concepts
- c) Interface concepts that span domains
- d) Cross-domain relationships
- e) Domain context documentation

A.9.3 Applications

- a) Interdisciplinary research
- b) Cross-sector collaboration
- c) Integrated information systems
- d) Translational science
- e) Policy development

A.9.4 Implementation considerations

- a) Identifying appropriate interface concepts
- b) Documenting domain-specific meanings
- c) Establishing clear cross-domain relationship semantics
- d) Handling terminology differences
- e) Managing conceptual misalignments

EXAMPLE Bridge between medical and legal domains:

Medical domain: Patient consent $\vdash$ is-a: Medical process $\vdash$ has-purpose: Treatment authorization $\vdash$ has-agent: Healthcare provider $\vdash$ has-recipient: Patient $\vdash$ cross-domain-relates-to: Informed consent (Legal domain)

Legal domain: Informed consent $\vdash$ is-a: Legal doctrine $\vdash$ has-purpose: Liability protection $\vdash$ has-requirement: Disclosure of risks $\vdash$ has-result: Legal authorization $\vdash$ cross-domain-relates-to: Patient consent (Medical domain)

Interface concept: Capacity to consent $\vdash$ medical-aspect: Mental competence assessment $\vdash$ legal-aspect: Legal capacity determination $\vdash$ bridges: Medical and legal domains

This pattern uses cross-domain relationships and an interface concept to connect related concepts from the medical and legal domains.

A.10 Versioning pattern

A.10.1 Pattern description

The versioning pattern represents relationships between different versions, editions, or states of concepts over time. It captures the evolution of concepts and maintains connections between successive versions.

A.10.2 Structure

- a) Concept versions as separate entities
- b) Temporal relationships between versions
- c) Version metadata (dates, status, etc.)
- d) Change documentation
- e) Optional branching and merging

A.10.3 Applications

- a) Standards versioning

- b) Terminology evolution
- c) Content management
- d) Product development
- e) Historical documentation

A.10.4 Implementation considerations

- a) Determining version granularity
- b) Documenting version differences
- c) Handling backward compatibility
- d) Managing concurrent versions
- e) Implementing version control policies

EXAMPLE Standard versioning pattern:

```

ISO 9001:2008 └─ has-previous-version: ISO 9001:2000 | └─ has-previous-version: ISO 9001:1994
└─ has-next-version: ISO 9001:2015 └─ has-status: Superseded └─ valid-from: 2008-11-14 └─ valid-to:
2018-09-14 └─ has-change: └─ Addition of risk management principles └─ Enhanced focus on process
approach └─ ...

```

```

ISO 9001:2015 └─ has-previous-version: ISO 9001:2008 └─ has-status: Current └─ valid-from: 2015-09-15
└─ has-change: └─ New high-level structure └─ Explicit risk-based thinking └─ ...

```

This pattern uses temporal relationships and metadata to represent the evolution of the ISO 9001 standard through multiple versions, documenting the changes between versions and their validity periods.

Annex B (normative)

Mappings to existing standards

B.1 General

This annex provides mappings between the abstract model for concept relationships presented in this document and various existing standards and frameworks for representing relationships. These mappings demonstrate how the abstract model can be realized in different implementation contexts while maintaining semantic consistency.

The mappings are not intended to be exhaustive but rather to illustrate the correspondence between the abstract model and common implementation approaches. They can serve as a guide for implementing the abstract model in specific technical environments.

B.2 Mapping to ISO standards

B.2.1 Mapping to ISO 704

ISO 704:2022 (Terminology work — Principles and methods) provides fundamental principles for terminology work, including the representation of concept relationships. The mapping between the abstract model and ISO 704 includes:

Table B.1

Abstract Model Element	ISO 704 Element	Notes
Concept	Concept	Direct correspondence
Concept relationship	Concept relation	Direct correspondence
Hierarchical relationships	Hierarchical relations	Direct correspondence
Generic relationship	Generic relation	Direct correspondence
Partitive relationship	Partitive relation	Direct correspondence
Instance relationship	Instance relation	Direct correspondence
Associative relationships	Associative relations	Direct correspondence
Equivalence relationships	Not explicitly covered	Can be represented through terminological synonymy
Relationship properties	Relation characteristics	Partial correspondence
Relationship context	Subject field	Partial correspondence
NOTE ISO 704 focuses primarily on hierarchical and associative relationships within a single language. The abstract model extends this with more detailed treatment of equivalence relationships, relationship properties, and cross-domain contexts.		

B.2.2 Mapping to ISO 24156-1

ISO 24156-1:2014 (Graphic notations for concept modelling in terminology work and its relationship with UML — Part 1: Guidelines for using UML notation in terminology work) provides guidance on using UML for representing concept relationships. The mapping includes:

Table B.2

Abstract Model Element	ISO 24156-1 Element	Notes
Concept	UML Class	Concepts are represented as classes

(continued)

Abstract Model Element	ISO 24156-1 Element	Notes
Concept relationship	UML Association	Relationships are represented as associations
Hierarchical relationships	UML Generalization	Generic relationships use the generalization relationship
Generic relationship	UML Generalization	Direct correspondence
Partitive relationship	UML Aggregation/Composition	Partitive relationships use aggregation or composition
Instance relationship	UML Instantiation	Instances are represented as objects
Associative relationships	UML Association	With appropriate stereotypes
Equivalence relationships	UML Dependency	With «equivalent» stereotype
Relationship properties	UML Association properties	Properties like transitivity can be represented as constraints
Relationship context	UML Package	Contexts can be represented as packages

B.2.3 Mapping to ISO 30042 (TBX)

ISO 30042:2019 (Management of terminology resources — TermBase eXchange (TBX)) provides a format for exchanging terminology data. The mapping includes:

Table B.3

Abstract Model Element	TBX Element	Notes
Concept	<termEntry>	Concepts are represented as term entries
Concept relationship	<xref>	Relationships are represented as cross-references
Hierarchical relationships	<xref type="broaderConcept">, <xref type="narrowerConcept">	Using appropriate relationship types
Generic relationship	<xref type="superordinateConcept">, <xref type="subordinateConcept">	Using appropriate relationship types
Partitive relationship	<xref type="wholeConcept">, <xref type="partConcept">	Using appropriate relationship types
Instance relationship	<xref type="instanceConcept">, <xref type="generalConcept">	Using appropriate relationship types
Associative relationships	<xref type="relatedConcept">	With more specific types as needed
Equivalence relationships	<termNote type="termType">	Within a concept for same-language equivalence
Relationship properties	Custom data categories	Can be represented using custom data categories
Relationship context	<descrip type="subjectField">	Context represented through subject field

B.3 Mapping to semantic web standards

B.3.1 Mapping to RDF/RDFS

The Resource Description Framework (RDF) and RDF Schema (RDFS) provide a foundation for representing relationships in the semantic web. The mapping includes:

Table B.4

Abstract Model Element	RDF/RDFS Element	Notes
Concept	rdfs:Class	Concepts are represented as classes

(continued)

Abstract Model Element	RDF/RDFS Element	Notes
Concept relationship	rdf:Property	Relationships are represented as properties
Hierarchical relationships	rdfs:subClassOf	For generic relationships
Generic relationship	rdfs:subClassOf	Direct correspondence
Partitive relationship	Custom property	E.g., ex:hasPart, ex:isPartOf
Instance relationship	rdf:type	Direct correspondence
Associative relationships	Custom properties	With appropriate semantics
Equivalence relationships	owl:equivalentClass, owl:sameAs	Using OWL extensions to RDF
Relationship properties	rdfs:domain, rdfs:range	Basic properties; others require extensions
Relationship context	Named graphs	Contexts can be represented as named graphs

B.3.2 Mapping to OWL

The Web Ontology Language (OWL) extends RDF with more expressive constructs for representing relationships. The mapping includes:

Table B.5

Abstract Model Element	OWL Element	Notes
Concept	owl:Class	Concepts are represented as classes
Concept relationship	owl:ObjectProperty	Relationships are represented as object properties
Hierarchical relationships	rdfs:subClassOf	For generic relationships
Generic relationship	rdfs:subClassOf	Direct correspondence
Partitive relationship	Custom object property	With appropriate semantics
Instance relationship	rdf:type	Direct correspondence
Associative relationships	Custom object properties	With appropriate semantics
Equivalence relationships	owl:equivalentClass, owl:sameAs	Direct correspondence
Relationship properties	Property characteristics	owl:TransitiveProperty, owl:SymmetricProperty, etc.
Relationship context	owl:imports, Named graphs	Contexts can be represented through imports or named graphs

B.3.3 Mapping to SKOS

The Simple Knowledge Organization System (SKOS) provides a model for expressing knowledge organization systems. The mapping includes:

Table B.6

Abstract Model Element	SKOS Element	Notes
Concept	skos:Concept	Direct correspondence
Concept relationship	skos:semanticRelation	Base class for all relationships
Hierarchical relationships	skos:broader, skos:narrower	Direct correspondence
Generic relationship	skos:broader, skos:narrower	SKOS doesn't distinguish between generic and partitive
Partitive relationship	skos:broader, skos:narrower	With optional extensions
Instance relationship	Not directly supported	Can be represented with extensions
Associative relationships	skos:related	Direct correspondence
Equivalence relationships	skos:exactMatch, skos:closeMatch	For cross-scheme equivalence
Relationship properties	Not directly supported	Requires extensions

(continued)

Abstract Model Element	SKOS Element	Notes
Relationship context	skos:ConceptScheme	Contexts represented as concept schemes

B.4 Mapping to knowledge organization systems

B.4.1 Mapping to thesaurus standards

Thesaurus standards like ISO 25964 provide models for representing relationships in thesauri. The mapping includes:

Table B.7

Abstract Model Element	Thesaurus Element	Notes
Concept	Concept	Direct correspondence
Concept relationship	Relationship	Direct correspondence
Hierarchical relationships	BT/NT (Broader Term/Narrower Term)	Direct correspondence
Generic relationship	BTG/NTG (Broader Term Generic/Narrower Term Generic)	In ISO 25964
Partitive relationship	BTP/NTP (Broader Term Partitive/Narrower Term Partitive)	In ISO 25964
Instance relationship	BTI/NTI (Broader Term Instance/Narrower Term Instance)	In ISO 25964
Associative relationships	RT (Related Term)	Direct correspondence
Equivalence relationships	USE/UF (Use/Used For)	For same-language equivalence
Relationship properties	Not explicitly represented	Implicit in relationship types
Relationship context	Microthesaurus, Facet	Contexts represented through structural elements

B.4.2 Mapping to taxonomy standards

Taxonomies typically focus on hierarchical relationships. The mapping includes:

Table B.8

Abstract Model Element	Taxonomy Element	Notes
Concept	Category, Class	Concepts represented as categories or classes
Concept relationship	Hierarchical relationship	Primarily hierarchical
Hierarchical relationships	Parent-child relationship	Direct correspondence
Generic relationship	Is-a relationship	Direct correspondence
Partitive relationship	Has-part relationship	When supported
Instance relationship	Instance relationship	When supported
Associative relationships	See-also relationship	Limited support in most taxonomies
Equivalence relationships	Synonym relationship	Limited to same-language synonyms
Relationship properties	Not explicitly represented	Implicit in relationship types
Relationship context	Facet, Domain	Contexts represented through structural elements

B.5 Mapping to database models

B.5.1 Mapping to relational database model

Relational databases can represent concept relationships through various table structures. The mapping includes:

Table B.9

Abstract Model Element	Relational DB Element	Notes
Concept	Entity (table)	Concepts represented as entities
Concept relationship	Relationship (foreign key)	Relationships represented through foreign keys
Hierarchical relationships	Self-referencing relationship	With type indicators
Generic relationship	Self-referencing relationship	With type="generic"
Partitive relationship	Self-referencing relationship	With type="partitive"
Instance relationship	Foreign key relationship	Between instance and class tables
Associative relationships	Many-to-many relationship	Through junction tables
Equivalence relationships	Synonym table	With equivalence type indicators
Relationship properties	Attributes in relationship tables	Properties stored as attributes
Relationship context	Context table	With relationships to context entities

B.5.2 Mapping to graph database model

Graph databases are particularly well-suited for representing concept relationships. The mapping includes:

Table B.10

Abstract Model Element	Graph DB Element	Notes
Concept	Node	Concepts represented as nodes
Concept relationship	Edge	Relationships represented as edges
Hierarchical relationships	Typed edge	With hierarchical type label
Generic relationship	Edge with "is-a" label	Direct correspondence
Partitive relationship	Edge with "part-of" label	Direct correspondence
Instance relationship	Edge with "instance-of" label	Direct correspondence
Associative relationships	Typed edges	With appropriate type labels
Equivalence relationships	Typed edges	With equivalence type labels
Relationship properties	Edge properties	Properties stored as edge attributes
Relationship context	Context nodes or edge properties	Contexts represented through additional structures

B.6 Implementation considerations

When mapping the abstract model to specific standards or frameworks, several considerations should be taken into account:

- Semantic precision:** Ensure that the semantic nuances of the abstract model are preserved in the target representation
- Relationship type granularity:** Adjust the level of detail in relationship types to match the capabilities of the target system
- Property representation:** Determine how to represent relationship properties that may not be directly supported
- Context handling:** Develop appropriate mechanisms for representing relationship contexts

- e) Extension mechanisms: Utilize extension mechanisms in the target system to represent aspects of the abstract model that are not directly supported
- f) Documentation: Clearly document the mapping to ensure consistent implementation and interpretation

NOTE No single implementation can perfectly capture all aspects of the abstract model. The goal should be to preserve the essential semantics while adapting to the constraints and capabilities of the target system.

Bibliography

- [1] ISO 860:2007, *Terminology work — Harmonization of concepts and terms*
- [2] ISO 12620:2019²⁾, *Management of terminology resources — Data category specifications*
- [3] ISO 23185:2009, *Assessment and benchmarking of terminological resources — General concepts, principles and requirements*
- [4] ISO 24613-1:2019³⁾, *Language resource management — Lexical markup framework (LMF) — Part 1: Core model*
- [5] ISO 26162-1:2019, *Management of terminology resources — Terminology databases — Part 1: Design*
- [6] ISO 26162-2:2019, *Management of terminology resources — Terminology databases — Part 2: Software*
- [7] ISO 29383:2020, *Terminology policies — Development and implementation*
- [8] Wüster1979, Wüster, E. 1979. *Einführung in die allgemeine Terminologielehre und terminologische Lexikographie*. Vienna: Springer.
- [9] Cruse1986, Cruse, D.A. 1986. *Lexical Semantics*. Cambridge: Cambridge University Press.
- [10] Winston1987, Winston, M.E., R. Chaffin, and D. Herrmann. 1987. "A Taxonomy of Part-Whole Relations." *Cognitive Science* 11(4): 417-444.
- [11] Sager1990, Sager, J.C. 1990. *A Practical Course in Terminology Processing*. Amsterdam/Philadelphia: John Benjamins.
- [12] Gruber1993, Gruber, T.R. 1993. "A Translation Approach to Portable Ontology Specifications." *Knowledge Acquisition* 5(2): 199-220.
- [13] Miller1995, Miller, G.A. 1995. "WordNet: A Lexical Database for English." *Communications of the ACM* 38(11): 39-41.
- [14] Wright1997, Wright, S.E. and G. Budin. 1997. *Handbook of Terminology Management*. Amsterdam/Philadelphia: John Benjamins.
- [15] Fellbaum1998, Fellbaum, C. (ed). 1998. *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.
- [16] Cabré1999, Cabré, M.T. 1999. *Terminology: Theory, Methods, and Applications*. Amsterdam/Philadelphia: John Benjamins.
- [17] Aitchison2000, Aitchison, J., A. Gilchrist, and D. Bawden. 2000. *Thesaurus Construction and Use: A Practical Manual*. 4th ed. London: Aslib IMI.
- [18] Temmerman2000, Temmerman, R. 2000. *Towards New Ways of Terminology Description: The Sociocognitive Approach*. Amsterdam/Philadelphia: John Benjamins.
- [19] Noy2001, Noy, N.F. and D.L. McGuinness. 2001. "Ontology Development 101: A Guide to Creating Your First Ontology." *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05*.

2) Withdrawn.

3) Withdrawn.

- [20] Kageura2002, Kageura, K. 2002. *The Dynamics of Terminology: A Descriptive Theory of Term Formation and Terminological Growth*. Amsterdam/Philadelphia: John Benjamins.
- [21] Murphy2003, Murphy, M.L. 2003. *Semantic Relations and the Lexicon*. Cambridge: Cambridge University Press.
- [22] Rector2004, Rector, A.L. 2004. "Modularisation of Domain Ontologies Implemented in Description Logics and Related Formalisms including OWL." *Proceedings of the 2nd International Conference on Knowledge Capture*: 121-128.
- [23] Soergel2004, Soergel, D. 2004. "Indexing Languages and Thesauri: Construction and Maintenance." *Information Resources Press*.
- [24] Smith2005, Smith, B. 2005. "Relations in Biomedical Ontologies." *Genome Biology* 6(5): R46.
- [25] Khoo2006, Khoo, C.S.G. and J.C. Na. 2006. "Semantic Relations in Information Science." *Annual Review of Information Science and Technology* 40: 157-228.
- [26] Hjørland2007, Hjørland, B. 2007. "Semantic Relations." *Knowledge Organization* 34(4): 207-217.
- [27] Green2008, Green, R., C.A. Bean, and S.H. Myaeng (eds). 2008. *The Semantics of Relationships: An Interdisciplinary Perspective*. Dordrecht: Kluwer Academic Publishers.
- [28] Guarino2009, Guarino, N. and C. Welty. 2009. "An Overview of OntoClean." In *Handbook on Ontologies*, edited by S. Staab and R. Studer, 201-220. Berlin: Springer.
- [29] Bean2010, Bean, C.A. and R. Green (eds). 2010. *Relationships in the Organization of Knowledge*. Dordrecht: Kluwer Academic Publishers.
- [30] Faber2012, Faber, P. (ed). 2012. *A Cognitive Linguistics View of Terminology and Specialized Language*. Berlin/Boston: De Gruyter Mouton.
- [31] W3C-OWL, World Wide Web Consortium. 2012. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. W3C Recommendation. <https://www.w3.org/TR/owl2-overview/>
- [32] W3C-RDF, World Wide Web Consortium. 2014. *RDF 1.1 Concepts and Abstract Syntax*. W3C Recommendation. <https://www.w3.org/TR/rdf11-concepts/>
- [33] W3C-SKOS, World Wide Web Consortium. 2009. *SKOS Simple Knowledge Organization System Reference*. W3C Recommendation. <https://www.w3.org/TR/skos-reference/>

