Systemic Federated
Enterprise Architecture & Transformation

Building Ontologies
-
The Syntactic Level

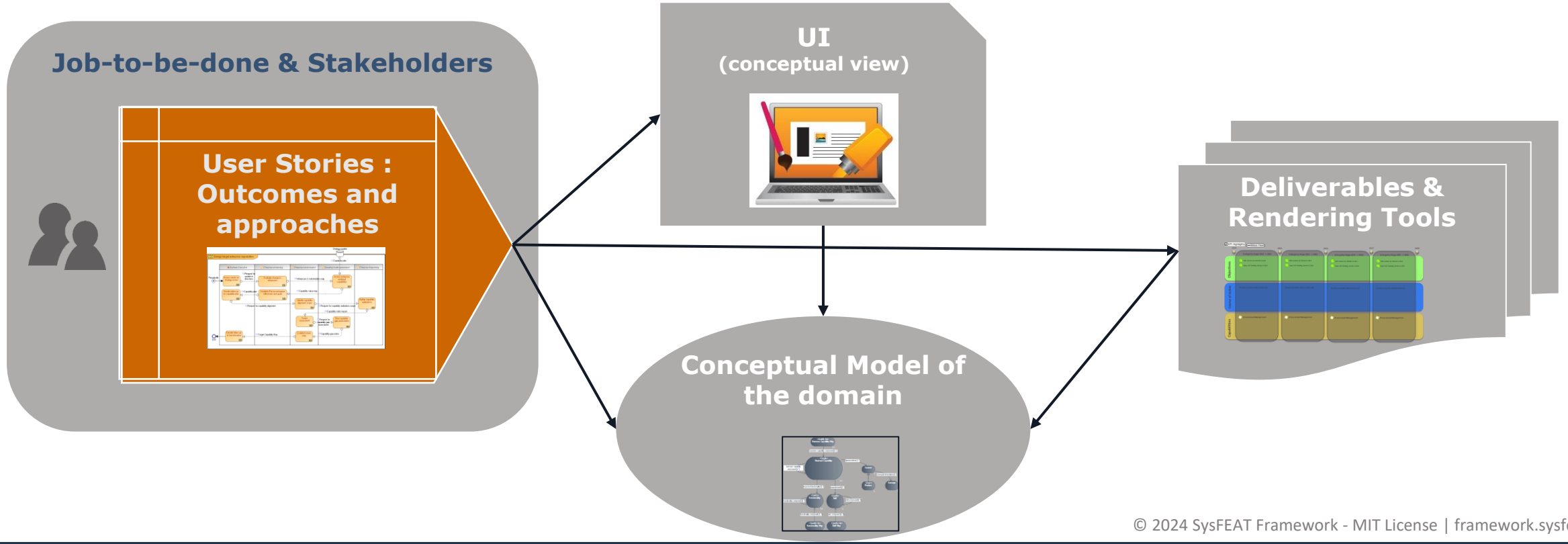
Building Ontologies

The conceptualization process:

Syntactic and Semantic Methodology for building Ontologies

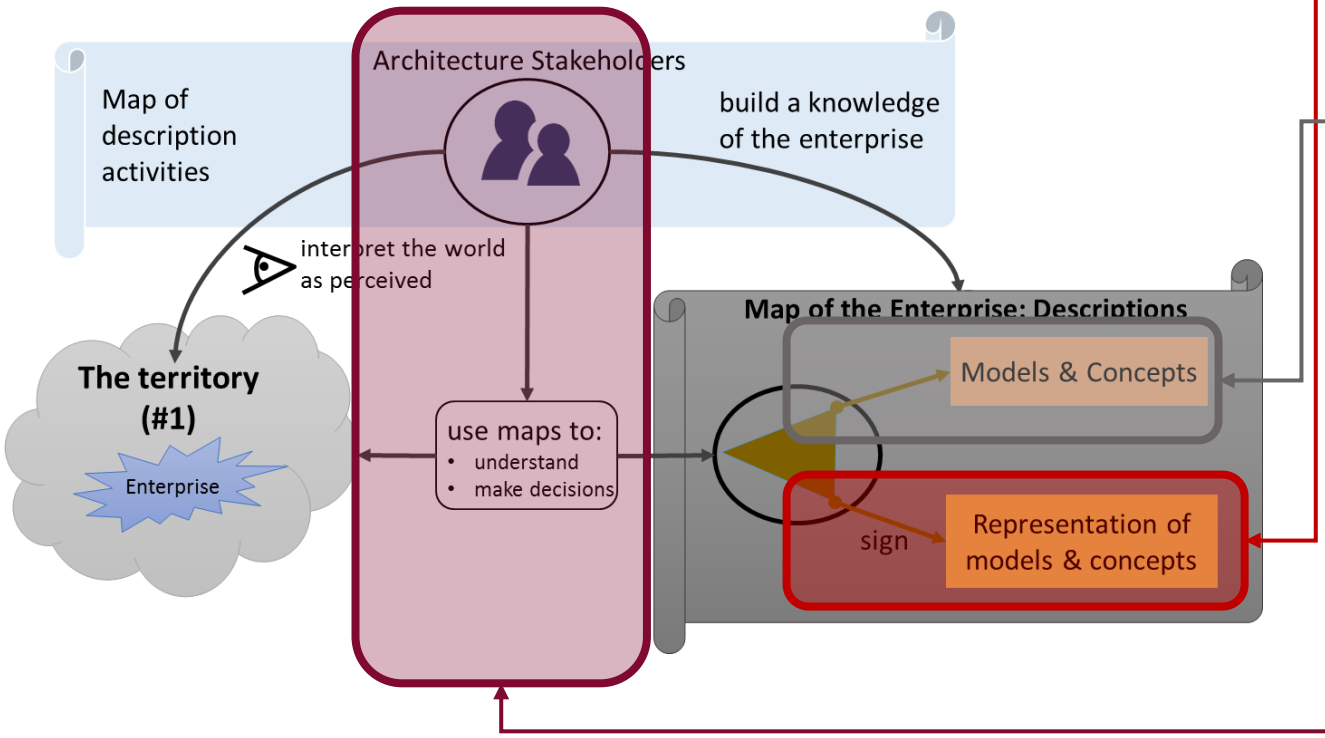
Reminder: the conceptualization process

- The conceptualization process defines syntactic and semantic approaches for building ontologies. It includes:
  - Gathering involved stakeholders and activities they need to pursue to achieved their jobs.
  - Defining input and outputs for stakeholder activity steps in terms of UI and underlying concepts to manipulated.
  - Indicating deliverables built and used at each source activity steps.



Modeling Language Aspects

- As any language, modeling languages have three aspects:



- **Syntax** is the required grammar and punctuation of the language.
  - The *upper-ontology* is about syntax.
- **Semantics** is about meaning - what do we mean by a Capability?
  - The SOF (System Operating Framework) is about semantic for Enterprise Architecture.
- **Pragmatics** has to do with:
  - how to use models (modeling technics)
  - What kind of model to use to address stakeholder concerns (method).

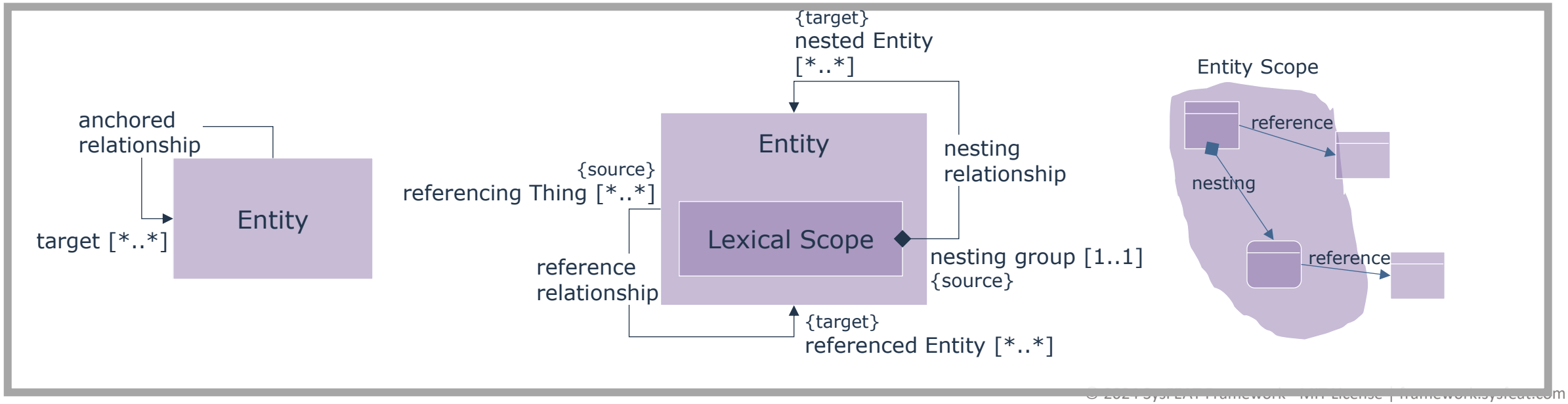
  Example: how to use capability modeling in enterprise transformation initiatives?

Purpose of this document

- Provide elementary syntactic rules required for building ontologies.
- Most syntactic rules deal with the right ways of creating Entities and their Relationships.
- The goal is to obtain Entity structures with well identified boundaries: what is in or out an Entity.
- The value is to provide modular ontologies that can be extended, plugged in and plugged out.

Modeling Syntax – scoping entities

- **Anchoring**: every relationship is anchored in its source entity — it lives inside the source, not floating between two nodes — a principle formalized mathematically as a fiber bundle.
- **Structural primitives**: any modeling language rests on two kinds of directed relationships:
  - Reference — relates an entity to another without enclosing it.
  - Nesting — relates an entity to another within the source's lexical scope, making the target structurally enclosed.
- Nesting ≠ Composition. Nesting is a syntactic relationship stating that some entities can enclose others. It defines spatial containment, not the whole/part semantic of mereology.
- **Entity scope**. Together, anchoring, nesting and reference define a syntactic boundary — what is inside vs. outside an entity. An entity's scope is the union of its nested sub-entities and its outgoing reference relationships.



Syntactic Rules Summary

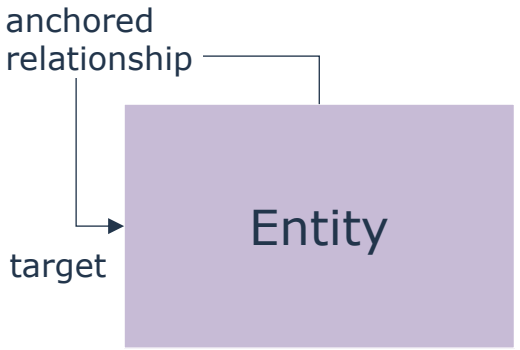
- Rule 1: Anchoring and directed relationships
- Rule 2: Existential dependency relationship
- Rule 3: Nesting relationships
- Rule 4: Attributes versus Relationships
- Rule 5: Common concept versus segregated concepts

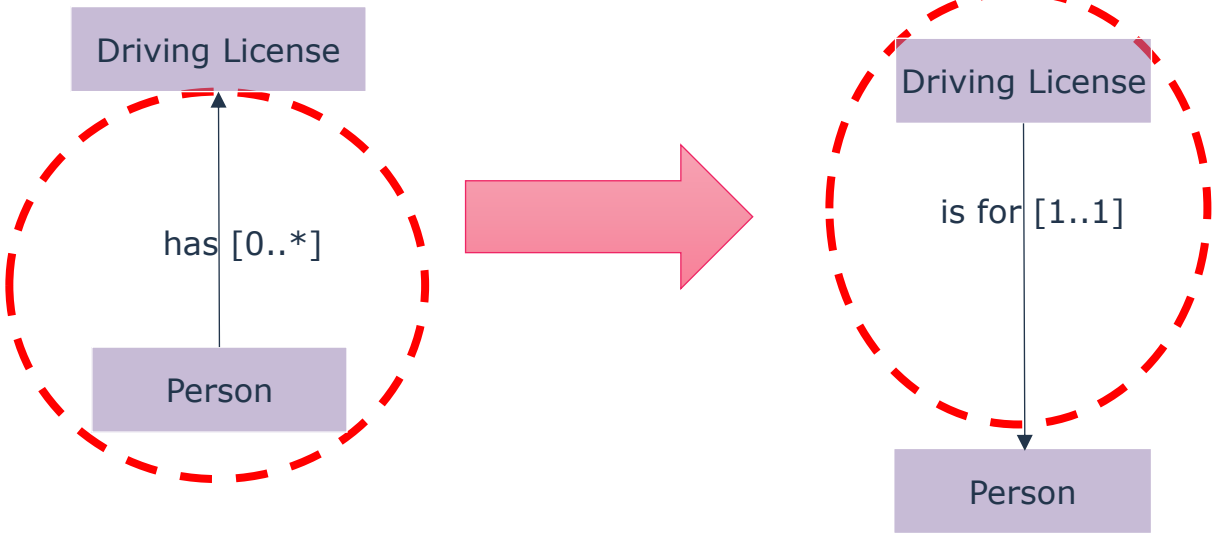
Rule 1

Anchoring: directed relationships

Rule 1.1 – Anchored relationships -> directed

- Because every relationship is anchored in its source entity, all binary relationships are inherently directed: from the source entity to the target entity.
- Therefore, every relationship falls within the scope of its source entity: it is internal to the source, not shared between two peers..





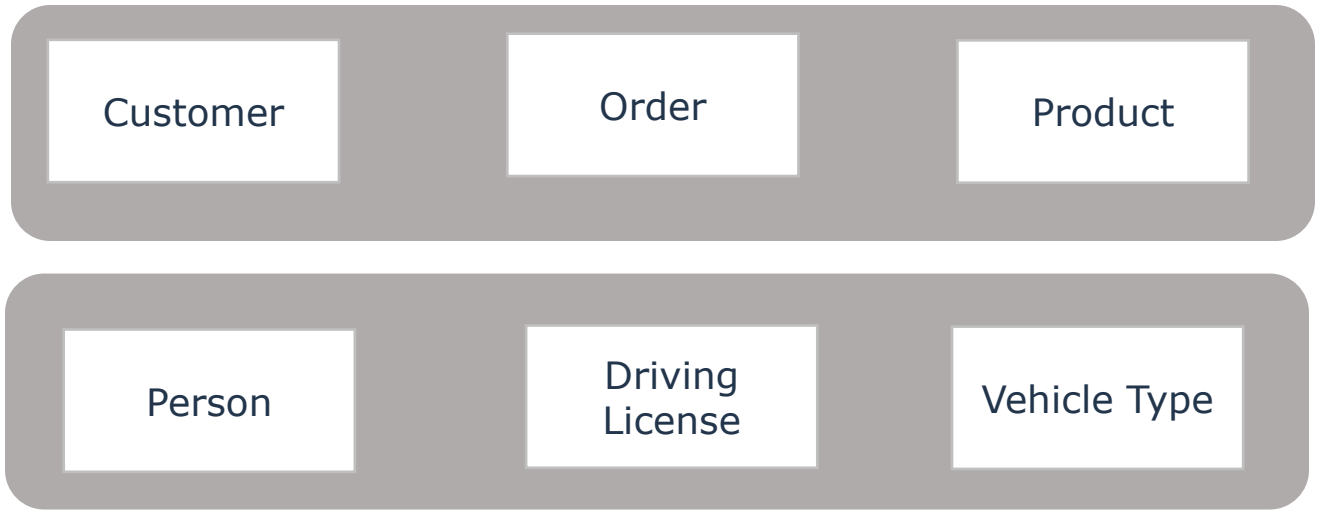
Persons have driving licenses … or …. Driving licenses are for Persons **?**

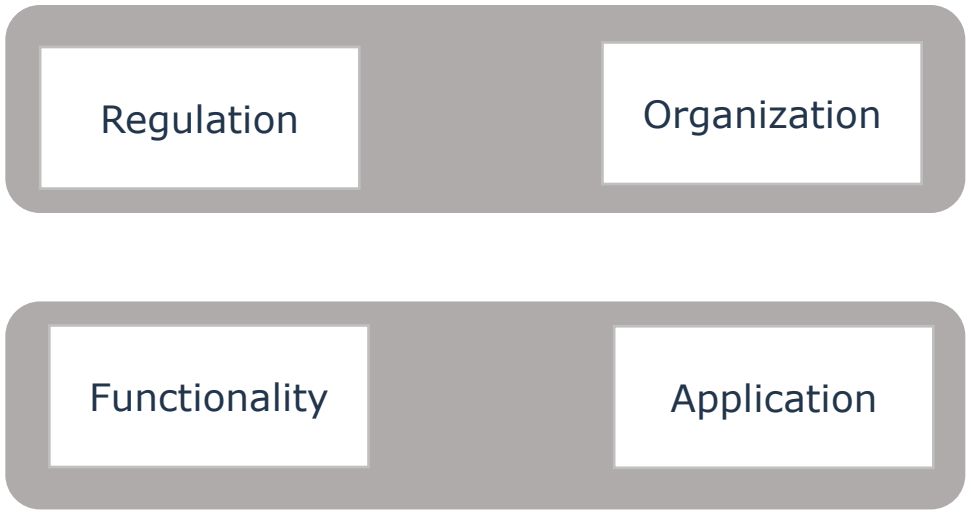
Tip : Finding the right direction might seem tough at the beginning. A good way to know is to wonder *"Which entity description is incomplete if I remove the relationship ?"*, or *"Which entity exists before the other ?"*

Exercise 1 – Determine relationships direction

- Setup relationship directions for the following examples:
  - Customers take orders.
  - Persons have driving licenses for specific type of vehicle.

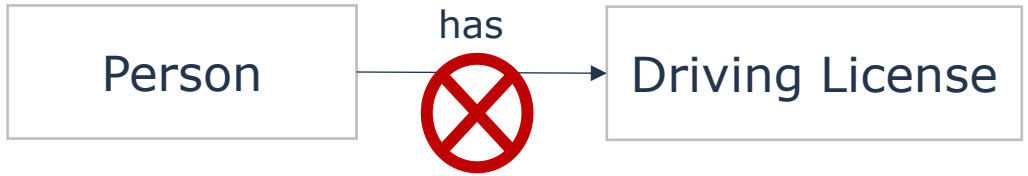
  - Functionalities are fulfilled by Applications.
  - Regulations applies to Organizations.





Directed Relationships – Tips – Verbalization pitfalls

- The discovery of concepts through verbalization is a common practice. While useful in many cases, it is also misleading when defining relationship directions.
- A common error is to confuse *direction from a navigation viewpoint* with *direction from an Entity definition viewpoint (what is in an out an Entity)*.
- For instance, one needs to get the orders of a customer and gets to a conclusion that the relationship is directed from Customers to Orders.



Tips: naming relationships

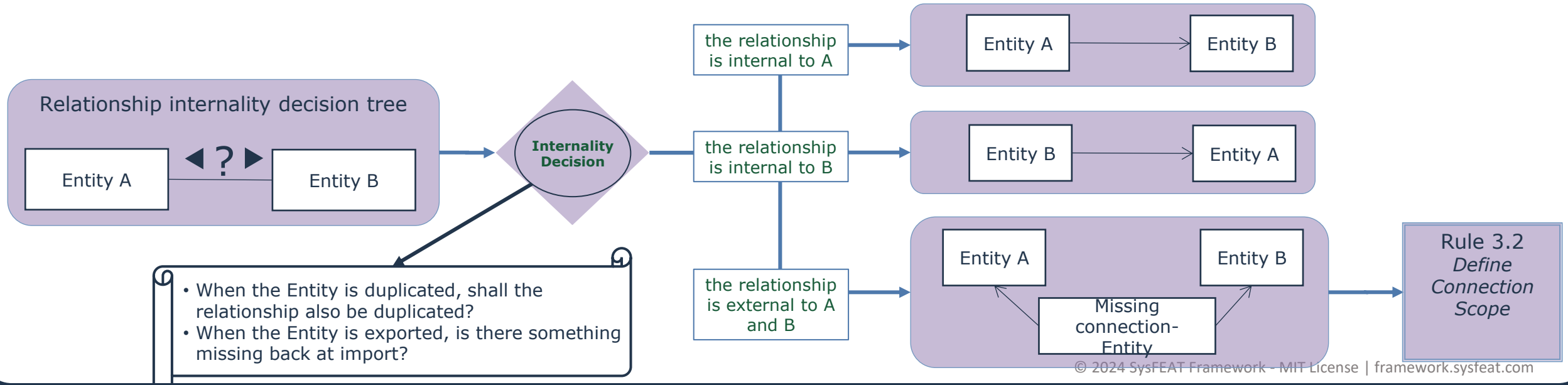
- Avoid:
  - Conjugated verbs - customer takes order, book has author,
  - The ambiguous "has" – unclear whether it indicates whole/part (composition) or classification (typing)

- Best practice: transform the predicate to clarify direction and meaning:

Technic	Example
Use passive voice – to anchor the relation in the source entity's perspective	Course hasInstructor Teacher → Course — instructed by — Teacher
Use role insertion – make the target's role explicit within the relation name	Student hasGrade Grade → Student — assigned Grade — Course
Use nominalization – turn the action or state into a noun that reifies the relation	Student isEnrolledIn Course → Student — enrollment — Course

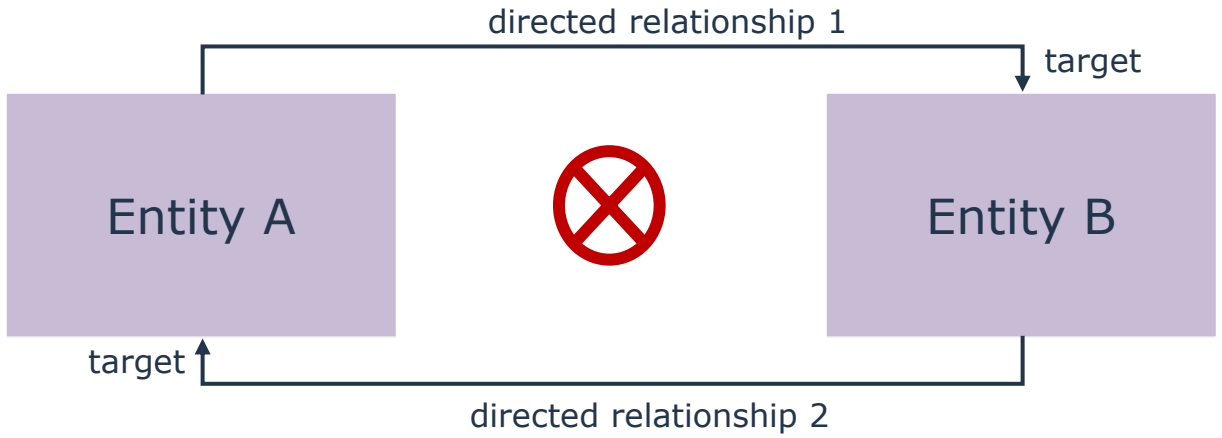
Rule 1.3 : the relationship internality decision tree

- When relationship directions are difficult to identified, the relationship internality decision-tree shall be applied.
- The internality decision-tree consists in evaluating the Entity scope through Entity manipulations:
  - When the Entity is duplicated, shall the relationship also be duplicated?
  - When the Entity is exported, is there something missing back at import?
- If the test concludes that the relationship doesn't belong to any of the related Entities, it means that a third *connection-Entity* is missing.
  - At this stage of the conceptualization process, the nature of the *connection-Entity* is still unknown. We just know that something is missing.
  - Once the *connection-Entity* has been identified, the next step is to determine its owner by applying the ownership rule.



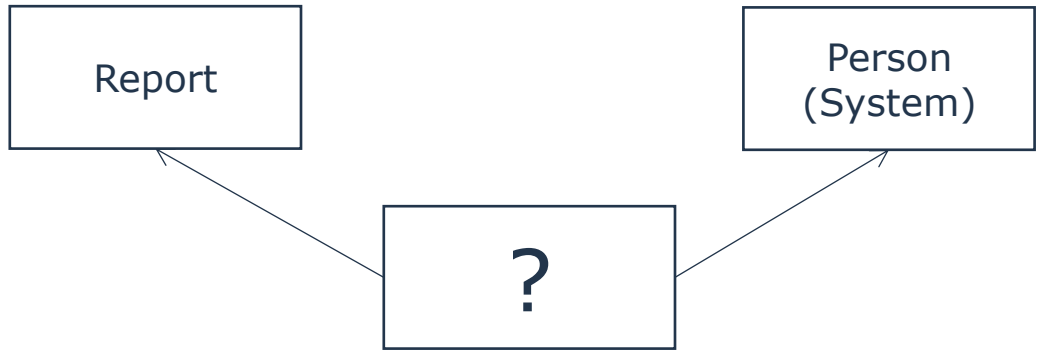
Rule 1.4 – No circular reference

- There shall not be circular references between Entities.
  - Recursive references are an exception.



Exercise 2: apply the relationship-internality decision-tree

- Does the relationship between Report and Person-(System) belong to Report or to Person-(System)?
  - The relationship expresses the fact that the Person-(System) has created the report.
- If the relationship belongs to Person-(System), shall it be duplicated when a Person (System) is duplicated?
- If the relationship belongs to Report, shall it be duplicated when a report is duplicated?
- What if the relationship belongs neither to Report nor to Person-(System)?



Directed Relationships – TOGAF 9.2

- Find semantic boundaries in the TOGAF 9.1 meta-model.

- Test:
- What is the nature of the relationship between “Process” and “Business Service”?

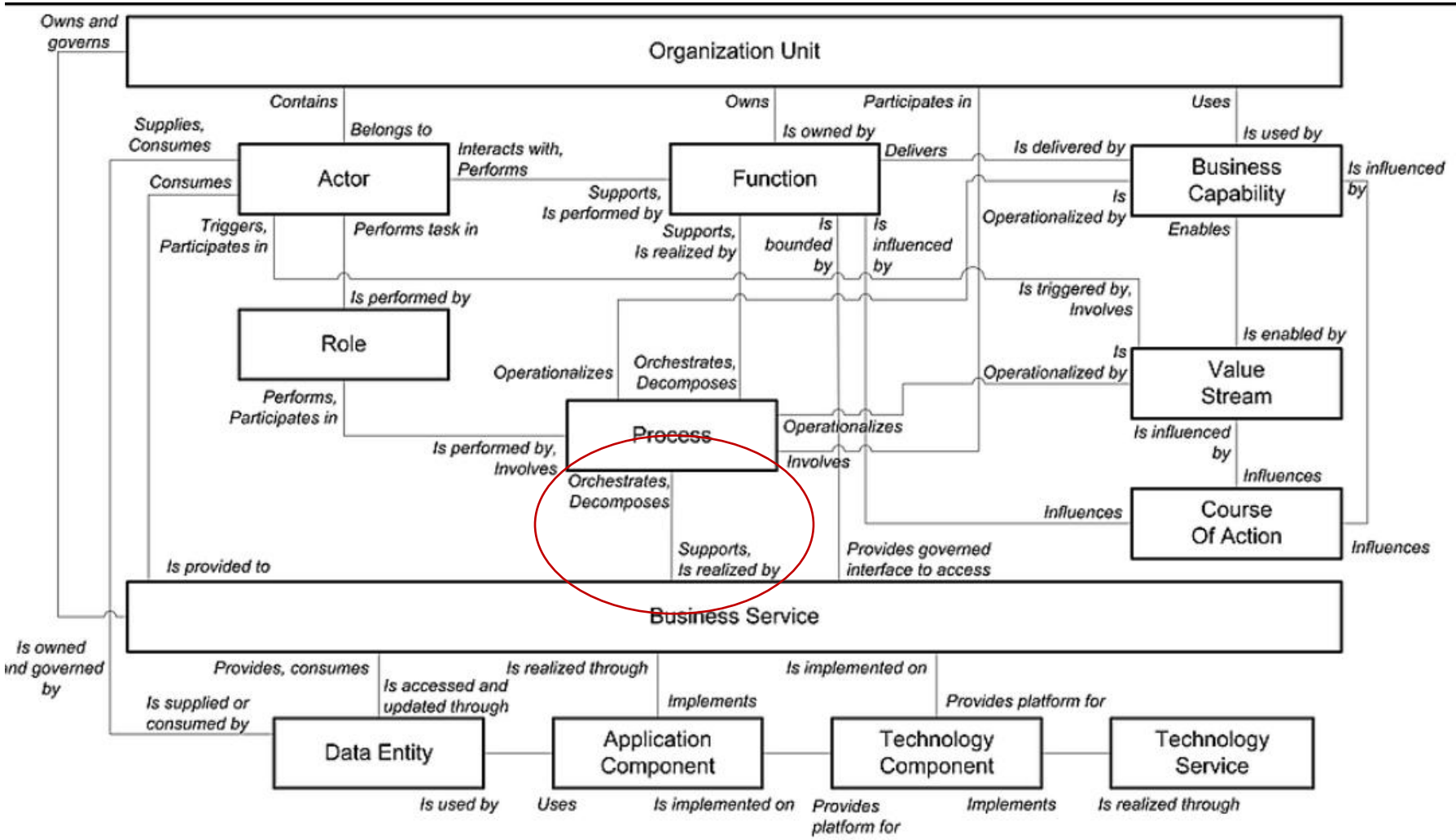


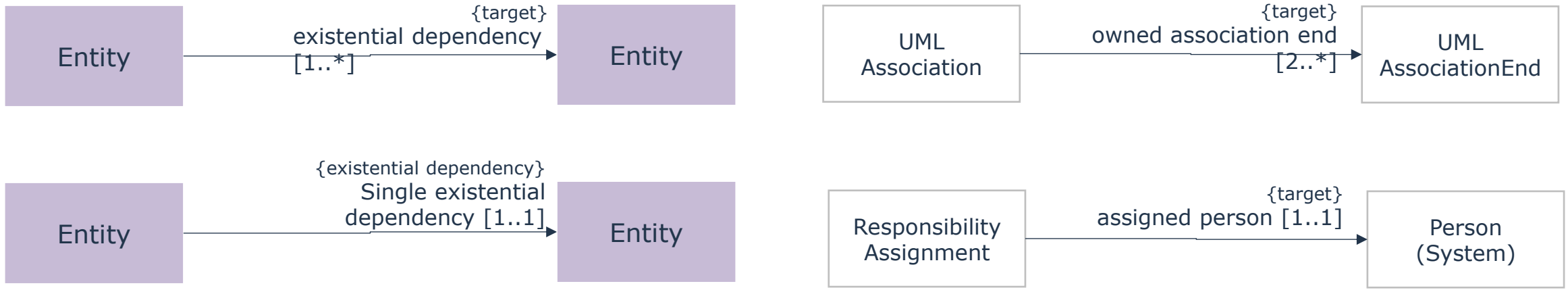
Figure 30-5: Entities and Relationships Present within the Core Content Metamodel

Rule 2

Existential dependency relationship

Existential dependency relationship

- Existential-dependency relationships are directed-relationships which are required for their source-Entity to exist. Their minimum target cardinality is equal to 1 (card-min = 1):
- Most existential-relationships also have their maximum cardinality equal to 1 (card-min = 1, card-max = 1).

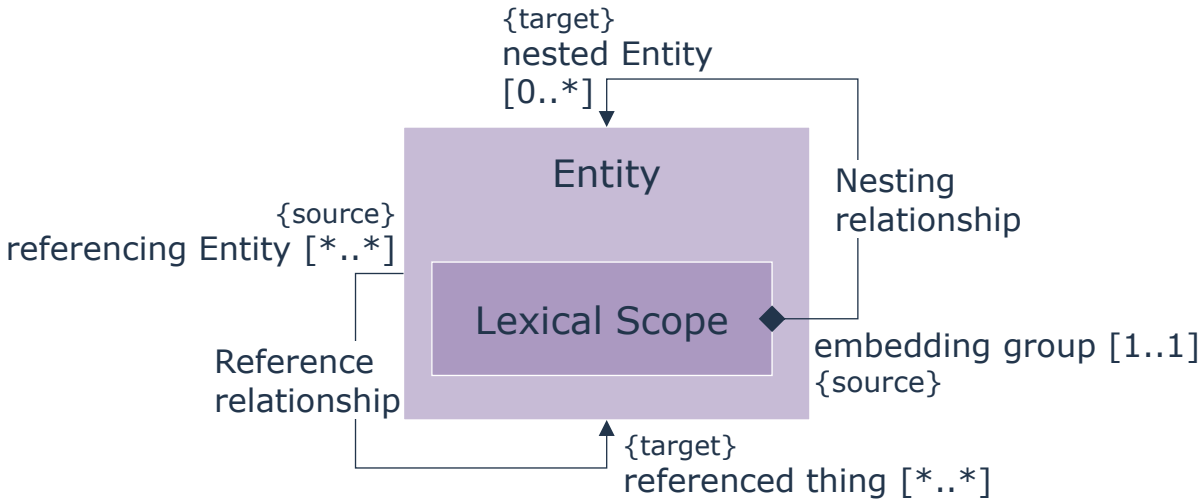


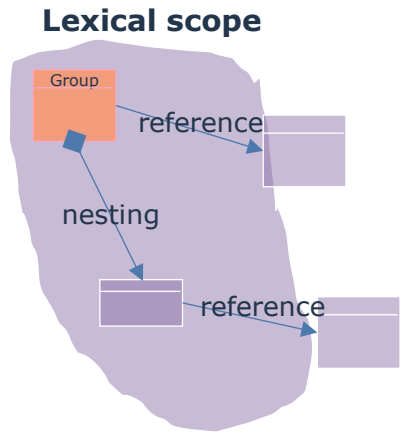
Rule 3

Nesting relationships

Nesting-relationship : scoping of entities

- Nesting relationships are directed relationships where the target Entity is nested into the source Entity. In other word, the target Entity exists only in the context of the source Entity.
- Entity-scoping partitions directed-relationships into two exclusive categories:
  - Reference relationships: Entities and their referenced Entities.
  - Nesting relationships: Entity Scopes and their nested Entities.
- Altogether, Nesting and reference provide a mechanism for a syntactic boundary of what is in or out of an Entity: an Entity scope is defined by its nested Entities and its directed relationships.
- All Entities belong to one and only one nesting group ([1..1] cardinality).
  - At this stage of the conceptualization process, the Group-of-Entity concept doesn't preclude the nature of the grouping. Group-of-Entities can be containers, blocks, hierarchies, etc.





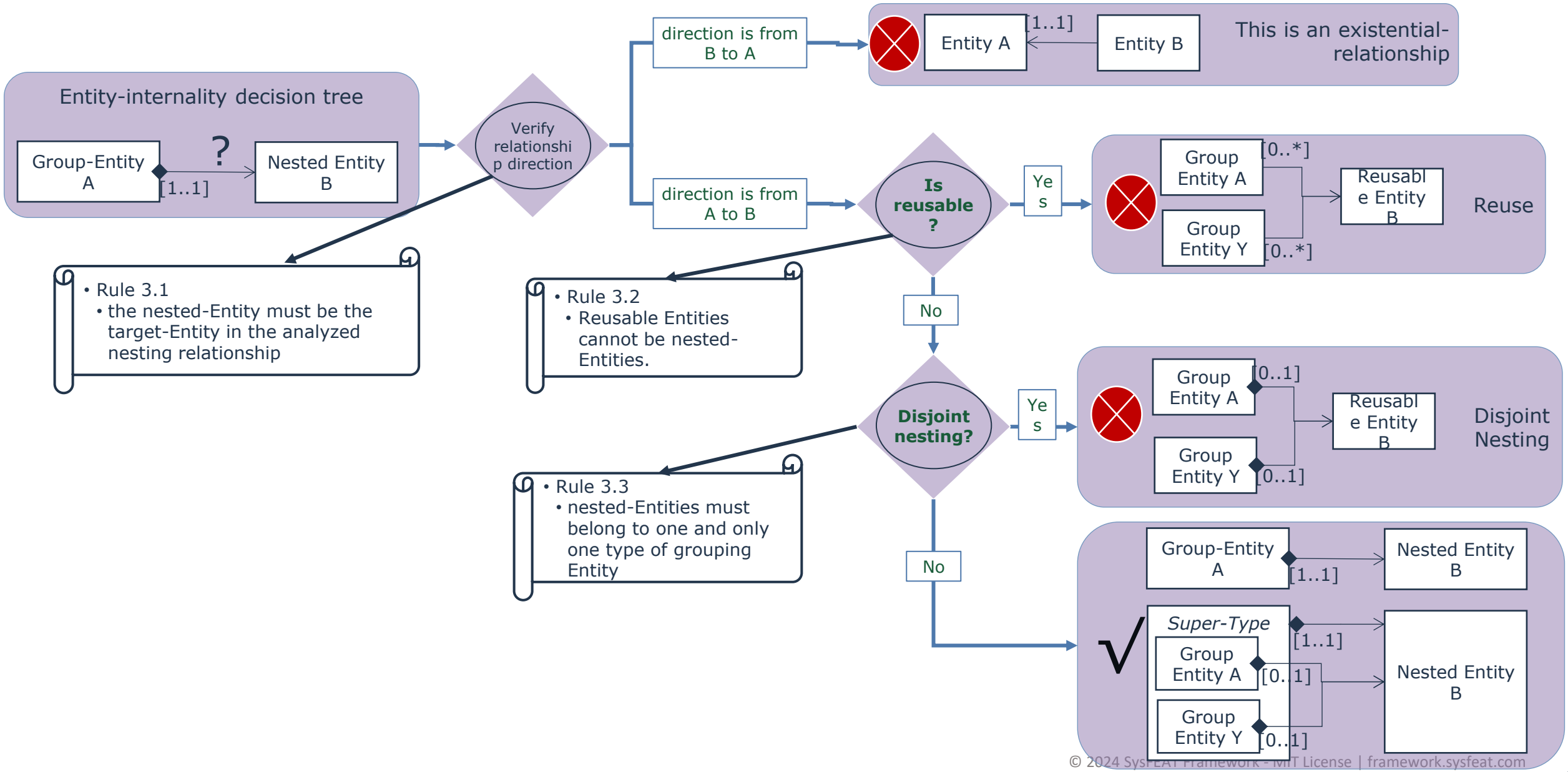
The Entity-ownership decision-tree

- Rule 3.1 - Verify relationship direction : the nested-Entity must be the target-Entity in the considered nesting relationship.
  - If the candidate nested-Entity is not the target Entity in the nesting-relationship, there is presumably a confusion with existential-dependencies (See next slide).

- Rule 3.2 - Are nested-Entities reusable?
  - When a candidate nested-Entity can be reused in multiple group, then the Entity is presumably not a nested-Entity.
    - For instance, because a process can belong to multiple parent processes => processes cannot “nest” processes.

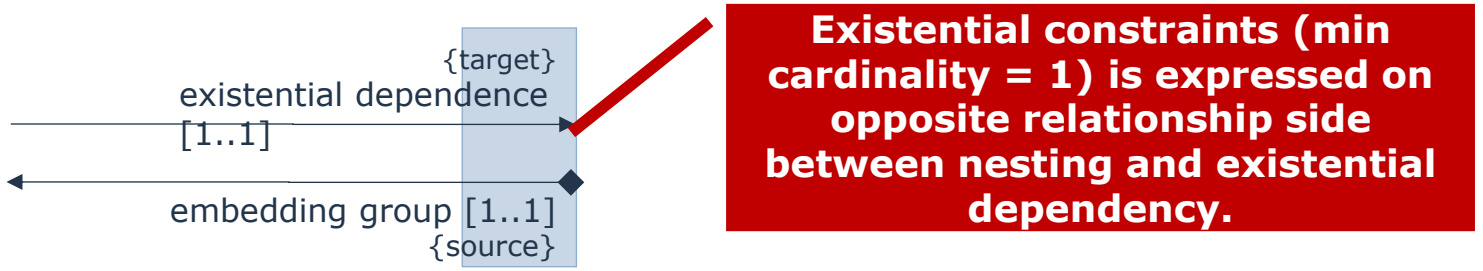
- Rule 3.3 – No disjoint nesting: nested-Entities must belong to one and only one type of grouping Entity.
  - A nested-Entity cannot be a nested in multiple kinds of grouping Entities unless these groups have a common super-type.
  - When there is a need for exclusive belonging to multiple groups there is presumably a missing abstraction between the two grouping Entities.

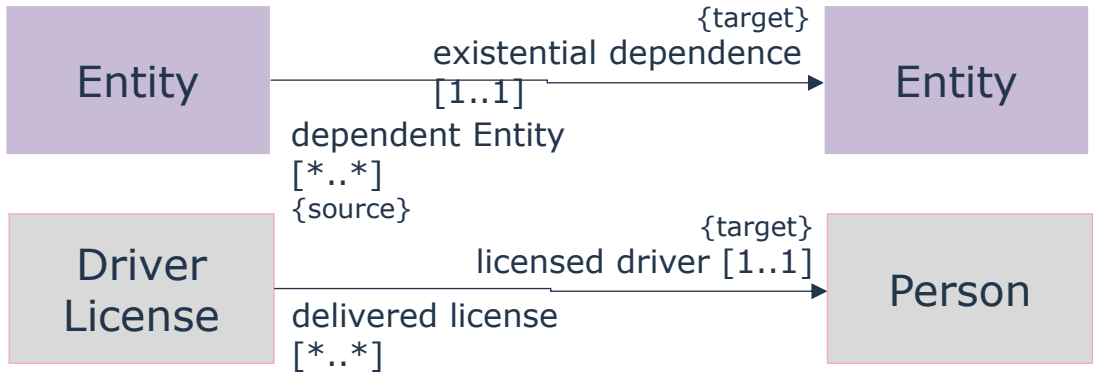
The Entity-Nesting decision-tree

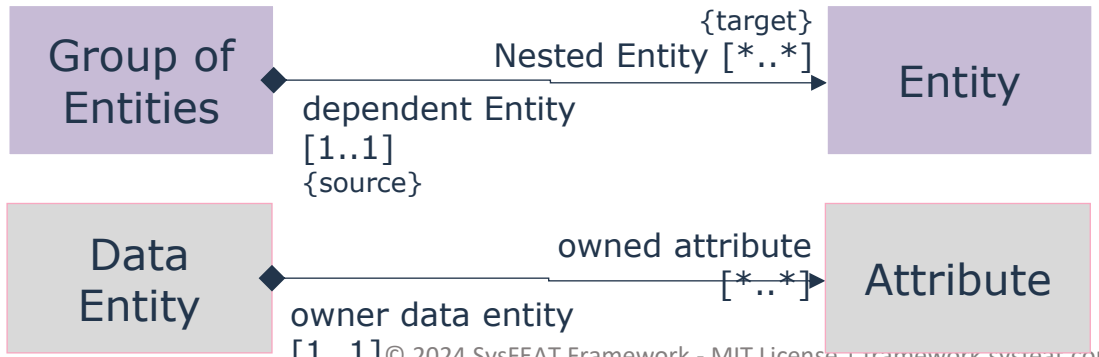


Nesting relationships versus existential dependencies

- Because *nesting-relationships* define the existence of their nested-Entities, they are often confused with *existential-dependencies* which state that one Entity needs to relate to another Entity to exist (A driver-license without an associated licensed-driver is meaningless).
- Both *nesting-relationship* and *existential-dependency* use the [1..1] cardinality to assert an Entity existence.
- The trick is that they go the opposite direction regarding the source of existence:
  - Existence depends on the grouping Entity in case of nesting.
  - Existence depends on the target Entity in case of existential dependency.

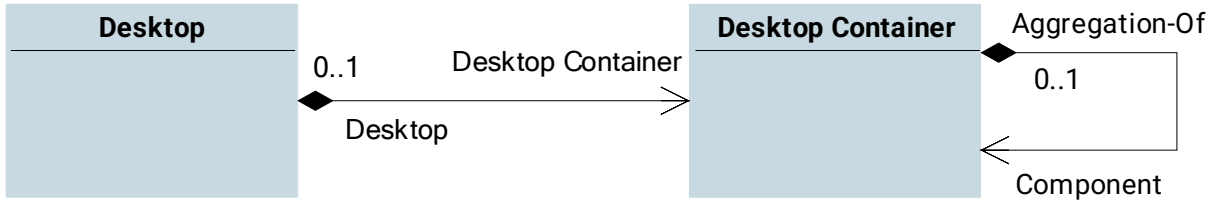


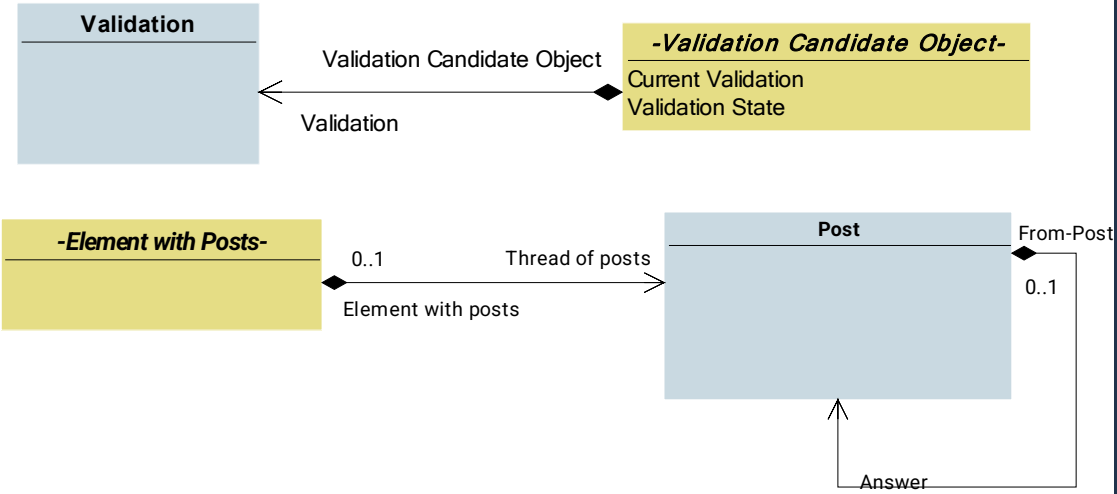


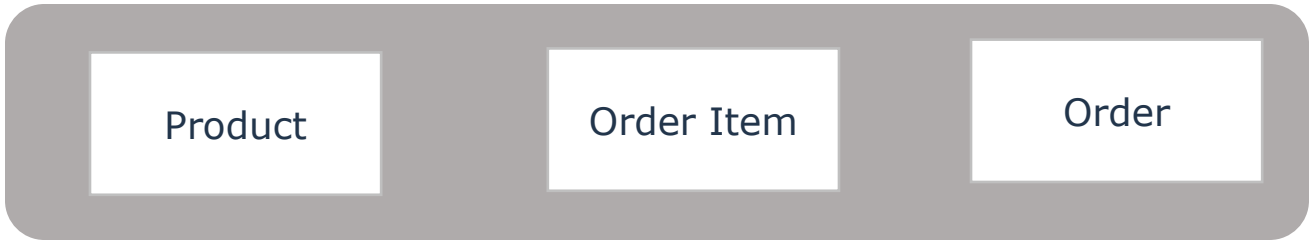


Exercise 3 – Nesting-relationships vs existential constraints

- Setup nesting-relationships for the following examples:
  - Products are parts of orders as order items.
  - Do desktops-container belong to desktops or parent-desktop-containers?
  - Is Validation part of “Validation Candidate Entity”?
  - Are Element-with-Posts owner of Posts?



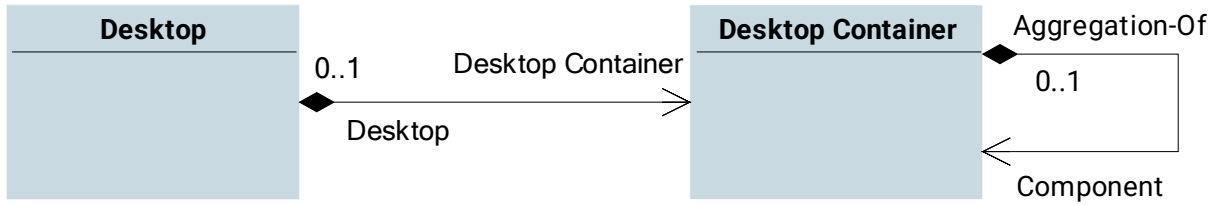


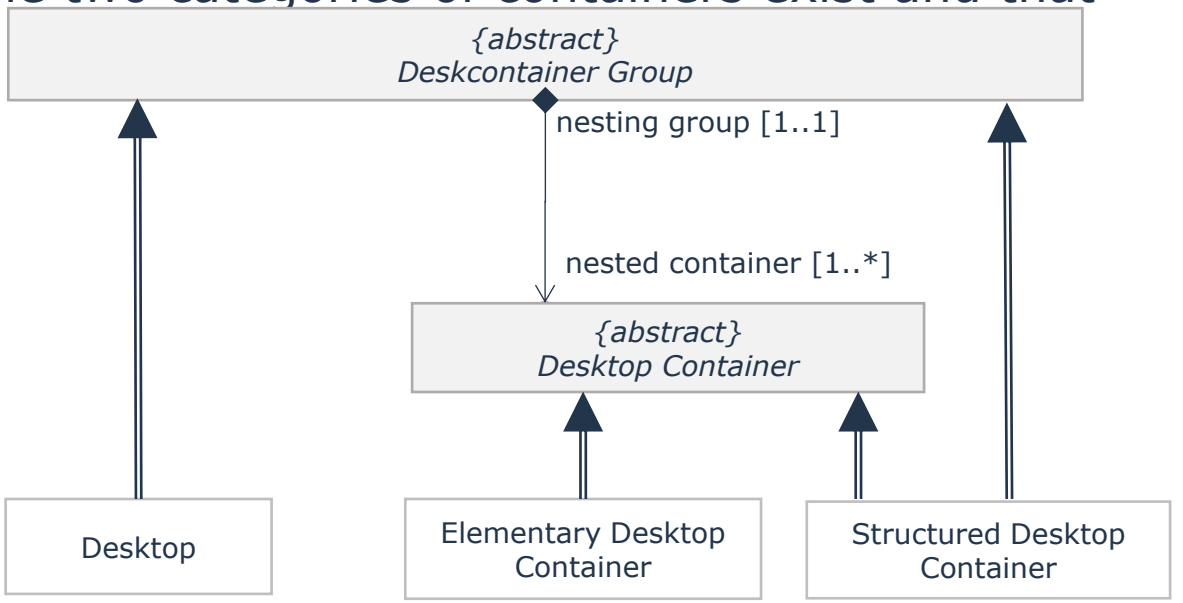


Exercise 3 – Solution for Desktop Meta-Model

- Application of Rule 3.3:
  - The *Desktop-Container* meta-class cannot have nesting-relationships with two different meta-classes (Desktop and itself).
- => There must be a common abstraction: Desk Container Group.
- => Two new categories of containers are discovered:
  - Structured containers that can be composed of other containers
  - Simple containers which host the final GUI items.

  An assessment in the code has confirmed that the two categories of containers exist and that they have distinct structures and behaviors.





Rule 3.2 – Entity ownership : a belonging context

- All Entities belong to one and only one group of Entity through a nesting relationship.

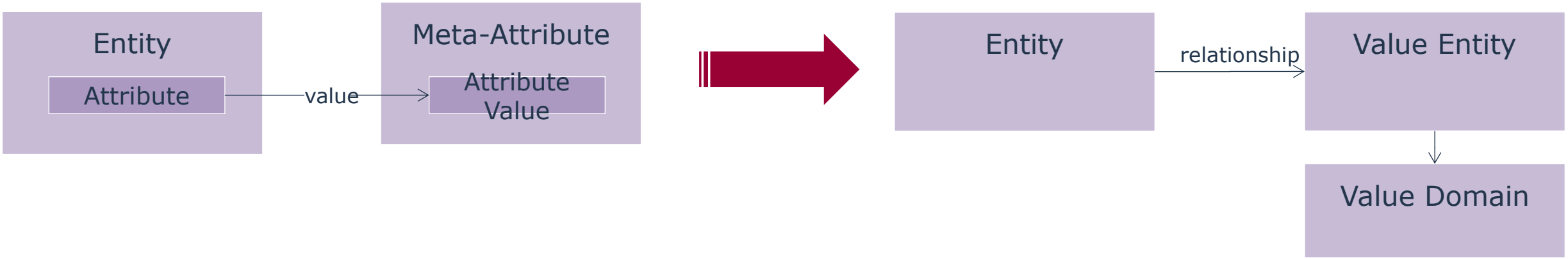
- Rational:
  - For an Entity to exist, it must exist somewhere: its belonging context.
  - This belonging context must be unique, otherwise there is a conflict for the existence of the Entity.
    - An attribute belongs to (exists in) one and only one class.
    - A class belongs to (exists in) one and only one package.
  - The “belonging context” becomes a repository management unit for all Entities that it owns/nests.

Rule 4

Attributes versus Relationships

Attributes versus Relationships

- Attributes should solely be used to store raw data values: text, numbers.
- Attributes on relationships should always be avoided (property graphs).
- Enumerated attribute values should be used exceptionally. In most cases, they are hidden relationships.
  - For instance, most “qualification” attributes such as “business value”, “technical efficiency” are not attributes but scores given by some governing entity when assessing the value of the considered asset.



Rule 5

Common concept versus Segregated concepts

Common concept versus segregated concepts

- The list below describes criteria used for hosting two distinct concepts into a single class or into multiple classes.
- Two concepts shall be represented by two distinct meta-classes when:
  1. They have distinct and exclusive set of characteristics (attributes and relationships).
  2. Their instances are primarily manipulated separately, while their union (the sum of both instances) is of secondary interest, or of no interest (true partition).
  3. There is no functional ambiguity, from a business-domain practice standpoint, to select between concept 1 or concept 2. If so, it shall be possible to requalify Entities from concept 1 to concept 2 and vice versa.

1. Example: shall “Department” and “Organizational Position” be hosted in a single “Org-Unit” meta-class?