Systemic Framework for Enterprise Architecture & Transformation

Semantic

Introduction

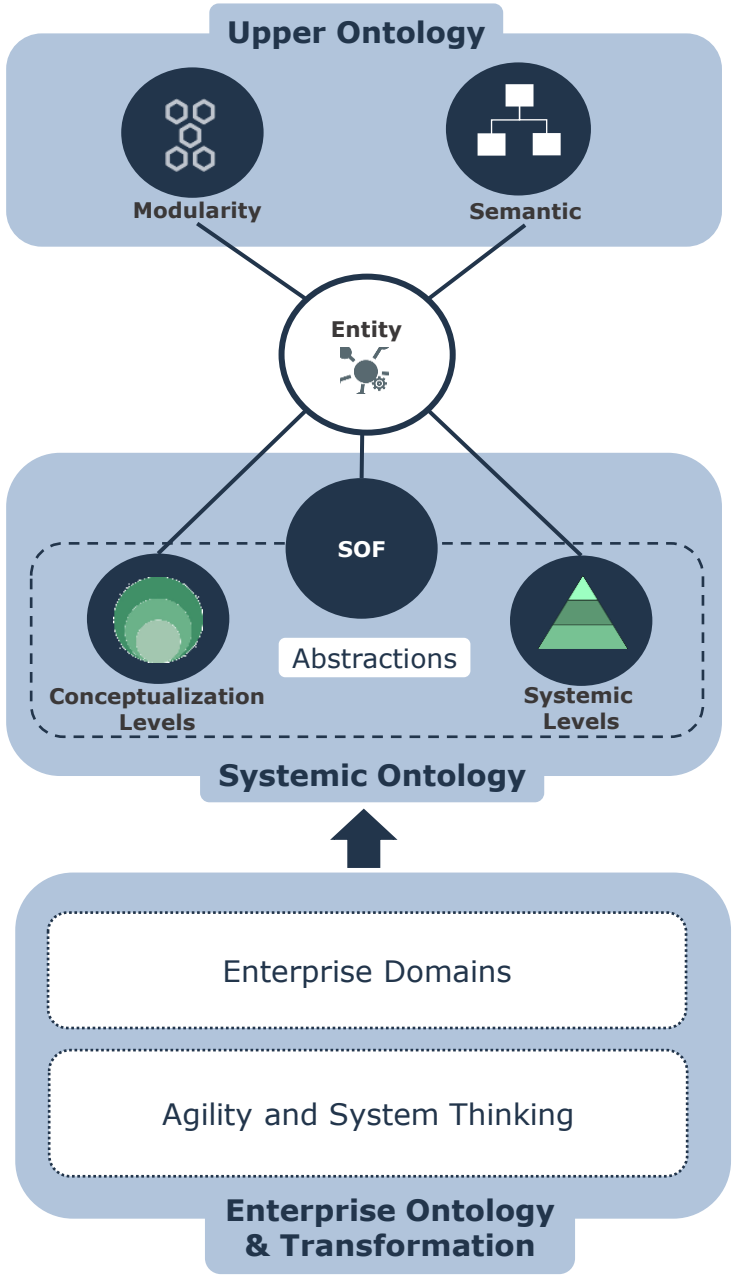
- This document is an integral component of the SysFEAT architectural framework. It provides foundations to address the challenges posed by Enterprise Architecture in the 21st century, which include :
  - Increasing complexity in system structures and behaviors.
  - Growing intricacy in architecture, management and governance of these systems.
  - The mission of the framework is to demystify these complexities, ensuring they are comprehensible to a broad audience, thereby facilitating the design and management of complex-systems across all scales, from micro-systems to enterprise level systems.

- Enterprise Modeling refers to the overarching language and conceptual framework used to describe, understand, and communicate the complex structures and dynamics of an enterprise.
- It integrates both the operating aspects of the enterprise (how it functions and interacts within its ecosystem), the transformational aspects (how it evolves and sustains over time through initiatives, asset management) and how these transformations are governed to ensure effectiveness, efficiency and reliability.

- The following slides present the foundations of enterprise modeling.

Foundations of enterprise modeling

- **Modularity** provides the syntax for building robust, manageable, and scalable architectures, based on the principles of composability and packaging.
- **Semantic** provides robust capabilities for classifying and composing entities, from time-bound entities (individuals) to families of concepts, enabling effective representation of meaning.
- The **Systemic Operating Framework (SOF)** serves as the overarching language that describes why and how a system operates and interacts within its ecosystems.
- **Abstractions** organizes systems and concepts in degree of abstractions, including systemic levels and conceptualization levels.
- **Enterprise Domains** formalize the various disciplines that make-up EA, ranging from enterprise road-mapping to System ArcDevOps.
- **Agility and System Thinking** ensure that the enterprise evolves and sustains over time through governed initiatives, architected for flexibility and responsiveness in complex and dynamic business environments.

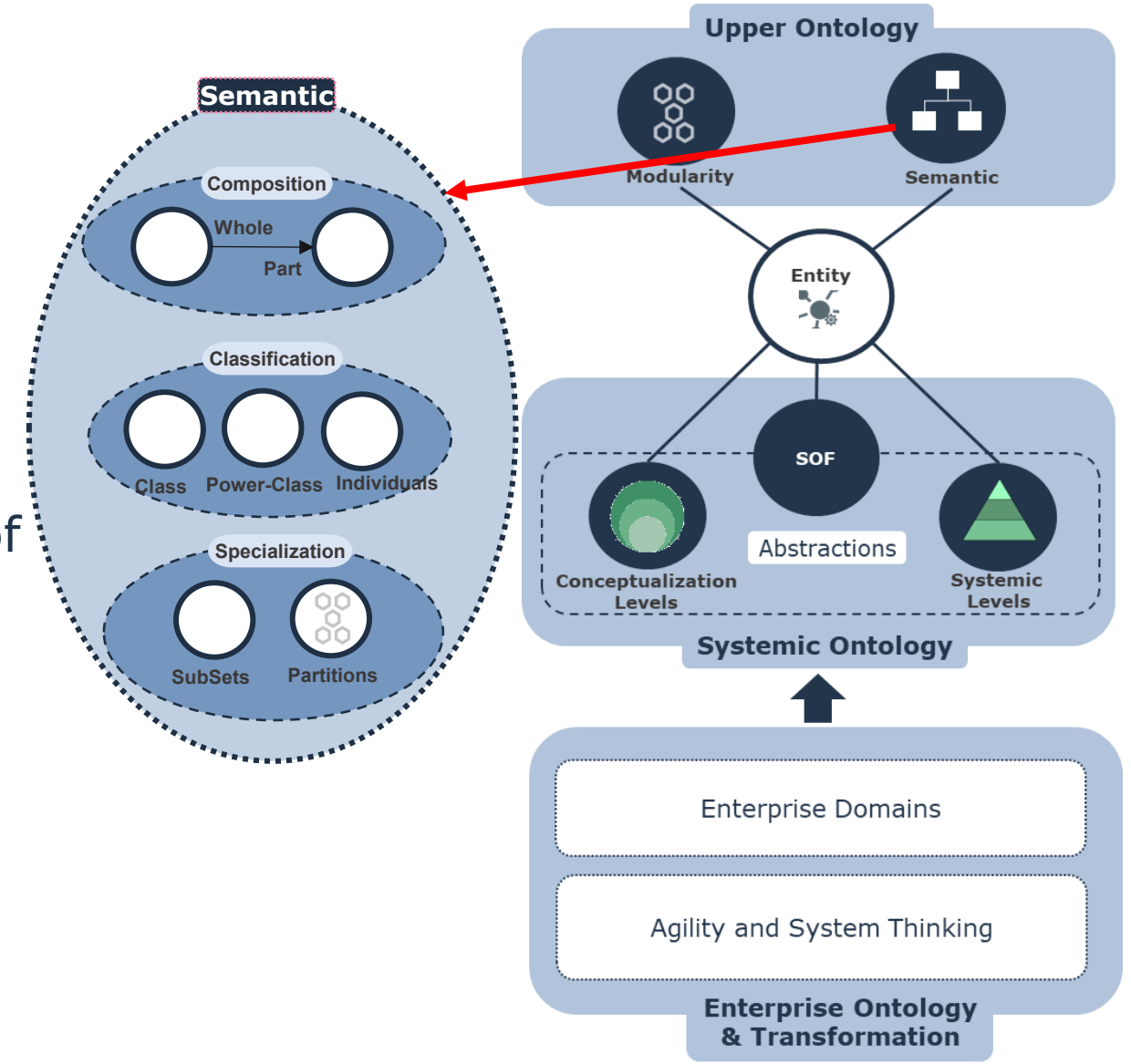


Semantic in the Architecture modeling landscape

- This document focuses on **semantic** which comprises two aspects:

- **Composition** is the ability to combine entities to form whole-part hierarchies.

- **Typology** is the ability to relate an entity to its categorical nature. It can take one of two forms:
  - Classification is the ability to organize elements in classes (instance to type relationship).
  - Specialization is the ability to form taxonomic hierarchies (sub-type to super-type relationship).



Composition

Composition – Whole-Part / Holonymy-Meronymy

- Composition is a whole-part relationship that describes how smaller entities – **parts (meronyms)** - combine to form a larger, more complex structure or system: the **whole (holonym)**.
- Composition follows the composability pattern meaning it can manifest in two distinct forms: elementary or aggregated.
  - **Elementary composition** establishes lightweight whole-part relationships between entities, where parts do not interact or interrelate with one another.
  - **Aggregated composition** offers internal structures to entities and thereby enables **Emergence**.
    - The whole exhibits properties or behaviors that arise from the interactions of its parts but are not reducible to the sum of the individual parts properties. This emergent characteristic distinguishes the whole as more than just an aggregation of its parts.
    - For example, a symphony orchestra is composed of various sections—strings, woodwinds, brass, and percussion—each made up of individual musicians. While each musician plays their part, the orchestra as a whole produces a harmonious performance that emerges from the coordinated interactions of its members.
- Aggregated composition is essential for modeling the structure of complex systems, such as IT systems, hardware systems, and organizations.

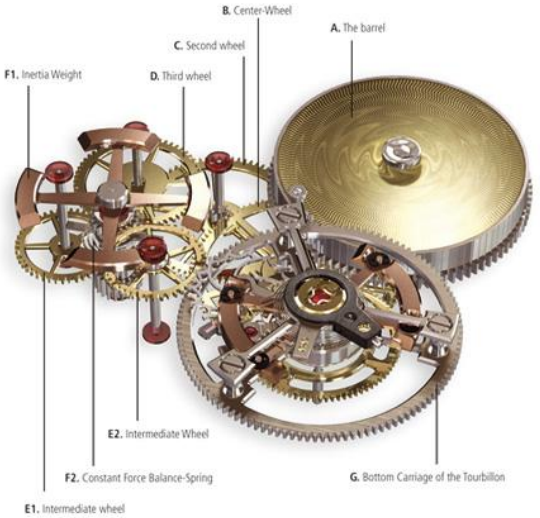
Herbert Simon and near-composition (Aggregated Composition)

- The parable of the two watchmakers was introduced by Nobel Prize winner Herbert Simon to describe the complex relationship of sub-systems and their larger wholes.

There once were two watchmakers, named Hora and Tempus, who made very fine watches. The phones in their workshops rang frequently and new customers were constantly calling them. However, Hora prospered while Tempus became poorer and poorer. In the end, Tempus lost his shop. What was the reason behind this?

The watches consisted of about 1000 parts each. The watches that Tempus made were designed such that, when he had to put down a partly assembled watch, it immediately fell into pieces and had to be reassembled from the basic elements.

Hora had designed his watches so that he could put together sub-assemblies of about ten components each, and each sub-assembly could be put down without falling apart. Ten of these subassemblies could be put together to make a larger sub-assembly, and ten of the larger sub-assemblies constituted the whole watch.

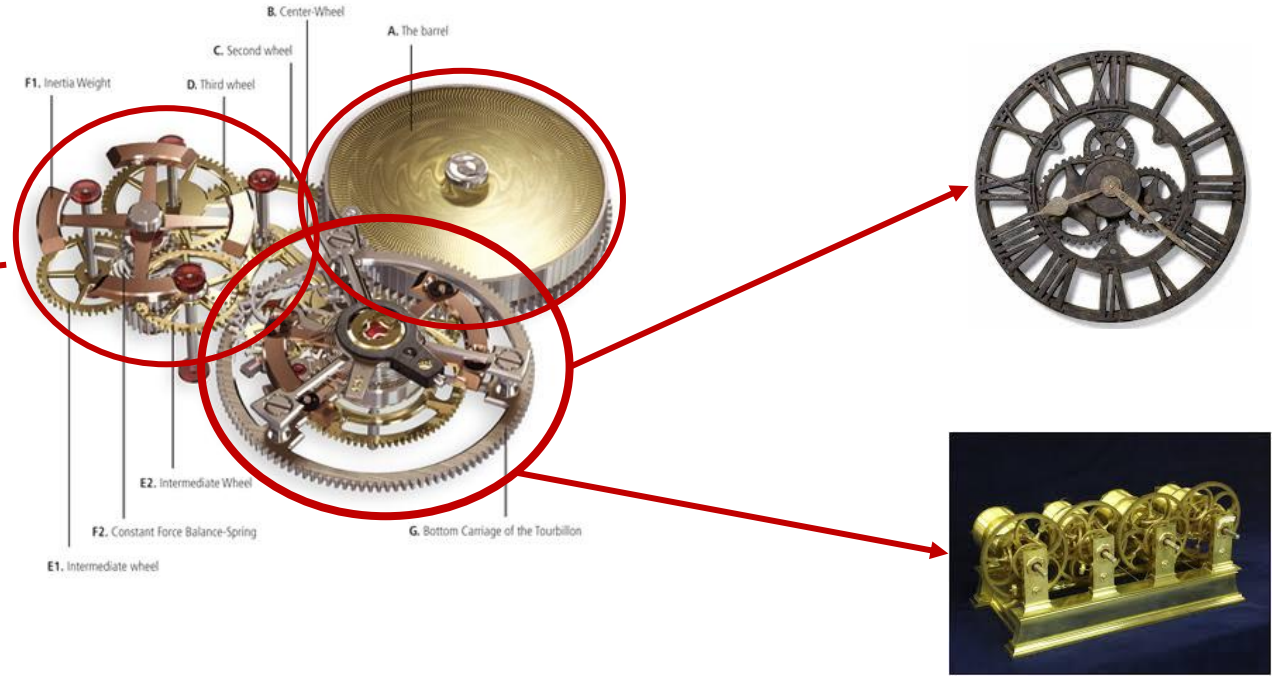


Reference : Herbert Simon and near-decomposability + the parable of the two watchmakers.

Aggregated Composition benefits

- Modularity is the primary benefit of aggregated composition, not reuse.
- The trick is to be able to handle complexity by delimiting autonomous building blocks that can be assembled and updated independently (like Hora).
  - This allows to decouple the various processes and life cycles of each piece.
  - Modular systems have the following properties: Maintainability, Sustainability, Repair Speed.

Managing complexity through composability:
=> modular building-blocks



Aggregated Composition - Benefits & Reuse Challenges

- Reuse can also be pursued, keeping in mind that *reuse* always comes at the price of *standardization* and increased *dependencies*.
  - Pure reuse leads to building similar products.
  - When businesses wish to differentiate from one another, they setup an integration process to assemble standard parts in a unique manner.
  - Smartphones are the typical illustration: they are built from standard parts which are then all hardwired to make-up unique phones.
  - For achieving "reuse + differentiation", building-blocks must have additional composability and integration properties that enable platform-based approaches:
    - Modifiability, Configurability, Adaptability, Extensibility

FORD Model T: complete standardization & reuse with no differentiation
Henry Ford: "*Any customer can have a car painted any color that he wants, so long as it is black*."

Reuse with full hardwired integration: once assembled the system is no more modular. Smartphones are a typical example.

Typology

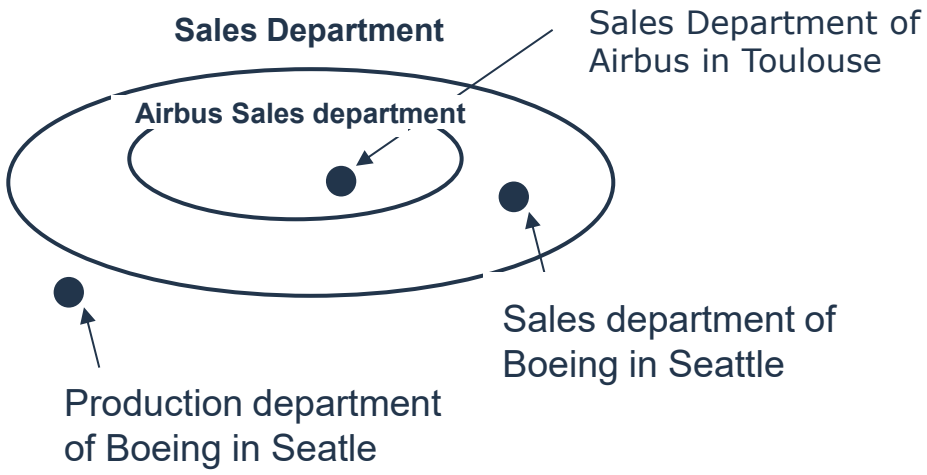
An introduction

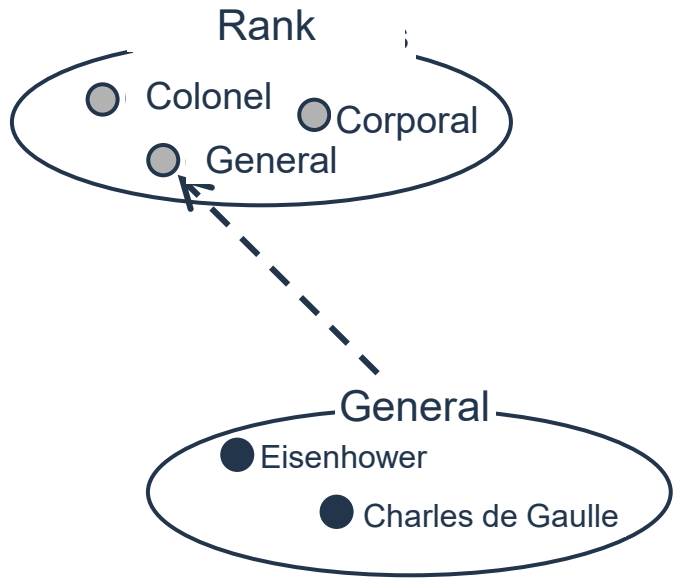
Typology: categorizing entities

- Typology defines how a conceptual entity relates to its categorical nature: the intrinsic properties or criteria that determine its membership within a class.
- Typology governs two distinct categorization mechanisms:
  - Classification: assigning an entity to a predefined class based on shared characteristics (e.g., grouping by common traits).
  - Specialization: refining a class into a subclass with narrower criteria (e.g., inheritance hierarchies or subtype relationships).
- While classification focuses on grouping entities into classes, specialization emphasizes hierarchical refinement of classes themselves. Both mechanisms operate under the umbrella of typology, which formalizes how entities are systematically categorized.
- Note that classification can also apply to classes themselves: there are classes of class. This leads to a hierarchy of classification, presented in the next slides.

Levels of classification: from Individuals to Class of Classes

- Instance->Class is a relationship between things and their classification: things are instances of classes which can also be formulated as “things are classified by classes”.
- A thing is either an individual thing or a class of things.
  - “Sales Departments” is a class of organizational departments.
  - “Airbus France Sales Department” is an individual which is an instance of the “Sales Departments” class.
- There are levels of classifications: “General” is an instance of Rank, which is a class of class





Source: BORO methodology:
https://en.wikipedia.org/wiki/BORO

Levels of classification: from Individuals to Class of Classes

Dog Breed
the class of all class of dogs

instance of

The concept of Dalmatians
(the class of all Dalmatians)

instance of

Spot
my dog born on July 1st 2022

Class of Class of individual

- Dog Breeds
- Business Line "BL"
- Functionality Meta-Class
- Business Function Meta-Class
- Application Meta-Class

instance of

Class of individual

- Functionality "F"
- Business Function "BF"

Abstract Class

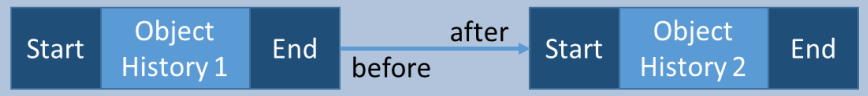
Concrete Classes : have direct individual instances

- Dogs / Dalmatians
- Application "A" supporting Functionality "F" and classified by "BL"
- Business Process "P" performed by "Department Type "D".

instance of

Individual

Only individuals have time and valued physical measures



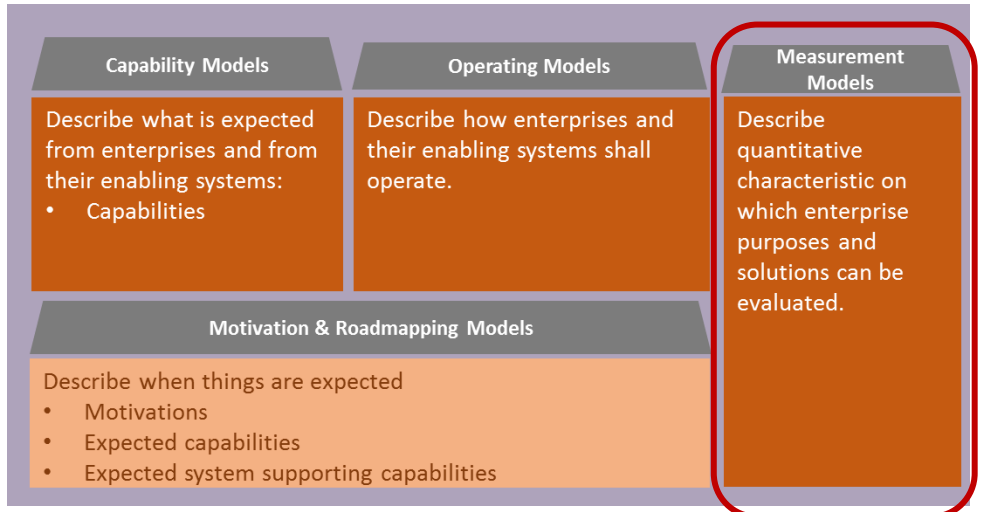
- Mirza / Spot
- Application "A" on John's computer
- Enterprise "Insur++"
- Portfolio "PI" of applications of "Insurr ++"
- Object Life Cycle of "Application A" in Portfolio "PI"

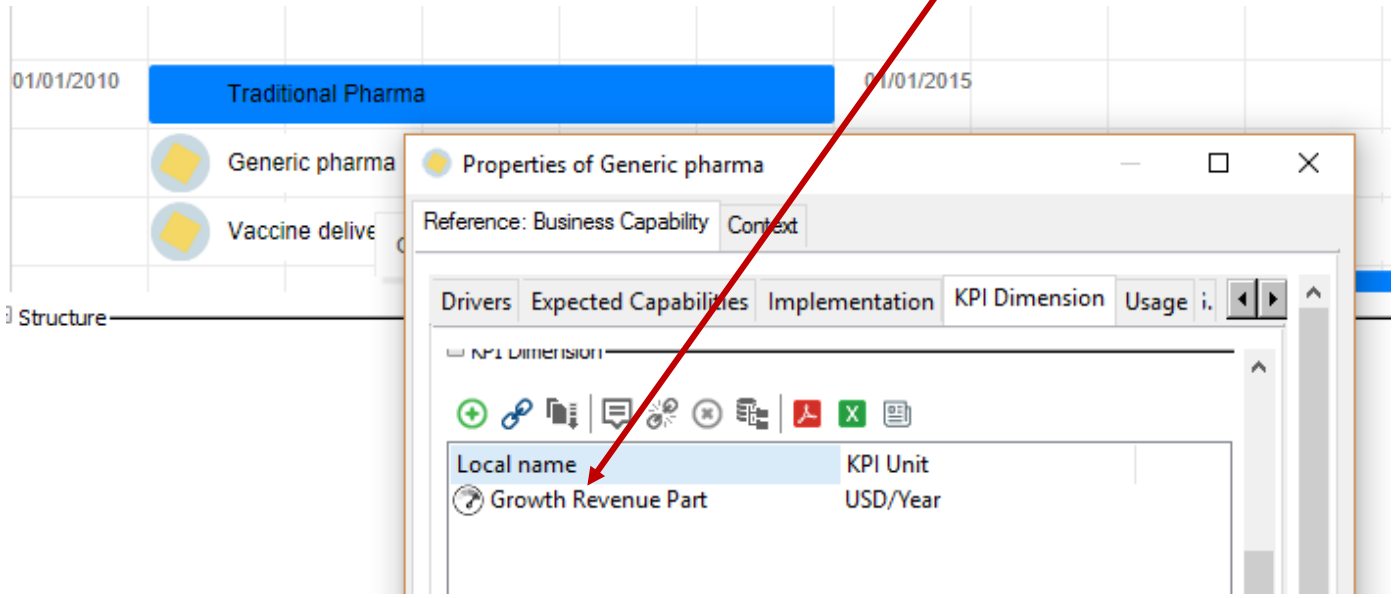
Qualification Classifications

Properties: Measure & Qualification

Measurable Properties

- Measurable Properties express a type of measure: % of growth-revenue, Temperature in Celsius, Time to deliver in minutes, Costs, etc.
- Qualifying values express a possible value for a measurable property: %40 of growth revenue in $,15 Celsius, 10 minutes, etc.

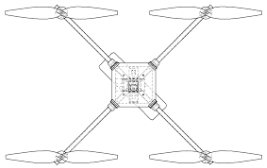




Measure, Classification & Individuals

the class of means of transport

instance of



The concept of Drone
(the class of all Drones)

instance of

Enterprise Pizza+ with
enabled Drone Transport
Capability in 2020

Class of Class of individual

- Means of transport

instance of

Class of individual

- Capability = Deliver Pizza
  - Measure Category = Time to Deliver

Abstract Class

- System = Deliver Pizza
  - Measure Category = Time to Deliver

Concrete Classes : have direct individual instances

instance of

Individual: Enterprise Pizza+

- Exhibited Capability = Deliver Organic Pizza
  - Time to Deliver = 30 min

Stage 1-

Stage 2 - 2020

- Exhibited Capability = Deliver Organic Pizza
  - Time to Deliver = 10 min

Only individuals have time and valued physical measures

Meta-Modeling

Partitioning and level of conceptualization

Power-typing: Multi-Level Conceptual Modeling

- Multi-level of classification leads to a flexible hiearchy of classes that allows the creation of open and adaptable metamodels.

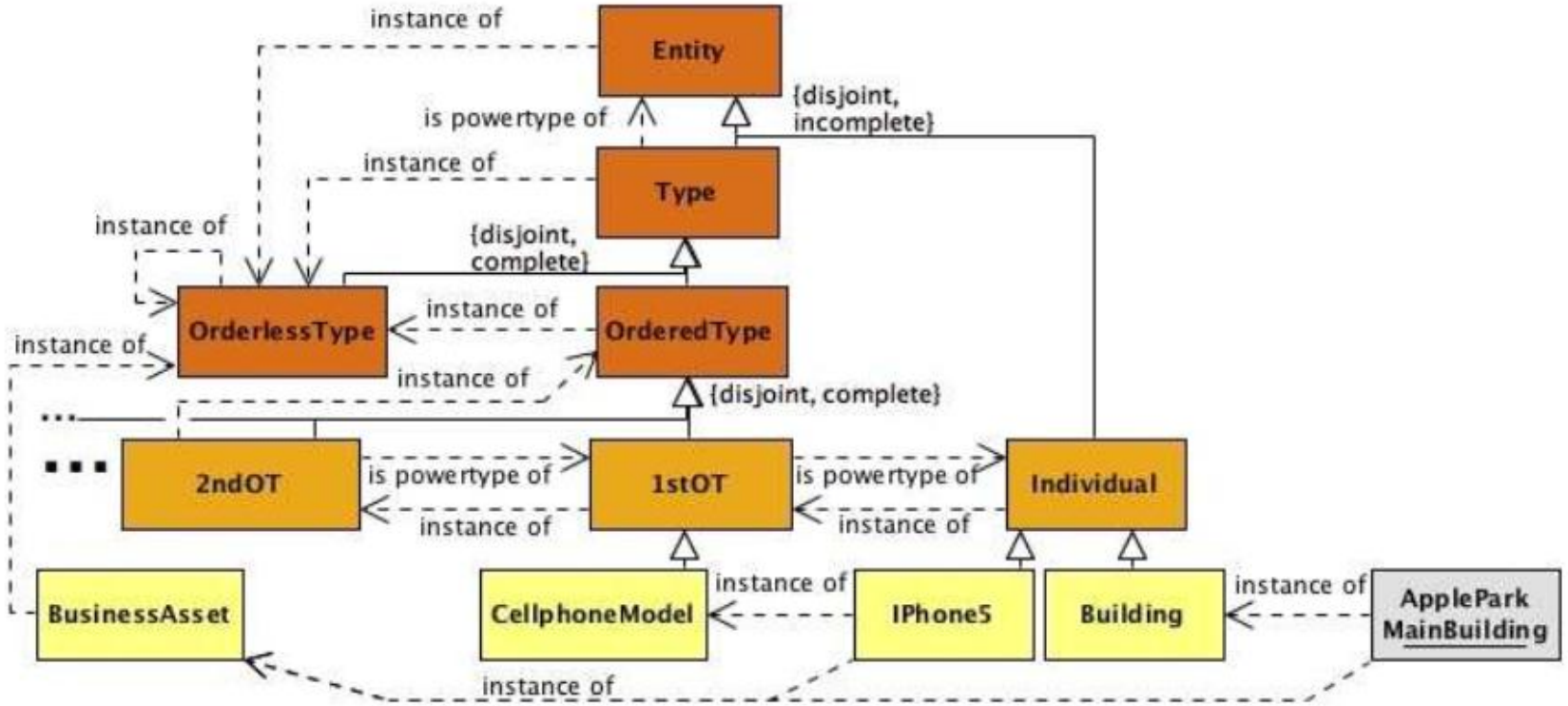


Fig. 5. MLT* basic scheme extended by a domain example.


Work In Progress