Basics of the Theory of Notions

Abstract:

An important industrial requirement is the ability to share computer interpretable product and process information between different computer applications, disciplines and organizations in a supply network. Many national and international standards aim at fulfilling this requirement. But the exchange and sharing of data across disciplines is still troublesome.

The interoperability problem is partially caused by the application of object- or class-based data modelling methods. Attempts to solve the problem through 'neutral models' such as neutral schemas or (upper) ontologies shift the problem from one place to another. Moreover, the concepts that are represented by object-types and classes are weakly defined in most standards and do not form a solid ground for the association of meaning to the data that we exchange or share.

In this paper we explain how human knowledge is formed, and how this results in different ideas about the world in which we live. The roots of human knowledge are formed by Notions, that will be defined here as sensory experiences that result in awareness. Notions can be considered as the elementary particles of knowledge.

Notion Theory rests on three pillars: (1) the theory of definition published around 300BC by Aristotle, (2) empirical science, such as shaped by the works of Galilei and Descartes in the 17th century, and (3) the theory of Cognition published in the late 20th century by psychologist Ulrich Neisser.

Notion Theory can be used for a more precise definition of terms in ontologies or conceptual schemas. But we will show also that the exchange or sharing of data by means of Notions is far more efficient. Notion Frames enable us to model different perceptions of a subject of discourse, thus preserving meaning and intent of the data. By making semantic differences and similarities explicit, it is now possible to model the conceptual union of data that is required for the specification of distributed data systems.

We will explain Notion Theory in the context of building and construction. But the theory itself is generic and should be applicable in other industrial sectors as well.

1. Introduction

1.1. Conceptual Modelling today

Data processing systems use and produce data in digital form. Data can be useful if it is clear what they represent. For that purpose, programmers define meaning by means of conceptual models, also known as conceptual schemas or ontologies. There are many languages and methods available to define these, but they are all based on more or less the same ideas.

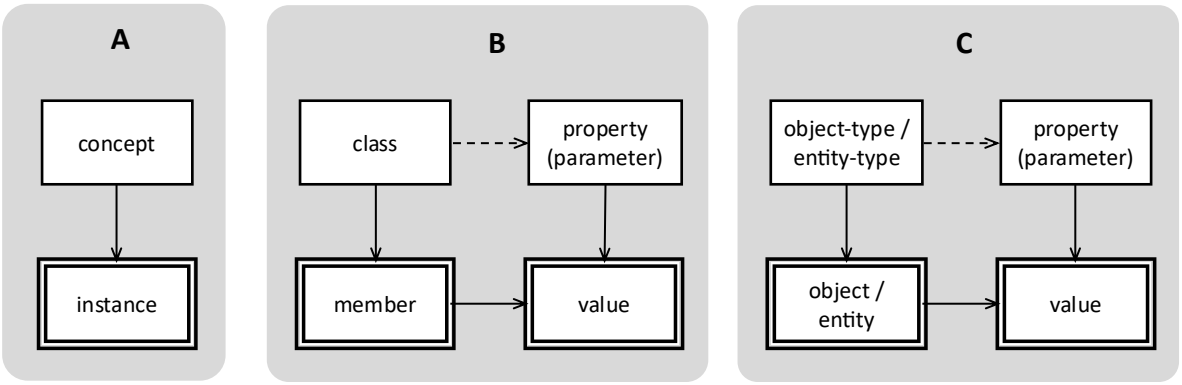


Fig 1 a-c. The rectangles with a single border represent memory cells, the ones with a double border the content of cells.

It starts with the fact that computers have memory cells which contain electric charges. In a binary expression these are considered to be 'on' or 'off', representing 'true' or 'false'. The cell – or group of cells - represents a concept, the content represents an instance of that concept, such as a character, a number, or the address of another memory cell. See fig 1A (left). Often, the concept is considered to represent a class. An instance represents then an element or individual member of that class. This individual member may have a property that is represented as a concept by another memory cell, and of which the instance represents the property value; see fig 1B (middle). Instead of classes, the memory cells may also be said to represent object-types of entity-types, in which case the instances represent objects or entities, respectively (fig 1C right).

The instance of a class may represent an element but instead also refer to a subclass, see figure 1D below. Depending on the modelling language applied, subclasses may also be called subtypes. In logic, subclasses are usually considered to be subsets of the superclass (the class that is on the top of the hierarchy).

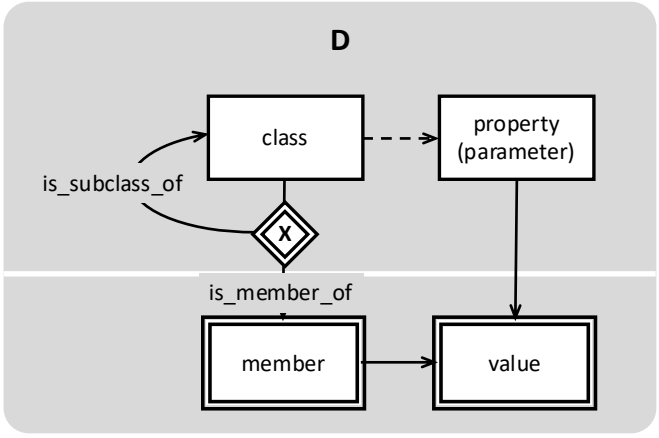


Fig. 1d.

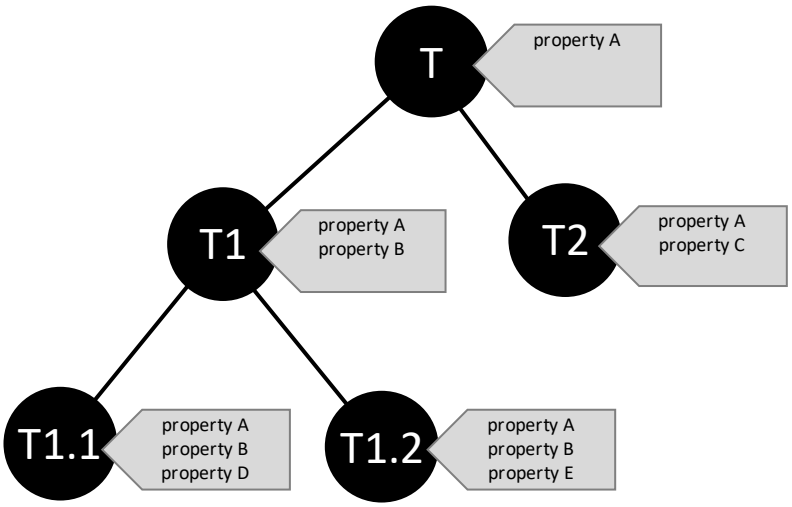


Fig. 2: When the upper part of the schema in 1d is instantiated, the result is a class-hierarchy such as shown here. Members of a class are characterized by common properties. All members of class T have property A. All members of subclass T1 have properties A and B, and so on. The rule that subclasses have the same property as the superclass (T in this example) is called inheritance.

Members of a class are characterized by common properties. In the example shown in figure 2, all members of class T have property A. All members of subclass T1 have properties A and B, and members of T1.2 have properties A, B and E. The rule that subclasses have the same property as their superclass (T in this example) is called inheritance.

The origin of this rule stems from historic attempts to classify things in nature. The Linnaeus method is based on the arrangement of species according to shared characteristics. As an example we give the classification of animals:

Table 1

- *Animal: a complex multi-cellular organism that does not produce its own food.*

There are seven subclasses of animals, amongst which:

- *Chordata, or vertebrates: animals that develop a cartilaginous skeletal rod that supports the body and can become a spine.*

A subclass of chordata is formed by

- *mammalia (mammals): warm-blooded four-legged animals that breath with lungs and give birth to living young.*

A subclass of mammalia is formed by

- *primates: mammals with prehensile hands and feet, commonly with opposable thumbs.*

And so on.

In this example, primates are mammals. Hence, they do not only have prehensile hands and feet, but are also warm-blooded four-legged animals that breath with lungs, give birth to living young, develop a cartilaginous skeletal rod, are multi-cellular organisms and do not produce their own food.

1.2. Informal modelling of meaning

The meaning of the concepts, classes, object-types, entity-types of properties is usually represented by a text – often a single word; see figure 2. What these words or texts actually mean is assumed to be clear for the programmer or the user. This part of the specification is informal in most modelling approaches. And this is where the problem of interoperability starts.

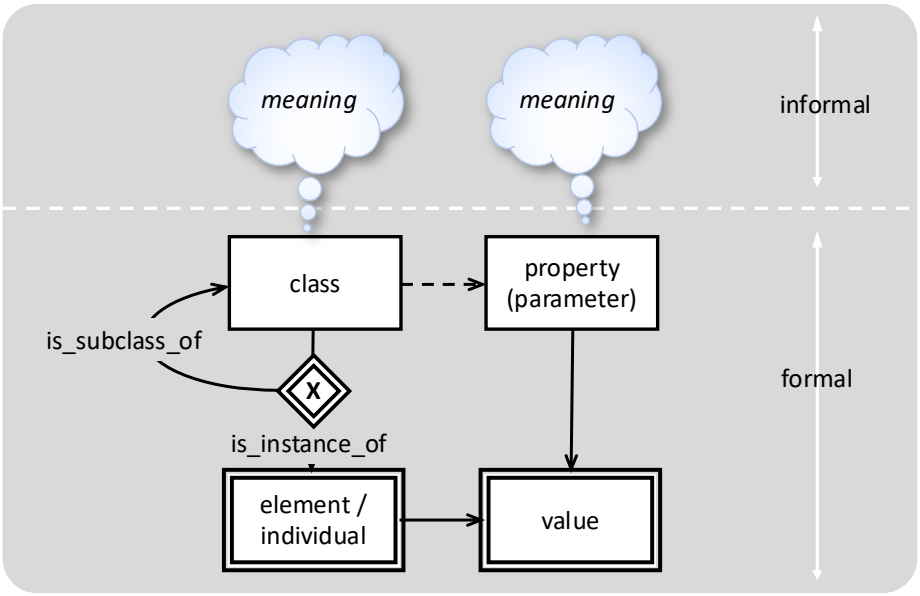


Fig.3.

Let us take a concrete example: the IFC (Industry Foundation Classes) standard (or: ISO 16739) defines the concept "Wall" as follows:

The wall represents a vertical construction that may bound or subdivides spaces. Walls are usually vertical, or nearly vertical, planar elements, often designed to bear structural loads. A wall is however not required to be load bearing.

- The first sentence suggests that Wall is a subtype of construction, the second that it is a subtype of element. However, in the IFC schema, ifcWall is a subtype of ifcBuiltElement, which is on its turn a subtype of ifcElement.
- The first sentence states that walls are vertical, the second that they are nearly vertical. What is meant with "nearly"?
- A wall may bound or subdivide spaces. That suggests that there may also be walls that do not bound or subdivide spaces. Hence, this phrase does not define a property of the wall.
- The same can be said about "load bearing": whether a wall bears load or not does not determine the meaning of wall.
- What is meant with "bounding" or "subdividing? Does that apply to physical access, climate control, or fire safety?
- Are non-planar walls no walls? So, what are they?
- Is a façade also a Wall?

ISO 6707 defines "Wall" as: *vertical construction usually in masonry or in concrete which bounds or subdivides a construction works and fulfils a load bearing or retaining function.*

- In this definition, a wall *does* bound or subdivide. But it doesn't bound or subdivide spaces: it bounds or subdivides a construction works.
- The explanation "usually in masonry or in concrete" does not add meaning to the definition. What should be concluded from the fact that materials such as gypsum, lime sand, glass or wood, which are quite common in modern buildings, are not mentioned?
- In this definition, walls are either load bearing or retaining.

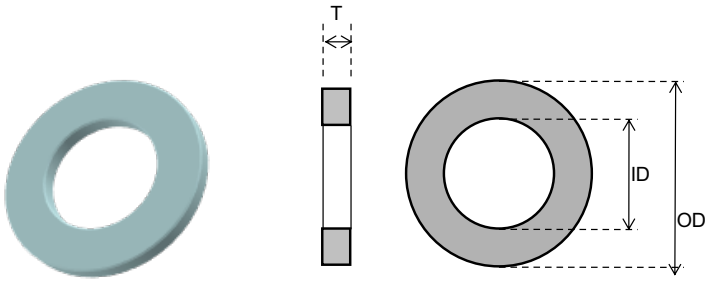
Dutch legislation, formulated as the 'Bouwbesluit', does not have regulations for Walls. In stead it uses the term "Space dividing construction". This is a broader definition because it is not limited to vertical space dividers. But it does make a distinction between internal and external space dividing constructions.

Apart from the fact that current definitions are not entirely consistent and leave room for interpretation, it must be noted that they are also expressed in informal English. The text is not meaningful for – nor interpretable by - a computer. It is therefore possible that properties that are associated with an object classified as "wall", such as a geometric model, may conflict with the definition.

1.3. Implicit or explicit data?

Another important notion is that classes are based on property values which are not part of the data specification itself. It is 'implicit' data.

We explain this with an example. Figure 4a shows an annulus shaped rubber part that can be used as a seal. There are different ways to model this object: it can be considered as an instance of entity-type 'seal', as an instance of 'annulus' or as an instance of 'rubber_part'. It depends on the preference of the schema-developer which aspect is used for classification.



Three incompatible ways to model a rubber, annulus-shaped seal:

```
ENTITY rubber_part
  function: seal
  shape: annulus(T,ID,OD)
END
```

```
ENTITY seal
  shape: annulus(T,ID,OD)
  material: rubber
END
```

```
ENTITY ring
  function: seal
  material: rubber
  shape: annulus(T,ID,OD)
END
```

Figure 4a. There are many ways to model a rubber, annulus-shaped seal. This example in the Express language shows just three.

In figure 4a, the property-values are made explicit. But in a real schema, property-values are instances of a variable. Examples of variables for this case are material, shape and function. The allowed values for each variable (for example, the fact that a valid value for ‘material’ is rubber’) are defined elsewhere in a schema. The result is shown in figure 4b.

```
TYPE
  function = ENUMERATION_OF ( spring, spacer, seal, ...)
  material = ENUMERATION_OF ( rubber, fiber, neoprene, teflon, steel, ...)
  shape    = ENUMERATION_OF ( cube, piramid, cilinder, torus, annulus, ...)
END
```

```
ENTITY seal
  shape: shape
  material: material
END
```

```
ENTITY ring
  function: function
  material: material
  shape: shape
END
```

```
ENTITY rubber_part
  function: function
  shape: shape
END
```

Figure 4b. In a conceptual schema, properties are defined as variables, thus hiding property-values. The variables 'function', ‘material’ and 'shape' are defined in this example as enumeration types. The entity-types 'seal' and ‘ring’ do not reveal that one or more of its instances are made of rubber. Their instances can be made of any material. Hence, it is not possible to see at the level of a schema that some of their instances describe one and the same object.

Property-values are invisible at the level of a schema. Hence, it is impossible to see that an object that is represented by each of the three entity-types may actually be the same object.

The class-hierarchy itself, however, is based on property-values! “Seal”, “Ring” and “rubber_part” are Entity-type “seal” is based on classification by function, entity-type “ring” on classification by shape, and entity-type “rubber_part” on classification by material. Which criterion is used for classification depends on the preference of the model developer.

Suppose that the example represents the 'views' of three different computer applications on a given subject. Even though these applications may contain exactly the same information about this object, it will not be possible to communicate it because their schemas are incompatible.

2. Rethinking the expression of knowledge

2.1. Design is about the union of knowledge

The purpose of data sharing in construction is to develop a common understanding of an artefact that does not yet exist in physical reality. There are many domain experts involved in this process, each adding a piece of knowledge to the common specification. The real goal is unify all that knowledge into a single model that answers all possible questions: does it meet its functional requirements, can it withstand the forces of nature, is it possible to build it, what do we need as equipment and resources, what is the environmental impact, what will be the return on investment?

Data exchange between experts is just a tiny part of that process. Data exchange can be troublesome because every expert has another view of the building. Expert A uses terminology that may not be understood by expert B, and vice versa. Hence, data exchange

supports only the intersection of knowledge (A ∩ B). But what we actually need in design is the unification of knowledge (A ∪ B). This is only possible with a building data model that also unifies different views.

2.2. About cognition

For a better understanding of this subject it is necessary to discuss the origin and nature of human knowledge. This is where we enter the scientific domain of cognitive psychology [a7].

Immediately after its birth, a baby has very little knowledge about the world that it has just entered. Its behaviour is determined by what is 'pre-programmed' in its genes. Soon after, it starts to explore and discover the world that surrounds it.

The senses play an essential role in this process. Sensory impressions are memorized and compared with new sensory impressions. Returning impressions lead to *recognition* and to the development of abstract structures in the mind that are called *perceptive schemas* in psychology. In order to avoid confusion with the term 'schema' in information technology, the term *perceptive frame* will be used here.

'Knowing' is thus the association of new sensory experiences with memorized experiences, which provide context and meaning. While the senses provide us with a huge amount of information, only a tiny part of that is relevant for us and results in awareness. Everything else will be forgotten. A sensory experience that results in awareness will be called a *Notion* in this paper.

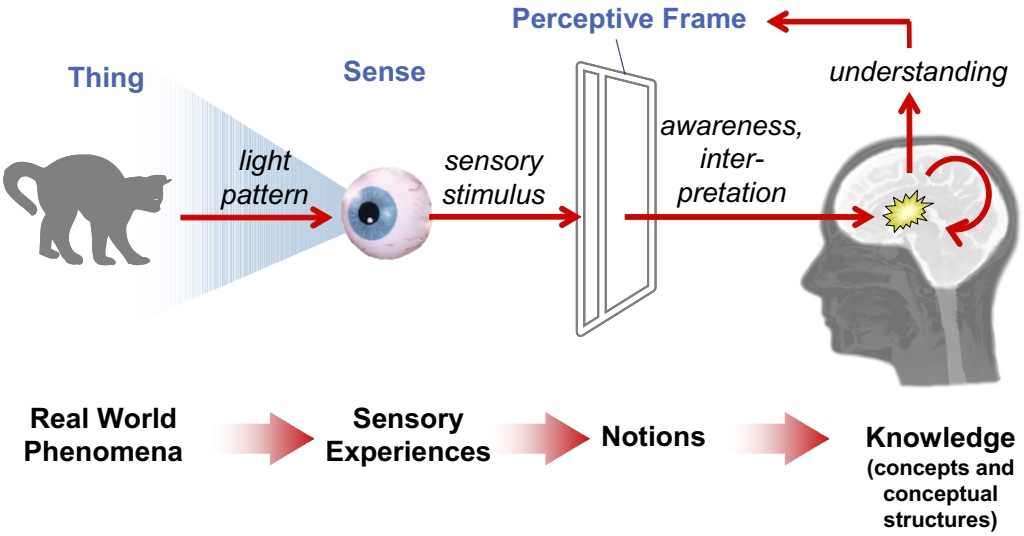


Figure 6. Simplified model of the process of human cognition.

Recurrent experiences that are memorized may become independent sources of information for the human mind. They become *concepts*: abstractions of real world experiences. According to Smith [a10], the primary function of concepts is to promote 'cognitive economy' (i.e. to understand the complex world around us with as few concepts as possible).

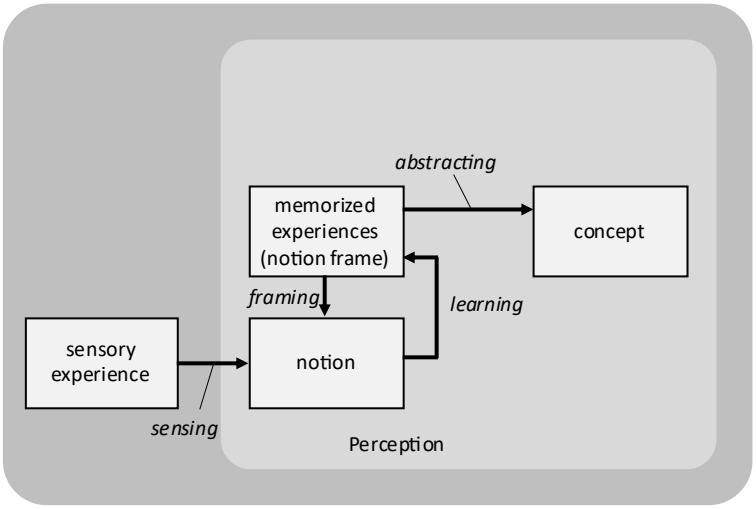


Fig. 7. Simplified depiction of the cognitive process. Memorized experiences add context and thus meaning to new sensory experiences. The combination is called a notion. This process may also be called recognition. The new experience is added to the already existing frame. This is a form of learning by experience. Concepts are formed by abstractions and combinations of multiple Notion Frames.

The human brain works the same for concepts that result from sensed experiences, and concepts that result from other concepts. The brain is therefore able to develop cognitive structures at ever higher levels of abstraction. Also in this process of abstraction, differentiation of concepts is based on notions. But in this case notions are abstract; they cannot be directly referred to observable experiences, but at best indirectly.

Mathematical concepts are examples of the latter ones. We cannot see 3, but we can see three apples and three oranges. We cannot see 'number', but we have developed notion frames for 1, 2, 3, 4, etcetera; what they in common is that discrete things can be counted. And if things can be counted, we are also able to add, subtract or multiply them.

The human mind is also capable to combine concepts into an idea of a non-existing reality: *imagination*. Imagination may lead to the creation of a new reality: *creativity*. And this is what designers do in industry. They imagine new products and processes that change existing physical reality.

2.3. Cognition in science

The scientific method is based on the same process as depicted here, but with three main differences:

a) The means of sensing in science are improved by all kinds of instruments, ranging from microscopes to telescopes, as well as electronic sensors that permit us to see what is normally hidden for the human senses;
b) The systematic documentation of scientific observations and experiments;
c) Standardized procedures for measuring and interpretation of observations.

Through the standardization of measurement procedures, the notions that result from scientific research are also fairly consistent. This doesn't mean that they are free from perception: scientific knowledge improves continuously, also because ever better instruments become available. Major scientific domains such as physics, chemistry, biology and earth sciences alle have different 'views' on reality. And within each scientific domain there are also more specific views, such as the mechanical view, field theory, relativity and quantum

physics. Nonetheless, the scientific method may provide a firm basis for the definition of concepts.

2.4. Communication

By giving names to concepts, humans are able to communicate with each other. The name is a kind of sign; this is why the science of giving names to concepts is also called semantics (sema = sign in Greek).

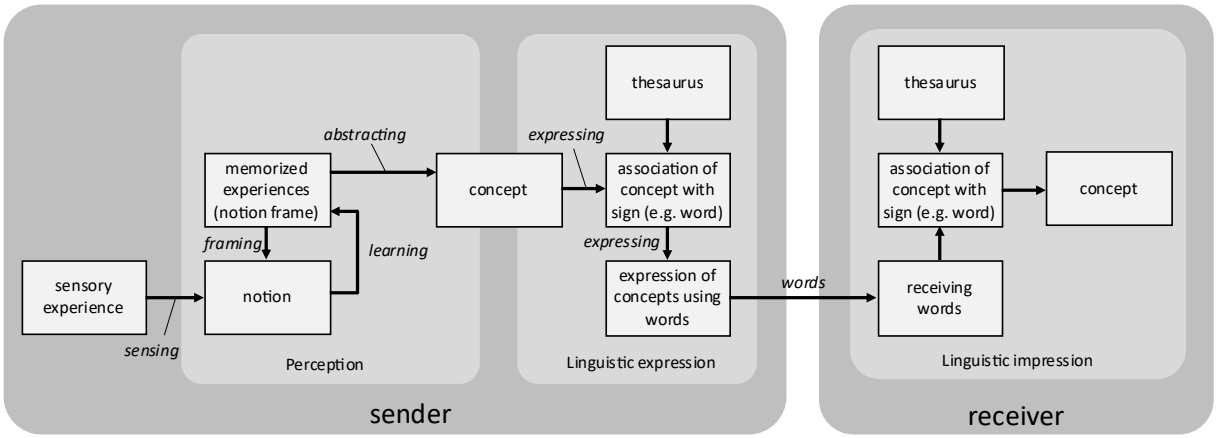


Fig 8.

2.5. Plato's Theory of Definition and Aristotle's Categories

Although the above theory is based on modern cognitive sciences, some parts of it were already understood by the Greek philosophers Plato and Aristotle. The Theory of Definition, which is described by Plato and used by Aristotle for his categories [a11, a21], is based on the following.

To determine the concept X, first determine the largest class G to which X belongs. Then, divide class G into parts (say G1 and G2) and determine to which (sub)class X belongs. Suppose this is G1. Divide G1 on its turn into parts (say G1,1 and G1,2) and locate X in one of these.

This procedure is continued until a subdivision is reached that is identical to X. The nature of X is then given by the entire division, which can be represented as an inverted tree; see figure 7 (left side). The class to which a thing belongs is called its Genus, and the characteristic that differentiates it within this class is called *Differens*. Combining a Genus with a Differens defines a (sub)class, and the entire set of differentia needed to arrive at X – indicated as d_a, d_b and d_c in figure 7 - defines X.

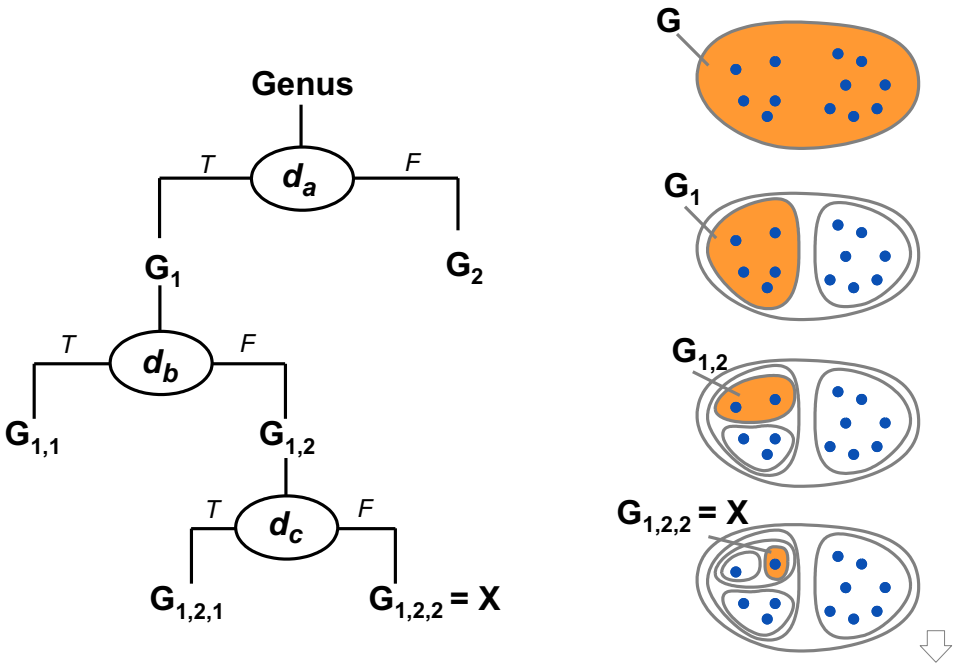


Figure 7. Theory of Definition. On the left side, a hierarchy of classes (Genus, G1..G1,2,2) is formed by subdivision through differentiae (da..dc). Each class corresponds with a set of individuals, such as shown on the right side. The set of individuals that belongs to a class is called the 'extension' of that class.

Figure 7 shows on the right hand side the respective (sub)classes that result from this division. Individuals that belong to each class are shown as dots.

The differentia that play an essential role in Plato's Theory of Definition are not different from the *notions* that were introduced in the previous chapter (chapter **Fout! Verwijzingsbron niet gevonden.**). Hence, notions can be used as differentia for the definition of concepts.

A definition of X may then take the following form:

$$\mathbf{X := G, d_a(\mathit{T}), d_b(\mathit{F}), d_c(\mathit{F})}$$

In this example, G represents the Genus, d_a the first Notion for division, and (*T*) the Notion value, where *T* stands for 'true'.

X can thus be defined through its Genus and the values of three Notions. This kind of definition is called an *intensional definition* [a9].

> *The Extension of a Concept is the set of phenomena (usually real world phenomena) to which the Concept refers.*
>
> *The Intension of a Concept is the set of Notion Values that characterises (or defines) the Concept.*

In figure 5, an example was given of four different definitions of the term 'wall'. For all four cases, the set of walls that exist in reality would form the extension of the concept 'wall'. The intension of these four definitions is however different. For the IAI/IFC definition 'a wall is a vertical construction that bounds or subdivides spaces', the concept 'construction' would be the genus. The Notions 'vertical' and 'bounds or subdivides spaces' are the differentia that distinguish walls from other constructions.

It is possible to define the term 'construction' on its turn via a genus and differentia. The most generic genus could be 'thing'. If 'construction' would be defined with 'thing' as a genus, then

the set of differentia needed to define 'construction' plus the differentia 'vertical' and 'bounds or subdivides spaces' would form the complete intensional definition of the concept 'wall'.

3. Cases

The theory is not yet transformed into an applicable method. Also, it is not yet possible to apply it a large scale as an alternative for existing standards. More additional research is needed to understand the system of Notions that is needed for industrial applications.

The four cases that will be discussed here are more intended for exemplification than for proof. For reasons of clarity, all Notion-Frames are associated with a Genus concept. Once it is possible to define the Genus completely in terms of Notions, a full Notion-based expression will be possible.

3.1. Notions support low redundant and precise semantics

This simple example demonstrates that, with only two Notion-frames, eight different concepts can be defined, using the concept 'Person' as a genus. It defines the concept 'boy' as a 'male person, of an age less then 18 years. An even more precise definition would be to replace the Notion-frame 'Gender' by a Notion-frame 'Legal-Gender' or 'Gender-by-DNA'.

If Notion-theory is applied in the form of Notion-chains, the Genus and values of the two Notion-frames are sufficient to replace the eight concepts. This leads a drastic simplification of a conceptual system, without any loss of semantic precision.

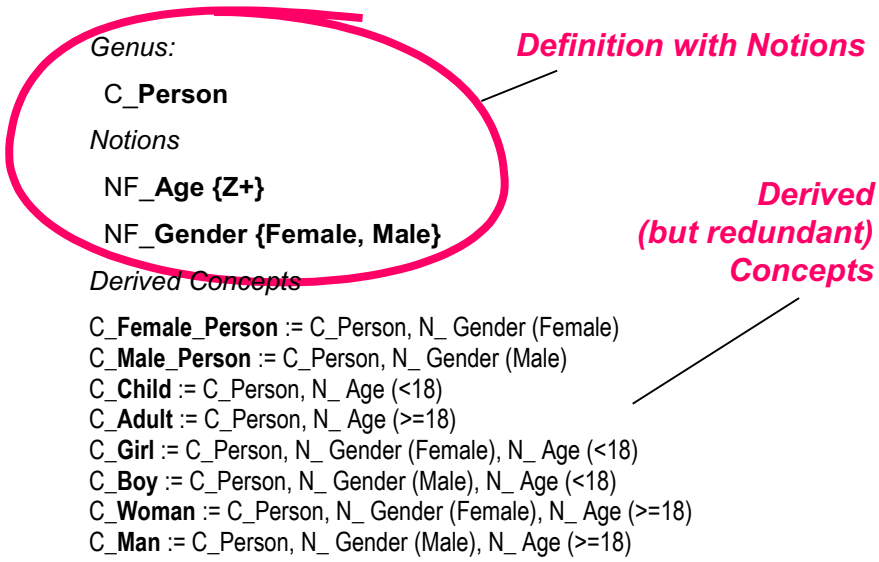


Figure 18. Two Notion Frames for the concept Person result in eight more specific concepts.

3.2. Application of Notion-theory for Computer Integrated Manufacturing

Between 1988 and 1992, the European R&D project IMPPACT developed a methodology and platform for Computer Integrated Manufacturing (CIM). For this purpose, the project developed an integrated Product and Process model [a13]. This model should be understood as an application schema for a database, through which several computer applications for design and manufacturing can share their data.

This huge and complex model could be developed fairly quickly, because its was found that several information structures were generic. The backbone of this integrated Product and Process model was a single concept, called 'definition'. In addition, five orthogonal 'dimensions' for specializations of this concept were identified. These 'dimensions' can be understood, in the context of the new theory, as Notion_Frames.

The heart of IMPPACT's Product and Process Model is expressed below according to Notion-theory.

Genus:

C_**Definition**

Notions

NF_**Change {Process, Product}**

NF_**Specification {Generic, Specific, Occurrence}**

NF_**Concretization {Required, Proposed, Realized}**

NF_**Lifecycle {As_Produced, As_Operated, As_Disposed}**

NF_**Aspect {Cost, Stability, Safety,….}**

Figure 19. The five Notion-Frames needed for integrated product and process modelling according to the IMPPACT project.

This model can be understood as follows.

The first Notion identifies two kinds of Definition, based on the Notion of Change: Product and Process Definitions.

The second Notion identifies Generic Definitions (for classes of products or processes, such as expressed in a parametric model), Specific Definitions (completely defined products or processes), and Occurrence Definitions (individuals).

The third addresses the phase of Concretization. It makes a distinction between Required, Proposed and Realized Products or Processes. Each of these could on its turn be specified as being Generic, Specific or Occurrence, thus leading to, for example, a 'Required Specific Product Definition'.

The fourth Notion introduces Lifecycle: As Produced, As Operated or As Disposed. Which leads, with reference to the above, to something like a 'Required Specific As_Operated Product Definition'.

Then, finally, the fifth Notion addresses Aspects, such as Cost, Stability and Safety. In combination with the aforementioned Notions this can lead to a 'Required Specific As_Operated Product Cost Definition'.

IMPPACT showed that practically all of these combinations make sense and are meaningful. In certain cases all Notions were used, but in several other cases partial generalizations were used, such as the concept 'Realized Process Occurrence'.

The five Notions can be used in any order and in any combination: they are independent, and thus orthogonal.

These Notions enabled the project to define very precise semantics.

The IMPPACT model was however implemented with an object-oriented database. The five Notions result in more than 500 different concepts. This 'explosion' of concepts made it impractical to implement the model using current technology.

This experience formed an important argument for the development of the current Notion-theory, as Notion-based implementations are more accurate and more compact than implementations using predefined concepts.

3.3. Engine taxonomy defined with Non-orthogonal Notions.

The following example shows some Notion-Frames that can be used to construct a taxonomy of engines.

It includes a few non-orthogonal Notions-Frames. Notion-Frames such as Gas_Expansion_Source and Gas_Expansion_Location are only relevant if the Power_Source has the value Gas_Expansion. And Notion-Frame Ignition is only relevant for engines of which the Gas_Expansion_Source is Chemical_Reaction.

This example uses six Notions for the definition of thirteen Concepts. Even in the case of non-orthogonal Notions, it appears that Notion-based notations are more compact than an enumeration of resulting concepts (i.e. the resulting taxonomy).

This example is derived from [a19] and reworked by the author.

Genus:

C_**Engine**

Notions

NF_**Motion {Linear, Rotation}**

NF_**Power_Source {Magnetic_Field, Gas_Expansion}**

If N_**Power_Source (Gas_Expansion)** then

NF_**Gas_Expansion_Location {Internal, External}**

NF_**Gas_Expansion_Source {Vaporization, Chemical_Reaction}**

If N_**Gas_Expansion_Source (Chemical_Reaction)** then

NF_**Ignition {spark, pressure}**

NF_**Gas_Change {2-stroke, 4-stroke}**

Derived Concepts

C_**Gas_Expansion_Engine** := C_Engine, N_ Power_Source (Gas_Expansion)
C_**Gas_Turbine** := C_Engine, N_ Power_Source (Gas_Expansion), N_Gas_Expansion_Source (Vaporization), N_ Gas_Expansion_Location (External)
C_**Piston_Engine** := C_Engine, N_ Power_Source (Gas_Expansion), N_Motion (Linear)
C_**Rotation_ Engine** := C_Engine, N_ Power_Source (Gas_Expansion), N_Motion (Rotation)
C_**Jet_Engine** := C_Engine, N_ Power_Source (Gas_Expansion), N_Motion (Rotation), N_ Gas_Expansion_Location (Internal), N_ Gas_Expansion_Source (Chemical_Reaction)
C_**Internal_Combustion_Engine** := C_Engine, N_ Power_Source (Gas_Expansion), N_ Gas_Expansion_Location (Internal), N_ Gas_Expansion_Source (Chemical_Reaction)
C_**Two_Stroke_Engine** := C_Engine, N_ Power_Source (Gas_Expansion), N_ Gas_Expansion_Location (Internal), N_ Gas_Expansion_Source (Chemical_Reaction), N_Gas_Change (2-stroke)
C_**Four_Stroke_Engine** := C_Engine, N_ Power_Source (Gas_Expansion), N_ Gas_Expansion_Location (Internal), N_ Gas_Expansion_Source (Chemical_Reaction), N_Gas_Change (4-stroke)
C_**Otto_ Engine** := C_Engine, N_ Power_Source (Gas_Expansion), N_ Gas_Expansion_Location (Internal), N_ Gas_Expansion_Source (Chemical_Reaction), Ignition (spark)
C_**Diesel_Engine** := C_Engine, N_ Power_Source (Gas_Expansion), N_ Gas_Expansion_Location (Internal), N_ Gas_Expansion_Source (Chemical_Reaction), Ignition (pressure)
C_**Four_Stroke_Diesel_Piston_Engine** := C_Engine, N_ Power_Source (Gas_Expansion), N_ Gas_Expansion_Location (Internal), N_ Gas_Expansion_Source (Chemical_Reaction), Ignition (pressure), N_Gas_Change (4-stroke)
C_**Wanckel_ Engine** := C_Engine, N_ Power_Source (Gas_Expansion), N_Motion (Rotation), N_ Gas_Expansion_Location (Internal), N_ Gas_Expansion_Source (Chemical_Reaction), Ignition (spark), N_Gas_Change (4-stroke)
C_**Steam_Engine**:= C_Engine, N_ Power_Source (Gas_Expansion), N_ Gas_Expansion_Source (Vaporization), N_Motion (Linear)

Figure 20. Engine taxonomy with non-orthogonal notions. Many more concepts can be derived and defined with the six notion-frames listed.

3.4. Wine Ontology

W3C (World Wide Web Consortium) uses a Wine Ontology [a24] for the explanation of OWL (Web Ontology Language) Description Logics. This ontology contains more than 70 classes.

The contents of this ontology, including its semantics, can also be expressed with just 9 Notion-Frames. The properties which are part of this ontology are interpreted here as Notions. But Notion-Frames such as NF_WineBody {Full, Medium, Light} and NF_WineFlavor {Strong, Delicate, Moderate} do not form a strong semantic foundation, as the classification of wine in one or the other category is very subjective. Firmer and more objective Notion_Frames could be based on measurable parameters, such as sugar-percentage and tannin-percentage.

The Wine Ontology demonstrates one other weakness of the ontologic approach. Specific wines, such as the 'Whitehall Lane Cabernet Franc', are considered as Individuals. They form the extension of the Class-hierarchy. But what about different vintages of this kind of wine? And what about different bottles of each vintage of this kind of wine? To declare classes at a particular level as individuals, is an arbitrary (subjective) choice of the model developer.

The Theory of Notions makes no difference between knowledge about individuals and classes: that what can be an individual from one perspective, can be a class from another perspective.

Genus
C_**Wine**
Notion Frames
NF_**Grape {Merlot, Cabernet Franc, Riesling, …}**
NF_**WineColor {Red, Rose, White}**
NF_**WineBody {Full, Medium, Light}**
NF_**WineFlavor {Strong, Delicate, Moderate}**
NF_**WineSugar {Dry, Offdry, Sweet}**
NF_**WineRegion {Saint_Emillion, Pomerol, Medoc, Bourgogne, Napa_Valley, France, …}**
NF_**Winery {Cheval_Blanc, Figeac, Petrus, WhitehallLane, …}**
NF_**VintageYear {Integer}**
Notion Frame from Topology
NF_**Spacial_Location {Enclosure, Boundary}**
Constraints
Constraint 1:= N_WineRegion (France), N_WineRegion (Bourgogne), N_Spacial_Location (Enclosure)
Constraint 2:= N_WineRegion (Pomerol), N_WineRegion (Saint_Emillion), N_Spacial_Location (Boundary)
Derived Concepts
C_TableWine := C_Wine, N_WineSugar (Dry)
C_WhiteBurgundy := C_Wine, N_WineRegion (Bourgogne), N_WineColor (white)
C_WhitehallLaneCabernetFranc := C_Wine, N_WineRegion (NapaRegion), N_Winery (WhitehallLane), N_WineFlavor (moderate), N_WineBody (medium), N_WineGrape (CabernetFranc), N_WineSugar (dry)

Figure 21. The W3C Wine Ontology, that has over 70 classes, can also be defined with just 9 Notion Frames. Except some additional rules, nothing more is needed. The derived concepts below are not part of the model but show that Notion Chains, which are data, are equivalent to - and thus can replace - the Classes that are currently incorporated in an ontology.