Basics of the Theory of Notions

Wim Gielingh, concept version 24 june 2024[1]

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[1] This article is based on a withdrawn paper submitted to the Journal of Computer Aided Design in 2007 and 2008, concurrently with reference [a1].

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.

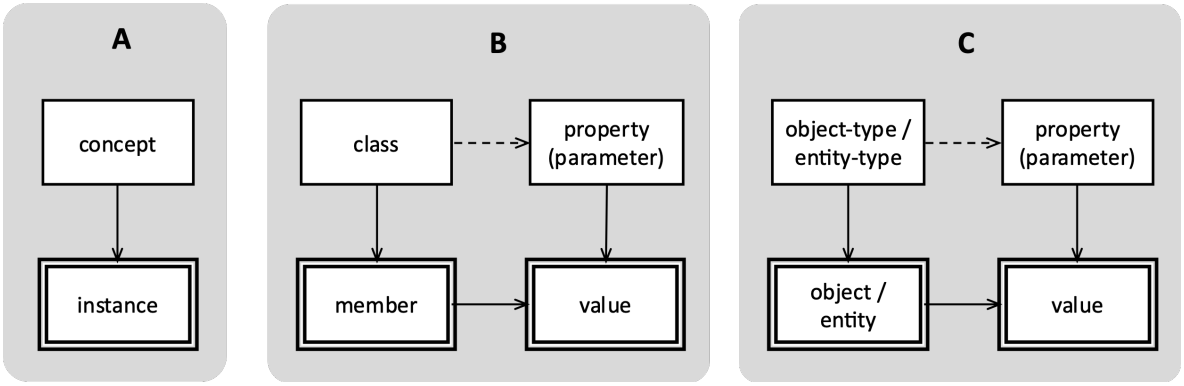

Fig 1 a-c. Three ways to use memory cells of computers for the representation of knowledge. The rectangles with a single border represent memory cells, the ones with a double border the content of cells.

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).

Class-based thinking

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).

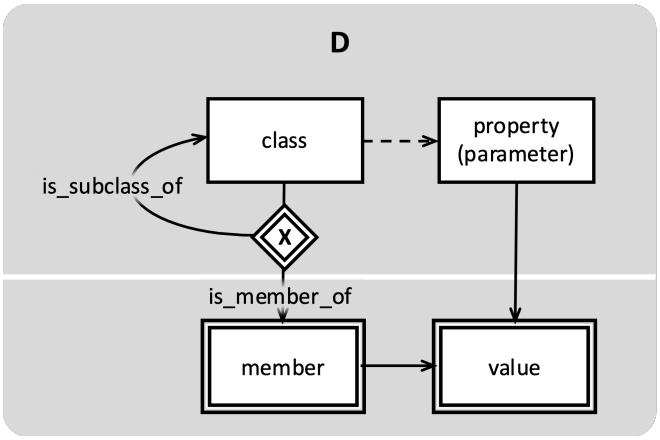

Fig. 1d.

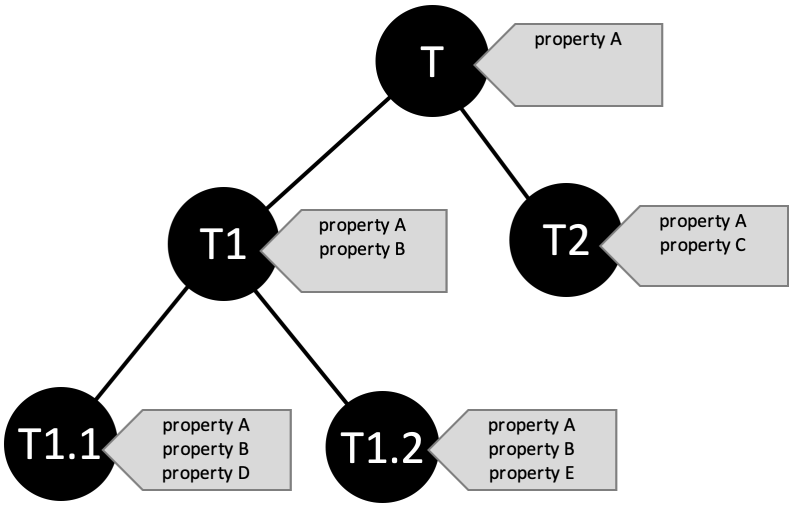

Fig. 2: When the upper part of the schema in 1d is instantiated, the result is a class-hierarchy such as shown here. Members of a class are characterized by common properties. All members of class T have property A. All members of subclass T1 have properties A and B, and so on. The rule that subclasses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

Object based thinking

Object-based thinking follows more or less the same pattern, but its origin lies in information technology. Objects are data elements that have characteristics than can be shared. These characteristics are represented by data but also by methods, i.e. algorithms for data conversion that can be shared by objects. Whatever is shared is attributed to an object-type. Object-oriented computer programming is an alternative for procedural computer programming, where procedures are independent of the 'objects' that are being processed.

Entity based thinking

The term entity has more or less the same meaning as 'object' but its origin lies in database technology, in particular relational databases. Entities are data elements with shared data structures, but without methods for data conversion.

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.

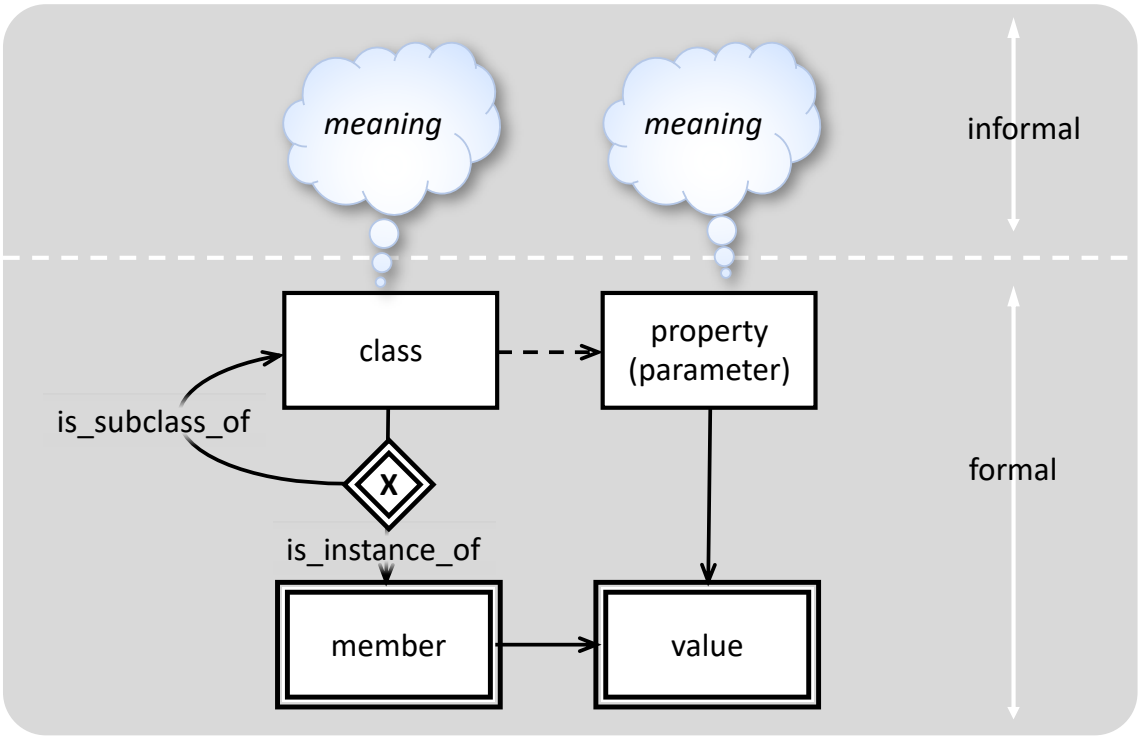

Fig.3.

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 observation 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 4 shows a seat in the form of a cube. There are at least five different ways to model this product: as a seat, as a cube, as a plastic product, as a product, or as a red_plastic_cubic_seat. It depends on the preference of the schema-developer which aspect is used for classification.

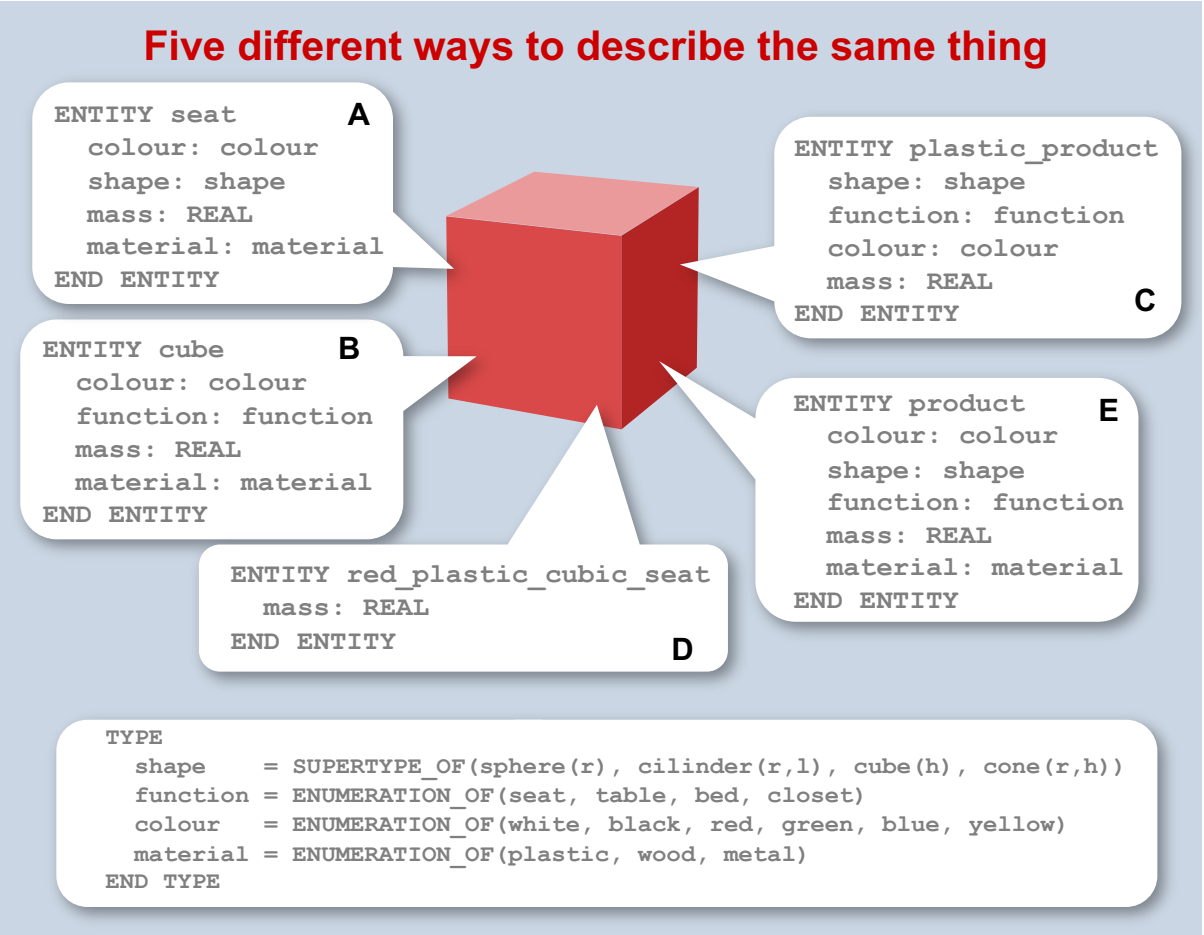

Figure 4. There are many ways to model a red, cube-shaped, plastic seat. This example in the Express language shows just five.

What we see also in this example, is that property-values are used for classification, while the corresponding property-parameters (i.e., the variables) are implicit. Function determines the classification of the object in schema A, shape in schema B, and the kind of material for schema C. For schema E a very generic (abstract) concept is used, so that the object itself has to be specified completely by means of variables, while concept D has all parameter-values defined as part of the definition of the concept. And as we have seen with the example of 'Wall' in IFC, such definitions are informal – they are not computer-interpretable. Hence, classifications that are used for the naming of object-types add no real meaning to a conceptual specification!

Furthermore, we must conclude from this example that it is impossible to exchange or share data between applications that are specified according to the schema's in figure 4. It is not possible to develop conversion software between two applications if they use different schema's. It is not possible to map schema B on schema C because the type of material is a variable of B and a given in schema C, while shape is a variable of C and a given in schema B. It is simply not possible to share or exchange data between applications using the five schema's shown in fig. 4, even though their data describe one and the same object!

Here we see what the real cause is of interoperability problems: these problems are inherent to the schema based approach. The same conclusion must be drawn for ontologies. Standardization of schema's or ontologies doesn't help us either!

~~1.4 Some intermediate thoughts~~

~~There are different ways to use computer memory for the modelling of knowledge. The assumption that memory cells represent classes of things, and that the contents of these cells represent members of a class, is just one of them. But there are many things that cannot be considered as a class. Is 'water' a class? A class of what? What is an instance of water: a bottle, a drop or a molecule? Is a thought a class?~~

~~In the class-based approach, cell-content may refer to a member (a discrete thing) but also to a sub-class.~~

~~Memory cells may also represent concepts so that their contents represent uses of these concepts. And cell-contents may represent objects, in which case the cells represent object-types.~~

~~Generally speaking, memory cells may contain data, but they may also contain meta-data (data about data). As this may cause confusion about the meaning of data, it is necessary to reconsider the way in which we model knowledge.~~

2. Rethinking the expression of knowledge

2.1. Design is about the union of knowledge

The purpose of data sharing in industry is to develop a common understanding of an artefact that may or may not 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 ($A \cap B$). But what we actually need in design is the unification of knowledge ($A \cup B$). 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.

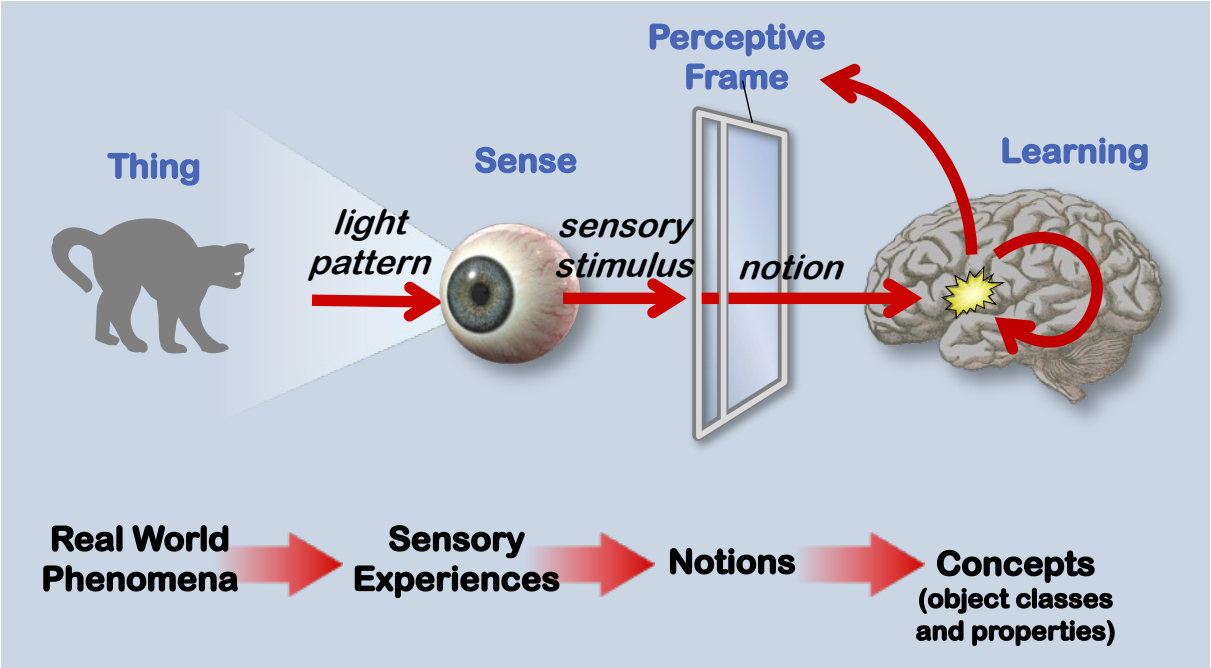

Figure 6. Simplified model of the process of human cognition. Notions are not pure sensory experiences; they are affected by the perception of the observer, which is shaped by former experiences. A new experience, and thus a new notion, may affect the perception. This is what is usually called learning by experience. Perceptive frames may develop into concepts that can be represented by words.

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).

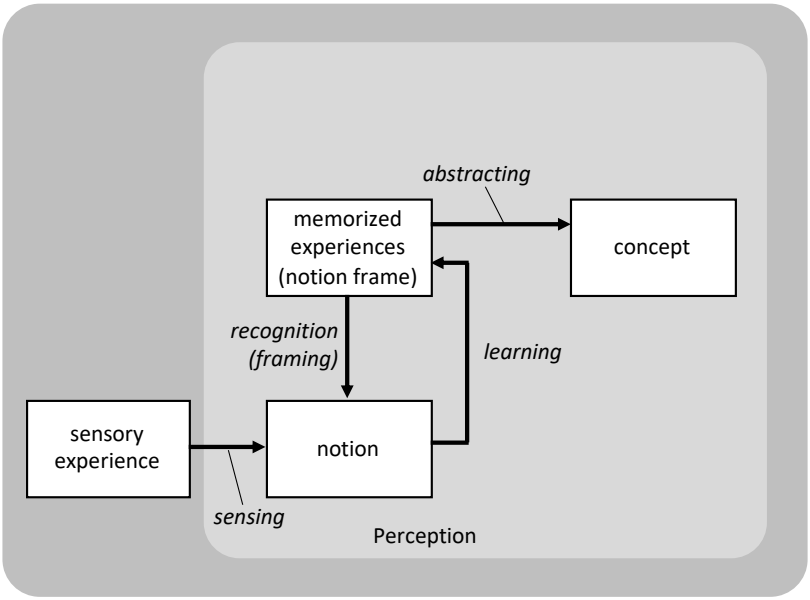

Fig. 7. Simplified depiction of the cognitive process. New sensory experiences are projected on memorized experiences which provide context. This process is also known as recognition. The result of a recognized phenomenon is called a notion. This process may also be called recognition. The new experience is added to the existing notion-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) Scientific observations and experiments are subject to standardized procedures and are systematically documented;

c) publications are peer reviewed to avoid subjectivity and to ensure quality;

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).

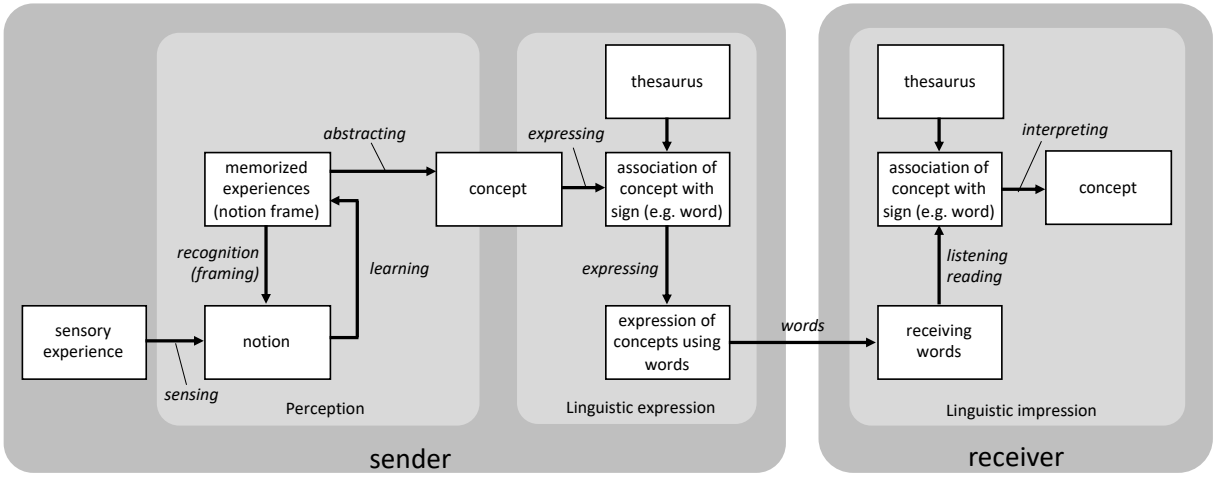

Fig 8.

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.

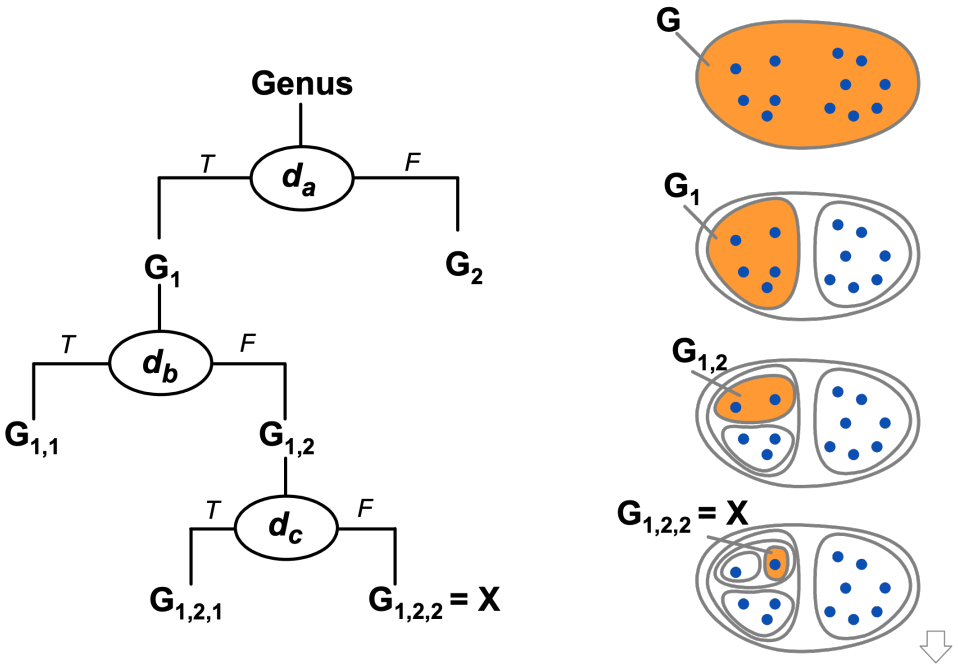

Figure 7. Theory of Definition. On the left side, a hierarchy of classes (Genus, G1..G1,2,2) is formed by subdivision through differentiae (da..dc). Each class corresponds with a set of individuals, such as shown on the right side. The set of individuals that belongs to a class is called the 'extension' 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 2.2). Hence, notions can be used as differentia for the definition of classes.

A definition of X may then take the following form:

$$X := G, d_a(T), d_b(F), d_c(F)$$

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.

Please note that we use the term 'concept' instead of 'class'. The reason is that Notions may form building blocks for the definition of classes, but also for members of classes, functions, processes, properties, or any other idea that is formed in the human mind. Nothing is excluded.

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'.

2.6. The difference between Properties and Notions

Notions may look like Properties, but they are not the same; the difference between them is subtle but important:

- Notions are inherent to an observer. They depend on (a) the capabilities of the observer to sense phenomena, and (b) how these sensory experiences are interpreted;
- Properties are assumed to be inherent to a phenomenon.

There is however no other way to acquire knowledge about a phenomenon than through observations, and thus via Notions.

Examples of notions are taste and colour. The human tongue distinguishes only four tastes: sweet, sour, salt and bitter. Other 'tastes' are actually sensed as odour. And although the spectrum of odours that humans smell seems to be much larger, the human nose is not as sensitive as that of dogs. The categories that a dog can derive from odour are much more detailed than that of a human being.

The human eye perceives colour by the sensitivity of cones in the retina in three distinct parts of the electromagnetic spectrum. This enables manufacturers of displays and printers to mimic practically all colours by a vector in 3-space, such as {red, green, blue} in the additive colour system or {cyan, magenta, yellow} in the subtractive colour system.

If we watch the world through an old-fashioned black-and-white television-set, we distinguish no colour at all; only shades of grey.

The reflection of electromagnetic radiation by the surface of an object is however much more complex and detailed than the human eye can detect.

Glass seems to have no colour: it is transparent for our eyes. But if our eyes were sensitive for ultra-violet light, it would not be transparent. Reversely, we, human beings, are transparent if our eyes were sensitive for extreme short-wave radiation, such as X-rays.

The notions of colour and transparency, just as that of taste and odour, depends therefore largely on our ability to sense and perceive.

The distinction between notion and property may be relevant for the classification of things. For example, we may state that there are two kinds of persons: male and female persons. But how is the gender of a person determined? There are at least three alternatives. The first is to determine gender by appearance. This is a rather imprecise and subjective method. The second is to examine the DNA of a person. This will give a precise and objective result. The third method is to take the legal registration, such as a certificate of birth. In most cases, the three methods will give the same result. But in the cases of transsexuals or transvestites, the results will be different.

'Gender' is therefore not appropriate as a Notion, because it doesn't clarify how a result is obtained. 'Gender-by-DNA' and 'Legal-Gender' are examples of more appropriate Notions.

2.7. Notion-Frames and -Values

To support a more detailed discussion of Notions, a distinction will be made from hereon between Notion-Frames and Notion-Values.

A Notion-Frame expresses the ability of a perceiving system to note differences in sensed input. *A Notion-Frame is thus inherent to (and part of) the perceiving system*.

Notion-Values provide information about the subject that is sensed by a perceiving system. Too easily, Notion-Values are claimed today to be Properties, and this is one of the causes of interoperability problems.

Notion-Values that originate from different Notion-Frames are in many cases incompatible. For example, the colour 'blue', means something different for a perceiver that uses the Notion Frame {red, orange, yellow, green, blue, violet} as for a perceiver that uses Notion Frame {red, green, blue}. That is because the second perceiver may interpret violet as a kind of blue, while the first doesn't. Further, a value of the wavelength of light, such as 450 nm, means something different than the term 'blue'. A wavelength permits a more accurate description of colour than a term. Clearly, the ability to make distinctions between colours is inherent to a perceiving system, and even values cannot be fully detached from that perceiving system.

Although Notion-Values may not always be convertible, there are also many cases that permit conversion. One of these is when different units are used for the expression of a quantity, such as meters versus feet or inches. But as part of the theory of Notions we recommend to keep the expression of quantities in the form in which they are observed, expressed and interpreted, so that conversions can be done in a later stage. The purpose of this is that conversions may result in a loss of accuracy or intent. Hence, the rule of retaining data in their original form should always be applied in a data sharing environment.

2.8. The role of Notions as differentiators

Does a Notion based definition system lead to more complexity? The answer is just the oppositie: it reduces the complexity of class-hierarchies!

First, it must be understood that a conceptual system, of which the concepts are defined through Notion-Frames that say something about the perceiving system, and which correspond with objective measuring methods, has a stronger semantic foundation then a system that uses plain English text, and of which the criteria for the distinction between concepts are vague.

Second, it appears that Notion-Frames used for definition and differentiation are often independent of each other, which makes Notion-based definitions less redundant and therefore very compact.

Examples of independent (orthogonal) Notion-Frames in animals are: gender (male, female), maturity (child, adult), food (herbivore, omnivore, carnivore), spine (vertebrate, non-vertebrate) and metabolism (cold-blooded, warm-blooded). These and several other Notion-Frames play a role in the classification of species and individual animals.

With n orthogonal and binary Notion-Frames, 3^n-1 Concepts can be defined. As an example: the genus concept 'person' and the two Notion-Frames 'gender (male/female)' and 'maturity(immature/mature)' result in the following (sub)concepts: male person, female person, child (= immature person), adult (= mature person), man (= male mature person), woman (= female mature person), boy (= male immature person) and girl (= female immature person). A total of eight different concepts.

Even if Notion-Frames are not orthogonal, $2n$ Concepts can be defined with n binary Notion-Frames. In the case of Notion-Frames that have more than two possible values that exclude each other, the possible number of concepts is $(p+1)^n - 1$, where p is the number of possible values.

The number of Notion-Frames in a conceptual system is thus considerably less than the number of Concepts; a few examples will be given in chapter X. A conceptual system based on scientifically determined Notions is objective, compact and less complex compared with a system based on Concepts or Classes.

Reversely, we may state that conceptual expressions that are *not* based on Notions contain semantic redundancy and are semantically incomplete.

3. The expression of knowledge through Notions

The philosophy that has been described up to this point is independent of a particular method for application. A discussion of this aspect reveals more implications of Notion-theory.

3.1. Perceiver Orientation

The conceptual framework that is proposed here, is not object- or subject-centred. In stead, it is based on models of perceiving systems: so-called ‘knowledge processors’. With a ‘knowledge processor’ we mean a human being, a discipline, an organization or a machine.

The set of Notion-frames that characterize the capabilities of a knowledge processor to sense, note and interpret the external world, forms together a *Perceptive Frame*. As each knowledge processor has a particular role in an industrial process, certain Notion-frames are used for knowledge input, while some or other Notion-frames are used for knowledge output.

Figure 9a shows an example of five knowledge processors with their respective Perceptive Frames (Indicated by PF*_name*). If one or more of their Notion-Frames (Indicated by NF*_name*) corresponds with Notion-Frames of other processors, they will be capable to communicate Notion-Values.

Current standards for the exchange or sharing of product data assume that communication between these knowledge processors is only possible via a "neutral model". A neutral model permits however only the communication of Notion-Values that knowledge processors have in common. This implies that they can only communicate the *intersection* of their (intensional) knowledge. Neutral models lead therefore to a *destruction* of knowledge.

In industrial processes, knowledge processors are however assumed to *unify* their (intensional) knowledge: each processor adds its knowledge to a pre-existing core of knowledge. The unification of Notion-Frames for the example in figure 9a is shown in figure 9b.

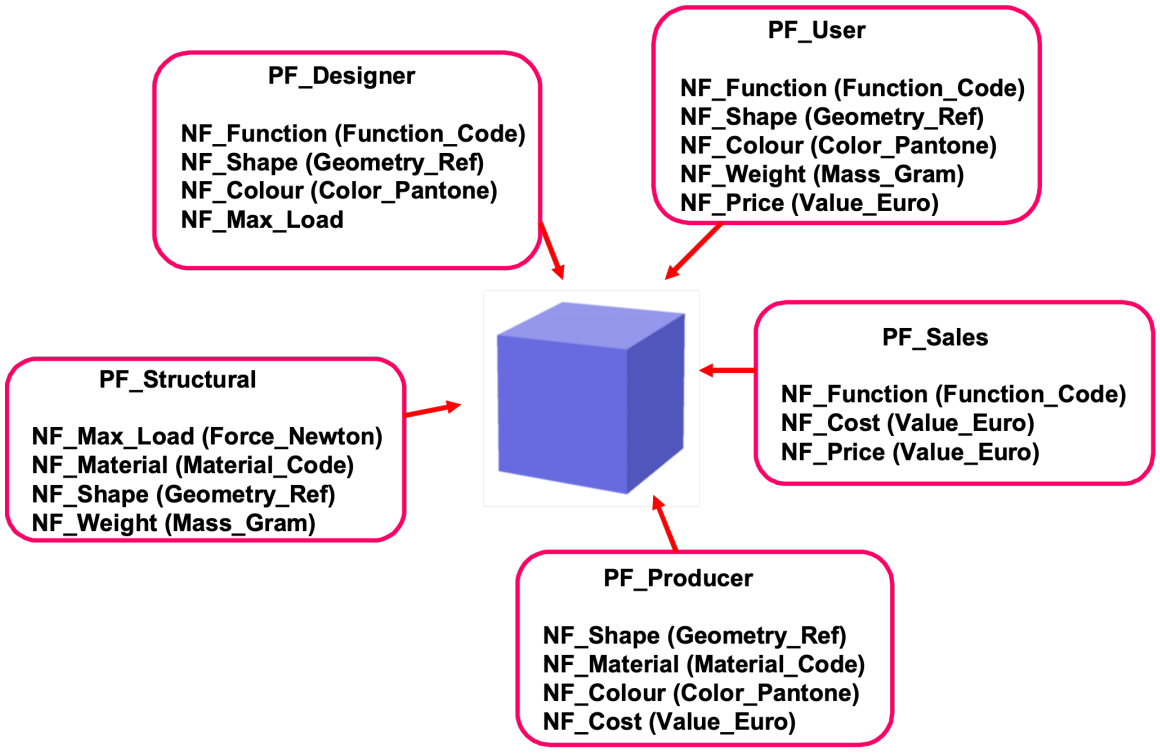

Figure 9a. Example of five actors in an industrial context that have a need to communicate. Each actor has a Perceptive Frame (PF) which consists of one or more Notion_Frames (NF). To enable the sharing of knowledge requires harmonization of the Notion_Frames.

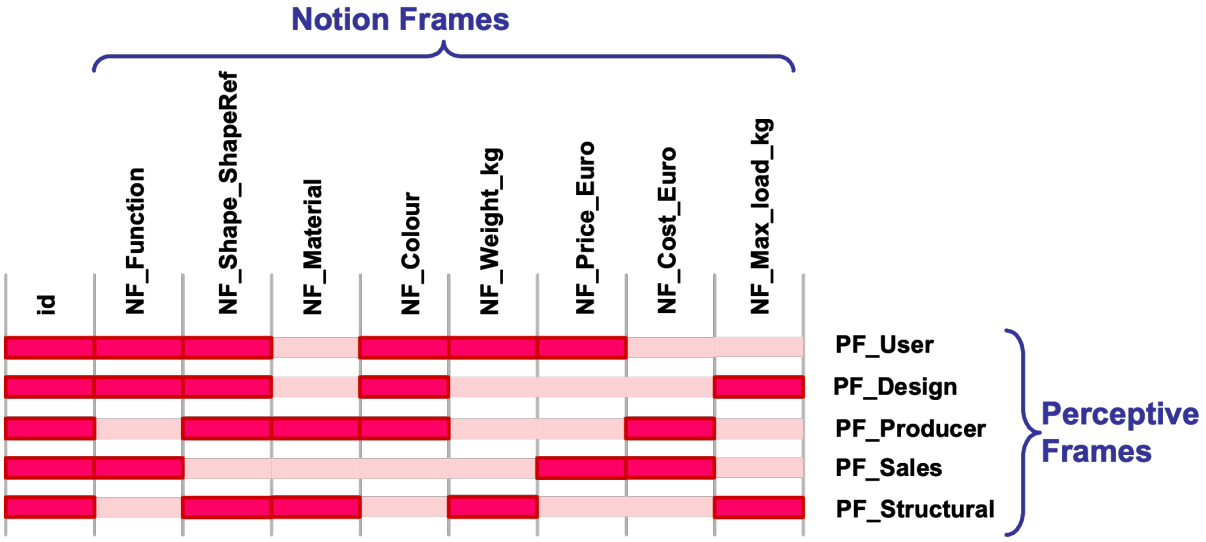

Figure 9b. Communication Matrix of the five actors. The five Perceptive Frames are depicted as horizontal bars. Notion Frames that are part of a Perceptive Frame are indicated dark/red.

3.2. Engineering the 'Knowledge DNA' of Products and Processes

Notion theory thus leads to a perceiver-oriented (or perceiver centred) way of structuring knowledge. The "subject of discussion" between knowledge processors is defined by the initiating knowledge processor through one or more notions, and it will be specified further by other processors by adding notions to it.

Knowledge creation is thus supported by a creative and constructive process in which Notions are combined into Notion-Chains. A Notion-Chain may be compared with a DNA molecule, where each Notion represents a gene. The objective of design and product development is then to assemble a complete 'Product-DNA'. Knowledge processors that are involved build this 'DNA' by adding their 'genes' to it. Each knowledge processor has a limited scope, but all *product-development-processors* together work on the full-scope specification.

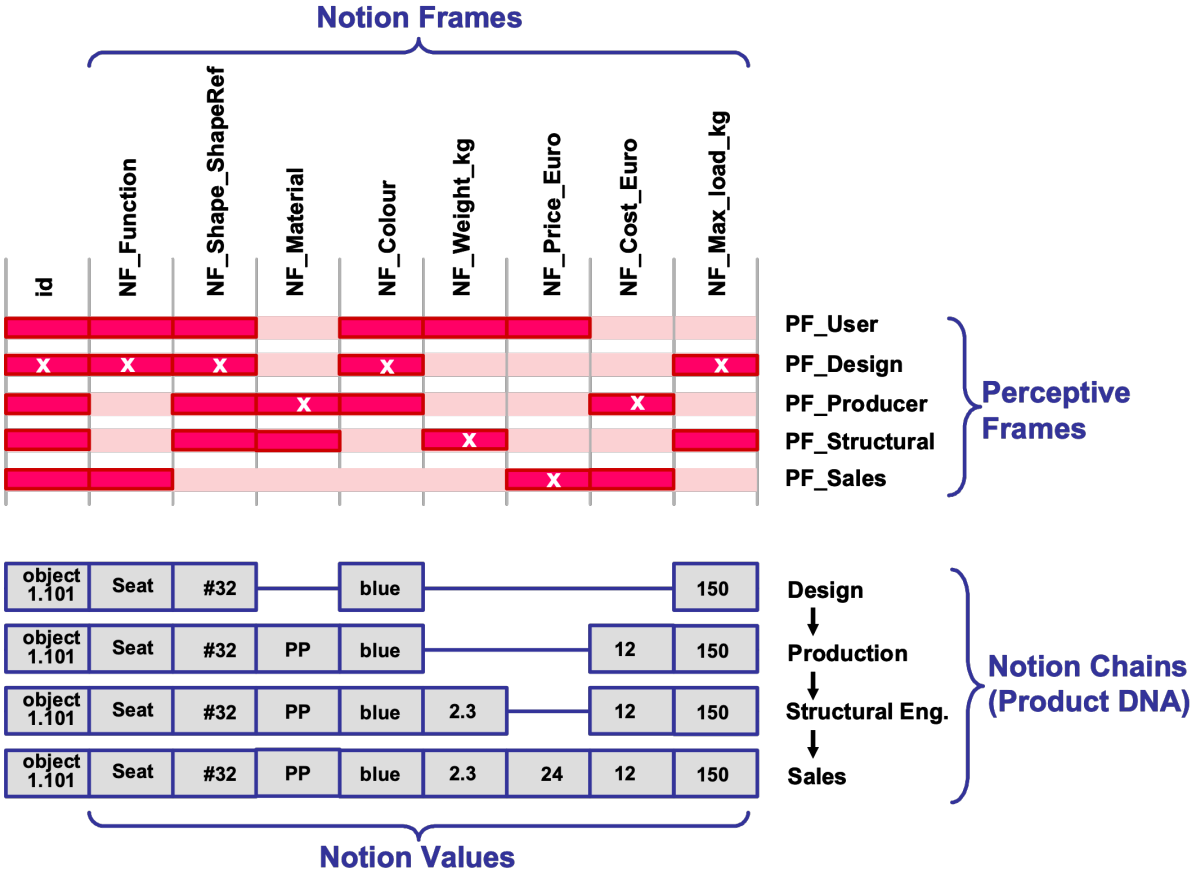

Figure 10. Example of four actors that produce and use chains of Notion Values (or Knowledge Genes) for a particular product, in this example the blue cube-shaped seat. The authors (and 'owners') of Notion Values are marked with an x. *The 'Product DNA', shown below, grows in every step of product development.*

This idea is illustrated by figure 10. The upper part of this figure shows five different perceptive frames depicted by horizontal lines. Only the Notion Frames that are marked dark/red on each line are part of a Perceptive Frame. In this example, the 'design frame' (the horizontal line marked with PF_Design) has four Notion Frames: function, shape, colour and maximum load. Production has also four Notion Frames: shape, material, colour and cost. If the design and production disciplines wish to communicate, they can exchange a string of notion values and use a common identifier for this string. A prerequisite for a correct interpretation of the exchanged knowledge is that they agree about the two Notion Frames (i.e. shape and colour) that they have in common.

Every time that a new discipline is involved in the process, new Notions can be added, ultimately leading to a complete 'Product DNA'.

Semantically, a Notion-Chain is principally equivalent with the instance of a concept in the class based paradigm. However, to produce this chain requires no equivalent of a concept, such as an 'object-type' or a 'class'. It doesn't require a schema or ontology either. The Notion

Chain is produced by one or more knowledge processors, each having a different view on a subject, as expressed by their Perceptive Frames.

In principle, there are no restrictions nor limits to the size or composition of Notion Chains; they can extend if more knowledge is needed and more knowledge processors become involved. The only prerequisite is that the Perceptive Frames of knowledge processors have some Notion-Frames in common.

In the process of design this means that one discipline starts to define a piece of the chain. Another discipline uses it, may modify some Notion-Values, and adds other Notions to it. For example, a structural engineer uses 'shape' and 'maximum load' defined by the designer, as well as 'material' defined by production, modifies 'shape' (such as the thickness of the walls of the seat), and may add the Notion 'Weight'.

No discipline sees everything; each discipline sees only that what is relevant for its own process.

To avoid redundancy in a chain, it is recommended to include references to shared specifications. 'Shape', for example, is a rather complex Notion that cannot be expressed by a single parameter. The Notion 'Shape' may therefore refer to another chain or an array of chains that describe shape.

Notion-Chains do not necessarily represent instances of objects. These chains may represent any kind of knowledge. They may be about objects, but also about processes, or generic reusable product specifications. One chain may represent the shape of a part, another may represent material properties.

3.3. Replacement of Concepts by pure Notion-Chains

Some of the examples that are given in the Appendix contain references to Concepts, such as Genus-concepts and derived concepts. But Notion-Chains do not have to be based on Genus-concepts, and do not have to be used to define concepts. They can be communicated as pieces of knowledge, or ‘knowledge genes’.

In the original Theory of Definition of Plato and Aristotle, the starting point of a Taxonomy was a Genus. A Genus can be considered as a 'super-class', or a 'super-concept'. If it is possible to define any concept by means of the most abstract - and therefore meaningless - concept, such as 'thing' or 'phenomenon', then there is no need anymore to use Concepts (or Classes, Types) in a Knowledge Specification. Concepts can then be derived - if desired - from the Notions that define them.

Hence, it is theoretically possible to replace Concepts entirely by Notions. Such Notion-sets are not yet available, because schemas and ontologies are today informally defined. However, if the right set of Notions for each relevant Concept (Class or Type) can be found, a major step forward will be made.

As concluded earlier, Notions are more precise alternatives for 'properties', but they can also be used as differentia for concept definition. If knowledge is actually based on Notions, it makes sense to drop the stage of concept definition.

By dropping concepts from a knowledge specification, it removes also the factor that causes the current schemas, dictionaries, repositories and ontologies to be view-dependent. Chapter 1.3 showed that current methods force model developers to use certain Notions for Concept definition, while others are classified as properties. By removing this artificial division line, knowledge specifications become independent of a particular modellers view.

For practical and explanatory reasons, examples in this paper still contain references to Concepts. A Concept is then intensionally defined by a Genus-concept and sequence of Notions.

To enable communication between knowledge processors, requires harmonization of Notion-frames between the processors involved. This harmonization can be local and does not have to be enterprise wide. It may therefore be that a particular Notion-frame is used by only two processors, and that other processors use different Notion-frames.

3.4. Authorship of Notions in Design and Product Lifecycle

In engineering practice, the responsibility for specific aspects or properties of a product is distributed over a number of disciplines. This is reflected in Notion-theory by making them (as knowledge processor) responsible for certain Notions. Thus, each Notion in a Notion-Chain has an 'author' who is permitted to define or modify Notion-Values. Other disciplines may access information about the Notion as readers but are not permitted to modify it. At most they can return suggestions for change to the author. If a Notion has multiple authors, all authors should use the same computer application.

This principle, which may seem restrictive, is already applied in many industrial practices for reasons of knowledge management and product data management.

The importance of this principle lies in the fact that authors use a single Notion-frame, and thus a single representation of Notion-knowledge, which then becomes the 'master-copy' of information. Other disciplines may copy Notion-information for their own process, and may even convert it into another representation. These copies are however derived from the master-copy and are considered as Secondary Notions.

This principle enables the concurrent usage of different Notion-frames for a single Property, but avoids loss of meaning or precision due to conversions back and forth.

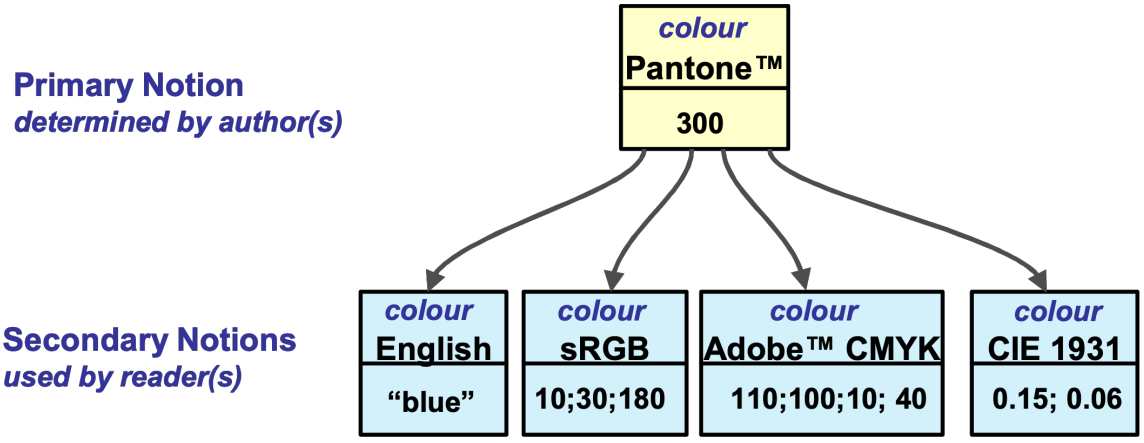

Figure 11. For each property, there may be various different Notions and representations of Notions. The Primary Notion and its representation is determined by the discipline that is responsible for the property. Other Notions and representations for different usages are derived from it. In this example, colour is defined by its 'owner' using the Pantone system. Other representations are derived from the Pantone colour.

3.5. Reconstruction of Concepts or Classes

A knowledge processor may use some Notions in a Notion-Chain to classify the thing that is being processed. Other Notions are then interpreted as the traditional "properties".

Different knowledge processors may use different Notions for classification. Thus, knowledge processors that use internally different taxonomies (or schemas) will still be able to communicate. The chain of Notions contains all relevant information, but is by itself free of interpretation. This enables knowledge processors to communicate at different levels of abstraction.

With reference to the example given in figure 4 a Notion-chain may take the form:

X:= N_Colour (blue), N_Function (seat), N_Shape (cube)

In this example, the name of each Notion is preceded by N_.

Knowledge processors with different 'views' on a particular subject, such as described in chapter, may extract their relevant Notion-values and convert them to their own internal schemas. As the intensional definitions of the concepts that they use are now explicit through a sequence of Notions, the information that they exchange is without loss of meaning and preciseness, and is also view-independent.

3.6. Reduced need for standards

The paradigm that is described here requires only standards for Notion-Frames. It requires no standards for schemas or ontologies. It even permits the *concurrent usage* of different standardized Notion-Frames for each property, including company-specific (internal) standards. Figure 11 showed that a property such as colour can be expressed in many different ways, and that such expressions may be used concurrently. The only condition for successful communication is that each pair of knowledge processors uses the same Notion-Frames for the knowledge that they share. Universal standards are not needed.

The Theory of Notions is relevant for the creation of *knowledge sharing environments*. Computer applications may still be based on conventional programming techniques. To make a bridge between conventional software and a Notion based communication system, requires that the concepts that are processed by each software application can be reconstructed through Notions.

As it is not feasible to jump directly from the current paradigm to the new paradigm, an intermediate solution would be to develop schemas or ontologies in which all object-types or classes are intensionally defined by Notions. This supports industry to develop semantically rigorous standards for knowledge sharing; a process that has already started with the emergence of knowledge bases and the semantic web. However, once sufficient Notions have become available, these schemas and ontologies may be dropped for the realization of a communication system as described in this paper. The advantages of making this final step are explained in chapter 5.

3.7. Formalizing Perception and Scope

In chapter 2.3, and in figure 4, the term 'scope' was used in an intuitive sense to explain why current standards for the communication of knowledge fail. 'Scope' is in fact nothing else than the Perceptive Frame of a particular knowledge processor (i.e. a computer, its application and its user). This Perceptive Frame is formed by a set of Notion-Frames. The 'scope' of an application can thus be expressed in terms of Notion-Frames.

For the example in figure 3, the blue cube-shaped seat, all three models result from the same Notion-Frames and thus have the same scope. However, on the schema-level it is not visible that these three models represent the same data about the same object. This is because current modelling methods force schema-developers to choose which Notion-values become part of the application-schema, and which become part of the application-data. Even if scope and data are the same, communication will not be possible in these cases.

3.8. Reconstruction of concepts and conceptual hierarchies

Conceptual hierarchies, more in particular generalization/specialization hierarchies such as taxonomies, are developed by humans to support cognitive economy. Abstractions or generalizations result from the dropping of Notions. Which Notions are dropped, is determined by the developer of a hierarchy. Hence, such hierarchies (including taxonomies) depend on perception.

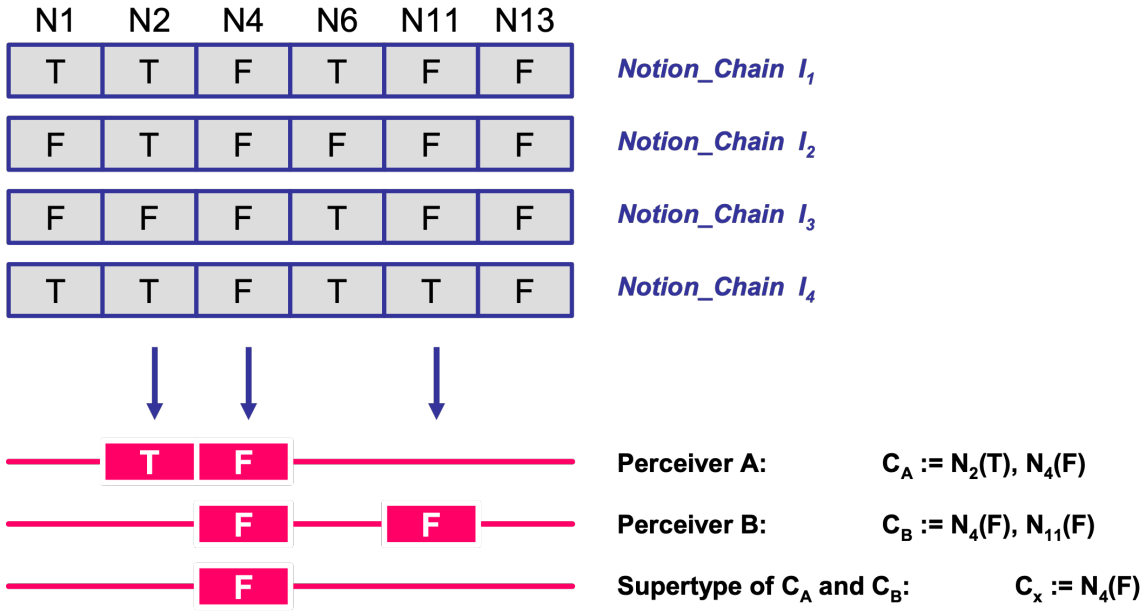

Figure 12. Knowledge about four subjects I_1 .. I_4 is expressed by Notion-Chains (top). Perceiving systems (bottom) may choose certain Notion-Values as criteria for classification and intensional concept definition. The corresponding Notion-Frames are part of their Perceptive Frame.

Figure 12 shows on top four Notion-Chains that contain knowledge about subjects I_1 ... I_4. Perceiver A considers the values of Notions N_2 (True) and N_4 (False) as relevant for the definition of Concept C_A. Perceiver B considers the values of Notions N_4 (False) and N_{11} (False) as relevant for the definition of Concept C_B.

We may also state that C_A and C_B are intensionally defined by:

$$C_A := N_2(T), N_4(F)$$

$$C_B := N_4(F), N_{11}(F)$$

The subjects that are represented by Notion-Chains I_1, I_2 and I_4 correspond with C_A, while I_1, I_2 and I_3 correspond with C_B. In the class based paradigm, it is said that I_1, I_2 and I_4 form an *extension* of C_A, while I_1, I_2 and I_3 form an *extension* of C_B.

As C_A and C_B have the Notion-Value N_4(False) in common, they have a common supertype (or superclass) C_X.

$$C_x := N_4(F)$$

Concepts and Conceptual hierarchies such as taxonomies and ontologies can thus be *intensionally* defined using the Theory of Notions. Also, concepts and conceptual hierarchies can be reconstructed from Notion-Chains if the Perceptive Frames of the perceiving applications are known.

In the present theory, intensionally defined concepts may replace strings of Notions that reoccur frequently.

3.9. About types, classes and their instances

Notion-Chains may represent any kind of knowledge. Therefore they may represent knowledge about types, classes, instances, properties, associations, and so on. This opens the gate to more flexible ways of knowledge modelling. Whether a Notion-Chain represents a class or an individual, is determined by Notions.

An example is described in [a8] which is based on the General AEC Reference Model (GARM) that was submitted as an early contribution to ISO 10303 STEP:

NF_Specification {Generic, Specific, Individual}

Generic definitions are parametric descriptions of products or processes. They have variables that are not fully specified. *Specific* definitions are fully specified descriptions. However, they may still have many instances. Each individual product or process is defined by the Notion-value *'Individual'*.

This 'three-level' model is important for design and engineering practices, where the design of a particular component is often repeated. A car may have, for example, four identical wheels; a building may have 24 identical windows. It is also an aid to represent things of which many of the same exist, such as a 'lot'.

The Generic/Specific/Individual principle (named Generic/Specific/Occurrence in the GARM) has been applied in many projects concerning Product and Process modelling, and is also applied in a more restricted sense in IAI/IFC.

4. A Comparison of Notion-theory with Class-based Expressions

4.1. Overview

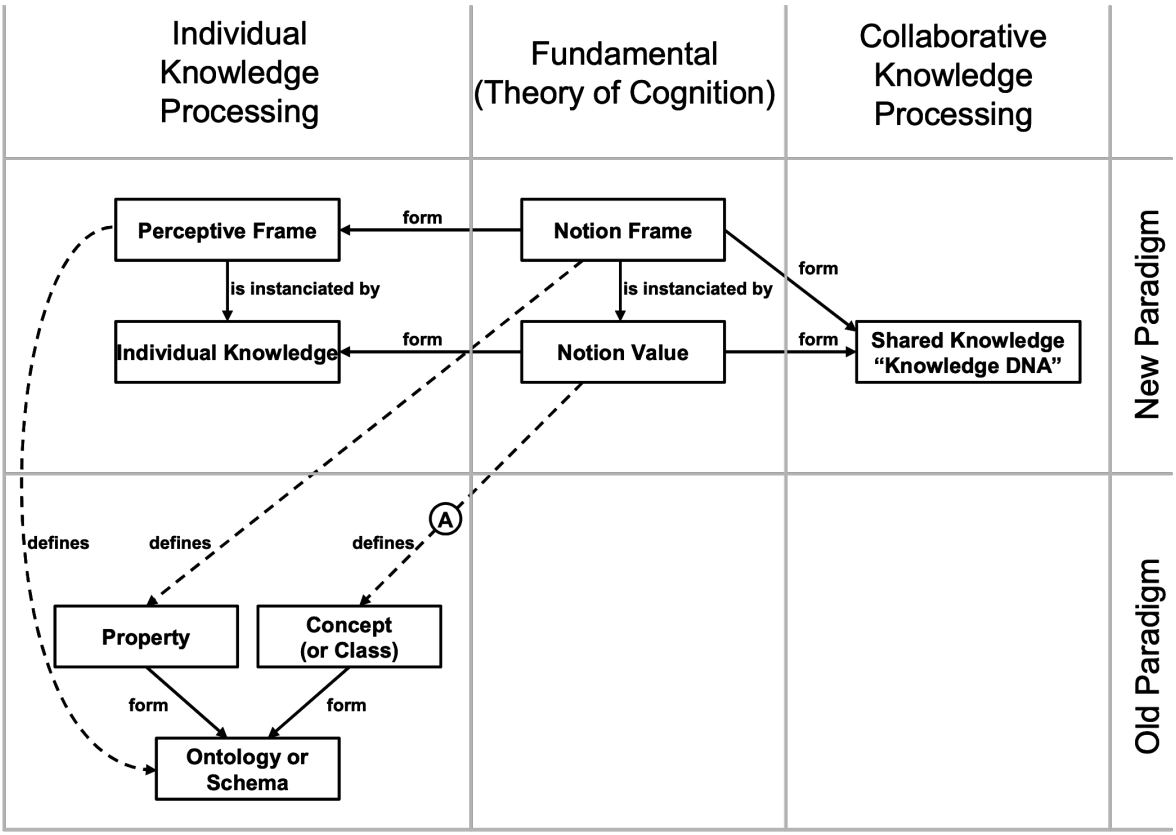

Figure 13. Conceptual graph of the new (top) and old paradigm (bottom), including mappings between the key concepts (dotted lines). The word 'form' means combination, for example 'notion frames form perceptive frames' by combination. The dotted line marked with an A refers the Plato's Theory of Definition, which laid a foundation for Aristotle's work, where Notion Values play the role of differentiae.

4.2. Class- or Type-based expressions are view-dependent

Class- or Type-based expressions are view dependent, because the selection of Notions to be used for classification depend on preferences of the developer. An example was given in Figure 3 using the language Express, but other popular languages such as UML (object-orientation) and OWL (ontologic approach) will give the same result.

A schema or ontology reflects therefore a particular 'view' on a domain of discourse. This makes it practically impossible to exchange or share knowledge between knowledge processors that have a different view, and which use different schemas or ontologies.

In the Notion-based paradigm, classes or types are not pre-defined in a meta-schema. But the knowledge from which classes can be derived is present in Notion-Chains. A Notion-Chain (or Knowledge-DNA) does not prescribe which Notions are used for classification. It is up to a knowledge processor to decide this. Two different knowledge processors may therefore select different Notion-Values for classification. Hence a Notion-Chain may be interpreted differently by different knowledge processors. This interpretation corresponds with the perception (or view) of the knowledge processor on a Domain of Discourse. The Notion-Chain itself is free from interpretation: it is a neutral description by nature.

Notion-Chains may be formed by any grouping of Notion-Frames or -Values, unless semantic rules prevent this explicitly. Such limitations occur in cases of non-orthogonal Notions. An example is given in figure 20.

The more generic or abstract a concept is, the less Notions are needed to define it. The most generic concept is the concept that has no Notion-value. It may be called 'anything' and is by definition meaningless.

Directly underneath this meaningless concept are abstract concepts that are defined by a single Notion-value. There are as many concepts at this first level, as there are Notion-values of orthogonal Notion-Frames. If we need 100 orthogonal binary Notion-Frames to define a conceptual system; then the highest level of a resulting ontology contains 200 different concepts. *Hence, the idea that semantic unification can be realised via an 'upper-ontology', which is the aim of many current initiatives [a27, a28, a29, a30], must be considered as naïve.*

4.3. Class- or type-hierarchies in the class based paradigm

Most conceptual modelling methods support the principle of generalization/specialization. In object-oriented applications this principle results in type-hierarchies; in ontological applications in class-hierarchies. Both approaches adopt an *extensional* definition of the principle, as opposed to the *intensional* definition that is supported by the Theory of Notions. Human and scientific knowledge is primarily (if not entirely) based on intension.

The extensional definition implies the following. Concept *B* is a 'sub-type' or a 'sub-class' of concept *A* if the set of all individuals (or instances) to which *B* refers is a sub-set of that of concept *A*. For example, 'horse' can be defined as a sub-type of 'mammal', and 'mammal' as a sub-type of 'animal'. A hierarchy of such conceptual relationships can be depicted by an inverted tree.

In object-oriented theory, properties are associated with the most generic concepts; i.e. concepts that are highest in the generalization-tree. This is done for reasons of economy. Sub-types inherit properties from their generic parents (called super-types).

Older methods support only tree-shaped hierarchies, but these appear to be too restrictive. It happens frequently that types or classes inherit from more than one super-type or -class. For example, we may define the type 'male animal' and the type 'horse'. The subtype 'stallion' may then inherit from both super-types. This multi-inheritance principle results in lattice-shaped conceptual hierarchies.

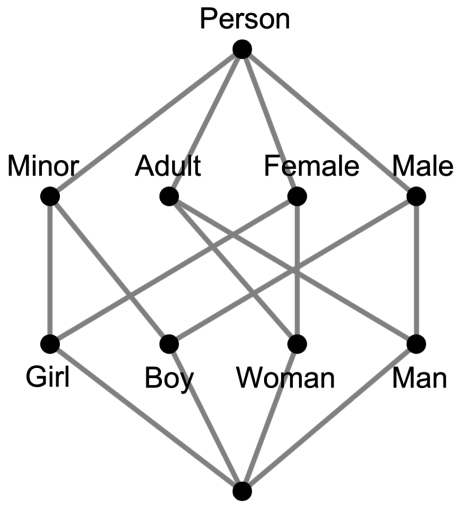

Figure 13. Example of a lattice shaped conceptual hierarchy using Formal Concept Analysis. The same example is modelled with the Theory of Notions in Figure 18.

Two more recently developed methods, namely *Formal Concept Analysis* (a method used for data mining) [a12] and *Description Logic* (a method for the specification of ontologies) [a5] support the description of lattice-shaped class hierarchies. Figure 13 shows an example, using Formal Concept Analysis. Description Logic will give a similar result.

Chapter 5.6 concluded that the number of classes grows exponentially with the number of Notions relevant for definition. Very precise semantic definitions are only possible at the cost of a huge number of concepts, which becomes at some point unmanageable.

This conclusion is supported by results from the European research project IMPPACT (1989-1992). The information reference model of this project is based on five orthogonal Notion-Frames, which exploded into more than 500 different concepts. Implementation of this simple model as a schema for a shared object-oriented database appeared to be very complex and expensive. See the case described in chapter 8.2 and [a13] for more details.

Knowledge expressions using Notion-Chains are less redundant and therefore compact. Because of this, they support fine granularity semantic definitions.

Moreover, Notion-Chains support the addition of new Notion-Frames - and thus the creation of new concepts - by users, without the need for reprogramming of an information system, nor the development of new standards.

Given the volatility and fine granularity of concepts used in industry, fully developed class-hierarchies will in practice be too complex to manage and are at the end less suited for practical use.

4.4. Concept Definitions

The intensional definition of concepts is hardly supported by current modelling methods. In most cases, concepts are informally defined using explanatory text.

Two methods, *IDEF1x* and *UML (Unified Modelling Language)* [a4] , had the possibility to express the differences (differentia) between concepts in a conceptual hierarchy. But as these differentia did not play a role in the formal definition of concepts, they were hardly used in practice.

Formal Concept Analysis [a12, a20] and *Description Logic* [a5] support the expression of differentia for the creation of lattice-shaped class hierarchies. It can be used for the specification of (sub)classes.

Both methods do not enforce full intensional definitions, however; probably for practical reasons. Most ontologies that are published today contain little or no intensional definitions, other than what is required for the modelling of proper conceptual hierarchies. Semantic rules are primarily used for *extensional* definitions, using set-theoretical conditions.

In the new paradigm, Notion-Chains can be considered as expressions of Concepts. Notion-Chains include Notion-Values that intensionally define the Concept. The intensional definition is thus part of the Concept itself. By doing this, information systems do not require predefined schemas or ontologies. The Concepts are 'self-explanatory', and can be classified and interpreted by computer applications at run-time.

This feature is of particular importance for industrial applications, and especially for design applications, where continuously new concepts are being developed.

4.5. Rethinking the Information Architecture

The Theory of Notions, according to the principles described in chapter 4, has an impact on the framework of information systems.

Standards for database management systems are based on a four-layer framework, which is incorrectly called the intension/extension dimension in ANSI/X3/SPARC [a15] and data levels in IRDS [a14]. Also the architecture of standards such as ISO 10303 and IAI/IFC recognize these four layers.

Knowledge DNA reduces this framework to three layers. This is because Knowledge DNA contains the Notions that are today dispersed over the meta-schema, the application schema and the application data.

Figure 14 maps an example from the classic to the new architecture. In this figure, the top-level layer of the architecture (the fundamental layer) is omitted.

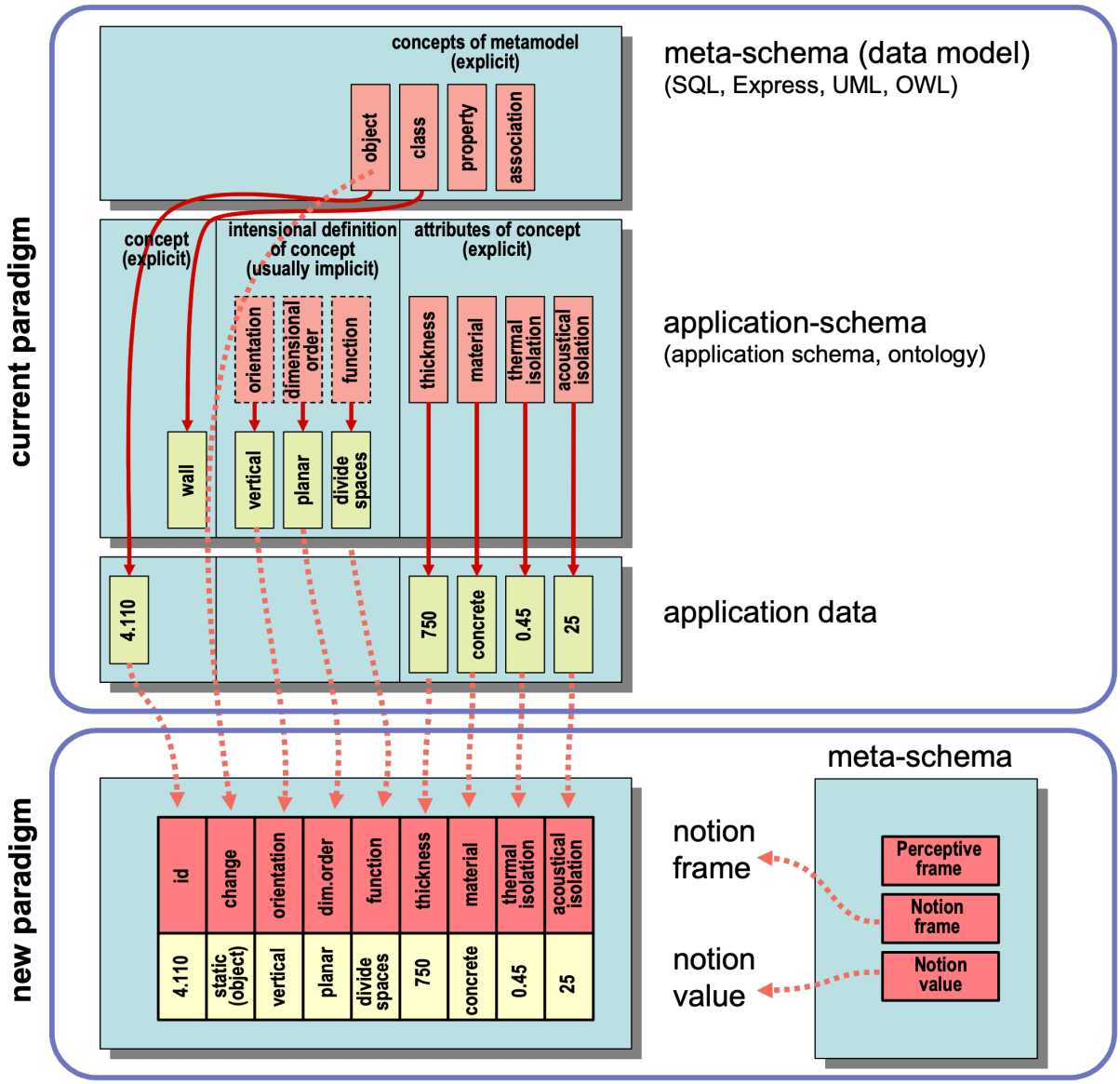

Figure 14. Data and meta-data are in the class based paradigm dispersed over three layers: meta-schema, application-schema and application-data. The intensional definition of concepts, which is implicit in standards such as STEP and IAI/IFC, contains property-values that may be relevant for certain applications. With the proposed idea of Notion-chains (or

Knowledge DNA) Notions and their values are placed on the same level. This makes Notion-chains independent of a particular view on a domain of discourse.

4.6. Class based standards for Product Data Interchange cannot support the interoperability of heterogeneous Computer Applications

The problem of communication between different enterprises, disciplines and computer applications (to be called *knowledge processors*) is often described as the problem of 'many-to-many' interfaces; see figure 15 left. If there are *n* knowledge processors involved, each processor may need to communicate with maximally *(n-1)* other processors, which results in maximally *n*(n-1)/2* communication lines between them. A "neutral model" (or, to be more precise: a "neutral schema" or "ontology", combined with a "neutral format") is assumed to reduce this problem to only *n* interfaces. The "neutral model" was therefore proposed as a solution for the communication problem, and formed the rationale behind the development of standards such as ISO 10303 (STEP) and IAI/IFC.

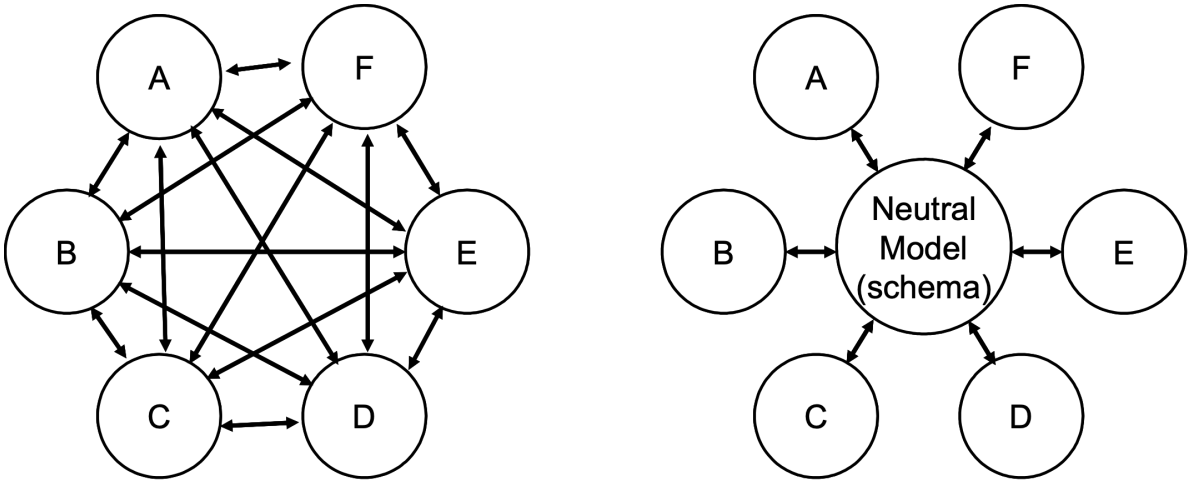

Figure 15. The rationale behind the development of standards such as ISO 10303 and IAI/IFC is to solve the problem of many-to-many interfaces between knowledge processors (left). These standards offer a neutral model for the expression of knowledge. This idea helps only if the knowledge processors have the same 'view' on - and process the same knowledge about - a particular subject. If knowledge and views are different, neutral models cause the destruction of knowledge.

But is this assumption really correct?

The "neutral model" assumes implicitly that all actors involved have the same knowledge about a given domain of discourse. It assumes that the communication problem is only caused by different expressions (or different representations) of that knowledge. In that case, the 'neutral model' offers a single way to express this knowledge.

But what if the actors involved have *different knowledge* about a given domain of discourse? In that case, the "neutral model" cannot solve the communication problem.

Using the current methods for specifications of "neutral models", it is only possible to communicate knowledge between knowledge processors that these processors have in common; see figure 16 left. In case two processors have different concepts, but with a common supertype, they can share knowledge only at the supertype level.

The scope of knowledge can be expressed in terms of Notions, such as shown in figure 16.

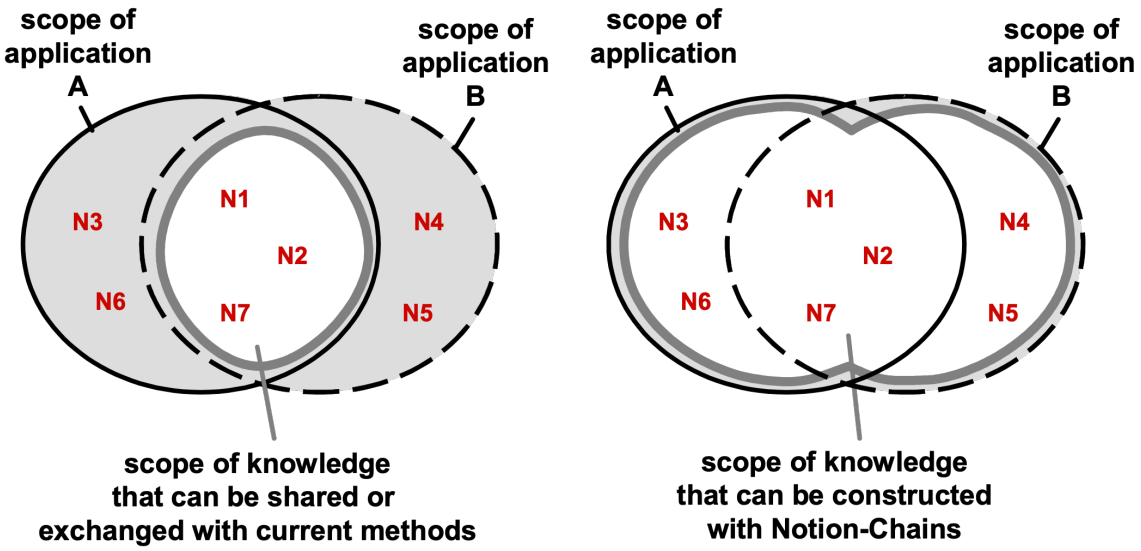

Figure 16.

Even if two processors have the same scope, knowledge may be lost if different expressions for the same concepts are used. An example of three incompatible ways to describe the same thing was given in figure 3.

As current methodologies do not make a distinction between knowledge and expressions of knowledge, they assume that identical expressions (i.e. symbols) refer to identical concepts, and different expressions/symbols to different concepts. Hence, they do not note the difference between the wall of an architect and the wall of a structural engineer, as long as both parties use the name 'wall'. Reversely, they do consider the three different expressions of the blue cube-shaped seat in figure 3 as being conceptually different, although the knowledge about it is actually the same.

With Notion-Chains, these problems should not occur. Notion-Chains are not imported completely by an application. For the input of information, applications copy Notion-Values that correspond with their Perceptive Frames, and ignore Notions that are outside their scope. As output, applications can add Notions or change Notion-Values for which the application is intended and authorized.

Figure 17 shows how two applications with different 'views' on a Domain of Discourse (in this example the Architects view and the Structural Engineering view) communicate via Notion-chains.

Hence, Notion-Chains are constructed in a collaborative process by multiple knowledge processors, where each processor adds its own knowledge.

In case two processors share Notions that would be interpreted in the current paradigm as a common super-type, all other Notions, including those that correspond with concepts at the subtype level, will be retained.

As Notion-Chains are principally independent of view and representation, such as shown for the blue cube-shaped seat (chapters 2.2 and 4.4), there should not be a loss due to different expressions either.

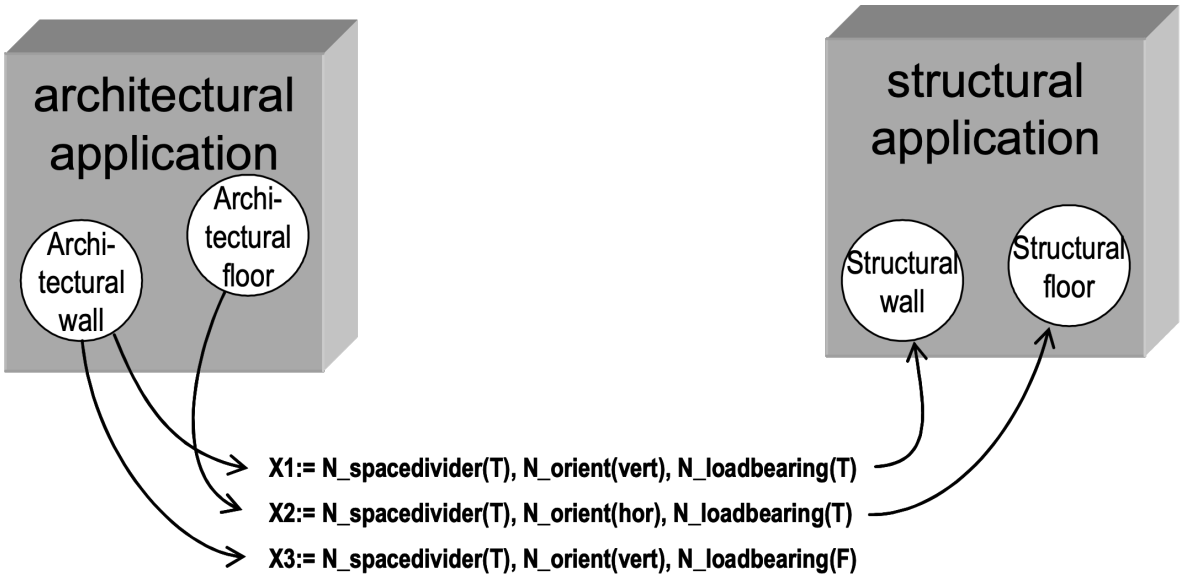

Figure 17. This example shows how two applications with different views on a given subject communicate via Notion-Chains. Objects X1 and X3 are seen as walls by an architectural application, while only X1 is seen as a wall by a structural application.

4.7. Why current integration methods fail

The exchange or sharing of information by computer applications is today only possible by some form of 'integration'. The term 'integration' may have several different meanings.

At the First International Conference on Enterprise Integration Modelling Technology (ICEIMT) in 1992 it was attempted to clarify terminologies [a16]. ICEIMT recognized three levels of integration: (1) "full integration", (2) "unification", and (3) "federation". Fully integrated systems make use of one and the same schema. Unified and federated systems make use of a schema that comprises only shared concepts. The difference between unification and federation is a matter of early and late binding, which is not relevant for this paper.

Full integration is in practice a difficult, if not impossible job. Only if an enterprise and its suppliers use fully integrated applications from one and the same vendor, we may speak of a true integrated solution. Industry uses however applications developed by different vendors, which incorporate different application schema's. Also ISO 10303 (STEP) did not manage to develop a single, fully integrated schema, and had to apply the "unification method" in order to obtain some level of harmonization.

"Unification" is placed between double quotes because Notion-theory makes a distinction between 'extensional unification' and 'intensional unification'. "Unification" such as meant by ICEIMT is, according to the Theory of Notions, 'extensional unification'. This corresponds with 'intensional intersection'. The opposite, 'intensional unification' is not possible with current methods, but is supported by Notion Chains (Knowledge DNA).

> *The architecture of ISO 10303 (STEP) aims at the extensional unification of application-schema's. It identifies three layers of abstraction: (1) Application Interpreted Models (AIM's) which are part of Application Protocols (AP's), (2) Applications Interpreted Constructs (AIC's) which contain partial schema's, shared by two or more AP's, and (3) Integrated Resources (IR's), which comprise schema's that aim at being context-free. Concepts and schema's in these three layers are interconnected via the principle of*

generalization/specialization, so that concepts in the AIM's are specializations of concepts in the AIC's, while the latter are specializations of concepts in the IR's [a25]. The goal of this architecture is to support some form of harmonization between the Application Protocols.

The Semantic Unification Meta Model (SUMM) [a17,a18] aims at the extensional unification of data-models. Hence, it presents a unified model at the meta-schema level such as shown in figure 14. As opposed to STEP itself it addresses semantic *(but extensional) unification.*

Integration means that applications use the same application schema and the same data model. Hence, their concepts are defined using the same set of Notions.

Unification and Federation means that applications share a generalized schema: a schema that is formed by the intersection of their Notions. They can only communicate knowledge which is defined with this limited set of Notions.

As already mentioned above, "unification" and "federation" address the *union of extensions*, but do this through the *intersection of intensions*. This will always result in a loss of knowledge. The inclusion of semantic rules cannot prevent this.

A method based on Notion-chains (or 'Knowledge DNA') enables the *union of intensions*, and should therefore not result in a loss of knowledge. It offers a new way of establishing semantic interoperability.

5. 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.

5.1. 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.

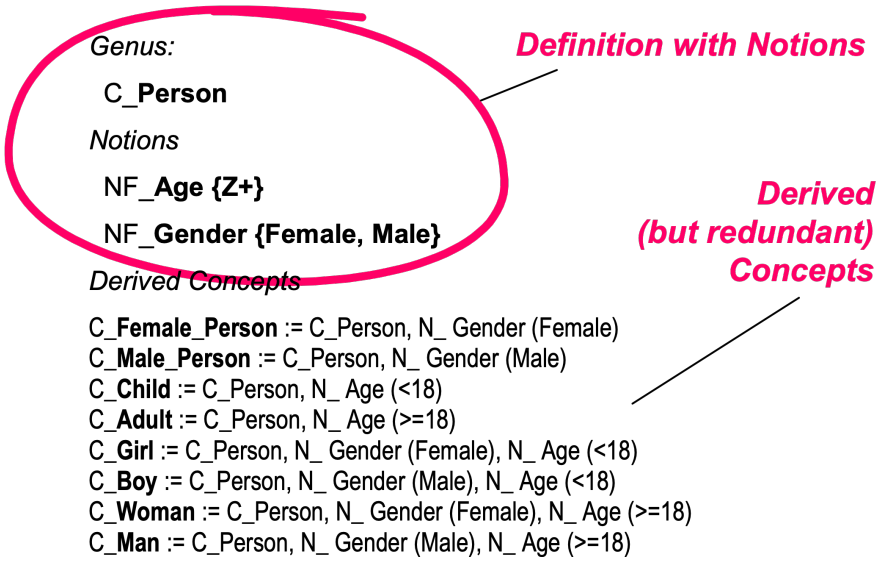

Figure 18. Two Notion Frames for the concept Person result in eight more specific concepts.

5.2. 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 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.

5.3. 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.

5.4. 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.

6. Phased Introduction

The envisioned new way of communicating cannot not be implemented at once. Several aspects need to be investigated further. Most importantly: there is not yet a stable, well accepted system of Notions available.

An evolutionary approach is however possible. Current schema-based standards and ontologies need a much firmer semantic foundation than they have today. A first step in this direction is the identification of Notions as part of ontologic specifications. Developers of standards should be asked to produce formal intensional definitions of types or classes.

The Theory of Notions can therefore be an aid for the development of a next generation ontologies and standards.

Once the required system of Notions for Product Data Technology becomes visible, attempts can be made to implement a pure system based on Notions, such as envisaged in this article.

7. Acknowledgement, Summary and Conclusions

7.1. Acknowledgement

A baseline for the theory presented in this paper was created by the European, EU-funded research project PISA (Platform for Information Sharing by CIME Applications), which ran between 1992 and 1995 [a26]. This project was initiated by BMW AG, Adepa, Cap Volmac, Digital, Caddetc, Pafec, TNO and the University of Karlsruhe, with the goal to address some of the fundamental (methodological) problems that were found in the development of ISO 10303 STEP. Although the current theory was developed many years later, the PISA framework helped to understand the issue better, and pinpointed also to the philosophical baseline of current information technologies as being the cause of the interoperability problem.

7.2. Summary and Conclusions

This paper presents a Notion-based paradigm for the expression of knowledge, as an alternative for the current Class-based paradigm.

The Class-based paradigm has serious limitations, if used for the exchange and sharing of knowledge between computer applications and organizations. This paradigm (1) freezes particular views on a domain of discourse into an information system, (2) limits the scope of shareable knowledge to the *intersection* of the intensional scopes of the applications involved, and (3) forces developers of information systems to distribute pieces of knowledge arbitrarily over two or three distinct levels in the information system architecture.

The current Class-based paradigm is also semantically weak. Recently developed methods support the expression of semantic rules between classes, but classes and other concepts are by themselves not formally defined. In particular, the *intensional definition* of concepts is not well supported and is usually omitted by developers of schemas or ontologies.

The class-based paradigm does not support the expression of fine-granularity semantics either, because detailed schemas or ontologies result in an explosion of different concepts. The number of concepts in a conceptual system grows exponentially with the number of Notions. Existing methods require therefore large, complex and difficult to manage standards.

Further, it is concluded that "neutral models" (i.e. neutral schemas or ontologies) are not adequate for the communication of knowledge between knowledge processors that have different views (perceptions) on - and knowledge of - a given subject. Communication via "neutral models" causes a destruction of knowledge. Hence, "Neutral Models"' are not suited for the communication of knowledge between different disciplines in industrial processes.

Standards based on "neutral models" will not be interoperable for the same reason.

As an alternative solution, this paper proposes the idea of Notion-Chains (or 'Knowledge DNA'). With this approach, the abovementioned problems should disappear. A Knowledge System based on the sharing of Knowledge-DNA (1) permits the sharing and exchange of knowledge between applications that have different views on a domain of discourse, (2) extends the scope of shareable knowledge to the *union* of intensional scopes of applications involved, and (3) brings all relevant knowledge onto a single level in the information system architecture.

Notion-Chains support full intensional semantics and the expression of fine-granularity semantics.

Notion chains should also reduce the need for standards. The inter-operation of heterogeneous applications requires only the sharing of Notion-Frames, not of Notion-Chains. In principle, standards for Notion-Frames can be developed locally, whenever and wherever they are needed.

The new paradigm cannot be implemented at once. A well founded system of Notion-frames is not yet available. However, the required Notion-frames can be found in a transition phase between the current class-based paradigm and the envisioned new paradigm. In this transition phase, Notions can support a more rigorous semantic definition of concepts or classes than what is possible with current methods. The Notion-paradigm unifies logical semantics with linguistic semantics, and bridges the gap between logic and empirical sciences.

8. References

[a1] W. Gielingh; *An Assessment of the Current State of Product Data Technologies*; Journal of Computer-Aided Design 40 (2008) 750–759.

[a2] *The EXPRESS language*. ISO 10303 part 11.

[a3] *Industry Foundation Classes*; www.iai-international.org.

[a4] M.Fowler, K.Scott; *UML Distilled,* 2nd edition; Addison-Wesley, 2000; ISBN 0-201-65783.

[a5] F.Baader, D.Calvanese, D.L.McGuinness, D.Nardi, P.F. Patel-Schneider (eds); *The Description Logic Handbook*; Cambridge University Press, 2002; ISBN 0521781760.

[a6] McGuiness, van Harmelen; *OWL Web Ontology Language*; www.w3.org/TR/owl-features.

[a7] U.Neisser; *Cognition and Reality - Principles and Implications of Cognitive Psychology*; New York 1976; ISBN 0-7167-0477-3.

[a8] W.Gielingh; *Improving the Performance of Construction by the Acquisition, Organization and Use of Knowledge;* ISBN 90-810001-1-X; 2005.

[a9] N.Swartz; *Definitions, Dictionaries and Meanings*; http://www.sfu.ca/philosophy/swartz/definitn.htm; 1997.

[a10] E.E.Smith; *Concepts and Thought*; in: *The Psychology of Human Thought*; Cambridge University press, Cambridge; ISBN 0-521-32229-4; 1988.

[a11] Barnes, J. (1984); *The Complete Works of Aristotle*; Princeton University Press, Princeton NJ, ISBN 0-691-01651-8.

[a12] U.Priss; *Formal Concept Analysis in Information Science*; in: Annual Review of Information Science and Technology, Vol. 40; 2006.

[a13] W.Gielingh, A.Suhm, eds.; *IMPPACT Reference Model for Integrated Product and Process Modelling;* Springer-Verlag, ISBN 3-540-56150-1 / 0-387-56150-1, 1993.

[a14] Burkhart, Dickson, Hanna, Perez, Sarris, Singh, Sowa, Sundberg; *IRDS Conceptual Schema*; working paper ISO/IEC JTC1/SC21/WG3, October 1991.

[a15] Database Architecture Framework Task Group of ANSI/X3/SPARC Database System Study Group; *Reference Model for DBMS Standardization*; NBSIR 85-3173; Gaithersburg 1985

[a16] Petrie, C. (ed); Proceedings of the First International Conference on Enterprise Integration Modelling Technology, Hilton Head, SC., MIT Press, June 1992.

[a17] J.Fulton, J.Tyler, J.Zimmerman, P.Eirich; *Semantic Unification Meta Model (SUMM)*; ISO TC184/SC4/WG3 N81, October 1991.

[a18] J.Fulton; *Enterprise Integration using the Semantic Unification Meta Model*; In: First International Conference on Enterprise Integration Modelling Technology, Hilton Head, SC., MIT Press, June 1992.

[a19] Suhm, A.; *Produktmodellierung in wissenbasierten Konstruktionssystemen auf der basis von Lösungsmustern*; Verlag Shaker, Aachen; ISBN 3-86111-547-3; 1993.

[a20] Wolff, K.E. (1993); *A First Course in Formal Concept Analysis*; in: Faulbaum (ed), SoftStat'93, Advances in Statistical Software 4.

[a21] Barnes, J.; *The Cambridge Companion to Aristotle*; Cambridge University Press, Cambridge, UK, 1995; ISBN 0-521-42294-9.

[a22] ISO 6707-1; *Building and civil engineering*, *Vocabulary - Part 1: General terms.*

[a23] http://www.bouwbesluitonline.nl.

[a24] http://ontolingua.stanford.edu/doc/chimaera/ontologies/wines.daml

[a25] Danner, Sanford, Yang; *STEP Resource Integration: Semantic & Syntactic Rules*; NIST publication NISTIR 4528, Gaithersburg MD, USA, 1991.

[a26] Gielingh, Los, Luijten, van Putten, Velten, eds.; *The PISA project, a survey on STEP*; Shaker Verlag Aachen; ISBN 3-8265-1118-2; 1996.

[a27] Bateman, J.A., Henschel, R., Rinaldo, F. (1995) *The Generalized Upper Model version 2.0*; www.fb10.uni-bremen.de/anglistik/langpro/webspace/jb/gum

[a28] SUMO (2007) *Suggested Upper Merged Ontology*, at: www.ontologyportal.org.

[a29] Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A. (2003) *Wonderweb Deliverable D18,* IST Project 2001-33052.

[a30] Obrst, Cassidy, Ray, Smith, Soergel, West, Yim (2006); *The 2006 Upper Ontology Summit Joint Communiqué*; www.mel.nist.gov/msidlibray/doc/orntology_summit.pdf

[a31] Bakis, N., Aouad, G., Kagioglou, M.; *Towards distributed product data sharing environments — Progress so far and future challenges*; Automation in Construction 16 (2007) 586–595. – survey paper

[a32] Amor, R., Faraj, I. (2000) *Misconceptions of an IPDB*, Proc: The UK National Conference on Objects and Integration, Watford, UK, March 13–14 2000, pp. 124–135. – on view integration see 31.

[a33] Gianluca Colombo, Alessandro Mosca, Fabio Sartori; *Towards the design of intelligent CAD systems:- An ontological approach*; Advanced Engineering Informatics 21 (2007) 153–168 -- on mereology, the part-whole Notion.

[a34] Pan Jiayi, Chin-Pang Jack Cheng, Gloria T. Lau, Kincho H. Law; *Utilizing Statistical Semantic Similarity Techniques for Ontology Mapping — with Applications to AEC Standard Models*; Tsingua Science and Technology, ISSN 1007-0214 35/67 pp217-222, Volume 13, Number S1, October 2008 -- semantic mapping

[a35] DIANE (1993); *Manufacturing Technology Program US Air Force*; DIANE Publishing, 1993; ISBN 1568066392.

[a36] Kosanke, K. (ed.) (1989); *Open System Architecture for CIM (CIMOSA);* Springer Verlag, ISBN 3-540-52058-9, ISBN 0-387-52058-9.

[a37] Richard J. Mayer, John W. Crump, Ronald Fernandes, Arthur Keen, Michael K. Painter (1995); *Information Integration for Concurrent Engineering (IICE) - Compendium of Methods Report. Version 1.0*; Knowledge Based Systems, Inc.; College Station TX 77840-2335

[a38] *ICAM Conceptual Design for Computer Integrated Manufacturing Framework Document* (1984), Air Force Materials Laboratory, Wright-Patterson Air Force Base, OH, USA.

[a39] McDonnel Aircraft Company (1984); *Product Definition Data Interface (PDDI) – System Specification Document*; Air Force Materials Laboratory, Wright-Patterson Air Force Base, OH, USA.

[a40] ISO TC184/SC4; Resolution #2, adopted in March 1986, Washington DC, USA.

[a41] Julian Fowler (1995); *STEP for Data Management, Exchange and Sharing;* Twickenham, UK, ISBN 1 871802 36 9.

[a42] Böhms M, Bonsma P, Bourdeau M, Kazi A S (2009) *Semantic product modelling and configuration: challenges and opportunities*, ITcon Vol. 14, Special Issue Next Generation Construction IT:Technology: Foresight, Future Studies, Roadmapping, and Scenario Planning, pg. 507-525, http://www.itcon.org/2009/33

[a43] Maturana, H.R., Varela, F.J.; *The Tree of Knowledge - The Biological Roots of Human Understanding*; Shambhala Publications; ISBN 0877736421; Revised edition (March 1998).

[a44] Varela, F.J., Thompson, E., Rosch, E.; *The Embodied Mind – Cognitive Science and Human Experience*; MIT-press, ISBN 0-262-72021-3 (1991).

[a45] R.Schuldt; *Universal Data Classification, the key to effective communications*; white paper ISO TC184/SC4/WG1; October 1989.

[a46] R.Schuldt, W.Gielingh; *How to transfer the unambiguous meaning of data*; presentation for ISO TC184/SC4/WG1; January 1990.

[a47] J.Barnes; *The Cambridge Companion to Aristotle*; Cambridge University Press, Cambridge, UK, 1995; ISBN 0-521-42294-9.

[a48] Guarino, Borgo, Masolo; *Logical Modelling of Product Knowledge: towards a well founded semantics for STEP*; European Conference on Product Data Technology, Sophia Antipolis (F), 1997.

[a49] I.Horrocks; *DAML+OIL: a Description Logic for the Semantic Web*; Manchester 2001.

[a50] H.A. Simon; *Rule Induction and Concept Formation*, in: *Models of Thought*; Yale University Press, New Haven/London, 1989; ISBN 0-300-04543-3.

[a51] J.F.Sowa; *Conceptual Structures - Information Processing in Mind and Machine*; Reading, Mass.; ISBN 0-201-14472-7; 1988.

[a52] J.F.Sowa; *Mathematical Background – Revised Version of 'Conceptual Structures'*; www.jfsowa.com, 2005.

[a53] Simpson, John, Robert Hocken, and James Albus, "*The Automated Manufacturing Research Facility of the National Bureau of Standards*." Journal of Manufacturing Systems, May l982.

[a54] Vergeest, J.S.M. and Horváth, I., *Where interoperability ends*; Proceedings of DETC2001 ASME Design Engineering Technical Conference, Pittsburgh, PA, USA

[a55] Goranson, H.T., *Dimensions of Enterprise Integration*; Proceedings First International Conference on Enterprise Integration Modelling Technology, Hilton Head, SC.; MIT Press (1992) ISBN 0-262-66080-6.

[a56] Catarina Ferreira Da Silva, Lionel Médini, Samer Abdul Ghafour, Patrick Hoffmann, Parisa Ghodous, Celson Lima; *Semantic Interoperability of Heterogeneous Semantic Resources*; Electronic Notes in Theoretical Computer Science, Volume 150, Issue 2, 23 March 2006, Pages 71-85

[a57] Hervé Panetto, Arturo Molina; *Enterprise integration and interoperability in manufacturing systems*: Trends and issues Computers in Industry, Volume 59, Issue 7, September 2008, Pages 641-646

[a58] Q.Z. Yang, Y. Zhang; *Semantic interoperability in building design: Methods and tools* Computer-Aided Design, Volume 38, Issue 10, October 2006, Pages 1099-1112

[a59] Josef Ingenerf, Jörg Reiner, Bettina Seik; *Standardized terminological services enabling semantic interoperability between distributed and heterogeneous systems;* International Journal of Medical Informatics, Volume 64, Issues 2-3, December 2001, Pages 223-240

[a60] David Ribes, Geoffrey C. Bowker; *Between meaning and machine: Learning to represent the knowledge of communities*; Information and Organization, In Press, Corrected Proof, Available online 21 May 2009.

[a61] William Wei Song, Paul Johannesson, Janis A. Bubenko Jr.; *Semantic similarity relations and computation in schema integration;* Data & Knowledge Engineering, Volume 19, Issue 1, May 1996, Pages 65-97

[a62] Ma H., Ha E., Chung J., Amor R. (2006). *Testing Semantic Interoperability*; Proceedings of Joint International Conference on Computing and Decision Making in Civil and Building Engineering, Montreal, Canada, 14. - 16. June 2006, p. 1216-1225. (www.icccbexi.ca/PDF/tf193.pdf)

[a63] Pazlar T, Turk Z (2008) *Interoperability in practice: geometric data exchange using the IFC standard,* ITcon Vol. 13, Special Issue Case studies of BIM use , pg. 362-380, http://www.itcon.org/2008/24

[a64] White, W.J.; O'Connor, W.; Rowe, B.; *Economic Impact of Inadequate Infrastructure for Supply Chain Integration;* RTI Project Number 07007.013 (May 2004); *www.nist.gov/director/prog-ofc/report04-2.pdf*

[a65] Brunnermeier, S., Martin, S.; *Interoperability Cost Analysis of the U.S. Automotive Supply Chain*; RTI Project Number 7007-03 (March 1999). http://www.rti.org/publications/abstract.cfm?pub=1390

[a66] Gallaher, M., O'Connor, A., Phelps, T.; *Economic Impact Assessment of STEP in Transportation Equipment Industries;* RTI Project Number 07007.016 (December 2002); http://www.rti.org/publications/abstract.cfm?pub=5250

[a67] Gielingh W.; *General AEC Reference Model (version 4);* Document N329 of ISO TC184/SC4/WG1, Oct. 1988. Published as TNO Report BI-88-150.

[a68] http://www.uml.org/

[a69] Douglas A. Schenck and Peter R. Wilson; *Information Modeling the EXPRESS Way*. Oxford University Press (ISBN 0-19-308714-3), 1994.

[a70] J.J.V.R. Wintraecken (1987). *Informatie-analyse volgens NIAM in theorie en praktijk*. Academic Serivce, Schoonhoven, (2e druk), ISBN: 90-623-3169-6.

[a71] Peter Chen (1976). *The Entity-Relationship Model - Toward a Unified View of Data*. ACM Transactions on Database Systems 1 (1): 9–36.

[a72] Sowa, J. (1997); *Glossary;* NCITS T2 Committee on Information Interchange and Interpretation, Ontology Working Group; http://www.jfsowa.com/ontology/gloss.htm - mappings

[a73] Chang, D., Kendall, E. (2005); *Ontology Definition Metamodel*; Third Revised Submission to OMG/ RFP ad/2003-03-40.

[a74] *Information technology — Common Logic (CL): a framework for a family of logic-based languages*; ISO/IEC IS 24707:2007.

[a75] Berners-Lee, T.; *Semantic Web Layer Cake*, at http://www.w3.org/2004/Talks/0412-RDF-functions/slide4-0.html

[a76] Staub, G.; and Grabowski, H.; *The Challenge of AP Interoperability within STEP*, Proceedings of the ISO TC184/SC4/WG10 Workshop on Architectures for Industrial Data, NIST, Gaithersburg, MD; 7 9 January 1997.

[a77] Ashworth, M., Bloor, M.S., McKay, A., Owen, J.; *Adopting STEP for in-service configuration control* (1996) Computers in Industry, 31 (3), pp. 235-253. doi: 10.1016/S0166-3615(96)00054-1

[a78] *Information Modeling Manual IDEF Extended (IDEF1X).* ICAM Project Report (Priority 6201); Air Force Systems Command, Wright-Patterson Air Force Base, OH; 1985

[a79] Nijssen, G.; and Halpin, T. *Conceptual Schema and Relational Database Design: A Fact Oriented Approach*; Englewood Cliffs, NJ: Prentice Hall; 1989

[a80] Danner, W. F. (1997). *Developing Application Protocols using the architecture and methods of STEP Fundamentals of the STEP Methodology*. NIST Internal Report (NISTIR) 5972, National Institute of Standards and Technology, Gaithersburg, MD

[a81] Batres, R., West, M., Leal, D., Price. D., Naka, Y.; *An Upper Ontology based on ISO 15926*; www.matthew-west.org.uk/Documents/batres-final.pdf

[a82] Sowa, J.F.; *Knowledge Representation: Logical, Philosophical, and Computational Foundations*; Brooks Cole Publishing Co., Pacific Grove, CA; ISBN 0-534-94965-7.

[a83] Wilson, P.; *PDES STEPs Forward*; IEEE Computer Graphics and Applications, vol. 9, no. 2, pp. 79-80, Mar./Apr. 1989, doi:10.1109/MCG.1989.10007

[a84] Wilson, P.R. and Kennicott, P.R. ; *ISO STEP Baseline Requirements Document (IPIM);* ISO TC184/SC4/WG1 Document N284, NIST, Gaithersburg, Md., 1988.

[a85] Kemmerer, S. J., *STEP, the Grand Experience*, NIST Publication Number: SP 939 - July 01, 1999, www.mel.nist.gov/msidlibrary/doc/stepbook.pdf.

[a86] Tenenbaum, J., Weber, J., Gruber, T.; *Enterprise Integration: lessons from SHADE and PACT*; Proceedings First International Conference on Enterprise Integration Modelling Technology, Hilton Head, SC.; MIT Press (1992) ISBN 0-262-66080-6.

[a87] Semantic Web based Open Engineering Platform - Product Modelling Ontology; http://www.swop-project.eu/swop-solutions/ontologies/pmo

[a88] Hars, A., Kosanke, K., Langemeyer, J., Petrie, C.;, (1992). *The Model/Application Link*. Proceedings of the First International Conference on Enterprise Integration. Petrie, C. J., (ed.), pp 17-22, ISBN 0-262-66080-6.

[a89] R.Burkhart, J.Fulton, W.Gielingh, C.Marshall, C.Menzel, C.Petrie, D.Zoetekouw, (1992). *The Notion of a Model*. Proceedings of the First International Conference on Enterprise Integration. Petrie, C. J., (ed.), pp 17-22, ISBN 0-262-66080-6.

[a90] W.Gielingh; *Rethinking Conceptual Structures and their Expression*; Proceedings of 22nd CIB W78 conference, Dresden, ISBN 3-86005-478-3; 2005.

Possible references, not yet integrated.

[Gleason 61] H.A.Gleason, *An introduction to descriptive linguistics*. Holt, Rinehart & Winston; New York, 1961.

[Guarino,Giaretta 01] N.Guarino, P.Giaretta*; Ontologies and Knowledge Bases, towards a terminological clarification*; LADSEB-CNR report, Padova Italy (update 2001).

[Guarino 97] N.Guarino; *Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction and Integration*; Summer School on Information Extraction, Frascati (I),1997.

M.R. Cutkosky, R.S. Engelmore, R.E. Fikes, M.R. Genesereth, T.R. Gruber, W. Mark, J.M. Tenenbaum, J.C. Weber, PACT: an experiment in integrating concurrent engineering systems, IEEE Computer 26 (1) (1993). – client/server type of integration for providing fit for purpose data (??) see 31.

[b] W.Gielingh, D.Beeckman, S.Braun, P.Willems; *The PISA Product and Process Model;* CAD 94 conference, Paderborn, April 1994.

[d] Guarino, Borgo, Masolo; *Logical Modelling of Product Knowledge: towards a well founded semantics for STEP*; European Conference on Product Data Technology, Sophia Antipolis (F), 1997.

[e] N.Guarino; *Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction and Integration*; Summer School on Information Extraction, Frascati (I),1997.

[h] Tae-Sul Seo, Yoonsook Lee, Sang-Uk Cheon, Soonhung Han, Lalit Patil and Debasish Dutta. *Sharing CAD models based on feature ontology of commands history*. International Journal of CAD/CAM. 5(1), 2005.

[i] Lalit Patil, Debasish Dutta, and Ram Sriram. *Ontology-based exchange of product data semantics*. IEEE Transactions on Automation Science and Engineering, 2(3):213–225, July 2005. Semantic Interoperability of Heterogeneous Semantic Resources

Bronsvoort, W.F., Noort, A.; *Multiple-view feature modelling for integral product development* (2004) CAD Computer Aided Design, 36 (10), pp. 929-946. doi: 10.1016/j.cad.2003.09.008