CONSOLIDATING SEMANTIC INTEROPERABILITY IN CONTEMPORARY ARCHITECTURAL PARADIGMS

version: v0.1-0

Paul Brandt,
Eindhoven University of Technology; Netherlands Organization of Applied Scientific Research TNO, Den Haag, The Netherlands,

Eric Grandry,
Luxembourg Institute of Science and Technology, Esch-sur-Alzette, Luxembourg,

Twan Basten,
Eindhoven University of Technology, Eindhoven, The Netherlands,

Abstract

Background: *Access-and-Play sIOP is the next glass ceiling in [interoperability/IT-based business collaboration]. We can think of two approaches to break through the ceiling, i.e., using either strong AI (a system that can* think *and has a* mind, *in the philosophical definition of the term) or weak AI (a system that can only* act *like it thinks and has a mind (Searle 1980)). Strong AI is not yet available, while weak AI, despite its current applications in Semantic Web or ontologies, has not yet been embedded in contemporary software architectural paradigms. Current approaches towards sIOP can be considered accepted folklore.*

Objective: *The objective of this study is to identify and define the (weak AI based) fundamental guidance towards access-and-play semantic interoperability in contemporary architectural paradigms.*

Method: *Our approach is based on the discipline of semiotics. After identifying semiotic shortcomings in MDA and view-based architectural paradigms and their subsequent definition as missing concerns, we develop the necessary guiding architectural principles. We finally consolidate their fundamentals as an ISO-42010 Architecture Viewpoint to disclose them for the various architectural paradigms. [We evaluate these principles by designing a reference architecture and proof its use in sIOP between two software agents.]*

Results: *The semiotic approach/discipline demonstrates/proves semantics in software to be the result of a reciprocity between data and the software code that operates on them. The major shortcomings in architectural paradigms to account for semantic interoperability are their negligence of semiotic fundamentals and, particularly, the absence of an explicit ontological commitment that stands at the root of semantics. Therefore, the concern about a semantic loose coupling should be added to the architectural paradigms. The supporting principles are (i) semantic transparency, (ii) semantic separation of concerns, and (iii) explicit computational semantics. In view-based architectures their consolidation implies a new semantic view, while the MDA paradigm requires an ontological commitment on M3. Both paradigms need to include a semantic alignment processing mediation capability.*

Conclusions: *Access-and-play sIOP can be achieved when considering semiotic fundamentals and adding loosely coupled formal semantics to contemporary architectural paradigms.*

Chapter 1

Introduction

Never before, data were so ubiquitous, and managed access to external data was so easy. Because current ICT is unable to *use* all that same external, non-native data as access-and-play service, agility in business collaboration is hampered in all domains. For instance, consider the following (allegedly real) example of an interoperability failure.

> A German steel producer upgraded its industrial process robot. Since the majority of the steel production process is dependent on time, from a security point of view the decision was made to not rely on their own internal clocks but to use the German *Braunschweig Funkuhr* time radio signal as source for the exact time instead. At the end of April 1993, when Germany went on summer time, the computer clock of the steel producer went from 1:59 AM to 3:00 AM in one minute. This resulted in a production line allowing molten ingots to cool for one hour less than normal. When the process controller thought the cooling time had expired, his actions splattered still-molten steel, damaging part of the facility.[1]

In this simple example a tiny difference in the meaning of `time` between the steel producer and the national time provider hampered interoperability to the extend of damaging the steel facility. This tiny difference rooted in the assumption by the steel producer that `time` expressed a continuous scale whilst for the Braunschweig Funkuhr, `time` denoted instant clock time for that time zone and therefore represented a non-continuous scale. In order to achieve that both collaborators, here the Braunschweig Funkuhr and the steel producer, can actually *use* their peers data, the need exists to design and implement wrappers that remove any inconsistency between the variations that may occur in terms, structures, dimensions and what have you. Many such variations exist, leading to a range of failures in so-called *semantic interoperability* (sIOP) and Section/Appendix ## provides for a short overview of sIOP-faults. Unfortunately, it is fundamentally impossible to automate the production of wrappers, because we need a genuine *understanding* upfront, which computers still cannot do.

The most disconcerting consequences of a lack of (automated) sIOP are time-to-deliver, flat interoperability failures, and even seemingly correct but quite invalid data analysis probably leading to devastating system behaviour. Current sIOP implementations are essentially based on the (time-consuming) process of establishing a (local) convention on the semantics of the terms that are exchanged during collaboration, requiring custom solutions and collaboration-dependent software adaptations. Such conventions can be considered a semantic monolith, which makes dealing with data outside the monolith impossible, unless again a time consuming (months) semantic adoption process is applied. Moreover, these semantic conventions consider semantic heterogeneity as a bug instead of a feature necessary to achieve semantic accuracy. But still, this conventions-based approach towards sIOP is accepted folklore in ICT. In view of the large uptake of the Internet, the

[1] Source: http://catless.ncl.ac.uk/Risks/14.57.html#subj1, accessed May 20, 2018

Internet of Things (IoT), cloud computing and big data, and in view of economical pressure to intensify enterprise collaboration, we consider this approach “too little, too late”.

In comparison, scalability was a big architectural concern in the past, requiring custom solutions as well. In response to this concern, scalability was standardised in the form of architectural patterns, and finally totally embedded and hidden into the infrastructure. Similarly, sIOP can be considered the architectural concern of this decade, and we first need to provide a standardised solution pattern to address semantic concerns, before we can embed it in a technological infrastructure so that sIOP becomes transparent to the developer. Where scalability resulted in a huge increase in performance-demanding applications against a fraction of the original costs and effort, business agility will emerge once the semantic monolith is removed and semantic services exist at the infrastructural level. Then sIOP becomes an access-and-play operation that achieves sIOP in due time with data not anticipated for during software design, at any point in their life cycle. Metaphorically speaking, we consider sIOP as a *bridge* overarching a (semantic) gap: with *bridgeheads* on each side of the gap, with a *spanning* resting on them to structurally support the bridge and its traffic, and with a *roadway* enabling the crossing of the traffic. Finally, architectural *principles* provide the necessary guidance to the architect for the various design decisions that effectively result in a particular bridge over a particular (semantic) gap. Our contributions to consolidating semantic interoperability in software architectures are fivefold, and represented as architectural principles and concerns, as follows:

- *Principles*: We base sIOP on establishing loose-coupling at the semantic level by introducing principles on semantic separation of concerns and semantic transparency (Section 6), and show how these principles can be operationalised;
- *Semantic concerns (bridgehead)*: Abstracting semantics from a tacit software implication into a tangible, computational and distinct artifact provides us with the potential to connect to it and to make comparisons with the semantic artifact of the peer software agent. Based on the discipline of semiotics, we explain the shortcomings of the current approach towards software semantics that rely on information models and information views. Instead, we provide for a fundamental notion on the application of ontologies [and ontological commitment] to remedy current semantic shortcomings, and we show [its/their] proper position in the total architecture (Section 3);
- *Weak AI concerns (spanning)*: Since “strong AI” does not yet exist, sIOP remains in demand of human intervention in order to reconcile the semantic differences between collaborating software agents. However, human intervention is time consuming. We reduce the necessary human intervention to complement weak AI to a task that suffices to achieve sIOP, viz. authoring semantic alignments only (Section 4);
- *Mediation concerns (roadway)*: We provide for a prototypical implementation of a mediator as the necessary component to automatically translate data when transferred between the collaborating software agents (Section 5);
- *ISO42010 Architecture Viewpoint*: We formulate the architectural consequences of the above concerns as a specific sIOP Viewpoint in order to consolidate sIOP for contemporary architectural paradigms (Section 7);

Based on these contributions we [argue/defend] that access-and-play sIOP can be embedded and hidden in infrastructural services when considering semiotic fundamentals and adding loosely coupled formal semantics to contemporary architectural paradigms. To that end, we first describe the semiotic fundamentals in Chapter 2.

Chapter 2

The semiotic and philosophical foundations of semantics

Synopsis: Purpose of this section: To establish an informal but concrete notion on (i) the semiotic triangle and (ii) the relationsbetween its nodes.

2.1 Semiotics

Synopsis: Present concrete notion on (i) the semiotic triangle and (ii) the relationsbetween its nodes. Show abstraction/generalisation as stacking of triangles vertically, and representational metalevels as stacking them horizontally.

The discipline of semiotics is the study of signs, reality and meaning. The meaning of a *token* (text, graphics, sound) ultimately relates to what it denotes in reality (the *entity*), whilst this relation cannot be deferred from the shape, structure or other characteristics of the token itself due to its total arbitrariness. In the early 1900s, De Saussure used a dyadic model that stressed that the token and the entity in reality were as inseparable as the two sides of a piece of paper (Saussure 1959). This piece of paper he called the ***semiotic sign***, denoting the whole. This 'self-containment of the sign' remains one of the major principles of semiotics. Constructing the semiotic sign from its distinct parts is called ***semeiosis***. The token, in combination with their ability for semeiosis, provides humans with the tool to converse with each other. The tokens provide humans with a vocabulary, the semeiosis makes them understand about what entities they talk about. Semantics, then, emerges as a result of the semeiosis that connects the distinct parts of the inseparable semiotic sign.

Sanders Peirce (in: Sowa 2000) developed another model to further investigate the semeiosis part of semantics. He built a triadic model of the semiotic sign, including a representamen (the token) and object (the entity), and introduced the *interpretant* which expresses the mental and, hence, individual sense making. This triadic model of the semiotic sign was coined by Peirce as the *semiotic triangle* (ibid.), depicted in Figure 2.1(a), and subsequently used and modified by Ullmann (Ullmann 1962), Ogden and Richards (Ogden and Richards 1989), and many others, also in recent years (Kuhn 2009). We introduce our modifications, as depicted in Figure 2.1(b), which mainly focus on naming conventions in IT architectures, as follows.

Where Peirce denotes the *object*, we prefer the use of *entity* due to the ambiguous nature of the former in IT modelling and architectures. We consider an entity to stand for a thing or an event, but also a category of entities, a relation between entities and a property of an entity. We will refer to the *interpretant* component as the *conceptualisation*, to underline the individual conceptualisation that is being formed during requirements analysis and conceptual modeling. And we prefer the use of *token* over *representamen*, and consider it both an atomic element as a particular composition of atomic elements. We include denotations for the edges that

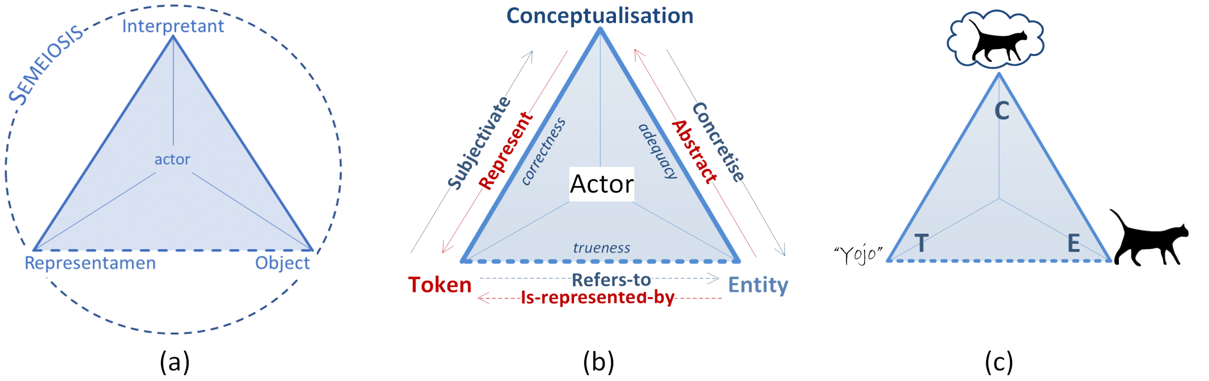


Figure 2.1. The triadic model of the semiotic sign, according to Peirce (a), and modified by us (b). Example (c) shows the concept of a cat named "Yojo"

are connected to the conceptualisation vertex, and use names that underline the individual and mental nature of the sense making. Note that these names are directional, and must be read as the transformations that takes place in that direction. Finally, we add the causal characteristics that the edges represent, introduced by (Ogden and Richards 1989), as *adequacy*, *correctness* and *trueness*. Observe that the connection between the token and the entity is drawn as a dashed line to stress that its existence is indirectly only through the conceptualisation and does not exist in any direct means. Whenever we use "sign" we refer to the semiotic self-contained sign. A well-known example of a sign is depicted in Figure 2.1(c), which shows that when we talk about "Yojo", our cognition interprets it as our cat.

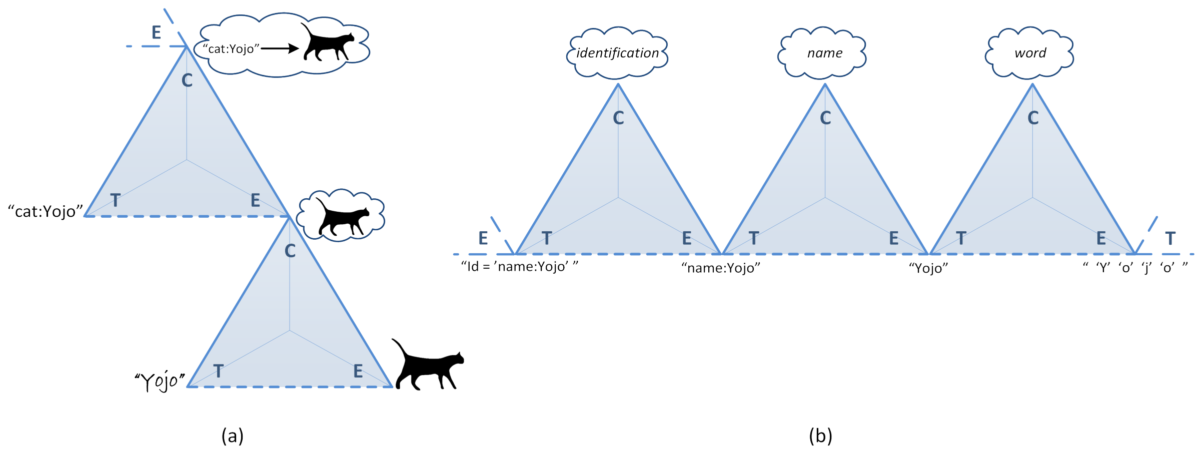


Figure 2.2. Linking triadic models together.

Peirce also recognised that multiple triangles could be linked together in various ways (in: Sowa 2000). By stacking them together, as depicted in Figure 2.2(a), a conceptualisation is made of "representing an entity": the original concept of a **cat** named "Yojo", depicted in Figure 2.1(c), is being conceptualised as the concept of a **cat named "Yojo"** and represented by **cat:Yojo**. In (Eco 1976), Eco uses the term *unlimited semeiosis* to refer to the succession of stacking signs that emerge from that, ad infinitum. We consider unlimited semeiosis as addressing a dimension of comprehension about abstraction and generalisation, with an eventual finish in the ultimate **Thing** concept. Linking the triangles horizontally results in different representational metalevels, depicted in Figure 2.2(b): From right to left, the characters "Y" "o" "j" and "o" are conceptualised as a single **word** and represented as "Yojo", which is conceptualised as a **name** and represented as "name:Yojo", which is conceptualised as an **identifier** that might be represented as "Id='name:Yojo'".

2.2 Ontological commitment

Synopsis: Purpose of this section is to show the relevance of a (modelling) language: a language is used to convey distinctions. The distinctions that are aticulated by a language are denoted as its ontologcal commitment. Modelling languages need to show distinctions that are of relevance to the purpose of modelling. The predominant purpose of modelling is to describe (a particular domain in) reality by distinguishing the entities of interest. The ontological commitment for a modelling language therefore should be of ontological nature, "(...) not in order to know *what there is*, but in order to know what a given remark or doctrine, ours or someone else's, *says* there is" [@Quine:1953er]. The ontological square or its extension into the ontological sextet [describe it] can be considered the most basic one, which have been specialized into several flavours, all called upper-level, or top-level, or foundational, ontologies, e.g., UFO, BFO, DOLCE and more.

Apart from this strict semiotics notion, semantics are also influenced by philosophy and need consensus on the question "when are we committed to the existence of certain entities?". It is relevant to acknowledge that humans maintain assumptions and background knowledge, both of which impact the semeiosis and, hence, semantics. This is where the conceptualisation plays an important role as frame of reference to our understanding, also denoted as the *ontological commitment*. Consider the following classical sentences:

- Sentence 1: "the king of France is bald". This sentence has got a useful meaning, being that in case of a king of France, the dear fellow is as bald as a coot. We did not say *that* a king of France exists, nor *that* bald men exist; we only used these two phrases to *refer* to things that might or might not exist. Hence, we do not need to commit to the existence of a king of France, nor to the existence of bald men, before we can formulate the sentence that *if* there is a king of France, he must be bald. Still, and despite being a meaningful sentence, "the king of France" does not refer to something (as France is a republic), and therefore we cannot render the truth of the sentence.
- Sentence 2: "the species *leo* (lions) is extinct". Despite the similarity with the linguistic structure of the previous sentence, in this case we do need to commit to the existence of the species leo. The reason for this is that we do not imply here something about one individual but about something that many individuals have in common, i.e., that which defines an entity as member of the leo species. Without accepting that "there is something" that we consider characteristic of the species of leo and leo alone, we defy the group as a whole, i.e., the universal type that each of them instantiates. And if we defy the existence of the universal type, we defy that "there is something". But if we defy that "there is something", it would be nonsense to even speak about any of its qualities, in this case extinction. Therefore, by defying the existence of the species, we defy the meaning of the sentence itself. We therefore are forced to commit to the existence of the species.

The contrast exemplified by these sentences shows that only when we commit to something (here "the species *leo*"), the theory that we propose (here "is extinct") can *refer* to that something in order to *establish its validity*; clearly, in our world the theory is invalid, i.e., renders `False`, given the many counter examples of lions being alive. We consider this the philosophical cornerstone for semantics: we can assess the semantic validity of any proposition if and only if the underlying ontological commitment can be referred to. Furthermore, any assessment towards semantic interoperability of two semantic theories cannot be made without an assessment of the similarity between their underlying ontological commitments. Note, however, that "We look to (. . .) Ontology not in order to know *what there is*, but in order to know what a given remark or doctrine, ours or someone else's, *says* there is" (Quine 1961).

Chapter 3

Bridgehead: Semantics

Synopsis: Purpose of this section:

1. From a semiotic perspective, explain what we mean with semantics in software agents, i.e., the reciprocity between data and data processing code (both represented as Term from the semiotic Sign). 1. Establish that for representing semantics, descriptive models (i.e., ontologies) trump prescriptive models (all 42010 models). 1. Conclude that ontologies need their place as single point of reference (trueness) in architectures, and identify their relationship with the rest of the architectures, i.e., all other prescriptive models. Note the issue on Open World Assumption (ontologies) and CWA (prescriptive models).

<!- end of list ->

3.1 What is software semantics

Synopsis: The purpose of this section is to explain software semantics as the "reciprocity between data and software code", and show that to some extent, set theory can replace the conceptualisation node. Conclude that the reciprocity between data and data processing code represents the smallest (atomic) semantic monolith.

Additionally, optionally, show:

1. the relationship with Grice's distinction in semantics as "what is said" and "pragmatic meaning" 2. OO as initial implementation to consolidate this reciprocity, and the class as implementation of the atomic semantic monolith;

We take the position that strong AI is not yet available, if ever (Xiuquan Li and Tao Zhang 2017), and conclude that weak AI is essentially a token-based machine without the ability to close the gap between token and reality. Also called the Grounding Problem (Harnad 1990), addressing this fundamental distinction in software engineering about semantics is at best extremely narrow (Steels 2012), or not present at all (Cregan 2007). This implies that the semiotic triangle is denied its conceptualisation vertex, and the sign remains incomplete. This is confirmed by the software engineering discipline herself implicitly, since it consistently speaks of 'models that represent reality' without factoring the conceptualisation into the equation, e.g., "*a model is a representation of reality intended for some definite purpose*" and similar quotes that are collected by (Aßmann et al. 2006). Consequently, the edges that connect the conceptualisation remain vague or necessarily conflate on the relationship between the model and reality, depicted in Figure 3.1. This beheaded sign cuts-off our "knowledge about our given remark or doctrine *says* there is". We have removed the "ontological level" (Guarino 1994), and with that, the fact that "terminological competence can be gained by formally expressing the ontological commitment of a knowledge base" (ibid.). Since we make do with weak AI and its beheaded sign necessarily, this suggests that genuine semantics can not ever exist in current software agents.

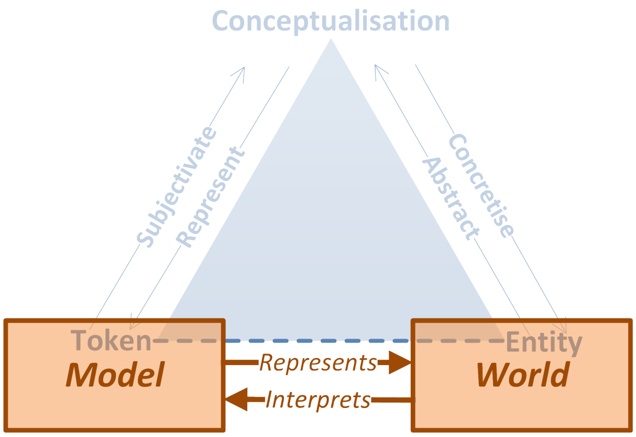


Figure 3.1. *Software engineering applies a beheaded semiotic triangle in which its edges remain vague or conflate in the single relation between model and reality.*

During the use of a software agent the semeiosis is taken care of by the human-in-the-loop, viz. the end user at the human-machine interface (HMI) whom interprets the tokens that are displayed (subjectivation). During development of a software agent the semeiosis is taken care of by another human-in-the-loop, viz. the software engineer whom implicitly performs the conceptualisation and explicitly represents this conceptualisation into tokens, i.e., *models*. Consequently, all models are representations of engineers' conceptualisations. From the many models that software engineering typically generates we focus on a pair of models: the information or data models that refer to the *information entities* in reality, paired with the process or business models that represent the *event entities* that operate on the information entities. Such data processing is in its bare form nothing more than tokens that follow a specific language grammar. This bare form is a representation of its quintessence, viz. a run-time notion on the proper way to operate on the data. Together, these models comprise the smallest atomic union that can represent meaning, indicated by (Grice 1989) as the semantic meaning, or "what is said" and the pragmatic meaning, what we like to understand as how it relates to our intentions with it.

brandtp, 8/30/2018 Make use of 42010 terminology

brandtp, 8/30/2018 ... ditto ...

Definition 3.1 (Atomic semantic monolith)
An Atomic Semantic Monolith (ASM) denotes the smallest, highest grained pair of models (a data model and a data processing model) that remains faithful to the entity in reality that it refers to. ■

At the modelling level, semantics still exist by virtue of the designer. However, when the software agent is subsequently compiled, its binary code originate from the process model of the model pair (operations, algorithms), and the memory allocation for the data originates from the information model of the model pair (size, format, encoding). At this binary level the software engineer has left the building, and with her the conceptualisation vertex and the subsequent capability for semeiosis and, thus, semantics. In other words, at binary level we have lost the capability to verify the semantic coherence between the code and the data while the reciprocity between data and software code determines the semantic validity of the data processing. For instance, consider a data element t to represent temperature, and an algorithm to establish fever, e.g., `t > 37.0` $\rightarrow$ `fever`. The one and only means to keep the software from failing is that both the data and the algorithm (i) are expressed in the same unit of dimension ($°C$ in this example), apply the same (ii) resolution and (iii) accuracy, to name a few obvious constraints. We, therefore, take the stance that semantics can only exist in software by virtue of the semeiosis by the human-in-the-loop, while in the software agent itself semantics are necessarily reduced to the reciprocity between data and software code. Still, the software agent acts as transport medium for the semantics as intended by the software engineer to the semantics as experienced by the end user at the HMI. We therefore consider the coherence between data models and data processing models essential for enforcing the software agent to maintain a semantic valid reciprocity between binary code and the data it operates on.

This leads to the definition of a (normative (Greefhorst and Proper 2011)) design principle to its effect:

Design Principle 3.1 (Semantic coherence principle)
Establish explicit coherence between the models that are contained in a semantic monolith.

Type of information: *business*

Quality attributes: *(semantic) accuracy, reusability, manageability, understandability*

Rationale:

- *Semantics in software agents are necessarily reduced to, and emerge from, the reciprocity between the data and the binary code that operates on them;*
- *Without explicitly addressing – at modelling level –* ***all*** *facets that influence the coherence between the data on the one hand, and the operations that apply on them on the other, the software agent cannot guarantee to maintain the reciprocity between them at the binary level;*
- *Without maintaining the reciprocity between binary code and the data it operates on, the semeiosis performed by the end user on the result of the data processing and their subsequent semantics cannot be guaranteed to be similar as intended by the software engineer.*

Implications:

- *The coherence principle is a necessary condition for supporting semantic interoperability;*
- *The scope of semantic validity & accuracy is addressed explicitly and can be referred to;*
- *Reuse of data often implies reuse of the data processing code, and vice versa. Having established explicit coherence improves the quality of data and code reuse, and facilitates the verification that the scope of the semantic validity & accuracy applies in the new context as well;*
- *manageability ...?*
- *understandability ...?* ◇

Coherence between models can be established with use of a single unique reference against which the truth of the expressions of both models can be verified. In semiotics, this single unique reference is considered reality, as indicated in Figure 2.1(b) by the *trueness* characteristic. Except as toy example in (Steels 2012), this is clearly not possible. The *correctness* characteristic is the only alternative left, taking the conceptualisation node as its principle point of reference. This is exactly what the mathematical branch of *formal semantics* achieves (Gamut 1991; Genesereth and Nilsson 1987) with its three main characteristics, viz. connecting (i) an abstract syntax of a language to (ii) a domain of interpretation (usually a set theoretic framework) by defining (iii) an interpretation function from the abstract syntax onto the set theoretic framework. In terms of the semiotic triangle, Figure 2.1(b), this implies the following:

(i) the *representation* node represents models that can be formulated by use of an abstract syntax (and grammar) as its modelling language. In this reading, a model is a particular constellation of tokens that represent a particular state of affairs;
(ii) a particular *conceptualisation* can be mathematically formulated as a specific constellation of (unnamed) individuals, sets of individuals, and sets of sets;
(iii) the *subjectivation* edge can be formulated as the interpretation function that assigns a mapping from modelling language tokens onto the set elements, enabling the evaluation of a specific model against the intended conceptualisation from (i).

Formal semantics thus provides a means to formulate a particular conceptualisation as principle point of reference to establish the coherence between two models. In the remainder of this text we will refer to the formulation of the reference conceptualisation as a *conceptual model*.

In conclusion, we explain software semantics as the reciprocity between data and software code, realised by maintaining the coherence between pairs of data and data processing models, by applying formal semantics to formulate a particular conceptualisation as semantic reference, and an interpretation function from the data and operation models to that reference.

<Elaborate on OO to consolidate the reciprocity; take the class as example of a semantic monolith, the minimal, atomic one.>

3.2 Explicit semantics

Synopsis: The purpose of this section is to establish that for representing semantics, descriptive models (i.e., ontologies) trump prescriptive models (all 42010 models)

First make the distinction between descriptive and prescriptive models [@Henderson-Sellers2012], and in [@Aßmann2006]: "Specification models focus on the specification, control, and generation of systems; ontologies focus on description and conceptualization (conceptual modelling) of things. Both kinds of models have in common the qualities of abstraction and causal connection."

FIrst argument shows that (i) the trueness of a model is laid in reality, which is impossible to achieve, and that (ii) the next best we can achieve is establishing the correctness of a model against its conceptualisation node. Finally, observe that (iii) 4 issues will influence the correctness of the model. Then conclude that the best tool to control these issues are the logics from descriptive models (viz. ontologies) at the one hand, and its use as ontological commitment for the prescriptive models (viz. the 42010 models)

We have seen how the cognitive quality of the conceptualisation can be substituted with a formulation in set theory. The resulting conceptual model essentially remains a representation, albeit a mathematical one. One can argue that such substitution does not resolve the grounding problem, and appropriately so. Still, mathematics provides for a very exact way to express oneself, reducing the ambiguity that comes implicitly with any other language. Furthermore, logical constructs used at the syntactical level can be interpreted into set theoretic operations, facilitating the evaluation of complicated expressions. And thanks to mathematics we can also indicate the exact issues that exist with conceptual modelling, depicted in **??**.

[Four different types of construction issues that come with formal semantics][def:constructissues]

This begs the question what we mean with model, and what criteria we should adopt to represent a conceptual model. This is especially relevant since in contemporary architectural paradigms models are being used as first class citizens to the architectures, MDA and ISP RM/ODP alike.

An appropriate definition for ontology is given by **Guarino:1998wq** as a "logical theory accounting for the intended meaning of a formal vocabulary".

However, because computers are unable to conceptualise or concretize, the connection between the software's conceptualisation and the entity does not exist. This "missing link" in artificial intelligence is called *the grounding problem*, named after the inability to ground a conceptualisation in what it refers to in reality. In literature, two exceptions to this rule exist, which we discuss in the box text below. Our stance towards these exceptions is that they are interesting, however currently irrelevant towards the resolution of semantic interoperability due to their many practical shortcomings in implementing an actual connection between the entity and the conceptualisation.

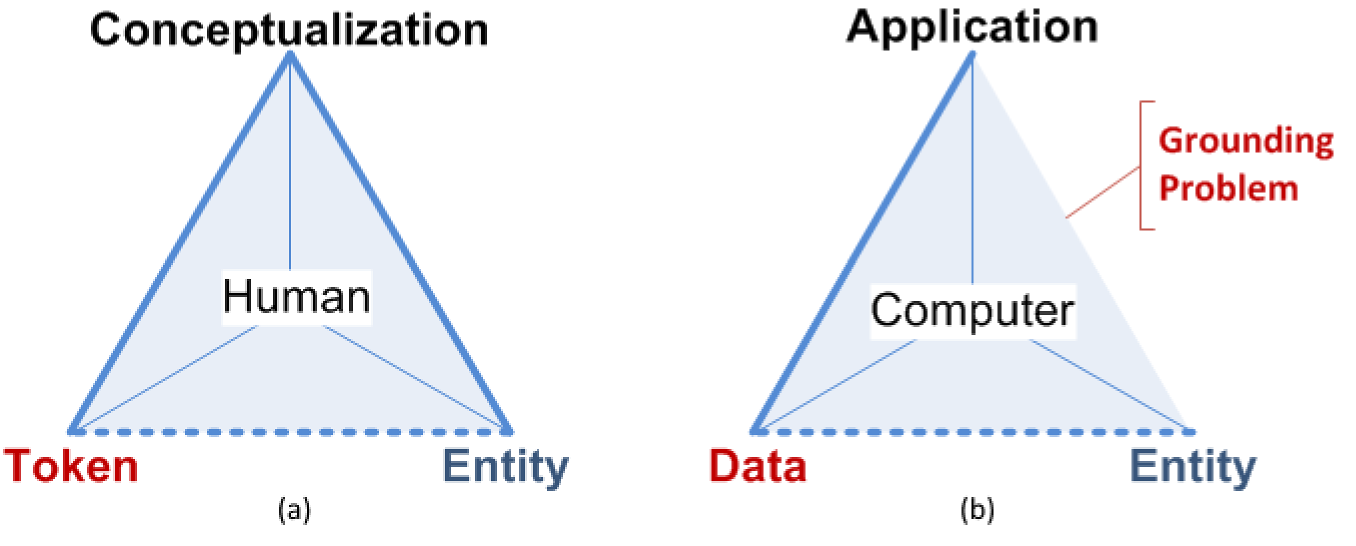


Figure 3.2. Semiotic differences in semantics for humans and computers

This is known as the *problem of reference*, a manifestation of the *grounding problem*.

In information systems, addressing the distinction between terms and reality is extremely limited (Steels 2012), or not present at all (Cregan 2007).

Artificial intelligence (AI) tries to tackle the grounding problem by building some form of understanding, also known as "strong AI". However, strong AI is expected to emerge on the long term only, if ever (Xiuquan Li and Tao Zhang 2017). Its counterpart "weak AI", characterised by logic and reasoning, relies on language only and can therefore never make the step to reality on its own (Scheider 2012).

Hierin duidelijk maken wat de verschillen zijn tussen modellen en ontologie. Semiotiek (eigenlijk de semiotische driehoek) gebruiken wij als methode om te verklaren wat semantiek is bij mensen en bij computers. En zonder semantiek in de architectuur, geen sIOP.

CONCLUSIE: Architectures will not be able to facilitate semantics and, hence, consolidate sIOP without including semiotics. Assumption 1: root cause for sIOP issues is the grounding problem: GP leads to absence of semantics, absence of semantics leads to absence of sIOP. Fact: Strong AI could solve GP, but doesn't exist Fact: Weak AI is based on language only, and can never solve GP on its own Observation: Humans can solve GP, semiotics explain why Fact: Semiotics studies relation between language (terms) and meaning

Thus, weak AI is our only option for the time being in order to achieve semantics and sIOP.

We therefore cannot neglect the existence of the grounding problem and its semiotic origins. Nevertheless, we do. For instance, when we are asked to explain how we address the grounding problem in the design of our software agent, we can't; when we are asked to point at the semantics parts in the code of our software agent, we can't. The same question however about, e.g., its scalability, will render a lecture with adequate references to the underlying architecture. We thus remain at a loss of how to engineer semantics into software agents. However, without a clear understanding on semantics and its contribution to the software agent, we are lacking the bridgehead within the software agent that is fundamental to the semantic interoperability bridge.

In fact, this is a question of philosophy while ICT is "only" faced with its consequence: computers can deal with language only and have no clue about reality.

It therefore remains impossible to ground the applied terms in reality, denoted as the *grounding problem*. Its resolution is a major subject in strong AI and in (geographic) information science in general **Scheider:2012tj**. Although **Steels:2008tr** provides for an alternative (weak AI) solution, that only shows the need to refer to general stance is that the grounding problem remains a big challenge .

3.2.1 Modeling

```
Synopsis:  Purpose of this section: present a list of differences, and
notably, that formal semantics can to some extend take the role as con-
ceptualisation Refer , [@Henderson-Sellers2012]

Conclusions: 1 - ontologies are more appropriate artefacts for concep-
tual models than system models 2 - as in [@Aßmann2006]: "Specification
models focus on the specification, control, and generation of systems;
ontologies focus on description and conceptualization (conceptual mod-
elling) of things. Both kinds of models have in common the qualities of
abstraction and causal connection."
```

In order to fully appreciate the results of our work, it is necessary to acknowledge the commonalities and notably the differences between ontology and models. This is especially relevant since in contemporary architectural paradigms models are being used as first class citizens to the architectures, MDA and ISP RM/ODP alike.

- Identify differences between prescriptive models and descriptive ontologies here? Maybe only "the" requirements that follow from the coherence principle.
- Second essential grain is the ontological commitment.

It is thus one of the responsibilities of the software engineer to keep the information model and the process model in strict coherence with each other. In the world of Model Driven Engineering (MDE), the object orientation (OO) way of formulating models indeed seems to enforce adherence to this principle. However, this is not completely true since OO enforces only some form of correctness between the methods and the data object they act upon.

Regarding the grain's process models, this falls apart in two other model categories, viz. models that directly operate on data in order to infer other facts (conclusions), and models that act on those conclusions and initiate some business oriented activities. We will call the latter the *pragmatics* of the software agent, and we will not further elaborate on that. Regarding the data model and the inference model, we explain them to represent the two subtypes of meaning according to (Grice 1989; Schulz 2007): firstly, *semantic* meaning, meaning as conveyed by the tokens, explained by Grice as *what is said*, are directly related to the data models. Secondly, *pragmatic* meaning, explained by Grice as what a speaker adds by implication and/or intention when uttering a sentence in a particular context (Smith 2003 p. 50), are directly related to models that infer new data. For instance, consider a heart rate reading of 128BPM. The semantic meaning that is carried is exactly what is said, viz. the number of times a heart beats during one minute. In case of the pragmatic meaning, though, the same bits will refer to a very different health condition in the context of a sleeping elderly (triggering an alarm) than in the context of a sleeping new-born (indicating perfect health). We like to consider the pragmatic meaning as the meaning that is required to draw conclusions, demanding the specific *context of use*.

However, as soon as the semeiosis has taken place, As we have seen in the previous section, linking between triangles exists horizontally and vertically alike. In this case, subsequent software engineering will conceptualise the models in different representational metalevels as well as built different levels of abstractions and generalisations.

the the *signification* of the software agent. Although genuine semantics does not exist in software, a software agent can definitely fail in its signification part. When such failure happen

one of her major responsibilities are to assure that the data and the code operating on that data remain coherent with each other. In fact, one of the main arguments for the introduction of object-orientation (OO) was to dispose of a means that could enforce this "data-code coherence" in a natural way. Introducing a class to represent a particular entity in reality, and its methods that operated on it, enforces the software engineer to maintain one single conceptualisation that is represented by a set of two tightly coupled tokens: the data and the code operating on that data.

Interestingly, according to (Grice 1989; Schulz 2007), two subtypes of meaning exist:

Elaborate on the reciprocity as software semantics

Chapter 4

Spanning: Alignments

Rephrase: despite the notoriously difficult philosophical questions involved, semantic interoperability can be seen as an engineering problem, namely that of effectively constraining interpretations towards the ones that are considered allowable (Kuhn 2009).

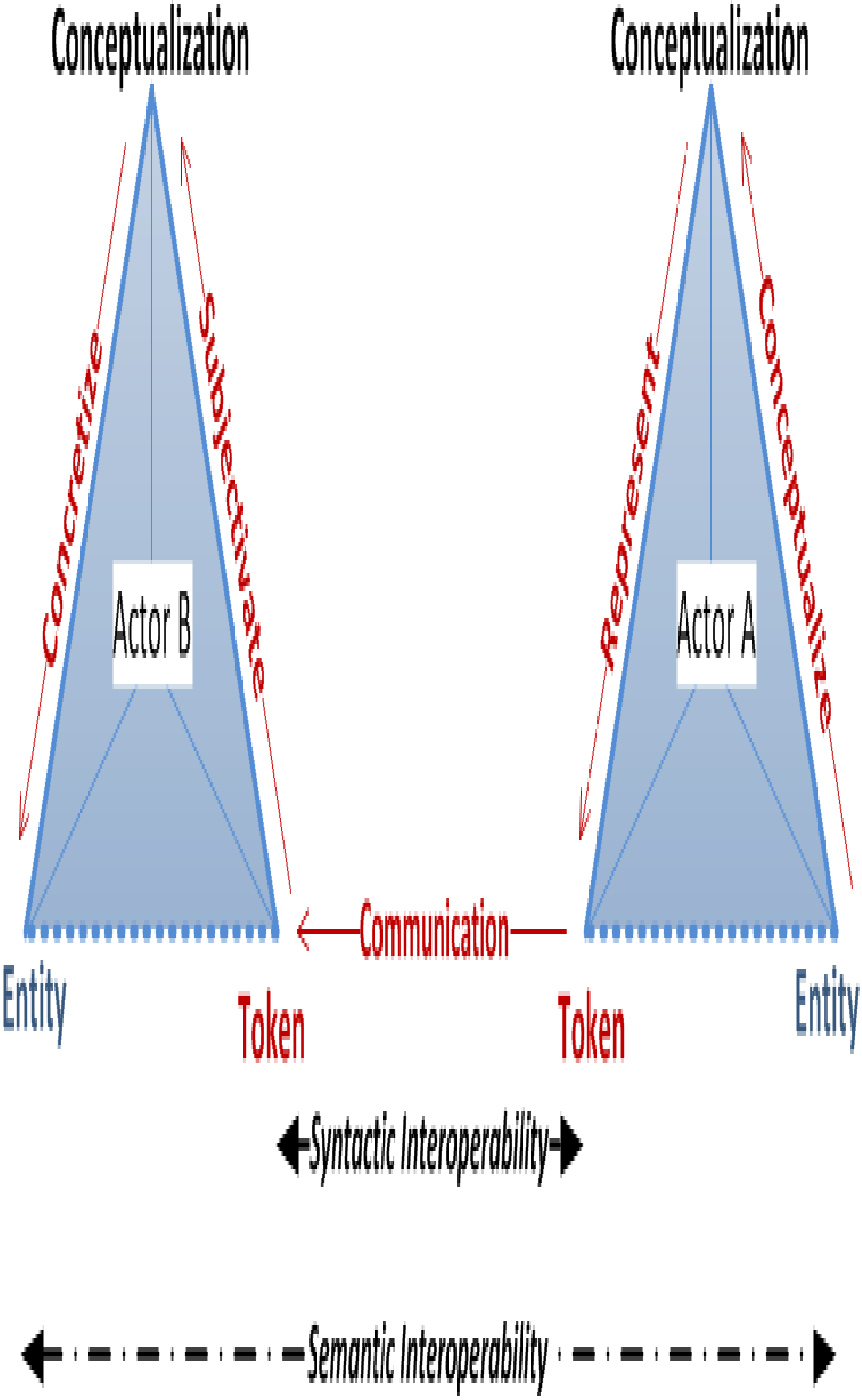


Figure 4.1. The various forms of interoperability

4.1 What is semantic interoperabiity

Synopsis: 1. Explain sIOP:

1.1. that consequence of data exchange = breaking the atomic semantic monolith = breaking the reciprocity by partitioning the data from its original code, and

1.2. that sIOP demands that despite this partitioning the reciprocity between the code of the receiving agent and the external data shall be reinstalled.

2. We therefore need to extend the semantic coherence Principe into a sIOP coherence principle with a sIOP rational that the result of the semeiosis on receiving agent A′ does not conflict withthe outcome of the semeiosis on agent A: Without maintaining the reciprocity between binary code and the data it operates on, the semeiosis performed by software engineer A′ on the result of the data processing and their subsequent semantics cannot be guaranteed to be similar as intended by the software engineer.

3. Explain the difference with semantic standards which are basically large semantic monoliths

4. Introduce Principle: ontological commitment as minimal standard for sIOP, "(...) not in order to know *what there is*, but in order to know what a given remark or doctrine, ours or someone else's, *says* there is" [@Quine:1953er]

Purpose of this section:

1. Explain sIOP as:
   1. that data exchange breaks the atomic semantic monolith = breaking the reciprocity by partitioning the data from its original code, and
   2. that sIOP demands that despite this partitioning the reciprocity between the code of the receiving agent and the external data shall be maintained, and
   3. that Phantom semantics (i.e., the difference in the original reciprocity and the new reciprocity) is to be prevented at all costs.
2. Explain that standards are large semantic monoliths, and introduce Principle: ontological commitment as minimal standard for sIOP. "We look to bound variables in connection with ontology not in order to know *what there is*, but in order to know what a given remark or doctrine, ours or someone else's, *says* there is" (Quine 1961)

4.2 Expicit sIOP by alignments

Synopsis: Thus:

1. Follow the coherence principle and conclude that the models from which *external* data and *receiving* data processing code are derived, need

to be brought into coherence with each other.

2. The coherence principle already enforced a single unique reference for each agent. Re-installing coherence demands a semantic alignment between those single unique references.

3. The purpose of that alignment is to establish how the truth of expressions that are formulated in terms of agent A, can be estabished by using formulations in terms of agent A′ against the single unique reference from A′.

4. The language that is used for expressing the alignment should be fit for its purpose. Refer to EDOAL [@Scharffe2011] as the currently most complete one.

Chapter 5

Roadway: Mediation

Synopsis: Purpose of this section: Establish requirements for a *generic* mediator.

With mediation we denote the process of transcribing a data term that originates from Agent A into a data term that matches a term familiair to Agent A′, based on both agents′ ontologies and the alignment between them. The main issue here is that although many different types of relation can be defined between the concepts of ontology A and A′, e.g., superset of, a transcription of a token from A into a token from A′ is a complete replacement and, hence, implements an equivalence relation. In [@Brandt2018b], we show a semantic valid transcription process. The requirements of a mediator are:

1. Being a generic service

2. Fully defined by two ontologies and their alignments

3. Allows for semantic valid transcriptions only, where ′validity′ refers to absence of inducing phantom semantics.

4. Appropriate behaviour for non-translatable content, which should apply only as result of an incomplete alignment, a logical incorrect alignment, or attempts to communicate content that is considered irrelevant for the receiving agent.

Chapter 6

sIOP Principles

Synopsis: Purpose of this section:

* Show the semantic architecture as an additional layer that is orthogonal to current layers, expressing a separation of concerns between syntax and semantics (see [@Brandt:2013jh])

* Define loosely coupled semantics as a result of applying the 2 principles 'semantic separation of concerns' and 'semantic transparency' (see [@Brandt:2013jh])

* Repeat the Coherence Principle, the sIOP Coherence Principle, and the Ontological Commitment Principle, and show how they fit in the semantic architecture

The main (business) requirement is to achieve sIOP as quickly as possible, with as minimal effort as possible, for collaborations that had not been foreseen and consequently could not be anticipated for during design time of the (two or more) software agents.

Consequently, the software agents have been developed totally and completely independent from each other.

This raises the following semantic concerns:

I. Loosely coupled semantics: * Define semantics once during software design phase, and achieve sIOP many times with many different peers * EW Dijkstra: Connected but as independent as possible: peer agents shall only publish *what* is meant with semantics, not *how* it is represented I. Scalable sIOP: * Variable in number of peers * Variable in level of semantic heterogeneity

1. Define the four semantic concerns (see Figure 6.2 for three related ones):
   i. Explicit and computational semantics by *conceptual modelling*: Bridgehead
   ii. Managed and controlled sIOP by *semantic reconciliation*: Spanning
   iii. Automated sIOP by *semantic mediation*: Roadway * Address semantic issue about the non-equivalence between an alignment and a transcription (refer to (Brandt et al. 2018))

Achieve loosely coupled semantics

Loose coupling is founded on principles about (i) separation of concerns, and (ii) transparency:

- Principle *Separation of concerns*:
  - Classical:

    i. Decompose system in parts
    ii. with minimal functional overlap
  - Semantical:
    i. Separate your own semantics (i.e., conceptualisations, viz. let each software agent manage its own abstraction from reality)
    ii. from establishing sIOP

- Principle *Transparency*
  - Classical:
    i. Agnostic to *how* its functions are being achieved
    1. Communicate with minimal mutual dependency
  - Semantical:
    i. Agnostic to *how* semantics are being achieved
    ii. Communicate with minimal syntactic dependency, i.e., without agreements on semantic representation

Formulate the principles in the format according to (Greefhorst and Proper 2011)

Ad.semantic separation of concern. Where in its classical application the result of applying the principle is that atomic functions are defined, designed and implemented only once and remain unique, in its semantic application the result of applying this principle is that every software agent maintains its own semantics. Semantics are, therefore, distributed all over the place. This seems counterintuitive or even plain wrong, however, it is necessary for complying with the concern about semantic scalability (in support of heterogeneous semantics). Besides that, it is a direct consequence of the demand to allow for independent software development

```
* Principle: specify ontological commitment as basic
* Refer to (and partly reuse?) semantic architecture from [@Brandt2013]
```

1. Complement weak AI with human brain:
   - use AI where possible (computational semantics for software agent; supporting semantic reconciliation)
   - use human brain where necessary (but not more): ontology engineering @ design time; alignment authoring @ pre-runtime

*** Achieve Scalable sIOP***

Ensure that different semantic topologies remain possible:

i. Star alignments (central domain ontology, aligned to local ontologies) for relative stable and homogeneous domain semantics
   - Good: easy semantic governance
   - Bad: very similar to a semantic monolith

ii. Mesh alignments (bilateral alignments) for very dynamic and heterogeneous (domain) semantics, or low number of peers
   - Good: quickly established bilateral sIOP; granularity-on-demand, viz. intricate where necessary, coarse-grained where possible
   - Bad: complicated semantic governance

iii. Mix-n-Match (coarse-grained star-alignment with specialised bilateral alignments) for the 70% bulk * Good: controllable semantic governance; after central alignment, quickly established bilateral sIOP * Bad: slightly more complicated mediation due to double alignment support

1. Explain: difference between ontologies and models
   i. Models lack an elaborate ontological commitment, domain ontologies naturally evolve on foundational ontologies (that express an ontological commitment)
   ii. For prescriptive models, truth lies in themselves (deterministic software); ontologies are descriptive models for which the truth lies in reality (good for semantics)
   iii. ontologies have open world assumption (semantic under-specification), models have closed world assumption (data remain consistent)
   iv. Models specify systems, ontologies conceptualise entities
   - Try to also connect the onto/model distinction with above principles
   - Induced problem: from OWA (domain ontologies) to CWA (information/data models)
   - Principle: use domain and business ontologies for CIM (Aßmann et al. 2006)
   - Principle: prescribe requirements model with concepts that are defined by CIM-DO and CIM-BO (see Figure 6.1).

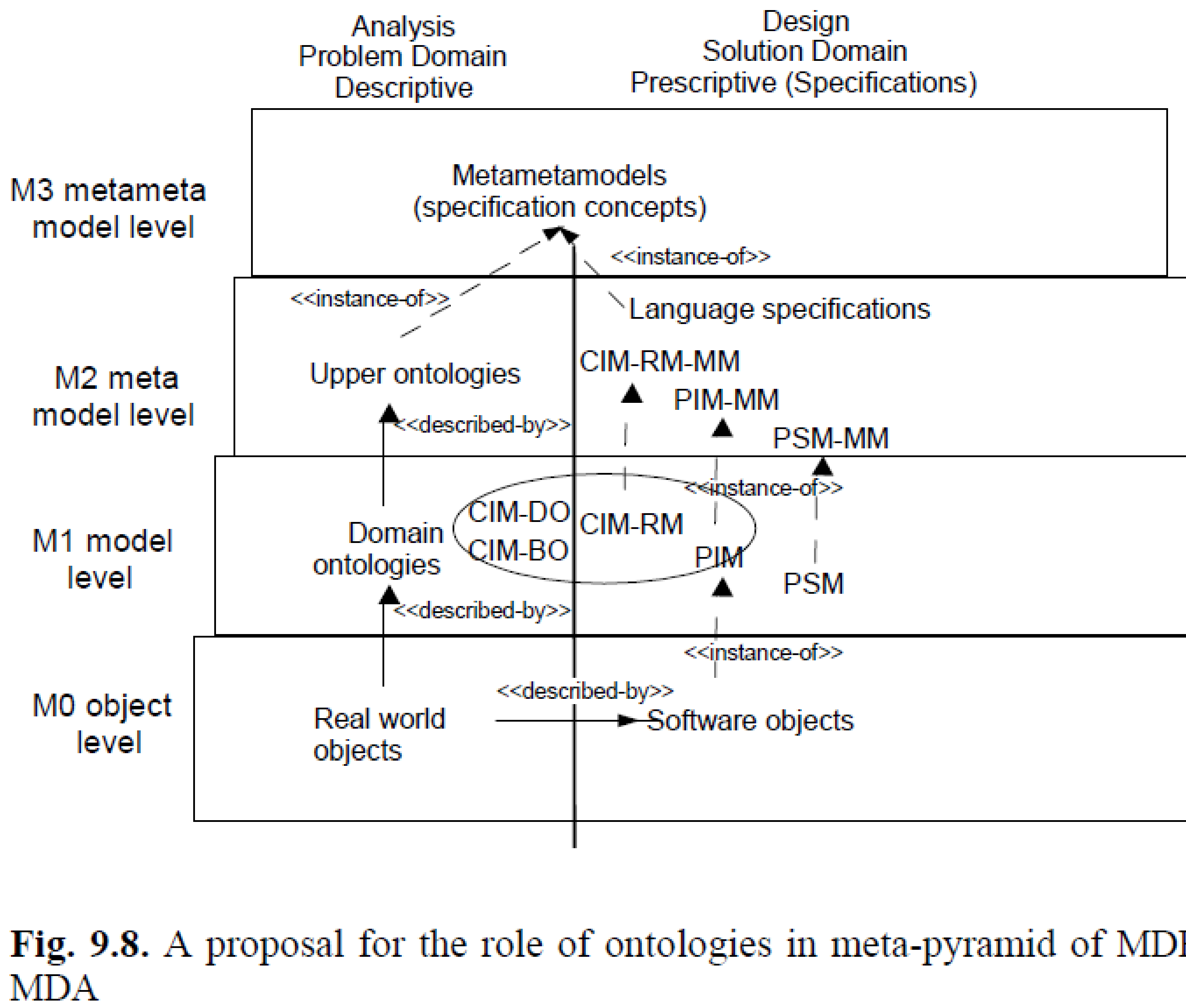
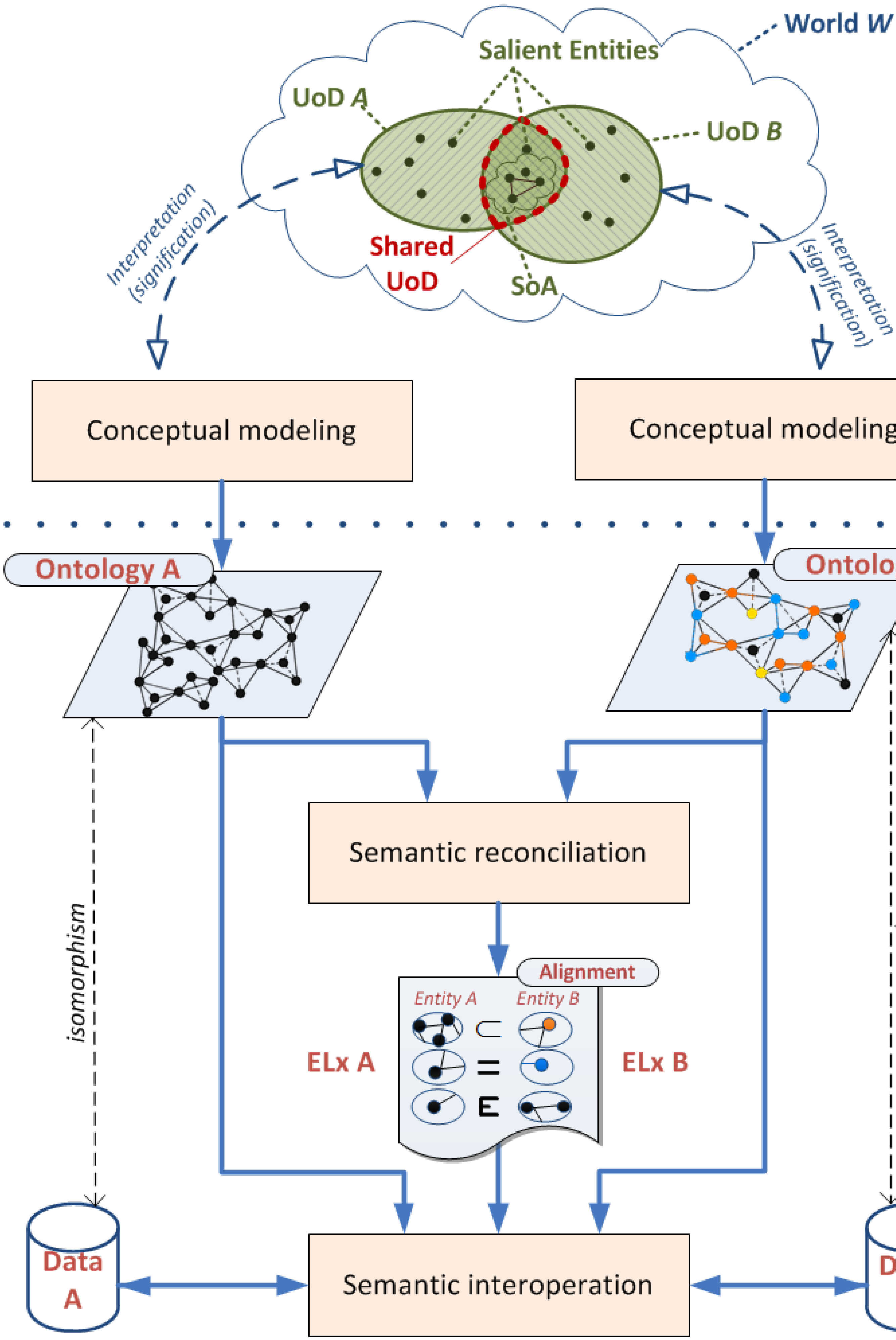


Figure 6.1. The use of ontologies in MDA, from (Aßmann et al. 2006)


World W
Salient Entities
UoD A
UoD B
Shared
UoD
SoA
Interpretation
(signification)
Interpretation
(signification)
Conceptual modeling
Conceptual modeling
Ontology A
Ontolo
Semantic reconciliation
Alignment
Entity A
Entity B
ELx A
ELx B
isomorphism
Data
A
D
Semantic interoperation

Chapter 7

Architectural viewpoint on sIOP

Synopsis: Consolidate the ideas on the bridgehead, spanning, roadway and principles into an additional ISO42010 Architectural Viewpoint (sIOP) that summarises all previous Sections as concerns on semantics and sIOP. Preferrably written by Eric.

Chapter 8

Related work

[1] M. B. Almeida, C. P. Pessanha, and R. Barcelos, "Information Architecture for Organizations: An Ontological Approach," in Ontology in Information Science, C. Thomas, Ed. IntechOpen, 2018, pp. 1–27.

[2] S. Yang, J. Guo, and R. Wei, "Semantic interoperability with heterogeneous information systems on the internet through automatic tabular document exchange," Inf. Syst., vol. 69, pp. 195–217, Sep. 2017.

[3] U. Aßmann, S. Zschaler, and G. Wagner, "Ontologies, Meta-models, and the Model-Driven Paradigm," in Ontologies for Software Engineering and Software Technology, C. Calero, F. Ruiz, and M. Piattini, Eds. Springer-Verlag Berlin Heidelberg, 2006, pp. 249–273.

[4] C. Atkinson and T. Kühne, "The Essence of Multilevel Metamodeling," LNCS, vol. 2185, pp. 19–33, 2001.

Chapter 9

Discussion & future work

Synopsis: Address shortcomings that we discover throughout writing the sections.

Conclude that by identifying a specific 42010 viewpoint on sIOP, a necessary condition towards the preparation of a sIOP capability in a software agent has been identified which can be applied to all MDE and view-based software architectures.

References

Aßmann U, Zschaler S, Wagner G. 2006. Ontologies, Meta-models, and the Model-Driven Paradigm. In *Ontol. Softw. Eng. Softw. Technol.* (C. Calero, F. Ruiz, and M. Piattinieds.), pp. 249–273, Springer-Verlag Berlin Heidelberg.

Brandt P, Sinderen MJ van, Basten T. 2018. Semantic mediation: from alignment relation to semantic identity.

Cregan AM. 2007. Symbol grounding for the semantic web. W. Franconi, E and Kifer, M and Mayed.. Semant. WEB res. Appl. Proc. 4519: 429–442.

Eco U. 1976. *A theory of semiotics*. Indiana University Press / London: Macmillam, Bloomington, IN.

Gamut L. 1991. *Logic, Language and Meaning, volume 1: Introduction to Logic*. The University of Chicago Press.

Genesereth MR, Nilsson NJ. 1987. *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers Inc.

Greefhorst D, Proper E. 2011. *Architecture Principles, The Cornerstones of Enterprise Architecture*. Springer Berlin Heidelberg.

Grice HP. 1989. Logic and Conversation. In *Stud. W. Words*, pp. 22–40, Harvard University Press, Cambridge, MA, USA.

Guarino N. 1994. The Ontological Level. R. Casati, B. Smith, and G. Whiteeds. Philos. Cogn. Sci. Proc. 16th int. Wittgenstein symp. 443–456; doi:10.1007/978-3-642-02463-4.

Harnad S. 1990. The Symbol Grounding Problem. Physica D 42. 335–346.

Kuhn W. 2009. Semantic engineering. In *Res. Trends geogr. Inf. Sci. Lect. Notes geoinf. Cartogr.* (G. Navratiled.), pp. 63–76, Springer Berlin Heidelberg.

Ogden CK, Richards IA. 1989. *The Meaning of Meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism.* with a pre. Harcourt Brace Jovanovich, New York, USA.

Quine WVO. 1961. From a logical point of view. Br. Dent. J. 195: 229.

Saussure F de. 1959. *Course in general linguistics*. C. Bally and A. Sechehayeeds.. Philosophical Library, New York, USA.

Scheider S. 2012. Grounding geographic information in perceptual operations. Dissertation, Westfälische Wilhelms-Universität Münster; IOS Press.

Schulz K. 2007. Minimal Models in Semantics and Pragmatics. Dissertation, Universiteit van Amsterdam.

Searle JR. 1980. Minds, brains, and programs. Behav. Brain Sci. 3: 417–424.

Smith CS. 2003. *Modes of Discourse: The Local Structure of Texts*. Cambridge University Press, Cambridge, MA.

Sowa JF. 2000. Ontology, metadata, and semiotics. Lect. Notes Comput. Sci. (including Subser. Lect. Notes

Artif. Intell. Lect. Notes Bioinformatics) 1867:55–81; doi:10.1007/10722280_5.

Steels L. 2012. The symbol grounding problem has been solved, so what's next. In *Symb. Embodiment debates mean. Cogn.* (M. de Vega, A. Glenberg, and A. Graessereds.), pp. 223–244, Oxford University Press, Oxford, UK.

Ullmann S. 1962. *Semantics: An Introduction to the Science of Meaning*. 1st ed. Basil Blackwell, Oxford.

Xiuquan Li, Tao Zhang. 2017. An exploration on artificial intelligence application: From security, privacy and ethic perspective. J. Zhu, E.-B. Lin, and T. Lieds. 2017 ieee 2nd int. Conf. Cloud comput. Big data anal. 416–420; doi:10.1109/ICCCBDA.2017.7951949.