A Novel Approach to Modeling Enterprise Services Leveraging Object Cloning and Multilevel Classification

Lam-Son Lê[1], Thai-Minh Truong[1], and Alain Wegmann[2]

[1]Faculty of Computer Science & Engineering, HCMC University of Technology (VNU-HCM), Vietnam
lam-son.le@alumni.epfl.ch, thaiminh@hcmut.edu.vn
[2]School of Computer and Communication Sciences, EPFL, Switzerland
alain.wegmann@epfl.ch

***Abstract*—Object-oriented modeling is concerned with capturing common properties of objects. The dominant thinking in this realm is to classify objects that share certain properties into what is called a class, which in turn enables us to instantiate additional objects. Deep modeling takes a step further by introducing the notion of clabject that might be instantiated multiple times until its instantiation potency runs out. This initiative has gained a lot of momentum of late, primarily due to the inadequacy of the classical mechanics of two-level object instantiation. There exists a less familiar way of reasoning in object-orientation that takes its root from the prototype theory. We believe that they co-exist as two sides of the same coin. Unfortunately, prototype-based modeling still stays on the sidelines in the mainstream of conceptual modeling and related areas (e.g., enterprise modeling). In this paper, we argue that the two methods actually complement each other. We propose a hybrid modeling suite that allows for both instantiation and cloning in enterprise modeling. We formally state that a clabject not only features the so-called potency (i.e., for how many levels this clabject might further be classified) but also carries the notion of characteristics (i.e., the extent to which this clabject resembles those being represented). We demonstrate our novel ways of modeling for capturing business processes in a service-oriented enterprise architecture.**

***Index Terms*—Multilevel Classification, Business Process Modeling, Prototype Theory, Enterprise Modeling.**

1. Introduction

> "...Essentially, all models are wrong,
> but some are useful..."
>
> – George E. P. Box

Conceptual modeling is about developing an understanding for a subject in a given domain. Modeling could be viewed as a mapping from the subject in question, to a representation expressed in a particular modeling language [1]. Object-orientation has proved its expressive power in many modeling languages in use today, which eventually lead to a dominant thinking of combining objects and relationships to best capture a business domain. Reference Model of Open Distributed Processing (RM-ODP) as a standard for being object-oriented [2] adopts the relatively sharp definition of type (i.e., predicative clauses capturing the defining properties of objects) and class (i.e., a set of objects that belong to a type). Practitioners, programmers and software engineers alike have overwhelmingly made this choice of modeling, resulting in the dominance of class-based programming languages and modeling techniques. Objects might be instantiated from a template that is implicitly equivalent to the concepts of type and class combined in many contemporary programming languages. This traditional 2-level classification mechanism is often critiqued for failing to yield a cascade of objects from a highly abstract example. In recent years, deep modeling has emerged as a remedy for this limitation [3]–[5]. The growing interest in this research line has contributed to the consensus that both objects and relationships are meant to be instantiated for multiple times but perhaps at different depths.

The fact that class-based ways of reasoning have prevailed in object-oriented modeling is perhaps due to an assumption that classification is effective for conceptualizing business domains. This standpoint takes its root in philosopher Aristotle's work that aimed to categorize all natural things into a comprehensive taxonomy. The so-called Aristotelian theory usually suffices to deliver a conceptually elegant model for many domains [6]. However, classification might not work well for graded categories where one cannot state that an object fully meets all defining predicates of a category. Instead, as pointed out by Rosch, we say that a category consists of concrete objects having unequal status [7]. A coexisting, albeit less popular, way of modeling is to allow objects to be derived from a representative example called *prototype*. In the class-based approach, whether an object belongs to a class could sharply be determined. All members of a class should share the same level of closeness to the class in question. In prototype-based modeling, objects being derived from a prototype may have different levels of closeness [8]. This way of reasoning is analogous to the notion of similarity that has been discussed in cognitive

This research is funded by Vietnam National University in Ho Chi Minh City (VNU-HCM) under grant number C2019-20-15

2325-6605/19/$31.00 ©2019 IEEE
DOI 10.1109/EDOCW.2019.00036

science and pedagogy. Cognitively speaking, we human being tend to seek similarity and semantic connections when reasoning about new concepts [9]–[11].

Central to object-oriented modeling is the ability to generate additional objects that resemble an original example. The rationale behind this practice is to make our conceptual models succinct, i.e. to avoid overly enumerating objects that have common properties. Thanks to the cross-level instantiation mechanism in the class-based approach (enhanced with deep modeling), we are able to generate objects across multiple ontological levels out of a clabject. In the prototype-based counterpart, objects, being treated as instances, are created by means of cloning an origin.

Conceptual models generated thanks to the above-mentioned dual techniques might not, quoting mathematician George Box, fully reflect what we attempt to represent our target enterprise. Yet some of them are still useful for capturing the essence of "pattern" in today's enterprise modeling. We believe that classification and prototyping co-exist in conceptual modeling just as two sides of the same coin. In this paper, we propose a hybrid approach to enterprise modeling that allows for both deep classification and prototype-cloning, which take their root from multilevel modeling and prototype-based reasoning, respectively.

The remainder of this paper is structured as follows. In Section 2, we give preliminaries of this work. Section 3 states our research motivation. Section 4 offers our novel approach to modeling business processes in enterprise context. Section 5 ends the paper by drawing some conclusion remarks and outlining our future investigations.

2. Background

2.1. Cognitive Issues in Conceptual Modeling

Frixione [12] have extensively researched the representation of concepts mainly from the perspective of knowledge engineering. They stress that conceptual representation has to deal with two conflicting requirements. On one hand, compositionality is a tool for reasoning about rigid definitions of concepts (i.e. in the form of necessary and sufficient conditions). On the other hand, reasoning about prototypical effects is helpful for understanding the human's psychological process of perceiving concepts. The authors argue that while compositionality works well in traditional ways of knowledge representation such as Brachman's semantic networks [13], it does not capture prototypical effects. Exemplar- and prototype-based approaches to modeling have come out as to address the representation of non-classical concepts [14], i.e. concepts that may not be defined in the form of necessary and sufficient predicates. Exemplar-based ways of modeling put the emphasis on the cognitive effects people may have on specific examples (e.g., the first elephant we saw in our childhood) of a modeling concept [15], [16]. Prototype theories are concerned with the extent to which an object is representative[1] w.r.t. to a domain concept regardless of its cognitive effect [8]. Prototype theories sound fuzzy in defining the notion of closeness of a given prototype, i.e., how it is cognitively close to a concept it represents [17]. Although we generally are not able to formulate this notion crisply, yet we are able to compare the closeness of a pair of prototypes, making the common sense for saying "a chair as a prototype is closer to the concept of furniture than a stool is". This relatively vague notion could be reasoned about in tandem with another term called the distance metric[2] that measures the conceptual distance between a pair of prototypes [17]. The exemplar-based paradigm contributes to the explanation of human's conceptualizing and cognitive process towards fully understanding the concept of interest, which might be complemented by the prototype-based counterpart. The latter eventually made its way to object-oriented programming, especially in the early stage of this evolution of programming languages. We will discuss this point further in the subsequent subsection.

2.2. Types and Prototypes in Modeling and Programming

Type/instance modeling is a mechanism to group and describe objects having common properties. This way of reasoning has widely been adopted in the object-oriented paradigm, perhaps based on an assumption that classification is effective for conceptualizing business domains. This standpoint takes its root in philosopher Aristotle's work that aimed to categorize all natural things into a comprehensive taxonomy [6]. When it comes to engineering, the standpoint has another advantage – the computational efficiency of class inheritance in enabling the programmers to extend their programming classes.

In the common sense, a type defines properties that could be shared between instances. A class refers to the set of all instances that can be created out of a type. People usually make an implicit assumption in reasoning about type and class that all instances are treated equal.

There exists a different vision on this matter. As Taivalsaari stressed, the so-called Aristotelian theory usually suffices to deliver a conceptually elegant model for many domains [6]. However, classification might not work well for graded categories where one cannot state that an object fully meets all defining predicates of a category. Instead, as pointed out by Rosch, we say that a category consists of concrete objects having unequal status [7]. The object that

[1]The most typical object that best represents the concept of interest is called a prototype. There might be multiple prototypes that characterize a given concept. In class-less programming, prototypes are simply objects.

[2]Closeness of a prototype could mathematically be regarded as a function over the said prototype whose value falls in the interval $[0, 1]$. Distance metric is a function from the Cartesian product of the set of all prototypes and itself into non-negative real numbers. Consider two distinct prototypes υ and ω both of which are originated from same concept. If the distance metric between υ and this concept is larger than that of ω, then υ's closeness is greater than that of ω.

best characterizes a category constitutes a *prototype*. When it comes to engineering, we may need a radically departed programming philosophy rather than the traditional object-oriented paradigm (e.g. cloning objects in place of class inheritance in Beta and Self) at the expense of computational efficiency.

Advocates of prototype-based modeling such as Lieberman [18] and Rosch [7] argue that psychologically, people tend to develop their conceptualization of a new subject by linking it to the closest examples they can think of. As such, the so-called type-based modeling seems to be artificial.

Consensus on these two theories is that the prototype theory might reflect real-world phenomena and business situations better than the Aristotelian theory does. However, the former limits the ability to communicate conceptual models among people (e.g., classes, instantiation and generalization are widely used in meta-modeling). Also, the former is technologically expensive and generally unfamiliar among programmers. When it comes to enterprise modeling, both of them have proved to be useful[3].

2.3. Multilevel Modeling

There has been a growing interest in multilevel modeling of late. Most object-oriented programming languages offer only two-level classification, i.e. being either at the class level or the object level, and therefore suffer from what advocates of multilevel modeling call shallow instantiation [4]. Due to this inadequate granularity of representation, traditional two-level classification techniques does not work with deep instantiation typically found in domain ontologies and linguistic structures [21]. To remedy this, conceptual modeling is enhanced with the notion of clabject that is supposed to be instantiated for multiple times until its instantiation potency runs out [22]. A clabject is a modeling metaphor that has dual facets: (a) being regarded as class from which an instantiation shall be made; (b) being an instantiation of another clabject. Like object-oriented classes, a clabject has its own attributes and is semantically connected with others via associations both of which are meant to be instantiated independently of the host clabject [22].

Other ways of multilevel modeling address the use of powertypes in conceptual modeling [23] [24] and powertype patterns [25]. This sort of reasoning would enrich modeling approaches to enterprise architecture that rely on the traditional 2-level classification such as the RM-ODP [2].

3. Dilemma of Conceptual Modeling

Strictly following class-based modeling real-life enterprise applications might not be effective, primarily due to the uncontrollable growth of classes to be defined over the course of modeling [26]. Practitioners might start with a coarse-grained classification for the sake of simplicity, for example, retail outlet as a modeling concept would suffice to represent a handful number of retailers that offer shopping space. As more types of outlet (e.g., warehouse, mall, pet store) find their way into this enterprise application, such a coarse-grained conceptual representation becomes naive. On one hand, we may find the fine-grained representation helpful for building up our models. On the other hand, populating the model with excessively detailed concepts in the early phase is likely to generate a biased classification model later on (e.g., being stuck when representing an e-commerce store). Traditional methods that solely rely on this classification generally do not epitomize at what level of classification we should start with to effectively build up our enterprise model [27]. Multilevel modeling approaches as mentioned in Section 2 somewhat mitigate this issue by allowing any modeling concept to be down-instantiated for a pre-defined number of times [5]. In another word, we are able to tell in advance how far the classification would develop down the track.

The aforementioned dilemma makes it challenging to define a set of necessary concepts during the course of enterprise modeling while avoiding overfitting the enterprise model in question. We aim to establish a minimum set of root concepts from which all elements in the enterprise model are derived from. Should such a set overgrows, there is a higher chance of overfitting later on. In case down-instantiation or any other class-based derivation mechanism is not helpful, we may resort to cloning as an alternative technique for creating additional model elements. We seek a novel approach to leveraging both down-instantiation and cloning in building enterprise models.

Growing an enterprise model up involves developing deep understanding of it as a complex system [28]. In pedagogy, modern ways of teaching facilitate this comprehension process by giving the learners a few conceptual variants of the target concept to reason about [29], [30], giving rise to the terms of meaning equivalence and surface similarity. Alternative enterprise models generated out of the above-mentioned dilemma are subject to be analyzed whether they are equivalent semantically or they are similar superficially [10].

4. Putting Instantiation and Cloning Together

4.1. A Running Example

We describe in this subsection a real case-study of a medium-sized company, namely Atadi[4], that retails low-cost domestic airfare in Vietnam sourced from a number of airlines including Vietnam Airlines[5], Jetstar Pacific Airlines[6]

[3]On one hand, the fact that many models created in the ArchiMate language [19] follow the well-known example of a fictitious insurance company named ArchiSurance reinforces the importance of the prototype-based method. On the other hand, RM-ODP Part 2 defines a number of foundation concepts many of which could be used in enterprise modeling following the type-based method [20].

[4]http://www.atadi.vn

[5]http://www.vietnamairlines.com

[6]http://www.jetstar.com

and VietJet Air[7]. Atadi has approximately 30 employees and four departments, namely finance, customer support, technical, marketing. The company deals with domestic airlines to allow its customers to book air tickets primarily over the Internet. It offers customer support services to those who wish to revise their air tickets at their own expenses. Atadi has the following business functions: `Financial Handling`, `Marketing Handling` and `Customer Support Handling`.

Figure 1 depicts the enterprise architecture of company Atadi in ArchiMate – a visual language for describing the structure of enterprises. One department of Atadi, responsible for the function of `Customer Support Handling`, handles post-sale changes requested by the customers to revise their air tickets (e.g., adding or removing luggage, changing passengers, rescheduling flights). The business logic of rescheduling is materialized in a process called `Reschedule airfare` that is viewed as whole the diagram of Figure 1. In this scenario, rescheduling an air ticket issued by, for instance, Vietnam Airlines for a customer whose name is David could be considered as another business process. An additional process in this space could be described as: to help Atadi's customers add more luggage to their previously booked tickets. Yet another process in this space would be about booking air tickets from a competitor of Atadi. These enterprise model elements, being viewed as a whole, share the abstract syntax of being a process towards a certain business goal. In terms of the concrete syntax, they might be expressed using representational variants, e.g, diagrams, text.

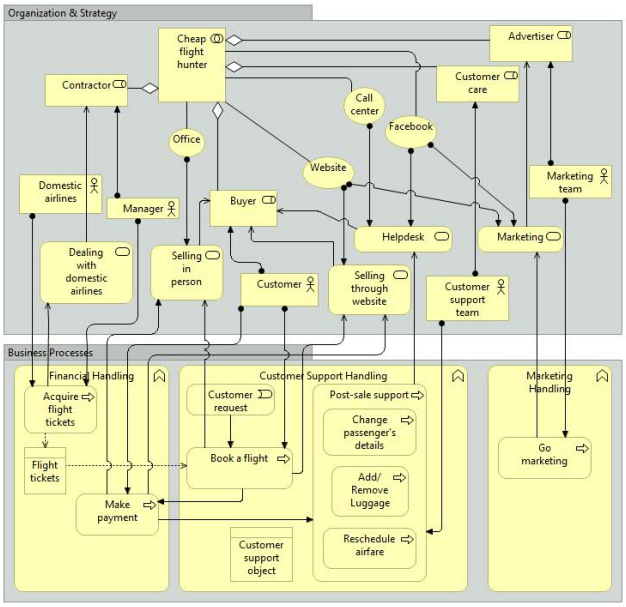

Figure 1: The enterprise architecture of company Atadi represented in ArchiMate

The business processes shown in Figure 1 contribute to business services that are named using gerunds. For example, `Selling in person` as a service is realized by the following two processes: `Make payment` and `Book a flight`. The processes might in turn be enacted using application services (e.g., microservices, IoT services – not represented in Figure 1 to save room). All in all, we represent in this running example a service-oriented enterprise architecture where business processes play a central role.

A technical question that arises here is that, whether are the aforementioned processes actually derived from an ancestor? If yes, thanks to what mechanism would they be derived? Semantically speaking, the following ways of derivation sound different.

- Deriving `Reschedule airfare` to obtain the process of rescheduling an air ticket issued by Vietnam Airlines for a customer whose name is David
- Deriving to obtain the process of adding luggage for a customer whose name is Jane
- Deriving to obtain the process of booking air tickets from a competitor of Atadi

We adopt the notion of clabject coined by Atkinson when coping with multiple ontological levels. To derive a new model element from a clabject, one chooses to either instantiate from or clone the said clabject. Each clabject comes with a non-negative integer value that explicitly tells us how many ontological levels one might instantiate from it. This value decreases by one per instantiation. We argue that a clabject may also serve as a prototype that is subject to be cloned. The said prototyping clabject and those cloned from it are all instances of the clabject from which the prototyping clabject was instantiated. In another word, they stay at the same level of classification. As such, they (the prototyping clabject in question and those cloned from it) have the same value of potency. As illustrated in Figure 2, instantiated clabjects and those cloned from them should share the same level of potency, embodying a row of clabjects.

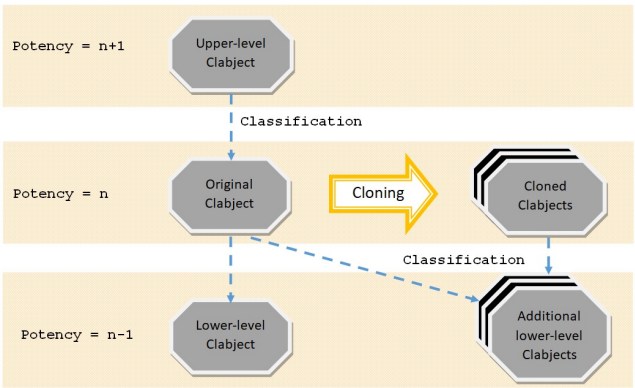

Figure 2: Cloning a prototyping clabject would yield another clabject both of which have the same level of potency. Instantiating a clabject essentially uses up its potency whereas cloning it leaves the potency unchanged.

To enable multilevel classification, Atkinson argues that

[7]http://www.vietjetair.com

most model elements[8] in fact have two facets: as a class and as an object, hence the term clabject [4]. We rather consolidate its object facet by introducing the cloning mechanism, which is originated from the prototype-based approaches, to a clabject. Note that cloning a clabject would result in another clabject that could itself be instantiated if the potency of the cloned one is greater than zero. In other words, we view a clabject simply as an object when cloning; as a class when instantiating. Exhaustively instantiating and cloning clabjects will generate a conceptual domain. One of the clabjects in this domain serves as the ancestor of all in this domain. The ancestor has highest level of potency and is not cloned from any other clabject.

Figure 2 offers a potency-centric representation of clabjects. What is not illustrated in this figure is the degree of closeness a clabject manifests w.r.t. to the ancestor. For the sake of formalization, we refer to this notion of closeness as the characteristic function. Instantiating a clabject will make its characteristic function unchanged. However, cloning a clabject will lower its characteristics. Measuring the loss of characteristics when cloning would make a topical domain-specific research. For example, the extent of being characteristic could be measured by leveraging the notion of similarity [31] [32] [33] [34].

Subsection 4.2 sums up our proposal for mixing multilevel classification and prototype-cloning. Subsection 4.3 formalizes our proposal.

4.2. Twin Track, Dual Facet and Bidimensionality

In short, we propose a twin-track, combined approach of (a) instantiating a clabject to reach a lower ontological level; (b) cloning a clabject to yield another clabject at the same ontological level. This combined approach is materialized by the dual facet of clabjects, i.e., any clabject could be treated either as a class or an object (a prototype). Moreover, each clabject is constrained by the following dimensions.

- Potency: as defined in multilevel classification. This dimension tells us how many levels we could classify the said clabject until we reach a clabject that is no longer a class.
- Characteristics: the degree of being representative of a clabject w.r.t the ancestor. Cloning a clabject would lower its level of characteristics.

4.3. Formal Definitions

A hybrid modeling suite is a quadruple $\langle \mathcal{A}, \mathcal{E}, c, potency \rangle$, where

- $\mathcal{A}$ denotes a conceptual domain, i.e., an exhaustive set of model elements that are originated from the an ancestor, denoted as $\mathcal{E}$;
- $c : \mathcal{A} \rightarrow [0, 1]$ is the characteristic function of an element w.r.t. the ancestor [17]. The ancestor has 1 as its level of characteristics (in math, $c(\mathcal{E}) = 1$);
- $potency : \mathcal{A} \rightarrow \mathbb{N}$ tells us the potency as a natural number of an element. The ancestor should have highest level of potency. Formally, $\forall e \in \mathcal{A} \mid potency(e) \leq potency(\mathcal{E})$

The characteristic function could in fact be related to a distance metric of two genuine model elements that describes the similarity of the couple [17]. Osherson accentuates that the distance metric of a pair of similar objects is lesser than the distance metric of a pair of dissimilar ones, as formalized in the following two conditions of distance metric and characteristic function [17].

$$\begin{aligned} \langle \mathcal{A}, d \rangle \text{ is a metric space, i.e.,} \\ (\forall x \in \mathcal{A})(\forall y \in \mathcal{A}) \\ d(x, y) = 0 \text{ iff } x = y \\ d(x, y) = d(y, x) \\ d(x, y) + d(y, z) \geq d(x, z) \end{aligned}$$

$$\begin{aligned} (\forall x \in \mathcal{A})(\forall y \in \mathcal{A}) \\ d(x, p) \leq d(y, p) \rightarrow c(y) \leq c(x) \end{aligned}$$

where p is an element of $\mathcal{A}$ that represents the concept's prototype.

Axiom 1. *If there exists υ cloned from ω (denoted as $\omega \Rightarrow \upsilon$ from now on), then υ will reflect the concept ($\mathcal{E}$) less precisely than ω and both of them will have the same value of potency.*

$$\begin{aligned} \forall \omega, \upsilon \in \mathcal{A} \mid \omega \Rightarrow \upsilon \\ \rightarrow c(\upsilon) < c(\omega) \wedge \ potency(\upsilon) = potency(\omega) \end{aligned}$$

Axiom 2. *If there exists υ instantiated from ω (denoted as $\omega \downarrow \upsilon$ from now on), then υ will retain all characters of ω and the classification level of υ will be decreased by one from that of ω.*

$$\begin{aligned} \forall \omega, \upsilon \in \mathcal{A} \mid \omega \downarrow \upsilon \\ \rightarrow c(\upsilon) = c(\omega) \wedge \ potency(\upsilon) = potency(\omega) - 1 \end{aligned}$$

Theorem 1. *If there exists υ instantiated from ω and κ cloned from υ, then κ will demonstrate the concept ($\mathcal{E}$) less precisely than ω and the potency of κ will be reduced by one from that of ω.*

$$\begin{aligned} \forall \omega, \upsilon, \kappa \in \mathcal{A} \mid \omega \downarrow \upsilon \wedge \upsilon \Rightarrow \kappa \\ \rightarrow c(\kappa) < c(\omega) \wedge \ potency(\kappa) = potency(\omega) - 1 \end{aligned}$$

Proof. According to Axiom 2, υ will keep all characteristics of ω and the classification level of υ will be decreased from that of ω if υ is instantiated from ω.

$$\begin{aligned} \forall \omega, \upsilon \in \mathcal{A} \mid \omega \downarrow \upsilon \\ \rightarrow c(\upsilon) = c(\omega) \wedge \ potency(\upsilon) = potency(\omega) - 1 \end{aligned}$$

[8]Objects instantiated from a clabject having potency of exactly one would not further be instantiated. They therefore possess only one facet.

Furthermore, based on Axiom 1, characteristic function of κ will be lesser than that of ω and they will be at the same level of classification when κ is cloned from υ.

$$\forall \kappa, \upsilon \in \mathcal{A} \mid \upsilon \Rightarrow \kappa$$
$$\rightarrow c(\kappa) < c(\upsilon) \wedge \ potency(\kappa) = potency(\upsilon)$$

Eventually, due to the transitive relation, we can deduce that κ will reduce characteristics of ω and the ontological level of κ will be decreased by one from that of ω.

$$c(\kappa) < c(\omega) \wedge \ potency(\kappa) = potency(\omega) - 1$$

□

Theorem 2. *If there exists κ cloned from ω and δ instantiated from κ, then δ will display the concept ($\mathcal{E}$) less precisely than ω and the potency of δ will be decreased by one from that of ω.*

$$\forall \omega, \kappa, \delta \in \mathcal{A} \mid \omega \Rightarrow \kappa \wedge \kappa \downarrow \delta$$
$$\rightarrow c(\delta) < c(\omega) \wedge \ potency(\delta) = potency(\omega) - 1$$

Proof. Similar to how we prove Theorem 1, as stated in Axiom 1, if κ is cloned from ω, then κ will reduce characteristics of ω and both of them will have the same value of potency.

$$\forall \kappa, \omega \in \mathcal{A} \mid \omega \Rightarrow \kappa$$
$$\rightarrow c(\kappa) < c(\omega) \wedge \ potency(\kappa) = potency(\omega)$$

Likewise, due to Axiom 2, characteristics of δ will be the same with characteristics of κ and the classification level of δ will be reduced from that of κ when δ is instantiated from κ.

$$\forall \kappa, \delta \in \mathcal{A} \mid \kappa \downarrow \delta$$
$$\rightarrow c(\delta) = c(\kappa) \wedge \ potency(\delta) = potency(\kappa) - 1$$

Ultimately, based on the transitive relation, we can infer that δ will reflect the concept ($\mathcal{E}$) less precisely than ω and the potency of δ will be decreased by one from that of ω.

$$c(\delta) < c(\omega) \wedge \ potency(\delta) = potency(\omega) - 1$$

□

4.4. Revisiting the Case-study

This subsection revisits the case-study described in Subsection 4.1 explaining how processes in the business domain of retailing domestic airfare could be derived from `Business process` (presented in BPMN[9]) – the ancestor in this conceptual model. The proposed hybrid modeling approach has resulted in a deep conceptual model as shown in Figure 3. These elements, being viewed as a whole, share the abstract syntax of being a process towards a certain business goal. In terms of the concrete syntax, they might be expressed using representational variants, e.g, diagrams, text. We assume that the ancestor has full characteristics (i.e., the value of its characteristic function is one) and a total of three levels of classification.

Clabjects	Potency	c function
Business process (BPMN)	2	1
Instructions (Text)	2	0.8
Workflow (UML)	2	0.7
Rescheduling process (BPMN)	1	1
Rescheduling documentation	1	0.8
Booking tickets (UML)	1	0.7
Rescheduling flight tickets (BPMN)	0	1
Adding luggage (BPMN)	0	0.9
Booking flight tickets (UML)	0	0.7

TABLE 1: The potency and characteristic function of all clabjects presented in Figure 3

The ancestor could be cloned into `Instructions`. As its name suggests, this cloned clabject refers to a textual, step-by-step procedure to achieve a business goal. At the same time, the ancestor is instantiated into `Rescheduling process`, shown in BPMN again. It captures the business that supports Atadi's customers in changing their flight tickets. Subsequently, `Instructions` might be cloned/instantiated into `Workflow`/`Rescheduling documentation`. On one hand, clabject `Rescheduling documentation` presents comprehensive guide for executing a rescheduling request. On the other hand, `Workflow` is a kind of activity diagram defined by the UML[10]. Furthermore, `Workflow`, having the potency value of 2, could be classified down into `Booking tickets` and further down into `Booking flight tickets`, both of which are depicted in UML. Note that the three clabjects stay in a descending order of potency. By the same token, clabject `Booking tickets` captures necessary steps for performing a booking request from Gotadi's customers. In addition, `Booking flight tickets` stands for the activity of booking an air ticket with Jetstar Pacific Airlines for Jane through retailer Gotadi.

Similarly, we clone (instantiate) `Rescheduling process` into `Rescheduling documentation` (`Rescheduling flight tickets`). The instantiated one fulfills a specific request of a genuine person (e.g., request submitted by David through company Atadi for rescheduling his Vietnam Airlines air ticket).

Interestingly, we might obtain clabject `Rescheduling documentation` from the ancestor in two different ways. On one hand, we apply instantiation (to the ancestor), followed by cloning. On the other hand, we instantiate after having cloned the ancestor.

In Table 1, we present both the potency and the characteristic function of the depicted clabjects. In fact, the goal

[9]BPMN stands for Business Process Modeling Notation. It is a modeling language defining a set of process elements. The specification of BPMN is managed by the Object Management Group http://www.bpmn.org/

[10]UML stands for Unified Modeling Language http://www.uml.org/

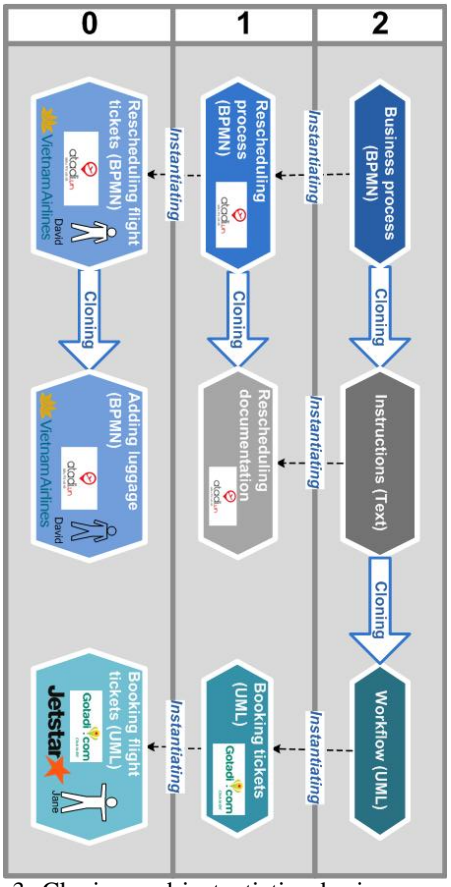

Figure 3: Cloning and instantiating business processes.

of having c function mentioned in this table is to illustrate how the characteristic function of a clabject would fluctuate when cloning/instantiating the said clabject (i.e., it could not be calculated precisely by our formulas), which is not embodied in Figure 3. The values of potency and characteristics given in Table 1 obey our theorems presented in Section 4.3. For the sake of readability, Table 1 is structured as follows. The leftmost column presents all the clabjects coming from the case-study. The rightmost column states the characteristic function of the said elements. The column in the center shows the potency of the clabjects in question.

5. Final Considerations

In this paper, we report our findings on the synergy between multilevel classification and prototype-cloning. While deep classification has gained research momentum and started to underpin engineering uptake of late, prototype-based programming appears to wind down despite having a long pedigree. We argue that a clabject as defined in multilevel modeling is subject to both instantiation (because it carries a class facet) and cloning (because it possesses an object facet). This hybrid modeling suite has proved to be useful for reasoning about the derivation of enterprise model elements in our example. This work, reporting some surprising findings along our research line in enterprise modeling based on RM-ODP [35], is yet to find its way into our full-scale service-oriented enterprise architecture of late.

Discussions. We address the usefulness of the proposed modeling suite but have not worked out the math properties of our instantiating and cloning operators. Whether the communicative, associative and distributive laws apply to these operators remains an open question. As explained by Box, this way of modeling might not generate truly sharp enterprise models. Usefulness in enterprise modeling is actually what drives us to the primary finding of this work.

To study the soundness of this hybrid modeling approach, we would devise a calculus for the whole modeling approach, in an analogous way to the λ-calculus and a family of object calculi [36] that serve as foundations for functional languages and object-oriented languages, respectively. Another way to address this topic would be assessing the quality of enterprise models constructed using the proposed hybrid modeling approach. We will carry this research direction out by leveraging existing metrics for the quality of conceptual modeling [37] [38].

References

[1] G. Guizzardi, "On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models," in *Proceedings of 6^{th} International Baltic Conference on Databases and Information Systems*. Vilnius, Lithuania: IOS Press, July 2006, pp. 18–39.

[2] ISO/IEC, "*ITU-T X.902 | ISO/IEC 10746-2 Information Technology – Open Distributed Processing – Reference Model – Foundations*," SC 7 and ITU, International Standard, 2010.

[3] T. Kühne, "Matters of (meta-) modeling," *Software Systems Modeling*, vol. 5, pp. 369–385, 2006.

[4] C. Atkinson and T. Kühne, "The Essence of Multilevel Metamodeling," in *Proceedings of 4^{th} International Conference on Unified Modeling Language. Modeling Languages, Concepts, and Tools*. Toronto, Canada: Springer, September 2001, pp. 19–33.

[5] U. Frank, "Multilevel modeling: Toward a new paradigm of conceptual modeling and information systems design," *Business & Information Systems Engineering*, vol. 6, no. 6, pp. 319–337, December 2014.

[6] A. Taivalsaari, "Classes vs. Prototypes – Some Philosophical and Historical Observations," *Journal of Object-Oriented Programming*, vol. 10, no. 7, pp. 44–50, 1997.

[7] E. Rosch, "Cognitive representations of semantic categories," *Journal of Experimental Psychology*, vol. 104, no. 3, pp. 192–233, 1975.

[8] H. Kamp and B. Partee, "Prototype theory and compositionality," *Cognition*, vol. 57, pp. 129–191, 1995.

[9] D. Gentnera and J. Medinab, "Similarity and the development of rules," *Cognitive*, vol. 65, no. 2-3, pp. 263–297, January 1998.

[10] M. Trench and R. Minervino, "The Role of Surface Similarity in Analogical Retrieval: Bridging the Gap Between the Naturalistic and the Experimental Traditions," *Cognitive Science*, vol. 39, no. 6, pp. 1292–1319, August 2015.

[11] A. Robins, "Transfer in cognition," *Connection Science*, vol. 8, no. 2, pp. 185–204, 1996.

[12] M. Frixione and A. Lieto, "Combining description logics and typicality effects in formal ontologies," in *Proceedings of 12^{th} International Conference of the Italian Association for Artificial Intelligence*, 2011, pp. 401–406.

[13] R. J. Brachman, "What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks," *IEEE Computer*, vol. 16, no. 10, pp. 30–36, 1983.

[14] M. Frixione and A. Lieto, "Representing non classical concepts in formal ontologies: Prototypes and exemplars," in *5^{th} International Workshop on New Challenges in Distributed Information Filtering and Retrieval*. Springer-Verlag, September 2011, pp. 171–182.

[15] A. Wedel, "Exemplar models, evolution and language change," *The Linguistic Review*, vol. 23, no. 3, pp. 247–274, 2006.

[16] R. Bod, "Exemplar-based syntax: How to get productivity from examples," *The Linguistic Review*, vol. 23, no. 3, pp. 291–320, 2006.

[17] D. Osherson and E. Smith, "On the adequacy of prototype theory as a theory of concepts," *Cognition*, vol. 9, pp. 35–58, 1981.

[18] H. Lieberman, "Using Prototypical Objects to Implement Shared Behavior in Object-Oriented Systems," in *Proceedings of the 1^{st} Conference on Object-Oriented Programming Systems, Languages and Applications*. ACM, 1986, pp. 214–223.

[19] M. Lankhorst, *Enterprise Architecture at Work - Modelling, Communication and Analysis*, 2nd ed. Springer, 2009.

[20] A. Wegmann, L.-S. Lê, G. Regev, and B. Wood, "Enterprise modeling using the foundation concepts of the RM-ODP ISO/ITU standard," *Information Systems and e-Business Management*, vol. 5, no. 4, pp. 397–413, 2007.

[21] C. Atkinson and T. Kühne, "Demystifying Ontological Classification in Language Engineering," in *European Conference on Modelling Foundations and Applications*. Vienna, Austria: Springer, July 2016, pp. 83–100.

[22] B. Neumayr, M.-A. Jeusfeld, M. Schrefl, and C. Schütz, "Dual Deep Instantiation and Its ConceptBase Implementation," in *Proceedings of 26^{th} International Conference on Advanced Information Systems Engineering*. Thessaloniki, Greece: Springer, June 2014, pp. 503–517.

[23] B. Henderson-Sellers and C. Gonzalez-Perez, "Connecting Powertypes and Stereotypes," *Journal of Object Technology*, vol. 4, no. 7, pp. 83–96, September 2005.

[24] J. Odell, "Power types," *Journal of Object-Oriented Programming*, vol. 7, no. 2, pp. 8–12, 1994.

[25] V.-A. Carvalho, J.-P. Almeida, and G. Guizzardi, "Using a Well-Founded Multi-level Theory to Support the Analysis and Representation of the Powertype Pattern in Conceptual Modeling," in *Proceedings of 28^{th} International Conference on Advanced Information Systems Engineering*. Ljubljana, Slovenia: Springer, June 2016, pp. 309–324.

[26] L.-S. Lê, H. Dam, and G. Aditya, "On Business Services Representation – the 3 x 3 x 3 Approach," in *Proceedings of the 21^{st} Australasian Conference on Information Systems*. Brisbane, Australia: AIS Electronic Library (AISeL), 2010, p. 58.

[27] A. Taivalsaari, "On the notion of inheritance," *ACM Computing Surveys*, vol. 28, no. 3, pp. 438–479, 1996.

[28] E. Crawley, B. Cameron, and D. Selva, *System Architecture: Strategy and Product Development for Complex Systems*. Pearson, April 2015.

[29] U. Shafrir and R. Kenett, *Concept Science Evidence-Based MERLO Learning Analytics*. IGI Global, December 2015, pp. 334–357.

[30] O. Robutti, F. Arzarello, P. Carante, R. Kenett, T. Prodromou, and U. Shafrir, "Meaning equivalence: a methodological tool for assessing deep understanding," in *Proceedings of 10^{th} International Conference on Technology, Education and Development*, Valencia, Spain, March 2016, pp. 7358–7367.

[31] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, and J. Mendling, "Similarity of business process models: Metrics and evaluation," *Information Systems*, vol. 36, no. 2, pp. 498–516, April 2011.

[32] M. Ehrig, A. Koschmider, and A. Oberweis, "Measuring Similarity between Semantic Business Process Models," in *Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling*. Ballarat, Victoria: ACM, 2007, pp. 71–80.

[33] T. Kurniawan, A. Ghose, H. Dam, and L.-S. Lê, "Relationship-Preserving Change Propagation in Process Ecosystems," in *Proceedings of 10^{th} International Conference on Service-Oriented Computing*, Shanghai, China, November 2012, pp. 63–78.

[34] T.-M. Truong and L.-S. Lê, "Towards a Formal Framework for Business Process Re-Design Based on Data Mining," in *Proceedings of 17^{th} Working Conference on Business Process Modeling, Development and Support, held at CAiSE'2016*. Ljubljana, Slovenia: Springer, June 2016, pp. 250–265.

[35] L.-S. Lê and A. Wegmann, "Hierarchy-Oriented Modeling of Enterprise Architecture using Reference–Model of Open Distributed Processing," *Computer Standards & Interfaces*, vol. 35, no. 3, pp. 277–293, March 2013.

[36] A. Martín and L. Cardelli, "An imperative object calculus," in *Proceedings of 6^{th} International Joint Conference CAAP/FASE on on Theory and Practice of Software Development*. Aarhus, Denmark: Springer, May 1995, pp. 471–485.

[37] H. Nelson, G. Poels, M. Genero, and M. Piattini, "A conceptual modeling quality framework," *Software Quality Journal*, vol. 20, no. 1, pp. 201–228, 2012.

[38] O. Lindland, G. Sindre, and A. Solvberg, "Understanding quality in conceptual modeling," *IEEE Software*, vol. 11, no. 2, pp. 42–49, March 1994.