A Missing Concept?

A Short Postulation About the Need for Human Understanding and an Often Only Implicitly Given, Fundamental Aspect of Modeling Efforts

Matthes Elstermann[1]

[1] Karlsruhe Institute of Technology, 76133 Karlsruhe, Germany
matthes.elstermann@kit.edu

Abstract. In almost all fundamental modeling or description approaches, especially those meant for the domain of (business) process, there is one description aspect often missing that is core to human understanding of the world: the aspect of active entity or the concept of the subject and its strict differentiation from passive entities (objects). This work postulates that this simple idea is of great importance when approaching any description/modeling, be it made by humans as input/instruction for information systems or AI engines, or be it generated by such a system to explain its inner workings to humans to enable better understanding and therefore acceptance. Therefore, it is proposed that any formal or informal modeling approach should consider these aspects in order to better facilitate human understanding of models and avoid humans working around the lack of expressiveness or try to erroneous interpret non-given information.

Keywords: Process Modeling · Fundamental Modeling Concept · Subject-Orientation

1 Introduction

Any kind of modeling or description is, in essence, always an act of communication. It is usually done to first structure thoughts and concepts and then convey them to other human beings or as input for human-made machines as operating instructions. However, any act of communication is potentially error-prone and may cause misunderstandings. Therefore having models that, while being as correct as possible, also are as easily conceivable as possible, is a goal to be strived for.

Considering the fundamental way in which humans communicate about the world and orienting modeling efforts accordingly (if possible) may give an indication of how to improve.

When humans convey information from and to one another, the simplest complete structure in all natural languages (spoken or written) is the simple active sentence. Shorter means of communication are possible in a context where with agreed-on or

Copyright © 2020 for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-2716/paper9.pdf

"common" knowledge that makes it possible to abbreviate the information exchange. Otherwise, full information exchange requires to state an active entity, the Subject (S), an activity, verb, or predicate (P) and an object (O) upon which an activity is acted upon (See **Fig. 1**).

The order of S, P, O, may differ. More than half of the world's languages have a subject-object-Predicate (SOP) structure. Among them Turkish or Japanese or Latin. Roughly 30% have the subject-verb-object (SVO) structure, e.g. the English or German languages [1] with very few of the rest not starting with the subject or active entity at the beginning of a sentence.

This leads to two conclusions or premises for this work: firstly, the existence and differentiation between active entities, activities and passive objects is a fundamental aspect of human perception of the world and secondly the subject is of especial importance and the first information expected to be given by human beings. Only if the context is clear[1] or in a passive setting[2], can it be omitted.

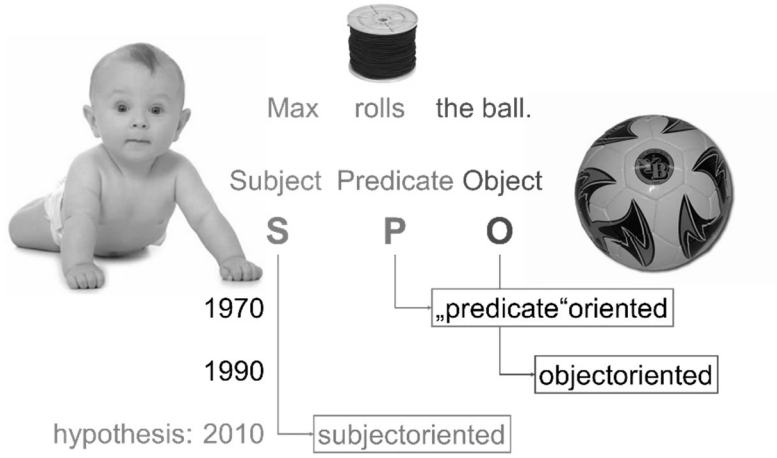

Fig. 1. The principle of conveying information using human (English) language compared to description paradigms typical found in programming (from [2])

1.1 Consequential Demand

If accepted, the consequence of this consideration is that when approaching any kind of modeling, there should always (!) be a fundamental, explicit, and strict differentiate between active entities *(subjects)*, passive entities (*objects* - including concepts, terms) and activities/instructions (*verbs/predicates*) before all other concerns or aspects. This is especially true and of importance when considering terminology or

[1] E.g. in a barber shop where subject (the barber) and object (hair and scissors a.o.) are clear, the activities can be described (modeled) by only using verbs (e.g. wash, cut, style)

[2] E.g. if the subject is a non-descriptive, generic "somebody" that

naming in any type of model – ambiguous terminology leaving interpretation of subject, object, or activity should be avoided.

Alternatively and more simply put, a modeler should, even if it is not his main goal, always be aware of and be able to answer the questions about his model (and ideally make that obvious by naming): Is this "*concept*" something to that does something on its own or in cooperation with other "*concepts*"? Is this a "*concept"* describing something that can be used by some entity for some purpose[3] or, is this the *"concept"* an action or activity that is to be executed? This may not necessarily be the focus of modeling, but it should always be clearly stated.

If this fundamental paradigm is adhered to, it should make legibility of models, be they human-created or be they output of technology (e.g., AI-based) systems describing their inner workings.

2 Differentiation

It is very likely that people always differ between *subject*, *object*, and *activity/verb*, since it is so ingrained into human communication and information exchange. The question is how well they can assign concepts and terms in models of any kind to the aforementioned terms. The problem arises if concepts are assigned erroneous or ambiguous and, e.g., properties are assigned to an activity that are more befitting to an active entity (subject vs. verb). The hypothesis here is, that if a clear differentiation is missing or at least not very obvious or ambiguous, model-perceivers will make (possibly wrong) implicit assumptions about what is an active entity, what an action, and what an object to be acted upon. And if these (erroneous made) assumptions do not fit together or, based on a perceiver's current level of understanding, are impossible to make at all, this may lead to a downright rejection of a description or model as stupid or simply wrong.

Equally, it can be assumed that if they are missing or not easily discernable in a modeling formalism or modeling concept, it becomes harder for humans to cope with the description mechanism and use them.

The second hypothesis here is that most modeling formalisms contain assumptions about *subject*, *object*, and *verb*, so to speak. However, in many cases they contain them so implicitly or are optional that consequential in the domain of a given modeling formalism, only for insiders it is obvious what is what and what fits together how.

The following section discusses these hypotheses with examples from different modeling formalisms.

2.1 Programming (General)

Programming is the *"art"* of describing activities to solve computational problems. While not modeling per-se, a program source code, especially in a high-level programming language, can be considered the model of an algorithmic workflow. Also,

[3] With „purpose" being the properties of an active entity.

the activity of programming is taught in almost all science, technology, engineering, and mathematic domains, so in contrast to more domain-specific approaches, this modeling approach is widely used, and many people are theoretically familiar with its description concepts. Therefore it is considered here.

Furthermore, in time of increasing usage of computer systems or AI solutions, it can be argued that the border between programming and describing aspect to be considered by the machines (modeling) is somewhat blurred, especially when most kinds of models are created on computers and are many are evaluated by them.

2.2 Procedural Programming

Procedural programming is the main paradigm of programming languages such as C, PASCAL, or (partly) BASIC (see among others [3]). These languages are already **Turing complete** [4], implying that it is theoretically possible to describe any computation process with them! This, in turn, implies two things: first, that modeling it is a description mechanism for computational problems or mathematical models, and secondly, that no other description mechanism is necessary in theory.

The whole concept centers on describing procedures/functions/methods, which contain instructions to modify some data or to call other procedures. All methods are basically instructions of *activities* (therefor verbs) that can be chained or use the subprocedure call that at the same time is the only available abstraction mechanism in this paradigm.

Since this description formalism is a programming paradigm, obviously, the *subject* conduct the activities is the computer or processor of the machine the instruction model is to be executed on. The *object* is also implicitly given in form of the *"data"* to be acted upon, which in original incantation was only the raw numerical or alphanumerical data. That data may implicitly be used to model complex circumstances or concept; however, no formal or conceptual modeling mechanism exists to express the according concepts.

However, the need to express/model descriptions for more and more complex circumstances and in larger teams of human beings led to the rise of another programming or rather modeling paradigm that nowadays predominant is the most widely used: Object-Orientation [5].

2.3 Object Orientation – The passive Tense

As stated before, object-orientation (OO) as a description paradigm allows to model individual (data) objects. It was praised as a better way to conceive computational models that fit the problems of the real world since everything can be described *"as an object."*

This simple statement may lead to somewhat of a misconception or misalignment in modeling. Namely, it may lead to the concept that in object-oriented programming, you can state that something is, e.g., a "car" and it can "drive." However, this is not what can be denoted exactly.

As a programming paradigm, in essence, OO is a means to describe instruction for an implicitly assumed *subject*: again, the processor of a computer. The objects are only data concepts that may represent concepts from the real world. Still, the methods/behavior (*the verbs*) described for an object are not its behavior but rather the behavior of the computer and how it is allowed to act upon the object.

If tasked to state a natural language equivalent to object-orientation, it would be that of the passive tense, as it can conceptually be modeled what is to be done to a (data) object but now how it acts. While possibly elegant and useful, the passive tense is never the first nor the easiest description means to learn a new (natural) language to describe the world with[4].

When using the passive tense for describing the real-world as well as for programming, there are some conceptual problems.

First, there are people not understanding the fundamentals of what can be modeled and for what purpose. People not versed well in OO but with access to UML-OO-class-diagraming tools creating conceptual non-sound models that are impossible to work with, in the programming domain because the model subject and objects mixed into each other. This, supposedly, could be the general cause for problems with OO as a modeling means.

A second problem is the conceptual impossibility to describe what is to be done with two objects to create a third (see **Fig. 2**). If strictly staying in the passive tense, the process of combining two objects into a third is neither really part of the first two nor the resulting object but rather somewhere in between. Workarounds for this situation exist; however, it can be speculated that this is one of the reasons why there for business processes modeling (see later section) there is no equivalent of object-oriented process modeling.

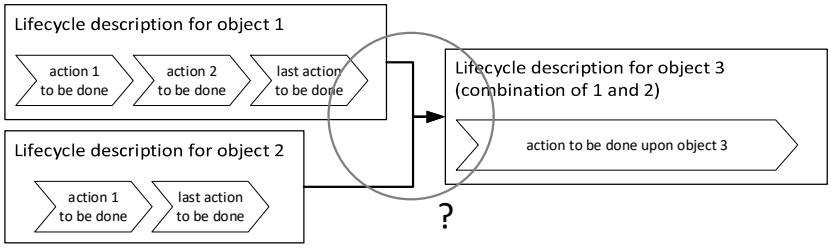

Fig. 2. Object-oriented process modeling concept (sketch). Processes are described in the passive tense for different objects individually. The context of transitions, termination, and object transformations is unspecific (from [6])

The third modeling or description problem is that of the modeling/programming threads. In essence, the problem arises when parallel activities need to be described for two or more processors or subjects. Most OO programming languages (e.g.,

[4] Microsoft Word, the software used to create this paper even has a default setting that discourages a user from using the passive tense and instead go for active sentence description as they easier, more precise and therefor better to understand.

JAVA) have the means to describe so-called thread objects that are to be executed independently from each other. However, in OO description languages, *"everything is an object."* Therefore no differentiation is done between active elements (*subjects*) and passive entities (*objects*) and no language has explicit means to differentiate the two. However, the need to express the concept of *subject* as something different from and *object* is there even though they are not named that way but rather, e.g., *"active objects"*[7] possibly to not have to deviate from the "object" concept[5]. The same can be seen in the programming directive that states that threads should not hold their own data, but need to be treated differently from classical passive data objects [8].

2.4 Functional Programming

Functional programming is a very powerful description means different from the previous two programming paradigms. Its possible pros or cons cannot be discussed here. It can be stated that (admittedly based on anecdotal evidence only) opinions are divided about it either as being a superior modeling concept for computational processes or alternatively as non-maintainable nightmare only mentally accessible to its creators, which necessarily need to be well versed in this mathematical modeling method.

However, it shall briefly be analyzed what is implicitly assumed in this paradigm about *subject*, *object*, and *verb*: Namely within this modeling concept, everything is considered a mathematical function that is necessary to be evaluated by a computer, the implicitly necessary *subject*. As instructions, all functions are computational *activities* with data inputs and outputs (the *objects*). However, describing aspects from the real world and/or mapping them to functional models is not necessarily easy or intuitive.

2.5 Business Process Modeling

In contrast to the domain of programming, the modeling of (business) processes is not in its origination geared towards describing instructions for implicitly assumed processors. Rather by its definition, it is about describing complex networks, interactions, and activities of several active entities of various types, sizes, and scopes. The need to express all three aspects of subject, object, and verb is obvious if not limited to the simplest of cases.

However, as **Fig 3** shows, almost none of the existing modeling languages explicitly used for business processes (as well as data) allow to formally express all three aspects.

[5] The argument here is that an „active object" basically is the description of a "subject" but since it is programmed using classes which get instantiated into "objects" may be mental willingness to identify them as such possible.

Considered Modeling Language	Development Paradigm	Consideration of		
		Subject	Predicate/Verb	Object
Natural Languages	n.a.	●	●	●
Flow Charts	Function-oriented	○	●	○
Event-driven Process Chains (EPC)	Function-oriented	○	●	○
Extended EPCs	Function-oriented	◔	●	◔
Entity-Relationship Model (ERM)	Data-Oriented	○	○	●
Unified Modeling Language (UML)	Object-oriented	◔	●	●
Calculus of Communicating Systems (CSS)	Subject-oriented	●	◔	○
Business Process Modell and Notation (BPMN)	Function-oriented	◔	●	◔
Parallel Activity Specification Schema (PASS)	Subject-oriented	●	●	●

Fig 3: Explicit consideration of the language elements of subject, object, and predicates in various modeling concepts (translated from [9])

Naturally, the notion of (different) actors is important for business process and many of these diagramming languages have the means to attach additional information about the active entities, however most often only as an informal and optional side note without formal implications. They mostly are task/verb in their based descriptions.

An example for additional information about active entities in business process models is the well-known swimlane concept that allows to group tasks according to their execution by the same entity. However, swimlanes are usually only graphical structuring means for human beings. While carrying the need to express the subject concept with them, rarely they have an impact on the conceptual modeling itself. They can be left off and the model would still be formally complete and consistent.

The only exception for business processes according to [9] is the Parallel Activity Specification Shema (PASS), an explicitly subject-oriented business process modeling language as introduced.

2.6 Fundamental Modeling Concepts

The final example discussed here are the Fundamental Modelling Concepts (FMC) [10, 11], a semi-formal modeling framework to describe software-intensive systems with a supposedly easily be understood graphical notation.

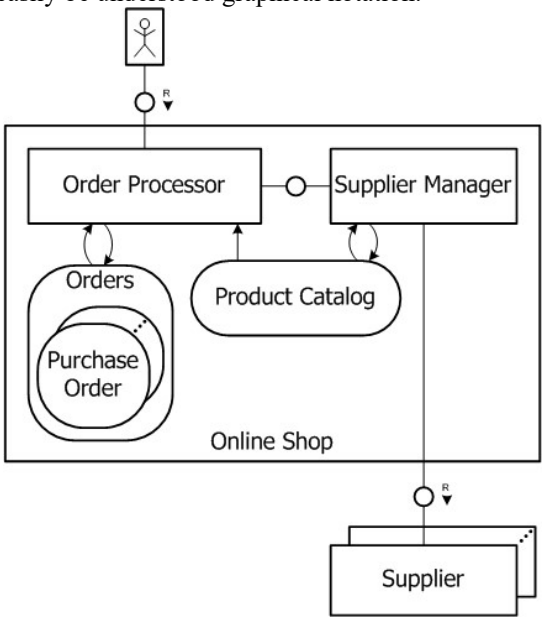

Fig. 4. Software Architecture Model using the Block Diagram Fundamental Modelling concepts [12] with active entities (squares) and passive entities (round border blocks) – the only activities expressed here are reading and writing as arrows and generalized *"communication"* between active entities.

While being semi-formal, similar to natural languages and the PASS process modeling language, it emphasizes the differentiation between active and passive elements, especially in its block diagrams (**Fehler! Verweisquelle konnte nicht gefunden werden.**).

2.7 Other Modeling or Description Means

The list of discussed approaches is far from holistic. Left out are, for example, very general and, therefore, very open modeling concepts like the Resource Description Framework (RDF) or the possibilities that, e.g., the related Web Ontology Language (OWL) offer. In both cases, it can be stated that the core element is always a name (or resource), and it is completely up to the freedom of the modeler how to use it and its interactions there. If they care for the differentiation, they can explicitly differ between *subject, object,* and *verb,* but they are not required to do so conceptually.

Beyond that, it is expected that modeling methods exist that also differ between the concepts. However, none is known that does so explicitly and consider it an important and essential modeling aspect.

3 A missing Concept?

As shown, implicitly, the concept or idea of *subject*, *object*, and *verb* exist in many modeling approaches, hypothetically in all. However, rarely explicitly nor even rarer as a must-be-used requirement. While descriptions of activities (or computations) essentially exist in every modeling method and due to object-orientation, the concept of *object* as modeling focus is not exactly unknown; it is the explicit and mandatory consideration of the subject that is missing from many. That does not mean that it is not there, but that it is not considered an elemental part or as the type of information necessary to be described explicitly at the "beginning" of a model, similarly to how an easy-to-understand active sentence is grammatically required to start with a subject.

The according paradigm that calls for exactly that is called Subject-Orientation: A modeling or description paradigm for originally intended for processes that requires the explicit and continuous consideration of active entities within the bounds of a process as the conceptual center of description. Active entities (*subjects*) and passive elements (*objects*) must always be distinguished and activities or tasks can only be described in the context of a subject. The interaction between *subjects* is of particular importance and must explicitly be described as the exchange of information [6] [13].

Next to the initially given hypotheses[6], a third one would that differentiating explicitly between all three may help avoid confusion in any kind of model, especially when it comes to summaries or abstractions. E.g., abstraction relationships in models typically are something like *"is-a"* (inheritance), *"contains/is collection of",* or "*uses"*. However, not all fit together with all modeled concepts: An activity may consist of other (sub) activities, but a subject *executes* activities; it does not *consist* of activities. Equally, an activity can *be* neither an active entity nor an object. However, a subject may consist of sub-entities or abstraction wise *be* of a type. Similar logical restrictions go with concepts of *objects.* E.g., a *subject* uses an *object,* but it does not *consist* of it[7].

Without a strict separation, small logical gaps are likely to occur in understanding, either because of erroneous modeling or because of incorrect interpretation when it is not clear what concept is meant. E.g., is a box in a graphical model labeled as "Business Planning" referring to the activity or to an organizational unit? One is executed at a certain point in time with a given amount of input data, while the other is a con-

---

[6] As in their nature those hypotheses cannot be proven but rather would needed to be disproven. The given examples, in their briefness and incompleteness tried to argue about their origins and indicate their relevance.

[7] Modelling wise, it could be argued here that, e.g., a computer as an active entity consist of many passive objects like memory, hard drives or processors. However, this refers to the non-animated object of the physical computer where the computer or computer system as an active entity in a model description is an abstract idea that makes uses of physical object.

tinuously operating entity or system that is to be handled and described quite differently. With expert support, errors or misunderstandings may quickly be resolved and, therefore, not even considered that harmful. However, if the conceptual differentiation of *subject*, *object*, and *verbs* is what is being done anyway when describing or modeling due to the way human beings perceive and especially describe and model the world, why not embrace it? Why not embrace subject-orientation as fundamental modeling concept in each and every modeling domain?

References

1. M. S. Dryer, “Chapter Order of Subject, Object and Verb,” 2017. [Online]. Available: http://wals.info/chapter/81. [Accessed 06 02 2017].
2. H. Buchwald, The Power of ‘As-Is’ Processes, Karlsruhe: Springer, 2009, pp. 13-23.
3. H. Buchwald and M. Elstermann, Propädeutikum Java : Ein Buch zum Selbststudium, Karlsruhe: KIT Scientific Publishing, 2012.
4. R. Herken, The Universal Turing Machine: A Half-Century Survey., Wien: Springer, 1995.
5. M. Parbel, "Programmiersprachen 2017: Vielfalt ist gefragt," 13 10 2017. [Online]. Available: https://www.heise.de/developer/meldung/Programmiersprachen-2017-Vielfalt-ist-gefragt-3861018.html.
6. M. Elstermann, Executing Strategic Product Planning, Karlsruhe: KIT Scientific Publishing, 2020.
7. G. Oprean and J. Pedersen, “Asynchronos Active Objects in Java,” in *Communciation Process Architectures 2008 - CPA conference*, New York , 2008.
8. D. Ratz, j. Scheffle, D. Sees and J. Wiesenberger, Grundkurs Programmieren in Java - 8. Auflage, Hanser , 2014.
9. W. Schmidt, A. Fleischmann and O. Gilbert, "Subjektorientiertes Geschäftsprozessmanagement," *HMD Praxis der Wirtschaftsinformatik,* pp. 52-62, 2009.
10. FMC-modeling.org, "Fundamental Modeling Concepts," [Online]. Available: http://www.fmc-modeling.org/. [Accessed 01 06 2020].
11. A. Knoepfel, B. Groene and P. Tabeling, Fundamental Modeling Concepts - Effective Communication of IT Systems, Wiley, 2006.
12. Ihpiv, "Wikipedia: Fundamental Modeling Concepts," [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/4/44/FMCBlockDiagram.png. [Accessed 01 06 2020].