

Towards On-The-Fly Creation of Modeling Language Jargons

Ilia Bider^{1,2}, Erik Perjons¹ and Dominik Bork³

¹Department of Computer and Systems Sciences (DSV), Stockholm University, 164 55 Kista, Sweden

²Institute of Computer Science, University of Tartu, Estland

³TU Wien, Business Informatics Group, Favoritenstrasse 11, 1040 Vienna, Austria

Abstract

When designing modeling languages, researchers and practitioners often take contradicting positions. While the former argue for formal soundness and precise specification, practitioners aim for more flexibility in using a modeling language, i.e., by adapting it to a specific project context. The same distinction applies to the differentiation between modeling tools and drawing tools. While the former enable model processing functionality and support modelers in creating only valid models, in practice, too often the latter tools (e.g., Visio or even PowerPoint) are used due to their ease of use and non-constraining nature. When aiming to introduce flexibility into modeling languages, one often needs to have metamodeling knowledge. Besides, changing the metamodel requires re-deployment of tools and hampers compatibility with previously created models. In this paper, we introduce the notion of *modeling language jargons* as a means to countervail these two contradicting positions. With such jargons, on-the-fly adaptation of modeling languages can be achieved by business users without requiring changes to the metamodel. To illustrate the conceptualization and use of modeling language jargons we report on the Fractal Enterprise Model (FEM) language and the experience of using the FEM tool in practice.

Keywords

Modeling language, Flexibility, Fractal Enterprise Model

1. Introduction

This paper has its origin in an ongoing project of designing and using a tool for Fractal Enterprise Models (FEMs) - the FEM toolkit. A FEM [1] connects an enterprise's business processes with assets used in these processes or being managed by them. As a result, the model depicts the enterprise's operational activities important for a particular goal, e.g., problem analysis, redesign, etc. The FEM toolkit is being developed on the metamodeling platform ADOxx [2]. Though the tool is specialized for FEM and the development is specific to ADOxx, the requirements we derived from using FEM and first tool prototypes in projects, such as [3, 4] are more general and can be encountered in other modeling tools being developed for other modeling languages on other metamodeling platforms – or even without such a platform.

The main source of the challenges that we encountered is connected to the fact that FEM is a general enterprise modeling technique that can be used for various purposes. Dependent on


ICTERI'21: 17th International Conference on ICT in Education, Research, and Industrial Applications

✉ ilia@dsv.su.se (I. Bider); perjons@dsv.su.se (E. Perjons); dominik.bork@tuwien.ac.at (D. Bork)

🆔 0000-0002-3490-6092 (I. Bider); 0000-0001-9044-5836 (E. Perjons); 0000-0001-8259-2297 (D. Bork)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

the purpose, additional means for expressing particularities of a specific project might emerge. Adding additional means to a generic language and tool as soon as they are encountered increases the complexity of the modeling language and impedes its use. This has already happened to many standardized languages, like BPMN [5, 6], which started with a small set of concepts aimed at business people, but ended up with hundreds of concepts, many of them too difficult to understand for non-technical people (see critique in [7, 8, 9]). There exists thus a need to provide instruments to add additional means to express domain- or project-specific concepts without overloading modeling languages and tools.

One obvious approach to adapt a modeling language is allowing direct manipulations of the metamodel and thus, explicitly introduce new concepts related to a particular purpose or project. This approach is particularly suitable if there is a need to have formal semantics of the language and its extension, i.e. in order to process the model by built-in algorithms, which is why most standardized modeling languages allow that (see the stereotyping mechanism for UML [10] and ArchiMate [11]). However, such changes require metamodeling expertise and most often also technical skills which modelers usually do not possess. Even if the required skills would be available, conformance of existing models with the new metamodel cannot be considered as granted. Eventually, there is a risk that extensions made for one project will follow to the next one etc. which leads to overarching metamodels and tools that hamper human comprehensibility and usability of the tool.

In the paper at hand we propose an approach that allows flexible adaptations of modeling languages to specific contexts without requiring changes to the metamodel. Such adaptations can be performed by business people and are primarily based on changing the visual appearance of the shapes representing the language concepts, e.g., as background colors. The interpretation of the adapted visual appearance are left to the modelers engaged in the project. In essence, this approach facilitates project and domain-specific *modeling language jargons*. This approach obviously has a disadvantage based on the informal convention accepted for the project at hand. However, this disadvantage is not severe if the models are meant to be interpreted by people of that project; not by formal algorithms. The contribution this paper aims to make is well positioned in the ongoing endeavors of the enterprise modeling community to elevate enterprise modeling for the masses [12, p. 229] where, amongst others, "*local practices in capturing knowledge*" and softening "*completeness, coherence and rigor requirements to models and modelling languages in some contexts*" are described as necessary steps toward reaching this goal.

The rest of the paper is organized as follows. Section 2 provides relevant background on metamodeling and FEM, and introduces the running example. In Section 3, we introduce the three ways of defining and using *modeling language jargons*: *Subclassing*, *Ghosting*, and *Nesting*. The proof-of-concept implementation for the FEM modeling tool is then presented in Section 4. In Section 5, we briefly refer to related works before we conclude this paper with a summary of the results achieved so far and an outlook to future research in Section 6.

2. Background

2.1. Metamodeling platforms

Metamodeling platforms are used for the development of modeling tools by raising the abstraction level to a more elaborate level that aims to be adequate also for non-programmers. This is achieved by providing pre-configured functionality attached to a generic meta-metamodel. The method engineer then only needs to adapt this meta-metamodel to her domain. Moreover, engineers can benefit from existing tool developments and reuse/extend existing implementations.

ADOxx [2] is one such metamodeling platform that has been widely used in academia and industry. The platform comes with a rich set of domain-independent functionality like model management and user interaction. Method engineers need to: 1) configure the specific meta-model by referring its concepts to the ADOxx meta-metamodel; 2) define a visualization for the concepts; 3) combine the concepts into logical chunks, i.e., ADOxx modeltypes; and 4) realize additional model processing functionality like model transformation or simulation.

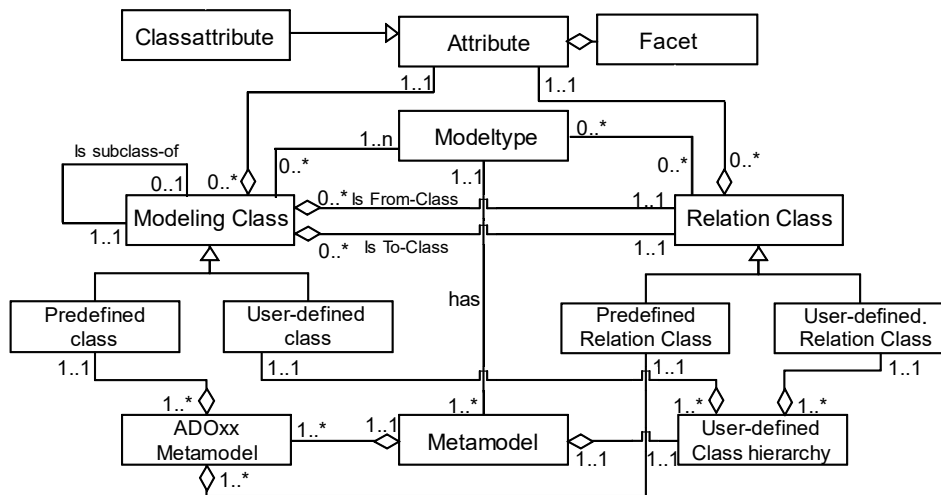


Figure 1: Excerpt of the ADOxx meta-metamodel [13]

An ADOxx-based modeling tool is composed of modeltypes which themselves comprise predefined and user-defined modeling classes and relation classes (see Fig. 1). Following a graph-based structure, modeling classes refer to nodes and relation classes to edges between nodes. Modelers instantiate a modeltype and use the provided classes to create models. Attributes define the semantics of all ADOxx classes. Amongst, basic attribute types like string and integer, ADOxx provides *interref* attributes that allow semantic relationships between modeling class instances of the same or of different models. The ADOxx meta-metamodel contains generic concepts and attributes that are inherited to the specific modeling language (e.g. the metamodel of FEM). These concepts and attributes, for example, the attribute "interref", can be applied by the modeler without the need to change the metamodel. This is the approach we are applying when conceptualizing modeling jargons, which will be detailed in forthcoming sections.

2.2. Fractal Enterprise Modeling

In this section, we present the main principles of building Fractal Enterprise Models (FEM) – more background on FEM can be found in [1, 14]. The basic components of a FEM model are: *business processes*, *assets*, and *relationships* between them. Fig. 2 shows a fragment of a FEM model that will be used for explaining the FEM concepts and that introduces the business case used throughout this paper.

Graphically, a process is represented by an oval, an asset is represented by a rectangle (box), while a relationship between a process and an asset is represented by an arrow. We differentiate two types of relationships in FEM. One type represents a relationship of an asset *used by* a process; in this case, the arrow points from the asset to the process and has a solid line. The other type, referred to as *affects* relationship, represents a relationship of a process changing the asset; in this case, the arrow points from the process to the asset and has a dashed line. These two relationship types allow tying up processes and assets in a directed graph. In FEM models, labels are used to further specify the semantics of the instances, e.g., the names of processes and assets placed inside the oval and rectangle, respectively. The labels of FEM relationships indicate the specific relationship type being used. For example, the label *acquire* identifies that the process *Sales* can/should increase the pool size of the asset *Stock of manufacturing orders* in Fig. 2.

In FEM, the same asset can be used in multiple processes playing the same or different roles. Likewise, the same asset may play multiple roles in the same process. Similarly, a process could affect multiple assets, each in the same or in different ways (see the two relationships between the *Sales* process and the corresponding assets in Fig. 2). Moreover, a single process may affect a single asset in multiple ways. Multiple relationships between assets and processes can be represented by having two or more labels on the corresponding arrows.

Labels defining the relationships between processes and assets are standardized in FEM. The method defines eight abstract relation types that further specify the semantics of the *used by*: *Beneficiary*, *Workforce*, *EXT*, *Partner*, *Stock*, *Tech & Info Infrastructure*, *Organizational Infrastructure*, *Means of Payment*, and *Attraction*. Furthermore, three abstract relation types that describe how an asset *is managed* by a process exist: *Acquire*, *Maintain*, and *Retire*.

Two new concepts were introduced to FEM in order to represent the business context of the organization and connect it to specific processes [14]: *External pool* and *External actor*. External pools, visualized by a cloud shape, represent a set of things or agents of a certain type (see the pool of *companies that need the manufacturing products* in Fig. 2). External actors, visualized by a rectangle with rounded corners, represent an agent (e.g., a company or person) acting outside the boundary of the organization. If a set of external actors shall be represented like the *competitor* external agents in Fig. 2, the border of the shape has a double line. External pools and external actors may be related to each other and to other elements of the FEM diagram.

This concludes the brief introduction to FEM. Readers interested to learn more about FEM and why it is called fractal are again referred to [1, 14].

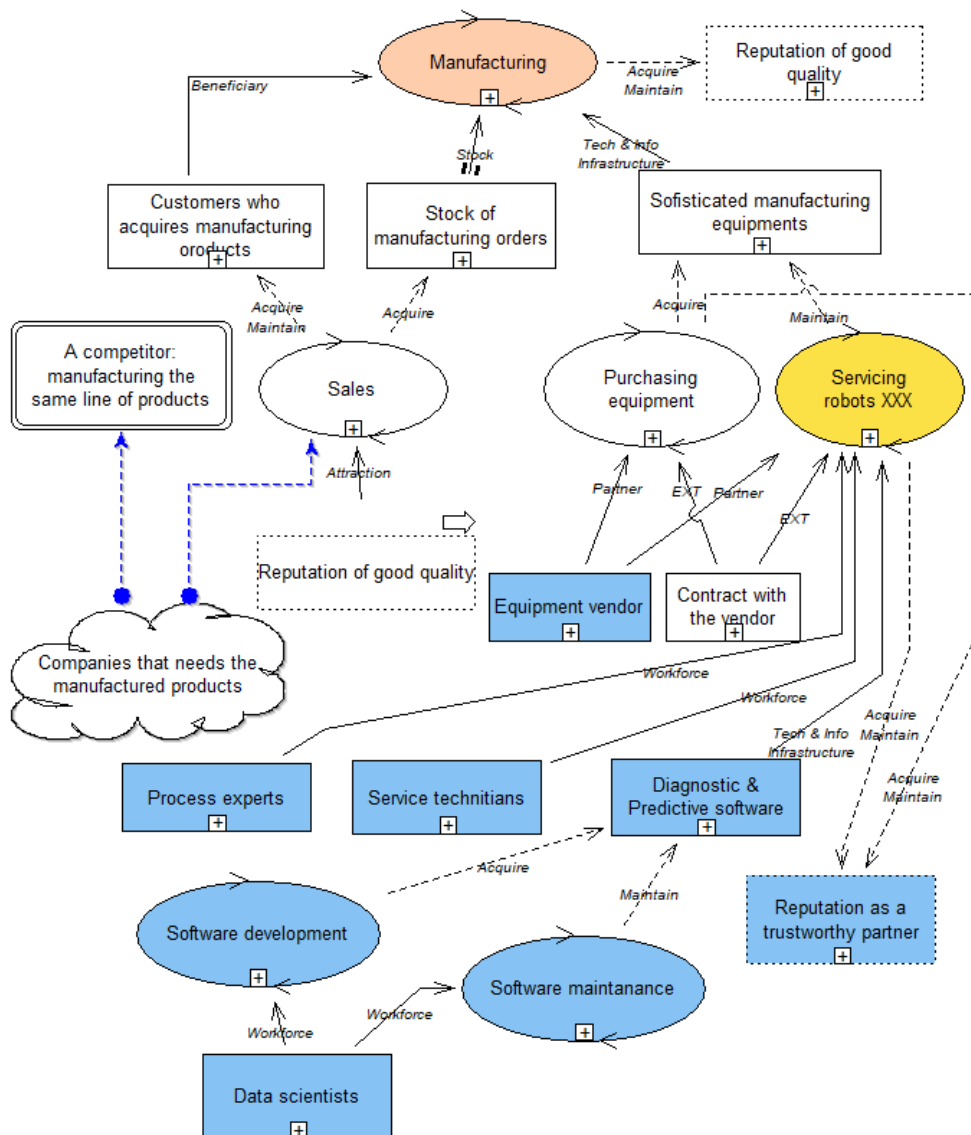


Figure 2: Fragment of a FEM for the manufacturing business model

2.3. Running example

To illustrate and discuss our proposal we will be using a running example taken from [15]. The example belongs to the field of Business Model Innovation (BMI), more specifically to the field of so-called industry level innovation according to the classification from [16]. This kind of innovation means developing products and/or services that are radically different from the current ones, and/or entering completely different markets. In the center of the case is a manufacturing company that uses an expensive equipment (robots) in their manufacturing process. To make the best use of this equipment, they have developed diagnostic predictive

software. The software is used for deciding when the equipment needs servicing, rather than relying on the standard recommendations from the maintenance plan. This allows to avoid unexpected breaks, while increasing intervals between the stops for planned maintenance.

Developing and supporting the software requires having specialists, and other types of investments. The company decided to explore a way to convert costs to profitable investment by starting a new business of licensing the software to other manufacturers that use the same equipment, including their current competitors.

A FEM model of the case company is represented in Fig. 2. The model is not complete, the purpose is to show how the company currently operates in general, and to show more details related to the diagnostic predictive software.

3. Modeling Language Jargons

In a modeling project, there often is a need to on-the-fly *graphically differentiate* elements that are represented using the same language concept. For example, in a FEM model, one might need to differentiate physical assets, e.g., computers, from the virtual assets, e.g., electronic documents or IT systems. Another recurring need we experienced in many FEM modeling projects is that modelers want to *reuse* already modeled elements inside the same, or even in a different FEM model. Eventually, a need for *nesting* was ubiquitous, i.e., modelers wanted to specify enterprises on different levels of granularity.

As one approach to tackle the three modeling needs stressed previously, the paper at hand introduces the notion of *modeling language jargons*. The Oxford dictionary defines 'jargon' as: "*words or expressions that are used by a particular profession or group of people, and are difficult for others to understand*" [17]. The Cambridge dictionary defines jargon as: "*the special slang of a particular group of people*" and "*special words or phrases used within a group, trade or profession*" [18].

A modeling language jargon thus provides additional means for those knowledgeable in the specific terms and their meanings while at the same time hampering the comprehension by a wider, unknowledgeable audience. Such a jargon is specific to a modeling project and the involved stakeholders and is expected to change when starting a new modeling project in a new context. In the following, we introduce three approaches for enabling project-specific modeling language jargons that all do not require any metamodel changes. We will refer to and further extend the FEM example depicted in Fig. 2 to exemplify the use of modeling language jargons in FEM.

3.1. Subclassing

The aim of the *Subclassing* approach is to enable modelers to on-the-fly introduce more specific semantics to the concepts provided by the FEM metamodel (e.g., Process and Asset). In a business model innovation project, like the one in our running example, the modelers could be interested to distinguish automated processes from manual processes, or to distinguish processes and assets that need to be introduced to realize a new business model.

The only requirement for creating subclasses of the existing language concepts in such cases is to have a unique visual representation of each subclass. A simple enough way for such

visualization would be assigning a specific background color to each subclass. In this respect, all subclasses still use the same shape, e.g., an ellipsis for FEM processes, but allow the identification of an individual subclass based on its unique background color. This differentiation is related to the type and inherent semantics dichotomy introduced in [19] where the metamodel concept defines the type semantics (e.g., what is the semantics of a process in FEM) and the labels of the process instances further specify the instance semantics (e.g., what is a primary process in Fig. 3).

Fig. 3 shows a set of potential subclasses for the new business model in our running business model innovation example. Labels inside the shape define the semantics of each subclass for the project. Most colors are already used in Fig. 2 to identify which elements of the FEM model are intended to be introduced when designing a new business, the red color will appear in Fig. 4 and explained subsequently.

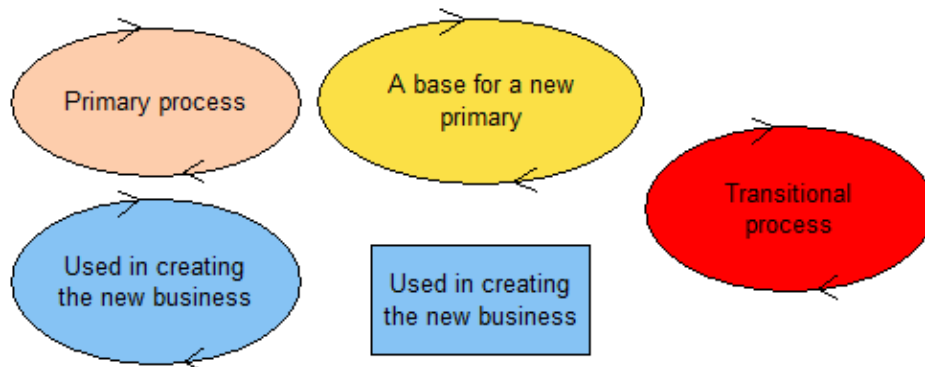


Figure 3: Subclasses defined for the case

3.2. Ghosting

Inspiration for the ghost concept was derived from a tool called InsightMaker [20], where it was introduced to solve the problem of a model becoming cluttered if too many connections are made to the same element. In InsightMaker, a ghost is a copy of an already existing element in the same model. Ghosts can be distinguished from their original by the use of a lighter background color.

Our extended notion of ghosts in modeling language jargons comprises the following three aspects:

- *Usage across models*, i.e., ghosts can relate to original elements in the same or in a different FEM model of the same modeling project. In Fig. 4, we have a number of ghosts that relate to elements that were introduced in Fig. 2 (like *Software development* process and *Service technicians* asset).
- *Navigation*, i.e., allowing to find all usages of a given FEM element starting from a ghost, or the original.
- *Visual distance*, i.e., ghosts need to be distinguishable from the original shape, but this is not constrained by relying on the background color.

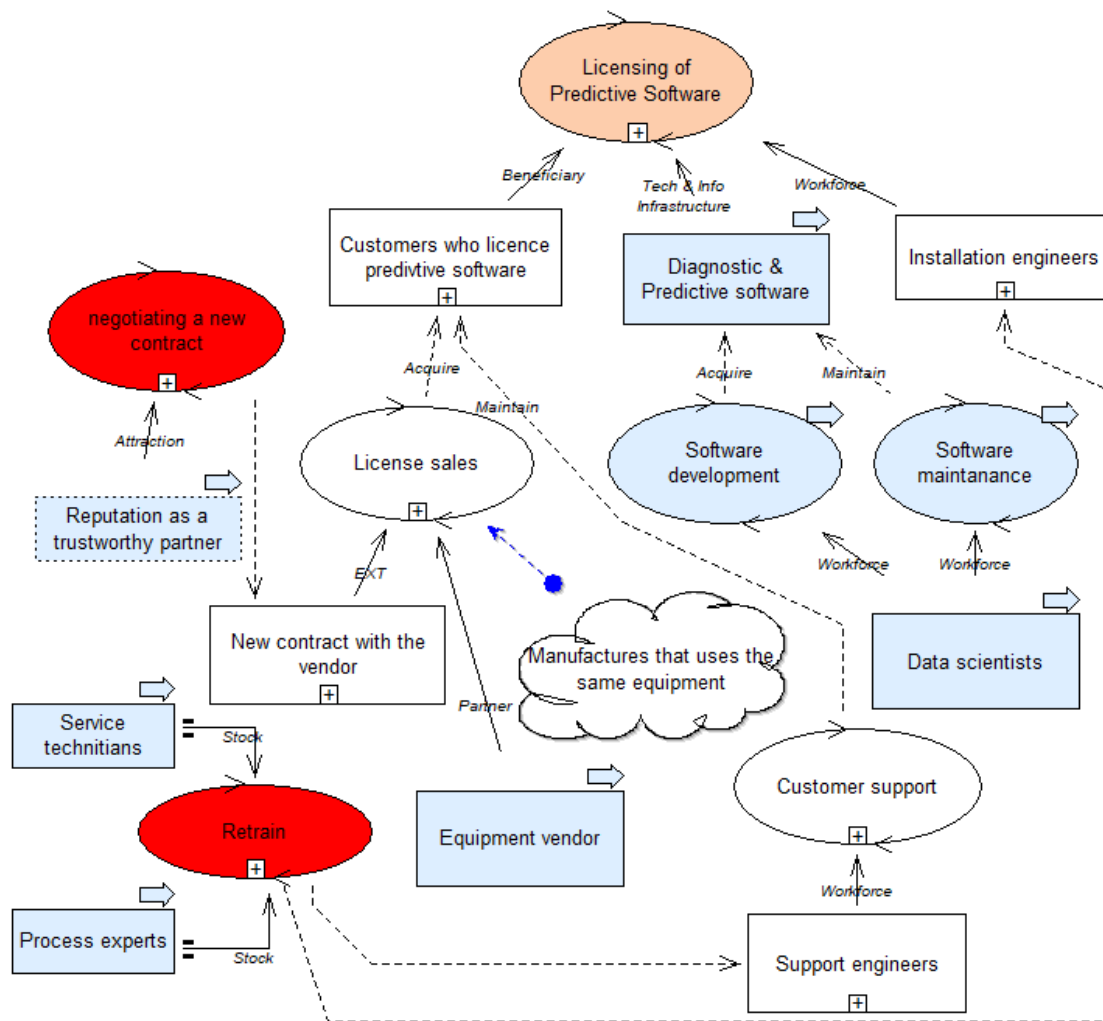


Figure 4: Fragment of a FEM that shows the new business models with many ghost elements relating to the elements proposed in the existing business model (see Fig. 2)

While subclassing adds a vocabulary to a jargon, the extended ghosting approach allows modelers to add specific modeling styles to the jargon. Ghosting, for example, can be used to create model views and visual repositories without introducing these concepts in the language. Fig. 3 and Fig. 4 demonstrate a usage of extended ghosting in the business model innovation project. First, one builds a model of the current business and marks elements that can potentially be used in a new business. Then one moves the identified elements as ghosts to another model, adds new nodes and configures a new business model. Note that in Fig. 4, we use an additional subclass from Fig. 3 - the two elements with red background color - to mark the transitional (temporal) processes in the model.

The extended ghosting approach thus not only supports reuse of existing model elements,

enabling the modeler to on-the-fly create a repository of elements, it also enables rearrangement of model elements in a more comprehensible way by reducing long-distance and crossing relationships by simply adding an additional ghost element at the position the original element would be required.

3.3. Nesting

Quite often, a modeler needs to use different levels of granularity when creating an enterprise model. For a high-level overview, the granularity will be coarse, while a view on the detailed aspects of a particular part of the business requires a finer level granularity. There needs to be a way to relate models on different granularity levels. For example, a business process that is depicted as one entity in one model may need to be related to an interconnected set of subprocesses in a more detailed model. Some languages, like IDEF0 and BPMN, introduce the concept of decomposition with strict semantics. Other languages, including FEM, may not introduce the concept explicitly, but the practice often needs some means for decomposition.

The semantics of the decomposition may vary from project to project. In one case, the decomposition may mean variants of the same thing, which is considered as specialization in some modeling languages, like UML. In other cases, the decomposition may mean depicting smaller elements and their interconnections that together makes the decomposed elements. There can be other alternatives of the decomposition concept as well. Instead of introducing the concept explicitly with all its variants, we suggest to use visual means that allow to present decomposition on-the-fly without introducing strict semantics. The semantics in this case, should be understood from the decomposition itself, or be agreed upon inside the project group.

We suggest using *nesting* in combination with the extended ghosts approach to represent a decomposition. This is illustrated in Fig. 5, which shows the decomposition of the *Sales* process from Fig. 2. The decomposed process is presented as special type of ghosts – *group ghost*, and it comprises two new processes, *Sales to existing customers* and *Sales to new customers*. To show the connection between the new processes and other elements of the model from Fig. 2, ghosts of the connected elements are presented in the new model. The decomposition in this example is of *specialization type*, a further type is referred to as *subprocess type*, where the decomposed elements are also connected inside the group. The ghosts navigation facilitates easy movement from the decomposed element to the model that depicts the decomposition and back. Nesting thus enables modelers to create new styles in a modeling language jargon on-the-fly.

4. Modeling Language Jargons for FEM

As a Proof of Concept (PoC) we will report in the following on the conceptualization of modeling language jargons in a concrete case: the Fractal Enterprise Modeling (FEM) [1, 14]. We will present and discuss the realization of the FEM modeling tool which runs as a project within the Open Models Laboratory (OMiLAB) [21]. Albeit the specific case, the conceptualization itself will be described in a way that shall enable not only comprehension but also facilitate the implementation of modeling language jargons in other modeling tools. Please note, that the conceptualization requires technical knowledge and only needs to be done once, while the

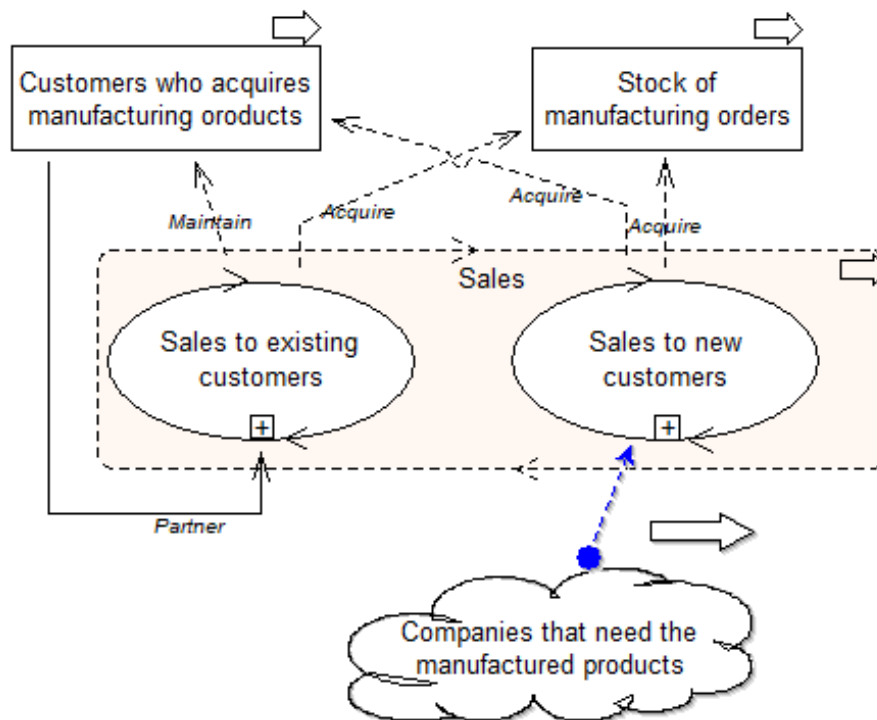


Figure 5: FEM fragment showing the nested Sales process (specialization type).

definition of arbitrary modeling language jargons can be done on-the-fly by business users without any further changes to the implementation or metamodel.

4.1. Conceptualization of Subclassing

For the conceptualization of Subclassing in the FEM tool we introduced an additional modeltype: *FEM Subclassing* which complements to conventional *FEM* modeltype one would use to create FEM models as the one in Fig. 2. The new modeltype basically acts as a repository of all subclasses of all FEM modeling classes. Modelers can instantiate FEM processes or assets and thereby define a new subclass of that type which can be used on-the-fly in all other FEM models that form part of the current modeling project. When creating a new subclass, the modeler can specify its properties (like the name) and its appearance.

When aiming to use a subclass, FEM modelers are assisted by a smart menu that lists all existing subclasses of the current project defined in the FEM Subclassing model. As soon as the modeler selected a subclass a reference is stored in the properties of the subclass and the appearance of the subclass is automatically propagated to the newly linked subclass, thereby replacing the default appearance. When removing the subclassing reference, the element is automatically returning to its default appearance. Likewise, synchronization has been realized to make sure that changes in the appearance of the subclass in the FEM Subclassing model are propagated to all referenced subclasses in all other FEM models of a modeling project.

Fig. 6 shows the use of the Subclassing approach. After opening the properties of a new Process element, modelers can select 'Subclass' in the Color Picker property (see number 2 in Fig. 6). An automatically generated dialog then lists all applicable subclasses for the Process. After the modeler selected one subclass, the representation of the subclass (in this case a red background color defined in Fig. 3) is automatically propagated to the new Process. In the shown example, the process on the right is now further specified as being a *transitional process*.

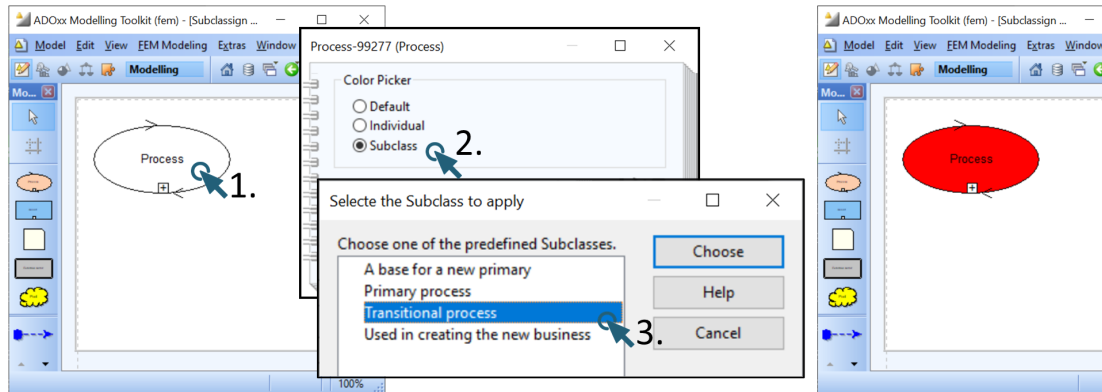


Figure 6: Conceptualization of the Subclass approach in the FEM tool

For the project participants, the subclassing model in Fig. 4 presents a convention on which they agree. It can also be viewed as an agenda, or vocabulary of the project's jargon. The scope of the jargon is a group of models that share the jargon, i.e. the same subclassing model. Consequently, any FEM modeling project will have one Subclassing model and multiple FEM models.

4.2. Conceptualization of Ghosting

The conceptualization of ghosting in the FEM modeling tool is primarily realized by an Interref attribute that is available for all FEM metamodel concepts. With this Interref, the modeler is enabled to link a selected FEM instances to an already existing instance of the same type in the same model or in a different model of the same modeling project. Similarly to the Subclass procedure introduced previously, the conceptualization of ghosting is build on two major functions: a smart menu that supports the modeler in selected an appropriate original, and automated adaptation of the graphical representation of the ghost instance. Consequently, the appearance of the ghost instance is automatically adopting the appearance defined in the linked original element. Noteworthy, for ghosts, the synchronization between original and ghost also concerns other attribute values like the name. If the modeler changes the name of the original, the new name is propagated and all ghost instances are updated automatically thereby keeping all FEM models in a consistent state [22].

In order to support comprehensibility of large FEM modeling projects that might require many FEM models, the FEM tool also comes with some processing functionality that enable the modeler to efficiently list all ghost instances of a selected original in the current model as well as in all other models of the FEM modeling project. When executing this functionality

an algorithm queries all FEM models to find the usage of the selected element as ghost and provides a list that has a navigable link to the specific model and element. The modeler thus only needs to click on an entry on the list of ghosts to navigate to its usage. Fig. 7 shows the navigable list generated as a result of executing the 'find usage in all FEM models' functionality on the process *Software maintenance*. The list shows three usages of this element in three FEM models.

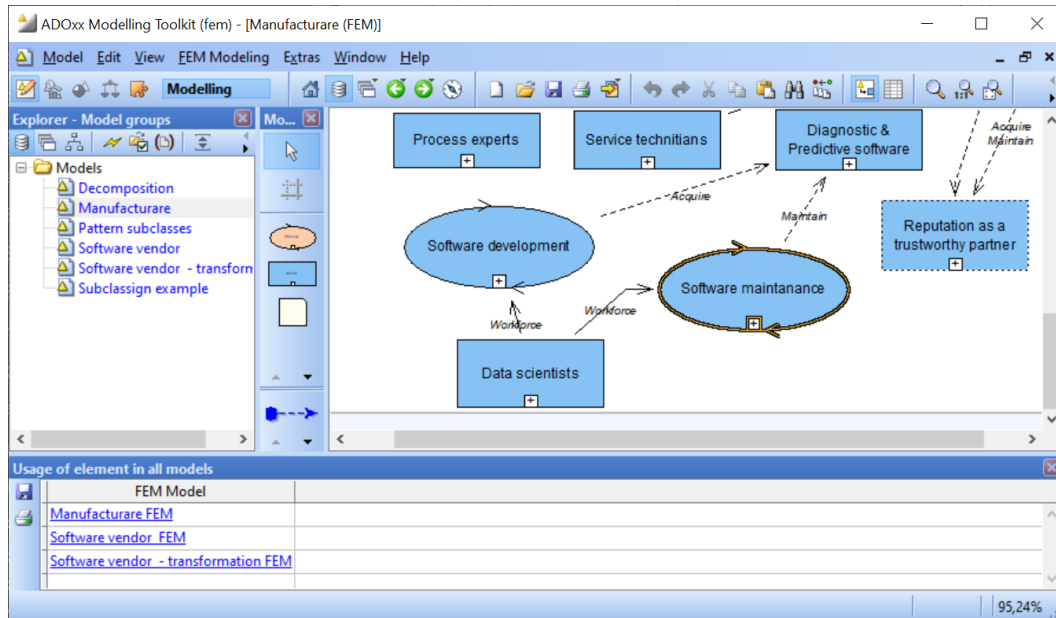


Figure 7: Conceptualization of the Ghosting approach in the FEM tool

4.3. Conceptualization of Nesting

The Nesting approach is conceptualized also on the FEM metamodel level and allows its on-the-fly use by modelers in any FEM modeling project. All FEM concepts provide an attribute that qualifies them for being a *group* or not. By changing the attribute value, the appearance of the selected element automatically changes, e.g., from a solid border line and a strong background color to a dashed border line and light background color. This change is exemplified when comparing the *Sales* process in Fig. 2 with the *Sales* nesting visualized in Fig. 5.

As introduced in Section 3.3, grouping can also be applied to ghosts. Again, the example of Fig. 5 can be referred to where *Sales* is not only a ghost (visually encoded by the dashed border and the light color) but also a ghost of the original *Sales* process in Fig. 2 (visually encoded by the arrow on the top right corner that would navigate the modeler to its original when being clicked).

Table 1
Contrasting modeling language jargon from profiling/stereotyping

Criteria	Modeling Language Jargon	Profiling / Stereotyping
Adaptation role	Modeler	Method engineer
Adaptation point	Model	Metamodel
Adaptation scope	Notation	Syntax
Adaptation time	on-the-fly	ex-ante
Adaptation effect	Expressivity and processing ↑	Expressivity ↑

5. Related works

The intention to introduce flexibility in modeling methods is not new in itself (cf. [5, 23, 24, 12, 25, 9, 26]). In the following, we will briefly elaborate on the most relevant related works in order to position modeling language jargons within the state of the art. Afterwards, we conclude the paper with a summary of its contribution and future research directions.

FlexiSketch is a very interesting approach comprehensively introduced in [25]. With FlexiSketch modelers are able to sketch models and automatically transform their model sketches into a metamodel and a corresponding modeling tool. In contrast to our approach, FlexiSketch primarily focuses on sketching new elements and designing the metamodel on-the-fly. This approach comes with promising possibilities while the problems of e.g., ever increasing metamodel size with every new modeling project and interoperability remain.

The AOAME approach proposed in [27] aims to enable business people with domain knowledge to adapt the modeling language to specific project contexts. AOAME binds a modeling language to an ontology that provides clear, unambiguous, and machine-interpretable semantics. This ontology can then be adapted on-the-fly thereby enabling the modeler to introduce context-specific semantics. Apart from the fact, that AOAME puts the ontology at the heart and users should have basic knowledge in ontology engineering, users can *remove unnecessary concepts* which is in conflict to our goal of retaining conformance to the core modeling language.

In contrast to the previously mentioned, existing, stereotyping and UML profiling [10, 11] approaches, modeling language jargons aim to be usable by business users that don't have any technical skills. Jargons are defined on-the-fly during the modeling process in the modeling tool instead of ex-ante implementation on the metamodel backbone as for stereotypes and profiles. Moreover, we propose means to navigate inside and between the models in order to find jargon usages in whole corpora of created models. Table 1 further contrasts our proposed modeling language jargon from profiling and stereotyping.

6. Conclusion

In the paper at hand we propose the notion of *modeling language jargons*, a means to on-the-fly adapt modeling languages to specific contexts. Key to our approach is that the language adaptations don't involve any changes to the metamodel, as such they can be performed

by regular modelers, and the fact, that the created models are still valid according to the underlying modeling language. The three approaches forming part of modeling language jargons, *Subclassing*, *Ghosting*, and *Nesting* are introduced on a generic level.

The three approaches listed above are fully implemented in version 0.7 of the FEM toolkit, which is publicly available for download and test¹. The toolkit has already been used in two practical projects. The first project was completed at Tartu Water Utility company; it was aimed at designing a new digitalization strategy at a company with an outdated IT systems park, as well as creating more detail requirements for completing the first step of the strategy. The FEM toolkit was successfully used in many tasks of this complex project, including understanding of how the company operated, developing "as-is" and "to-be" models, tying them together via the ghosting feature, and marking changes via the subclassing feature.

The second project's practical goal was to suggest improvements for a so-called sourcing process in an international concern. The sourcing project aimed at signing long-term purchasing contracts for all branches of the concern. The project employed all three features introduced in this paper. Analysis of the produced FEM diagrams resulted in a number of suggestions for process improvements.

Besides testing in practical projects, the FEM toolkit was employed in teaching enterprise modeling at University of Tartu. We experienced, that the students had no problems in understanding the features discussed in this paper, as they successfully used them when completing project and exam assignments.

The results of testing our suggestions implemented in FEM toolkit are quite encouraging, although a comprehensive empirical evaluation, e.g., with respect to ease of use, usability, and comprehensibility of the resulting models is on a route we plan to follow in our future research. It will be interesting to see, how the gains in flexibility might influence comprehensibility. One could imagine that by adding additional semantics, the models are more expressive while on the other hand adding several different appearances for the same concept might result in an increase of cognitive load [28, 29].

Another important part of future research is to investigate transferability of the modeling language jargon to other modeling tools. We thus hope this paper motivates the modeling community to gain interest in our concept and the combined generic and case-based introduction of the concept eases the implementation of it for other modeling tools. Eventually, we see the possibility that metamodel platform vendors would be interested in ideas that enable a balance between flexibility and context-adaptability on the one hand, and conformity to a formally defined metamodel on the other. Implementing the modeling language jargon concept in a way such that modeling tool developers only need to define the metamodel concepts they want it to be enabled for seems feasible.

In the practical terms, we are working on the next version of FEM toolkit (0.8) with an additional classification option, which will be orthogonal to the existing subclassing, and which will have visualization via coloring the shapes' borders. This kind of classification might be useful in multi-organizational contexts of the type discussed in [3]. Another feature that is considered for the new version is allowing the usage of several subclassing forms along with a switching mechanism to have several, mutually exclusive perspectives on the same FEM diagram.

¹FEM tool: <https://www.fractalmodel.org/fem-toolkit/>, last visited: 09.07.2021

Acknowledgement: The authors are very grateful to Steven Leego, Erik Falenius and August Carlsson who independently tested the FEM toolkit in practical projects. The work of the second author was partly supported by the Estonian Research Council (Grant PRG1226).

References

- [1] I. Bider, E. Perjons, M. Elias, P. Johannesson, A fractal enterprise model and its application for business development, *Software & Systems Modeling* 16 (2017) 663–689.
- [2] ADOxx.org, Official homepage of the ADOxx meta-modeling platform, 2021. <http://adoxx.org>, last visited: 10.03.2021.
- [3] M. Henkel, G. Koutsopoulos, I. Bider, E. Perjons, Using the fractal enterprise model for inter-organizational business processes, in: *Proceedings of the ER Forum and Poster & Demos Session co-located with 38th International Conference on Conceptual Modeling (ER 2019)*, volume 2469 of *CEUR Workshop Proceedings*, 2019, pp. 56–69.
- [4] V. Klyukina, I. Bider, E. Perjons, Does fractal enterprise model fit operational decision making?, in: J. Filipe, M. Smialek, A. Brodsky, S. Hammoudi (Eds.), *Proceedings of the 23st International Conference on Enterprise Information Systems, ICEIS 2021, Volume 2*, SciTePress, 2021, pp. 525–533.
- [5] C. Atkinson, R. Gerbig, M. Fritzsche, Modeling language extension in the enterprise systems domain, in: *2013 17th IEEE International Enterprise Distributed Object Computing Conference*, IEEE, 2013, pp. 49–58.
- [6] R. Braun, Towards the state of the art of extending enterprise modeling languages, in: *2015 3rd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, IEEE, 2015, pp. 1–9.
- [7] J. Recker, Opportunities and constraints: the current struggle with bpmn, *Business Process Management Journal* (2010).
- [8] M. zur Muehlen, J. Recker, How much language is enough? theoretical and practical use of the business process modeling notation, in: J. A. B. Jr., J. Krogstie, O. Pastor, B. Pernici, C. Rolland, A. Sølvsberg (Eds.), *Seminal Contributions to Information Systems Engineering, 25 Years of CAiSE*, Springer, 2013, pp. 429–443.
- [9] E. Van Wyk, M. P. E. Heimdahl, Flexibility in modeling languages and tools: A call to arms, *International journal on software tools for technology transfer* 11 (2009) 203–215.
- [10] L. Fuentes-Fernández, A. Vallecillo-Moreno, An introduction to uml profiles, *UML and Model Engineering* 2 (2004) 72.
- [11] M. M. Lankhorst, A. Aldea, J. Niehof, Combining archimate with other standards and approaches, in: *Enterprise Architecture at Work*, Springer, 2017, pp. 123–140.
- [12] K. Sandkuhl, H.-G. Fill, S. Hoppenbrouwers, J. Krogstie, A. Leue, F. Matthes, A. L. Opdahl, G. Schwabe, Ö. Uludag, R. Winter, Enterprise modelling for the masses—from elitist discipline to common practice, in: *IFIP Working Conference on The Practice of Enterprise Modeling*, Springer, 2016, pp. 225–240.

- [13] D. Bork, Metamodel-based Analysis of Domain-specific Conceptual Modeling Methods, in: R. A. Buchmann, D. Karagiannis, M. Kirikova (Eds.), IFIP Working Conference on The Practice of Enterprise Modeling, Springer, 2018, pp. 172–187.
- [14] I. Bider, Structural coupling, strategy and fractal enterprise modeling, in: International Conference on Research Challenges in Information Science, Springer, 2020, pp. 95–111.
- [15] I. Bider, A. Lodhi, Moving from manufacturing to software business: A business model transformation pattern, in: International Conference on Enterprise Information Systems, Springer, 2019, pp. 514–530.
- [16] E. Giesen, S. J. Berman, R. Bell, A. Blitz, Three ways to successfully innovate your business model, *Strategy & Leadership* 35 (2007) 27–33.
- [17] Oxford Learner’s Dictionary, Definition of the term ‘jargon’, 2021. <https://www.oxfordlearnersdictionaries.com/definition/english/jargon>, last visited: 17.03.2021.
- [18] Cambridge Dictionary, Definition of the term ‘jargon’, 2021. <https://dictionary.cambridge.org/dictionary/french-english/jargon>, last visited: 17.03.2021.
- [19] P. Höfferer, Achieving business process model interoperability using metamodels and ontologies, in: H. Österle, J. Schelp, R. Winter (Eds.), Proceedings of the Fifteenth European Conference on Information Systems, ECIS 2007, St. Gallen, Switzerland, 2007, University of St. Gallen, 2007, pp. 1620–1631.
- [20] <https://insightmaker.com/>, Official homepage of Insightmaker, 2021. <https://insightmaker.com/>, last visited: 22.03.2021.
- [21] D. Bork, R. A. Buchmann, D. Karagiannis, M. Lee, E.-T. Miron, An Open Platform for Modeling Method Conceptualization: The OMiLAB Digital Ecosystem, *Communications of the Association for Information Systems* 44 (2019) pp. 673–697.
- [22] D. Bork, R. Buchmann, D. Karagiannis, Preserving multi-view consistency in diagrammatic knowledge representation, in: International Conference on Knowledge Science, Engineering and Management, Springer, 2015, pp. 177–182.
- [23] M. Bjeković, H. A. Proper, J.-S. Sottet, Embracing pragmatics, in: International Conference on Conceptual Modeling, Springer, 2014, pp. 431–444.
- [24] D. Bork, S. Alter, Satisfying four requirements for more flexible modeling methods: Theory and test case, *Enterprise Modelling and Information Systems Architectures (EMISAJ)* 15 (2020) 3–1.
- [25] D. Wüest, N. Seyff, M. Glinz, Flexisketch: a lightweight sketching and metamodeling approach for end-users, *Software & Systems Modeling* 18 (2019) 1513–1541.
- [26] Z. Zarwin, M. Bjekovic, M. Favre, J.-S. Sottet, H. A. Proper, Natural modelling, *Journal of Object Technology* 13 (2014) 4:1–36.
- [27] E. Laurenzi, K. Hinkelmann, A. Van der Merwe, An agile and ontology-aided modeling environment, in: IFIP Working Conference on The Practice of Enterprise Modeling, Springer, 2018, pp. 221–237.
- [28] D. L. Moody, Cognitive load effects on end user understanding of conceptual models: An experimental analysis, in: A. Benczúr, J. Demetrovics, G. Gottlob (Eds.), *Advances in Databases and Information Systems*, Springer Berlin Heidelberg, 2004, pp. 129–143.
- [29] S. Yusuf, H. Kagdi, J. I. Maletic, Assessing the comprehension of uml class diagrams via eye tracking, in: 15th IEEE International Conference on Program Comprehension (ICPC’07), IEEE, 2007, pp. 113–122.