

# Questionnaire Template

## First Part

**When:** .....

**Where:** .....

**Expert:** .....

**Role:** .....

**Company:** .....

**Company Website:** .....

In consulting projects:

1. *Did users face difficulties in learning standard modeling languages and using them in “name of the tool”?*
  - a. *If yes, what are the reasons for that?*
  - b. *How could this issue be overcome?*

In both research and industry, there is the recent trend of adapting standards (or existing) modeling languages to address a specific domain. In result a domain-specific modeling language is developed (DSML). This has the following benefits:

- It decreases the error prone while modeling. This is due the injection of semantics (i.e. abstract syntax and constraints) in the metamodel, which decreases the degree of freedom of modelers.
- It enhances understanding of models by domain experts. This is due to the graphical notations targeting a specific domain.

Developing a DSML through domain-specific adaptation of existing modeling languages has the following benefits:

- Modeling expert-friendly. This is due to the reference to already existing modeling standards.
- Total or partial reusability of the resulting language (i.e. DSML) within the modeling community. This is also due to the reference to already existing modeling standards.
- It fosters the quality of the modeling language. Established experience and lessons learned from existing modeling languages can be taken into account. Additionally, semantics and syntax can be borrowed.

2. *Could a DSML address one of the issues/problems identified in question 1? Are there more problems that can be addressed?*
3. *Have there been situations, where the modeling languages were not sufficient and where an adaptation could have made sense? Namely, adapting language constructs to fit a more specific domain?*
  - a. *If not, Why?*

- b. Could a functionality for an on-the-fly customization of modeling constructs be useful in projects with domain-specific target?*
- 4. For the situations, where the language was not adequate, what kind of modifications on the modeling language would have helped? e.g. creating/deleting/update a modeling construct (i.e. class, attribute and relation on the metamodel level)*
- 5. Could you provide at least a use case where domain-modeling adaptation would have made sense?*

Comments:

.....  
.....  
.....  
.....

## Second Part

- 1) Do the operators derived in the paper (see below) make sense for you?
- 2) Do you suggest other operators/actions on the language for adaptation purpose?

### Operators

**Operator 1:** Create sub-class. This operator is applied on modeling elements and modeling relations to create new modeling elements and new modeling relations. This operator is also applied to integrate modeling elements (classes) from different modeling languages. For example, the operator would allow to connect "Discretionary Task" from CMMN as a subclass of the "User Task" from BPMN.

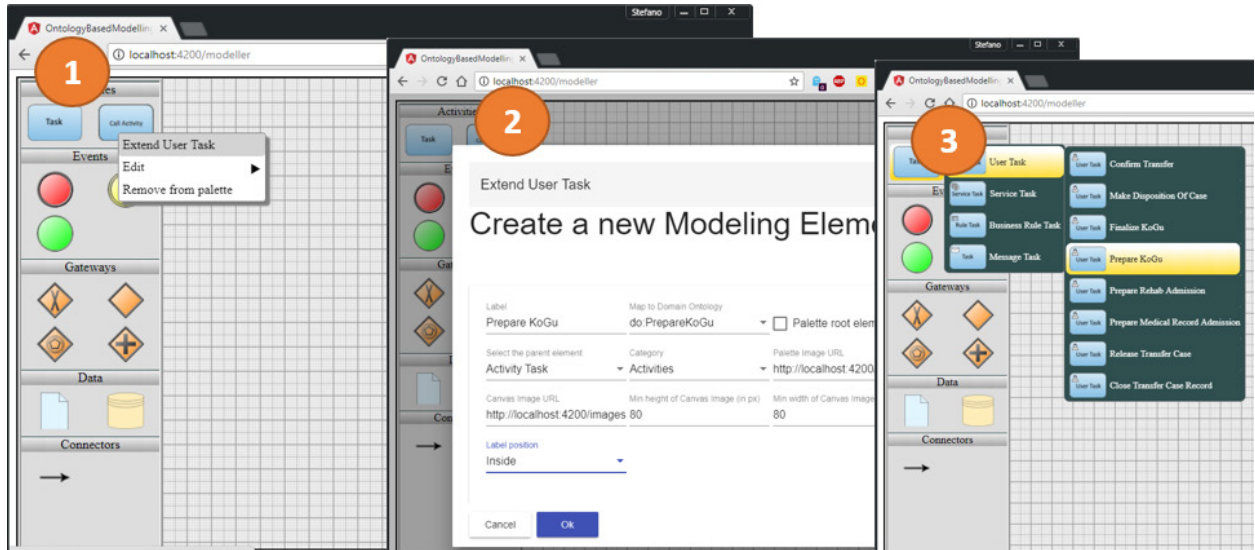


Figure 1. Operator 1 applied on "Data Object"

**Operator 2:** Delete sub-class. This operator is applied on modeling elements and modeling relations to remove unneeded modeling elements and modeling relations from the modeling language.

**Operator 3:** Create relation (object properties). This operator connects modeling elements and modeling relations to the related Domain Ontology concept.

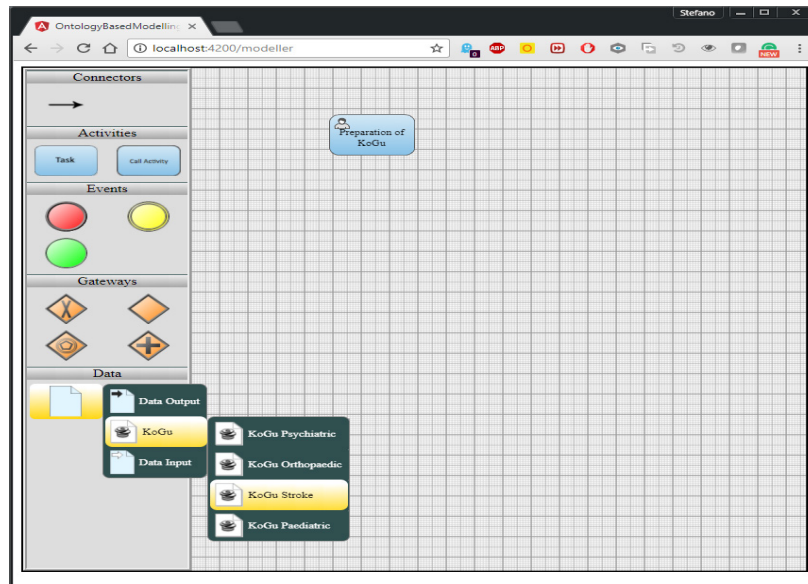


Figure 2. Operator from 1 to 3

**Operator 4:** Update relation (object properties). This operator is applied on as it allows updating existing connections between modeling elements/relations and the related Domain Ontology concepts.

**Operator 5:** Delete relation (object properties). This operator allows deleting existing connections between modeling elements/relations and the related Domain Ontology concepts.

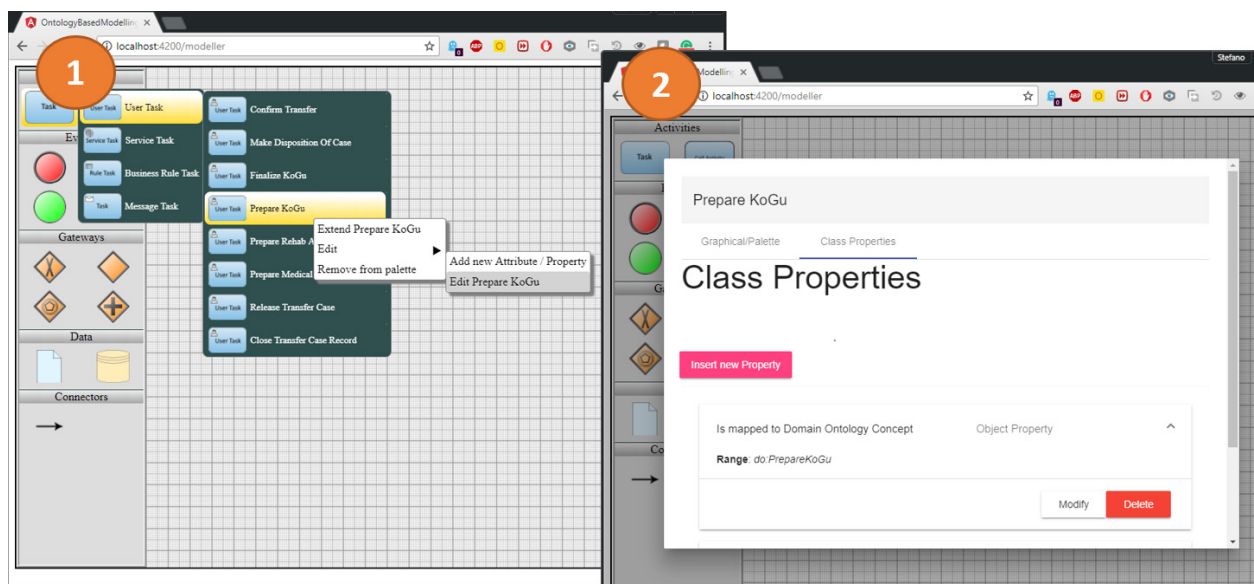


Figure 3. Operators 4 and 5

**Operator 6:** Create attribute (datatype properties). This operator allows adding new attributes to modeling elements and modeling relations.

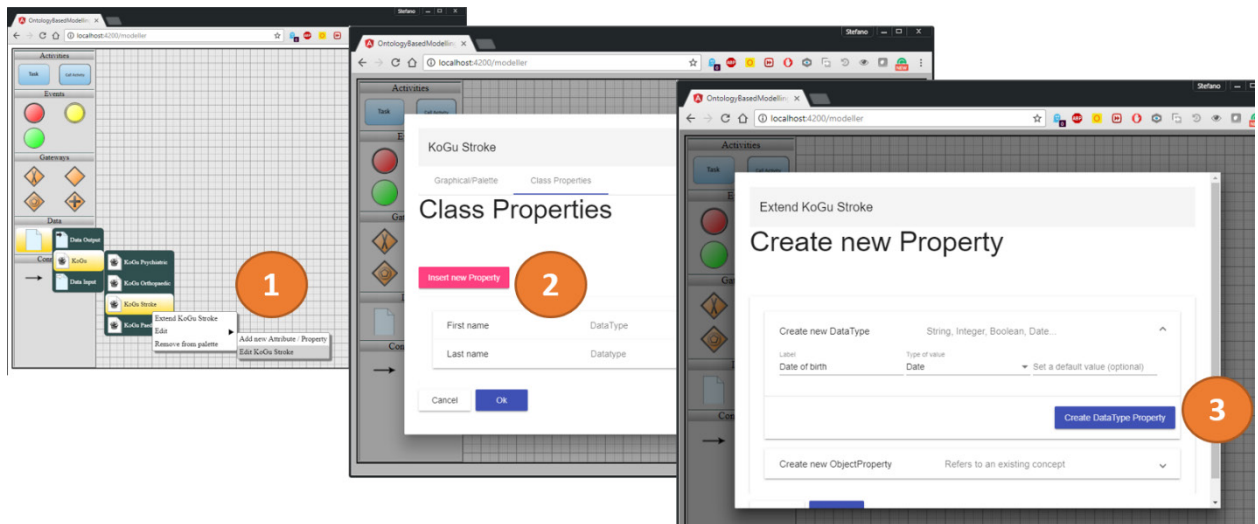


Figure 4. Operator 6 (and 3)

**Operator 7:** Update attribute (datatype properties). This operator is allows updating existing attributes.

**Operator 8:** Delete attribute (datatype properties). This operator is allows deleting existing attributes.

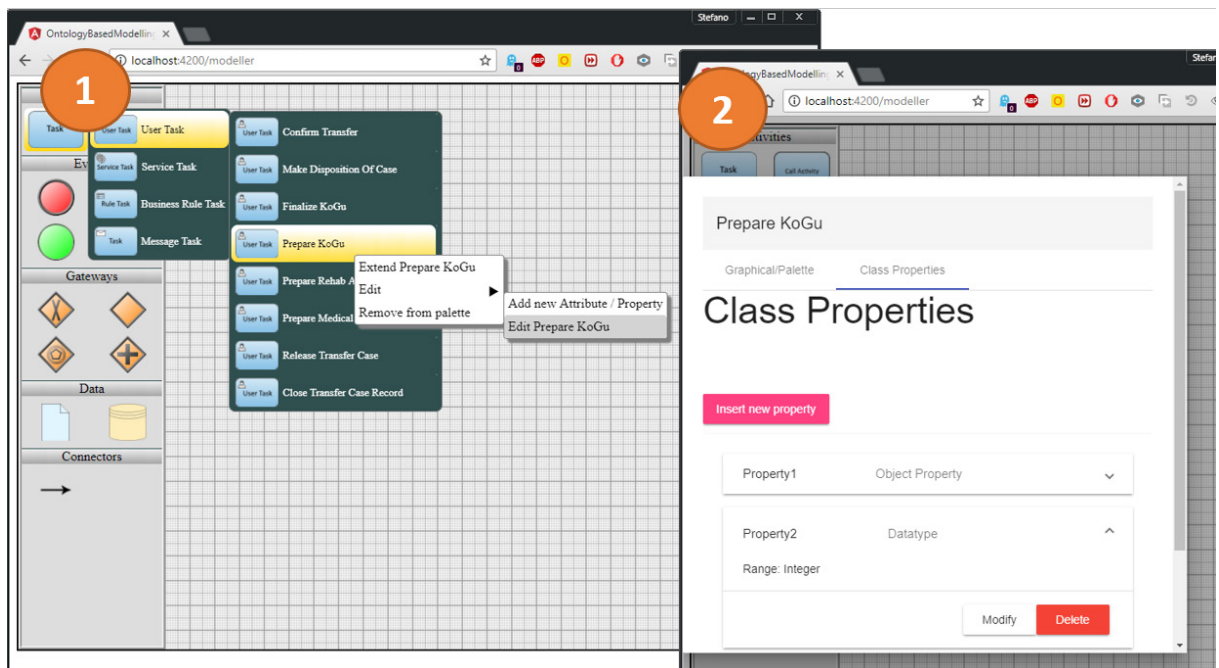


Figure 5. Operators 7 and 8

**Operator 9:** Assign attribute type and attribute value. This operator allows assigning value types String, Integer, Boolean to attributes as well as concrete values to attributes of modeling elements.

**Operator 10:** Update attribute types and attribute value. This operator allows updating types and/or values that are assigned to attributes of modeling elements.

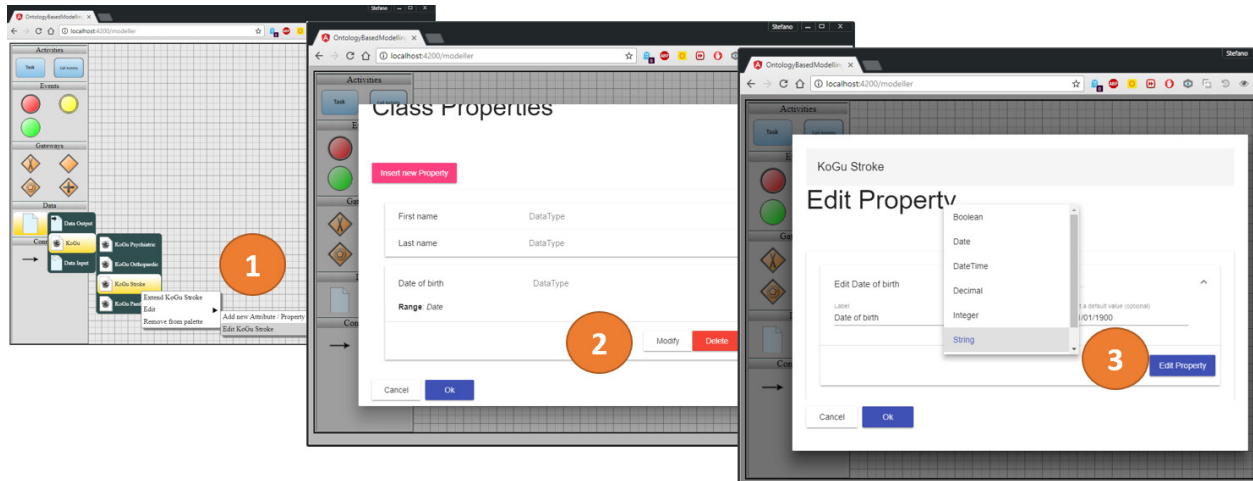


Figure 6. Operator 9 and 10

**Operator 11:** Enable representation level. This operator allows specifying whether a modeling element is a type or instance. Hence, the modeling environment can model instances as well as classes. Since a class can be an instance of another class, the modeling environment enables to distinguish between different levels of abstractions and not restricted to the class-instance dichotomy of description logics representation.

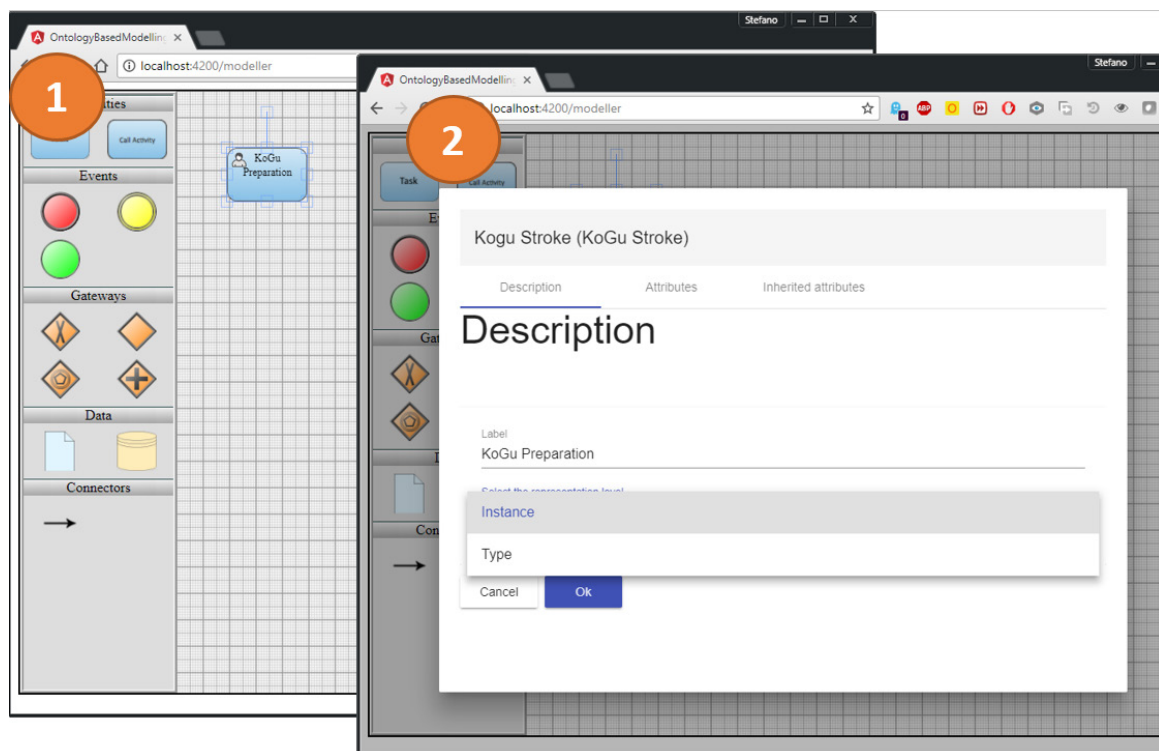


Figure 7. Operator 11





Figure 9 shows the overall new approach agile and ontology-aided approach.

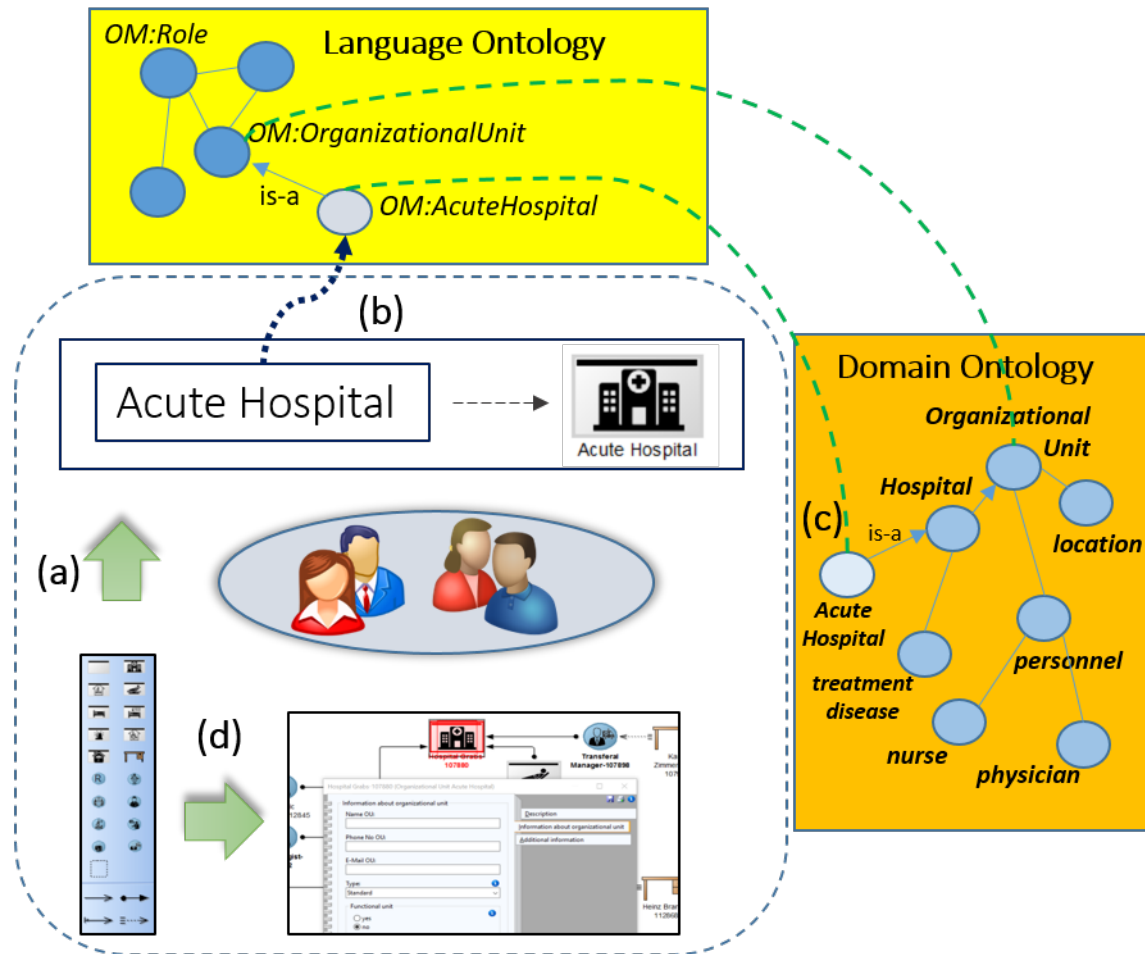


Figure 9. The Agile and Ontology-aided Modeling Environment