ITEA Office
High Tech Campus 69 - 3
5656 AG Eindhoven
The Netherlands

T + 31 88 003 6136
E info@itea3.org
W www.itea3.org

ITEA 3 is a EUREKA strategic ICT cluster programme

BUMBLE Deliverable D3.3 (Version 1)

BUMBLE Methodology

Edited by: BUMBLE Team
Date: November 2021

Project: BUMBLE - Blended Modeling for Enhanced Software and Systems Engineering

Contents

1. Introduction 3
2. Topology 4
3. Taxonomy 7
4. Conclusion and next steps 10

1. Introduction

The purpose of this deliverable is to provide a technology-independent overview of the BUMBLE approach to realize blended modeling. It also includes aspects of the collaboration functionality. The goal is to clarify how all required functionality fits together and to prevent problems because of misinterpreted terminology.

This document is a start for the deliverable for the task called ‘BUMBLE Methodology’. Any methodology addresses subjects that need to be defined first. That is why this first deliverable will focus on the functional parts and terminology. In later versions more will be written about the modeling process.

This document contains two major parts. The first part is topology. The topology defines the building blocks of the solutions and how they fit together. The topology is not ment as a technical architecture, but more as an overview of the functional parts of the solutions. The second part is the taxonomy. The taxonomy is a definition of the terminology by means of an ontology that depicts the relations between the different terms.

2. Topology

BUMBLE is about blended and collaborative modeling. We follow common terminology and approaches used in model-driven practices as far as possible. Since it is common to use modeling for different purposes we distinguish the common *meta-levels* of models:

- M1 are the actual models that model non-language aspects like printer cars, or income-tax laws. Here is where the users enjoy the blending and collaboration functionality.
- M2 are the models that define the languages of the M1 models and the way different languages blend together.
- M3 are the models that define the modeling-languages in which the M2 models are written.

Figure 1 shows the topology of an example of the functional parts that blend and synchronize M1 models. Figure 2 shows the topology of an example of the functional parts that define the languages and their transformations. It also shows how these M2 models relate to the M1 topology. It is shown that the functionality of M1 is derived from the M2 specifications, by means of generators.

Figure 1 depicts two modeling client environments that are involved in a collaboration session where their models are immediately synchronized with each other. Furthermore, the modeling languages (and possibly MDSE technologies) are different from each other. There is also blending of different concrete syntaxes happening in both client environments. To make this happen there are several transformations happening, both in the client- as in the server environments. The synchronization between the different environments is done by two model distribution services that exchange the mutations for the synchronized models. Mind that remote synchronization only takes place between models that conform to the same metamodel (are written in the same language).

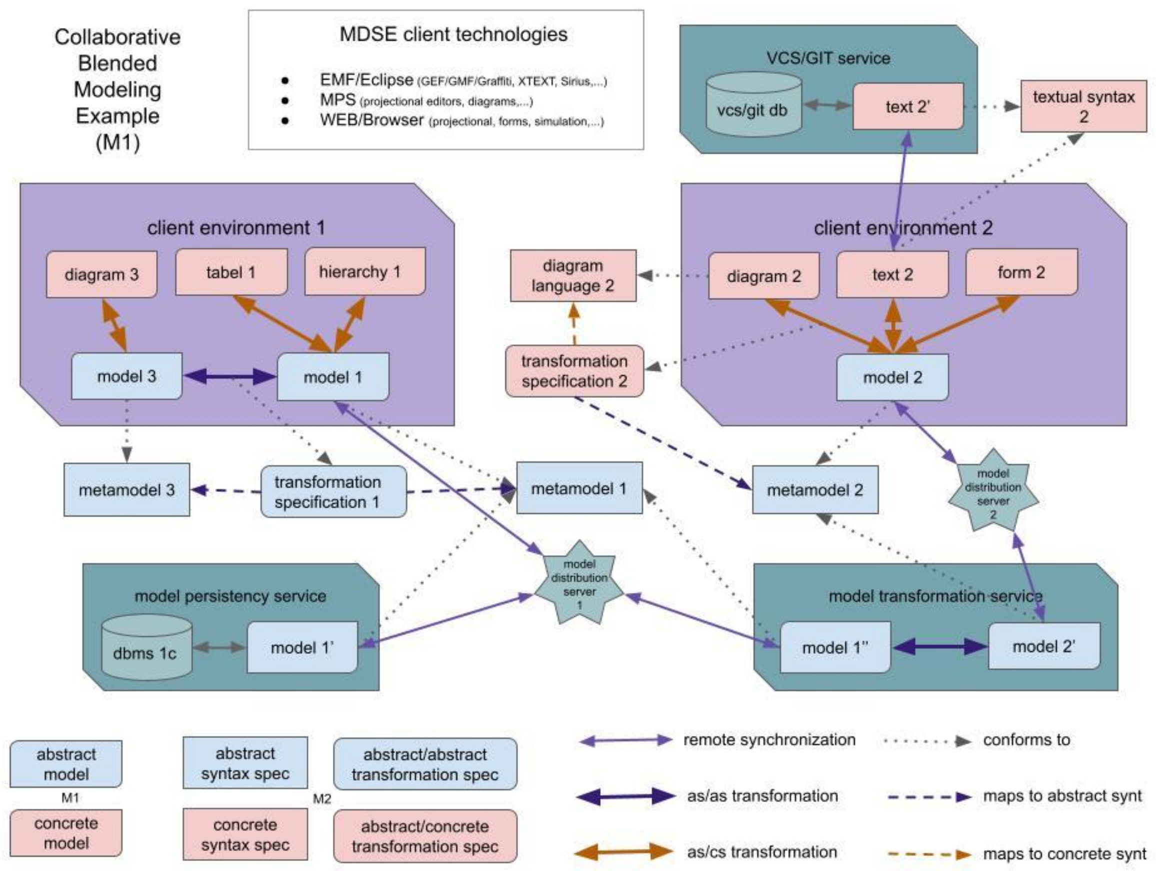

Figure 1. M1 topology

The goal of the BUMBLE project is to develop solutions for blending for different MDSE technologies, and that are usable for all possible modeling languages. MDSE technologies already have generic functionality to define modeling languages. We need to extend this functionality to define transformations between those languages. It very much depends on the MDSE-technology involved if and how this should be realized.

Figure 2 shows that the transformations of the different syntaxes (blending) is realized by an modeling-environment that is (partly) generated based on language and transformation specifications. These M2 level models are also modeled in a modeling environment (the yellow one in figure 2). A generator (the yellow arrow) generates parts of the M1 modeling environment (the purple part). How and which solutions need to be developed in the BUMBLE project for defining and executing the transformations is very much dependent on the involved MDSE technologies (e.g. PMS or EMF).

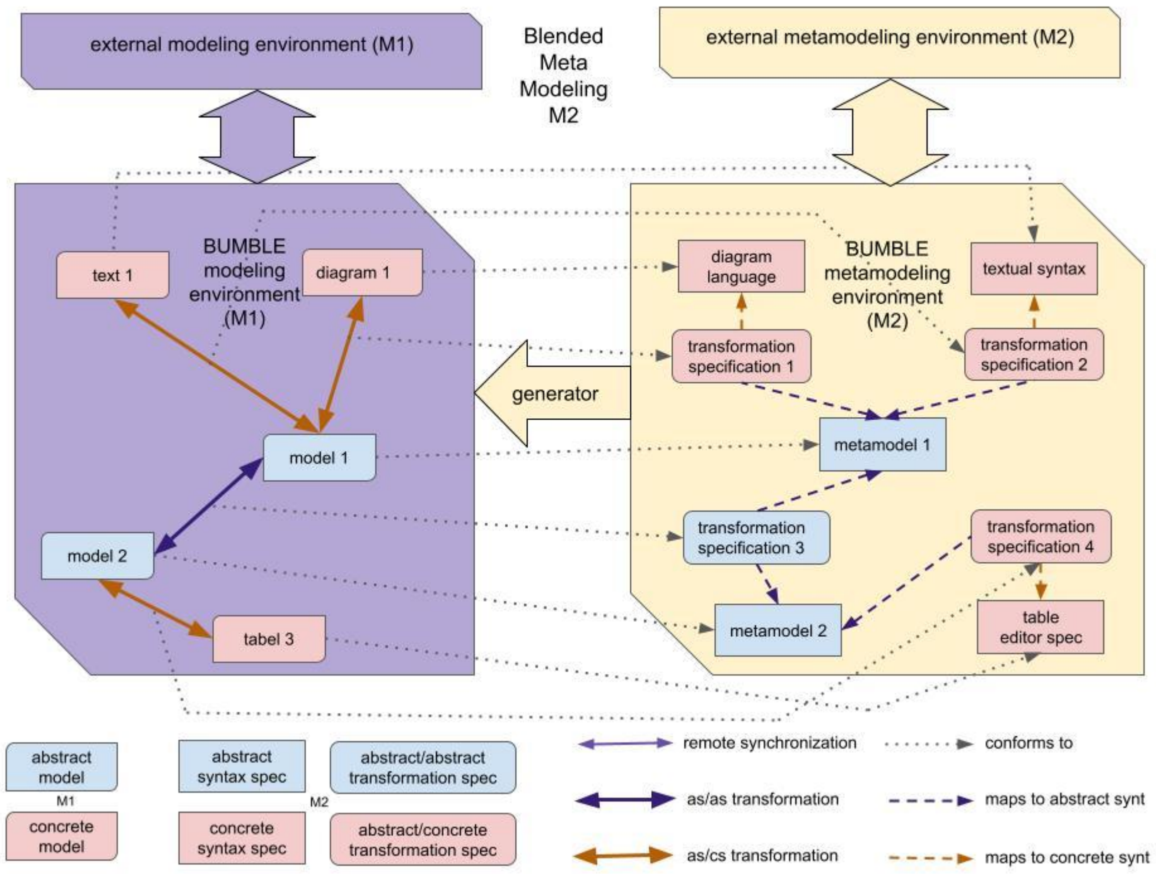
external modeling environment (M1)
Blended Meta Modeling M2
external metamodeling environment (M2)
BUMBLE modeling environment (M1)
text 1
diagram 1
model 1
model 2
tabel 3
generator
BUMBLE metamodeling environment (M2)
diagram language
textual syntax
transformation specification 1
transformation specification 2
metamodel 1
transformation specification 3
transformation specification 4
metamodel 2
table editor spec
abstract model
concrete model
M1
abstract syntax spec
concrete syntax spec
M2
abstract/abstract transformation spec
abstract/concrete transformation spec
remote synchronization
as/as transformation
as/cs transformation
conforms to
maps to abstract synt
maps to concrete synt

Figure 2. M2 and M1 relation topology

3. Taxonomy

Figure 3 shows the terms and the relationship between the terms.

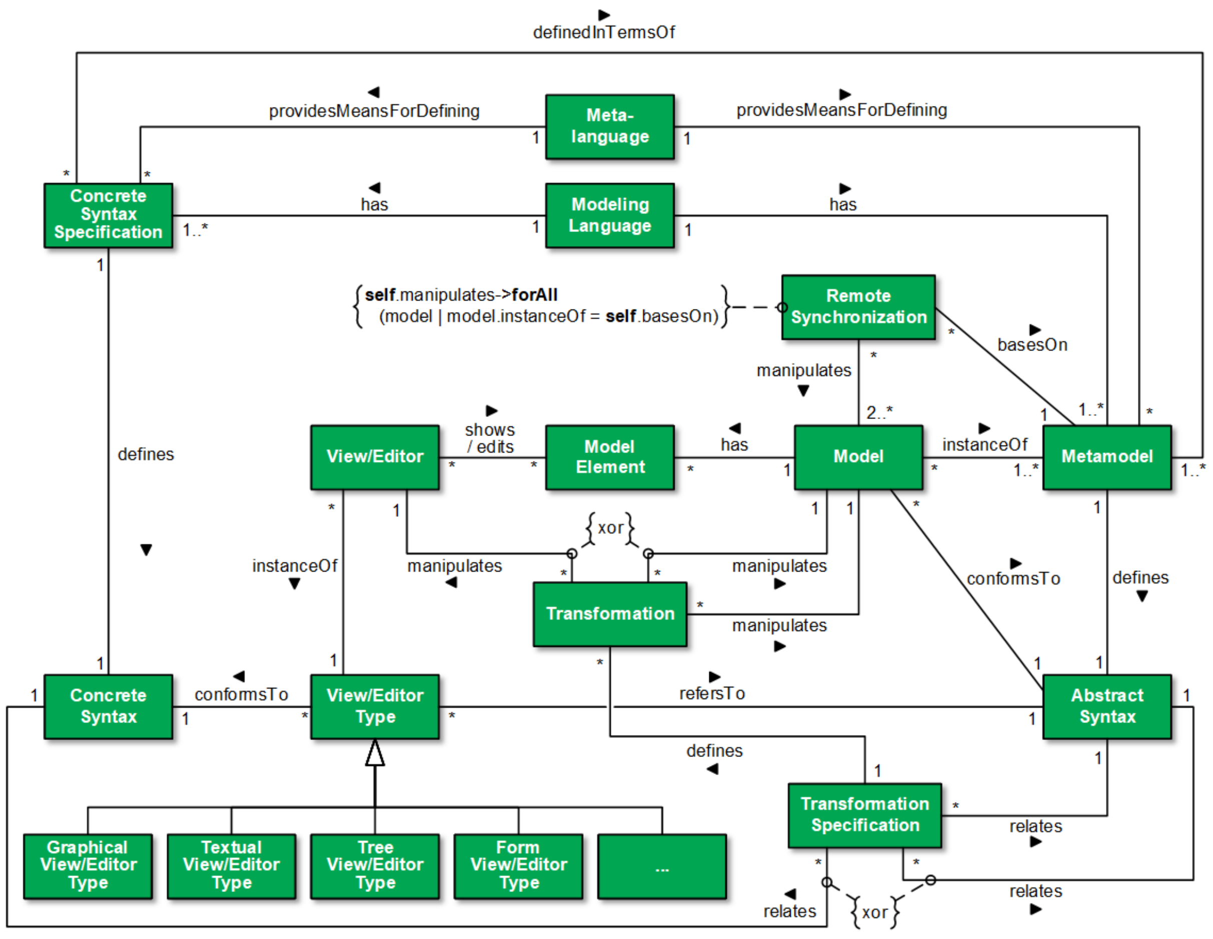

Figure 3. taxonomy

Model

“A model represents an aspect of a system under development captured in a specific instance of a [machine-processable] ***[modeling] language*** that serves a purpose within the development lifecycle.”[1]

[1] J. Holtmann, J.-P. Steghöfer, M. Rath, D. Schmelter: Cutting through the Jungle: Disambiguating Model-based Traceability Terminology. RE 2020: 8-19

Modeling Language
A language defines the ***syntax*** and static semantics of ***model***s. The syntax defines the concepts and rules to be used and conformed to, for any ***model*** to be a well-formed instance of that ***modeling language***.

Metamodel
A meta-model defines the ***abstract syntax*** of a ***modeling language***.

Abstract Syntax
The abstract syntax defines the concepts and rules by which structure models shall be written in a specific ***modeling language***. The abstract syntax is mainly useful for the static semantic aspects of the ***model***s. The abstract syntax is defined by and has to conform to an abstract syntax specification (i.e., a ***metamodel***), which is part of a ***modeling language***.

Concrete Syntax Specification
A concrete syntax specification defines a ***concrete syntax*** for a ***modeling language***.

Concrete Syntax
The concrete syntax defines how humans read and write ***model***s of a specific ***modeling language***. The concrete syntax is defined by and has to conform to a ***concrete syntax specification***, which is part of a ***modeling language***.

Metalanguage
A metalanguage is a language that provides means for defining an aspect (***metamodel***, ***concrete syntax specification***, static semantics, ...) of a ***modeling language***.

Remote Synchronization
A synchronization mechanism that keeps two or more models of the same language edited by different editors the same across a network. Changes in one model are propagated to equivalent changes in the other model.

Transformation
A transformation is a manipulation of a pair of one ***model*** and another ***model*** or a ***view*** that preserves the relation between them according to a ***Transformation Specification*** for the two syntaxes involved.

Transformation Specification
A transformation specification is a specification that defines the meaning-preserving relation between two ***syntaxes***. One of the transformed syntaxes must be an ***abstract syntax*** and the other syntax can either be an ***abstract syntax*** or a ***concrete syntax***.

View/Editor Type
"A view[/editor] type defines rules according to which ***view***s[***/editor***s] of the respective type are created"[2] based on the ***concrete syntax*** and thereby its ***concrete syntax specification*** of a ***modeling language***. "It defines the set of metaclasses whose instances a ***view***[/***editor***] can display [and can be edited]."[2] A view/editor type can be of graphical, textual, tree-based, form-based, … nature.

View/Editor
"A view[/editor] is the actual set of objects and their relations [(i.e., the elements of a ***model***)] displayed using a certain representation and layout [and providing the allowed editing commands]. A view[/editor] resembles the application of a view[/editor] type on the [...] ***model***s. A view[/editor] can therefore be considered an instance of a ***view***[/***editor***] ***type***."[2]

[2] T. Goldschmidt, S. Becker, E. Burger: Towards a Tool-Oriented Taxonomy of View-Based Modeling. Modellierung 2012.

4. Conclusion and next steps

We can conclude that it is possible to define the generic BUMBLE concepts in a technical-independent manner. We have also clarified that a complete BUMBLE solution can be created by combining relatively independent 'partial' solutions for different aspects.

This is the first version of the BUMBLE methodology deliverable. In the following versions we will add a more detailed explanation of the concepts depicted in the topology, and a description of the BUMBLE methodology process.