JOT
JOURNAL OF
OBJECT TECHNOLOGY

Journal of Object Technology | **RESEARCH ARTICLE**

A model-driven approach to align heterogeneous models of a complex system

Mahmoud El Hamlaoui[*], Sophie Ebersold[†], Saloua Bennani[*,†], Adil Anwar[§], Taoufiq Dkaki[‡], Mahmoud Nassar[*], and Bernard Coulette[†]

[*]IMS Team, ADMIR Laboratory, ENSIAS, Rabat IT Center, Mohammed V University in Rabat, Morocco
[†]SM@RT Team, IRIT Laboratory, University of Toulouse-Jean Jaurès, France
[‡]SIG Team, IRIT Laboratory, University of Toulouse-Jean Jaurès, France
[§]Siweb Laboratory, EMI, Mohammed V University in Rabat, Morocco

ABSTRACT To understand and manipulate a complex system, it is necessary to apply the separation of concerns and produce separate models, called viewpoints models. These models represent views on the system that correspond to distinct business domains. They are generally heterogeneous, i.e. conform to different meta-models. The management of the system's global model (a complete view of the system) requires the identification of the existing correspondences among the viewpoints models. However, in practice these correspondences are either incompletely identified or not sufficiently formalized to be maintained when models evolve. This restricts their use and does not allow their full exploitation for managing the global model. To fix this issue, we propose AHM (Alignment of Heterogeneous Models), an approach to organize the source models as a network of models through a virtual global model called M1C (Model of correspondences between models) that conforms to a Meta-Model of Correspondences (MMC). AHM proposes also a process, allowing for both the creation of the global model, and its consistency control. Partial automation of this process is done through a refining mechanism supported by a semantics expression described in a Domain Specific Language (DSL). The application of AHM is illustrated by the example of a conference management system. A prototype of a tool called Heterogeneous Matching and Consistency management Suite (HMCS) has been developed to support this approach.

KEYWORDS Alignment, heterogeneous models, refining mechanism, correspondences, semantic expression, multiparty graph

1. Introduction

The development of complex software systems is usually based on a varied set of languages, tools and environments that are generally used separately by different modeling practitioners (designers) working on different aspects of the system (Yang et al. 2004; France & Rumpe 2007; Mernik et al. 2005). In addition, these modeling experts can be located in distant geographical areas, as it is the case in distributed collaborative development in big software companies.

JOT reference format:
Mahmoud El Hamlaoui, Sophie Ebersold, Saloua Bennani, Adil Anwar, Taoufiq Dkaki, Mahmoud Nassar, and Bernard Coulette. *A model-driven approach to align heterogeneous models of a complex system*. Journal of Object Technology. Vol. 20, No. 2, 2021. Licensed under Attribution 4.0 International (CC BY 4.0) http://dx.doi.org/10.5381/jot.2021.20.2.a2

Several approaches have been developed to face complex systems modeling. One of the most efficient and widely used in industry consists in elaborating separate models that correspond to different points of view (Hilliard 2001), (Koning & van Vliet 2006), (Boulanger et al. 2010). This general approach - called multi-modeling (Boronat et al. 2009) - is widely used in software engineering and allows designers to focus on different parts of the system in isolation.

Among problems that typically arise in this multi-modeling situation, we can mention the fact that different terminologies and terms are used to represent the same concept or that the same term can be used to express different concepts. This issue, typically known as heterogeneity problem is a common problem widely shared by the community of complex software systems (Yousafzai et al. 2016), (Mosterman & Zander 2016).

GEMOC provides a classification of the different levels of heterogeneity in software engineering in (Baudry et al. 2012). Authors propose a three-level classification: heterogeneity of systems (such as subsystems orchestration), heterogeneity of execution platforms (this is the case for example with simulation engines), and finally heterogeneity in modeling which deals with coordination of domain specific models. The latter is at the heart of the work handled by GEMOC. The first solution that comes in mind is to compose those different points of view into one single representation.

Globally, composition approaches proposed in the literature (Drey et al. 2009), (Kolovos et al. 2006a), (Zito et al. 2006) rely on the elaboration of one global model and have three major drawbacks related to models heterogeneity. The first one concerns the structure of the meta-model associated to the composed model; indeed, there is no consensus on how it should be constructed: from the union of all elements coming from source models or from their intersection, or from another combination? The second issue concerns the semantics used to represent an element of a composed model given that the source models may use different semantics. Finally, a global model obtained by composition may be huge in case of complex system and therefore really difficult to maintain.

In this context, megamodels have emerged as one means to cope with the problem of managing a multitude of development models and relations. In MDE, a megamodel is considered as a model that contains models and relations between these models or between elements of these models (Vogel et al. 2010). We have been working for years on the composition issue by providing a view-based methodology and a UML profile called VUML (Nassar 2003). This profile proposes the concept of multiview class to represent different points of view on a given concept. VUML has been used to represent the output of exogenous transformations taking UML models as input (in particular class diagrams (Anwar et al. 2010), or state-charts (Ober et al. 2008)). However, when facing complex systems, this approach turns out to be too restrictive and not scalable since views must be transformed into UML before applying the profile. The complexity of this model-to-model transformations depends on the degree of heterogeneity of the points of view. Furthermore, the transformations must be re-executed even for the slightest change on source models.

We are considering points of view in which heterogeneous models are described in different domain specific languages by designers with different skill areas. So we are addressing alternative modeling languages than UML. We propose an approach called AHM (Alignment of Heterogeneous Models), supported by a methodology ((El Hamlaoui et al. 2013)) and a tool ((El Hamlaoui, Ebersold, Anwar, et al. 2014)) that aims at (i) aligning heterogeneous models (in a megamodel), (ii) managing consistency when models evolve at design time, or after ((El Hamlaoui, Bennani, Nassar, et al. 2018), (El Hamlaoui, Ebersold, Coulette, et al. 2014)). Instead of building a single global model, our proposition consists in organizing the different source models as a network of models that provides a global view on the system through correspondences. This will improve the global consistency of the system and ease its consistency management when evolutions occur.

This paper is a synthesis of AHM. We will first present the key elements of this approach. After that we will tackle the following points: the matching process, the meta-model of correspondences, the construction of models of correspondences (which can be considered as mega-models according to (Bézivin et al. 2004), (Salay et al. 2016) since they contain references to model elements and links between them), the refinement of correspondences from the meta-model level to the model level, the consistency management process, and the tool support. We will also detail the formalization of AHM by providing a DSL, **S**emantics **E**xpression **DSL**, dedicated to the expression of the correspondences' semantics. This semantics will be useful for the construction of the correspondence model but also for the checking and management of the system consistency. AHM has been successfully applied to different case studies and application domains (El Hamlaoui, Bennani, Ebersold, et al. 2018). It will be illustrated all along this paper by the example of a Conference Management System.

The rest of this paper is organized as follows. Section 2 introduces a conference management system case study that was chosen to illustrate AHM through out this paper. Section 3 presents the matching approach and its first steps. It highlights how the core of the approach, the meta-model of correspondences, is enriched by semantic annotations. Section 4 shows how models are matched through a model of correspondences. It deals with correspondences at model level and how they can be established from correspondences established at meta-model level, through the refinement mechanism. The tool support and the application of the approach are described in Section 5. Section 6 presents the evaluation of the tool. Section 7 investigates the related works and Section 8 presents a discussion on the weak points of AHM. Conclusion and some perspectives are presented in section 9. Finally, Appendix A presents the implementation of different methods used to express the correspondences' semantics in the CMS system.

2. Running example to illustrate AHM: Conference Management System

Production and reviewing of documents are widely used in various contexts such as project management, software development, conference management, etc. We place ourselves in the context of this last application domain to illustrate our approach.

A Conference Management System is a system designed to automate the main functions needed to manage a scientific conference, namely: call for papers, paper submission, paper assignment for evaluation, notification of the final decision, registration, etc. Even if it is not a large complex system, CMS has been chosen because it is firstly well known to researchers; and secondly and most importantly, because it involves different designers which are working with different points of view. We consider that there are three business domains in the CMS (Figure 1), covering various aspects of modeling. Each business domain is described by a dedicated model that conforms to a given meta-model.

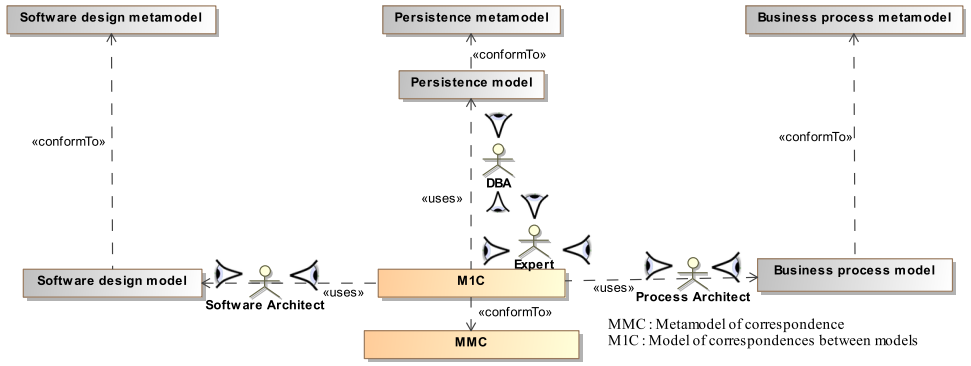

Figure 1 Overview of CMS models

Viewpoints models are manipulated by designers with specific roles:

- Software Architects: responsible for the conference management design, they create a model – called Software Design model – expressed through a specific software design meta-model,
- Database Administrators: responsible for storing data, they create a model – called Persistence model – expressed through a persistence meta-model,
- Process Engineers: responsible for describing the management process of CMS, they create a model – called Business Process model – expressed through a business process meta-model.

Besides these designers, an expert on the three viewpoints has to be involved to conduct the AHM approach. In the next sections, we present as an example the three models and an extract of their respective meta-models.

2.1. Software Design viewpoint model

In Figure 2, we show a software design meta-model that contains entities and stereotyped entities. Entities are connected either to an association or to a generalization. They may also have attributes and operations.
Figure 3 shows an overview of the software design model. It consists of four types of users (organizer, attendant, author and reviewer) and their personal information. The conference to manage is described through the entity *Conference*. It contains general information about the conference; such as *research area*, *submission deadline*, *notification date*, *conference venue*, etc. An entity called *paper* is also presented in this model. A *paper*, which can be written by several authors and evaluated by several reviewers, is characterized by several data including: *title*, *abstract*, *keywords*, *body*, an assigned identifier once the article is submitted, etc.

2.2. Persistence viewpoint model

The meta-model of Figure 4 represents the core elements of a relational database and the relationships among them. A schema includes tables and views of the system. Both of them

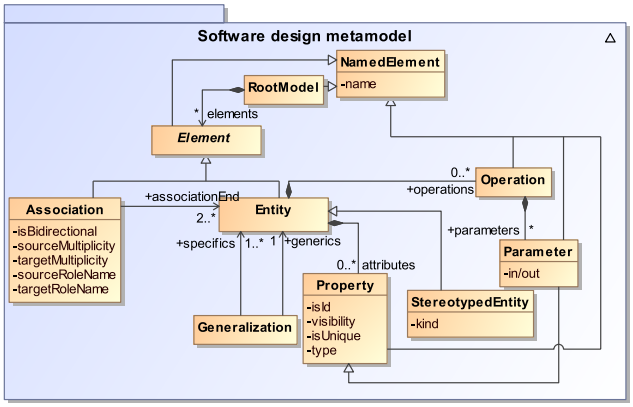

Figure 2 CMS Software Design meta-model

contain columns that can be specified as primary or foreign keys. Figure 5 shows the persistence model of the CMS. The Table *AuthorTable* stores information about every author such as its identifier, its affiliation, etc. *ArticleReviewsTable* contains data concerning the evaluation of an article. For example, the columns *review* and *decision* are used respectively for remarks and acceptance/rejection notifications. In the same table, a primary key column type is used to uniquely enumerate the various evaluations. Foreign keys columns type, *reviewerId* and *articleId*, are used to identify respectively the reviewer and the article.

2.3. Business Process viewpoint model

CMS can also be modeled in the business process point of view. Various actors must follow a well established process, to ensure the good management of a conference. The process engineer uses a business process model conforming to a simplified BPMN (OMG 2013) meta-model. This latter presented in Figure 6 comprises the following concepts: *lane*, *pool*, *task*, etc. Figure 7 presents an instance of the business process meta-model of Figure 6.

Currently, the CMS process involves one designer, namely

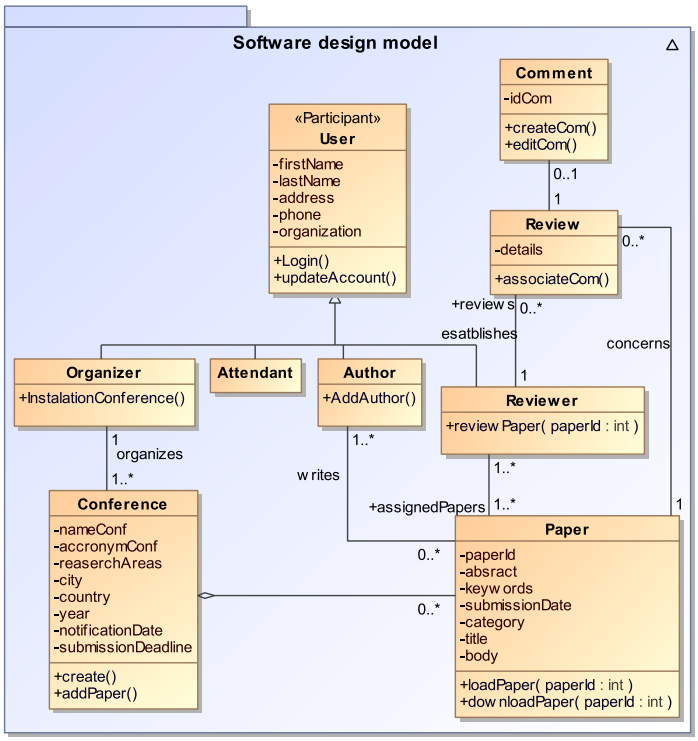

Figure 3 CMS Software Design model

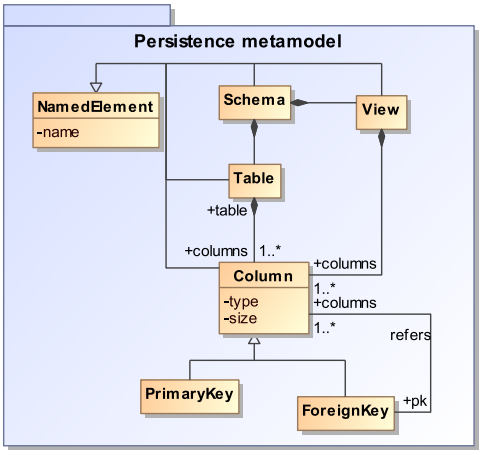

Figure 4 CMS Persistence meta-model

the reviewer (others will be added later). When the deadline of article submissions is reached, reviewers indicate, via the CMS, the papers they wish to review. Afterwards, they declare potential conflicts of interest. The described process model (Figure 7) put the emphasis on the evaluation procedure. Once the review date is reached, reviewers first choose papers among those they previously selected. Then, they may either assess the paper or subcontract it to a third person The result of the review is subsequently entered with any comments.

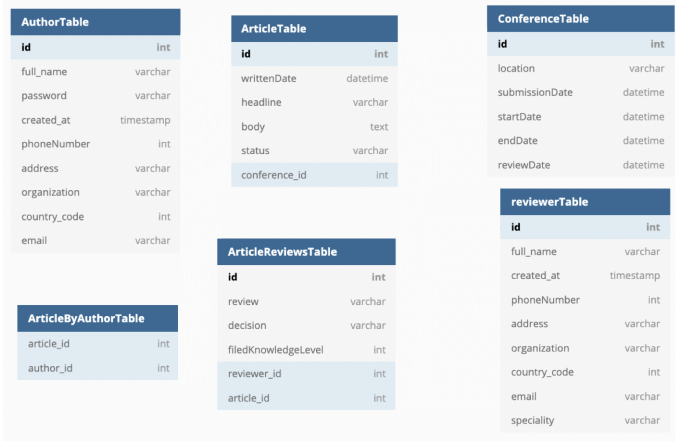

Figure 5 CMS Persistence model

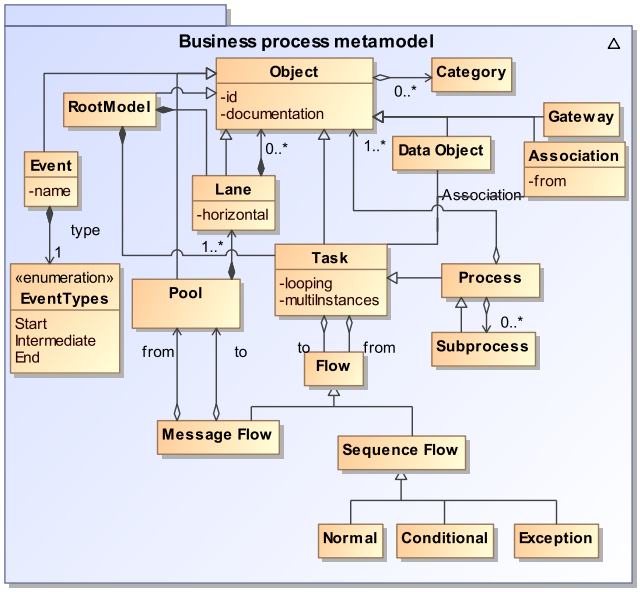

Figure 6 CMS Process meta-model

3. Matching approach description

The AHM approach consists in analyzing source models (and their respective meta-models) to identify correspondences that exist among them. We only consider external correspondences among the different viewpoints models. Internal correspondences (links between elements of the same model) are not in the scope of our research study.
Identified correspondences are then stored into a model of correspondences (M1C) that conforms to a meta-model of correspondences (MMC). We discuss below the elaboration of M1C as well as the proposed iterative matching process.

3.1. Meta-Model of Correspondences

MMC is presented in Figure 8. The *CorrespondenceModel* is composed of a set of correspondences. A correspondence can

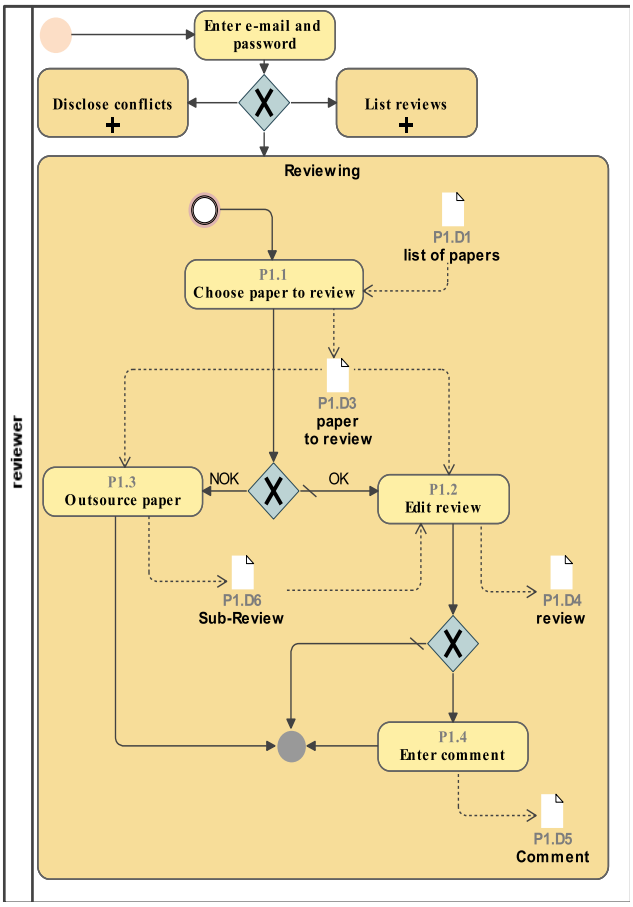

Figure 7 CMS Process model

either link elements of meta-models - in such case we speak about High level Correspondences (HLCs) - or elements of models - in this case, they are Low Level Correspondences (LLCs). The *Correspondence* meta-class has at least two ends (*RefElement*), that are elements from different (meta-)models (*RefModel*), and a *Relationship*.

Relationship is an abstract generalization of *DIR* (Domain Independent Relationship) and *DSR* (Domain Specific Relationship) meta-classes. The specialization of the first one allows for representing generic relationships that may exist in different domains. However, these relationships may be insufficient for a given domain. In this case, it is possible to represent specific relationships by a specialization of the DSR meta-class. The abstract meta-class *Level* is associated to a relationship to describe whether it represents a *High Level Relationship* (HLR) or a *Low Level Relationship* (LLR) or both. *HLR* are relationships that are used in HLCs, correspondences at meta-model level, whereas *LLR* are relationships that are used in LLCs, correspondences at model level. Any relationship used in an *HLC* will also be used in an *LLC* as specified by the OCL rule number 2 (on Figure 8), the opposite case is not always correct.

The *bidirectional* attribute of a relationship specifies whether this latter is bidirectional or not. If the relationship is bidirectional, the concerned (meta-)elements are all source ones and there is no target (meta-)element, as specified by the OCL rule number 1.

3.2. Matching process

The model of correspondences cannot be constructed in a monolithic manner. It follows a process, that we call *matching process*. Figure 9 shows an iteration of the proposed process. It aims at describing the steps required to perform the matching of heterogeneous source models, to obtain a model of correspondences. The process involves one actor (the integrator expert) and a support tool for assisting the expert and performing the automated phases.

The process takes as input the various models and their respective meta-models and also the meta-model of correspondences. Firstly, a check is performed (Step 1) to inspect and ensure that the MMC contains all needed relationships to set up correspondences for the given application domain. If the integrator expert considers that the proposed relationships are not sufficient, the *DSR* meta-class of MMC is specialized (Step 2). In the case of CMS, we added the following three types of relationships: *Require*, *Contribution* and *Play*. The first one gives the required input of a task. The second one identifies the operations that contribute to the success of a task. The third one highlights the role, for example, of the participants of a conference in the business process.

Activities 1 and 2 are mainly manual, depending on the experts' knowledge of the domain. In the second activity he or she defines missing DSRs and implements them through the tool support. We consider that this two phases are the starting point of AHM process. In the next sections, we will focus on the following steps.

Once all relationships are well identified, they are defined throughout the Step 3. The third activity of this process aims at enriching the MMC with a Semantic Expression (SE) for each relationship. This SE describes the semantics of the relationship and makes it possible - if implemented - notably to check whether the relationship fits among the elements it connects. The implementation of this activity will be detailed in section 3.3.

Once all the work on the MMC is done, the core of the matching operation can be launched. It begins, in the fourth activity (Step 4), by identifying correspondences between meta-elements so as to produce a model of correspondences called M2C. M2C (Model of correspondences between meta-models) stores High Level Correspondences (HLC) that contain meta-elements linked by High Level Relationships (HLR) (section 3.4).

HLCs are then refined (Step 5) in order to produce Low Level Correspondences (LLC) that are stored in M1C (Model of correspondences between models). LLCs link model elements through Low Level Relationships (LLR). HLCs are established only once and LLCs, which can be numerous and are to be defined for each new system, are produced semi-automatically by the developed support tool. This will be detailed in section 4. Note that if an M2C already exist for the studied system, it will be used at the beginning of the matching process as a reference model of correspondences.

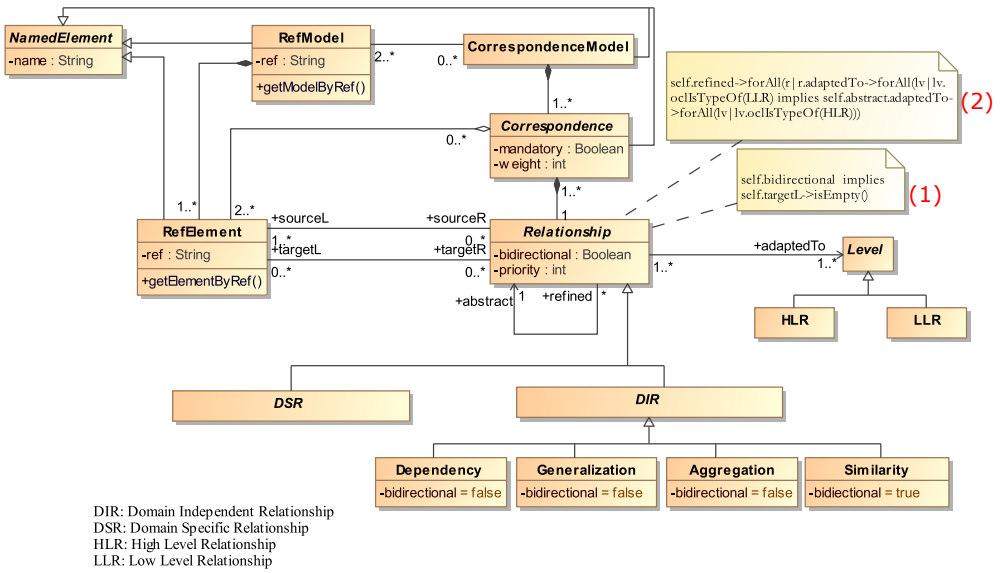

Figure 8 Overview of the kernel part of the meta-model of correspondences (MMC)

3.3. Semantic Expression DSL (SED) - Step 3

In the third activity of the process (Figure 9 (3)), the integrator expert has to specify the semantics of each relationship. This specification consists in enriching the MMC with expressions, presented as UML annotations, to assist and facilitate the matching. For this purpose, a model conforming to a proposed semantic expression DSL (Figure 10) must be created. It starts with a run-through the MMC to retrieve automatically the list of relationships (sub-classes of DIR and DSR). Thereafter, for each relationship a semantic expression is defined. This latter is composed of the source element(s) type and/or the target element(s) type. Giving an example, the *similarity* relationship, as it is not a directed one (*bidirectional* = *true*), will have source elements only. The condition allows for defining an action that contains the body to be executed in a language such as OCL (OMG 2014), Java (Gosling 2000), Alloy (Jackson 2002), B (Lano 1996), etc. The *format* attribute specifies whether the body of the condition is based on model elements (*MBF: Model Based Format* - see section4.2.2) or on ontology's ones (*OBF: Ontology Based Format*.

In this paper we do not address OBF since exploiting the body of a condition whose format is OBF, implies to switch from the technological space of models to the technological space of ontologies. Transformation from Ontology field to Model field (that we have done for some relationships like similarity), is still a work in progress.

The condition also defines two operations for retrieving the source and the target elements of a correspondence. If the semantic expression cannot be described in a formal language, the expert has the ability to use a structured natural language.

Figure 11 shows the semantic expression model for CMS. For example, the *Similarity* relationship (top left of the figure) is used in a correspondence involving two elements (recovered by the *importSourceElts()* [1] operation) with indifferent types (Any, Any). It is described by an expression in Java. The condition explicitly states, using the *sameAs* function, that the connected elements are similar. The *Dependency* relationship is expressed in natural language since we cannot describe the action with a language. *Require* is an example of DSR that it is used to link an element of *Task* type with an undefined type (Any). The condition, written in Java, is based on a function called *contain*. Details of the implementation of the methods *sameAs* and *contain* are given in Appendix A.

The model of semantic expression (SE model), is then woven with MMC using aspects (Filman et al. 2004). Here, it consists in grafting the different expressions on MMC as annotations as shown on Figure 12. We can notice that three sub-meta-classes of DSR have been added to MMC to express CMS specificities.

The advantage of using the SED DSL is primarily to have a structured common definition of each relationship. Secondly, it helps build the M2C in an assisted manner using information on the connected elements types (*sourceEltType*, *targetEltType*). Thirdly, by exploiting the actions, it helps filter out the correspondences in the refinement step to keep only those correspondences that match the semantics of the relationship.

3.4. Defining correspondences at meta-model level - Step 4

According to AHM process (Figure 9), in this fourth step correspondences at the meta-model level (HLC: High Level Correspondence) are created. As explained in section 3.2 above, HLCs are defined only once during the modeling cycle. The identification of a correspondence begins with the selection of

[1] importSourceElts () gets the value of the property *name* of source elements.

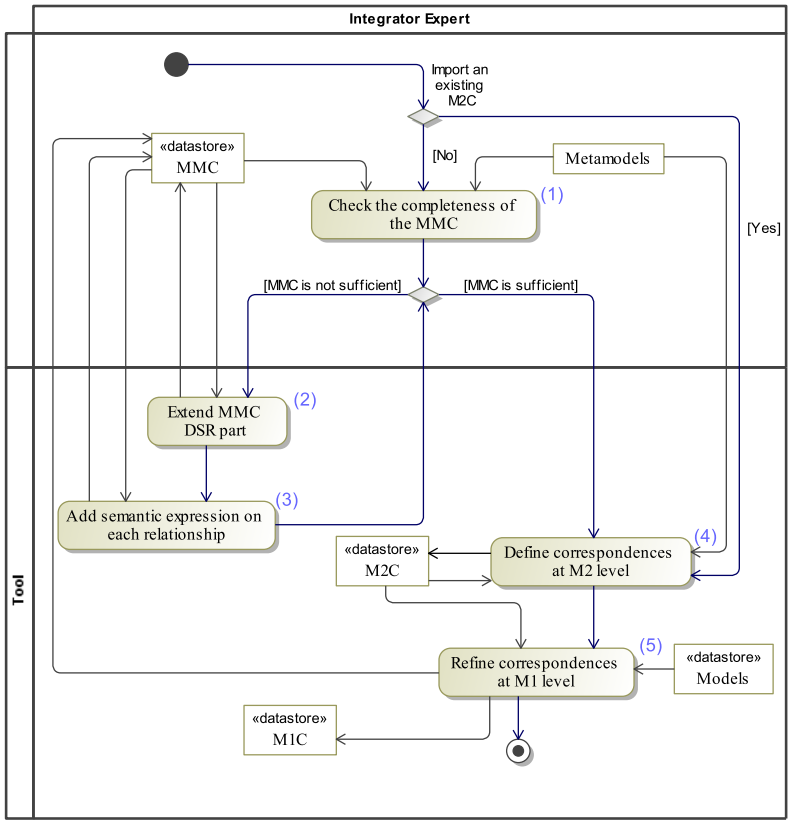

Figure 9 Activity diagram of the matching process

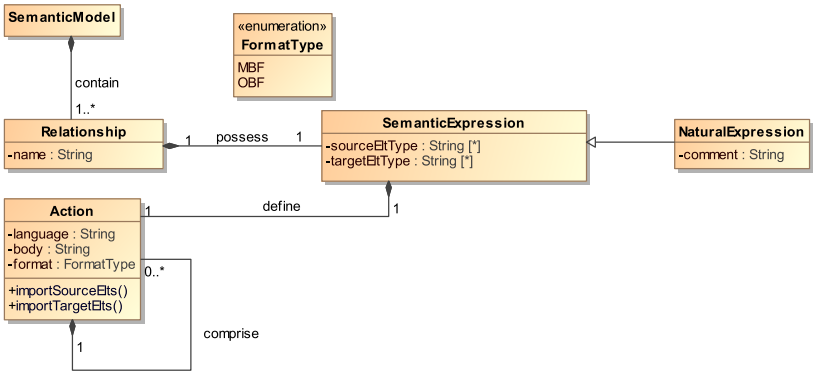

Figure 10 Semantic Expression DSL (SED)

the relationship and continues with its assisted creation, based on the annotations previously added to the MMC. More precisely, through information on the element types, previously filled in the annotations, it is possible to apply a filter on all meta-elements by showing only the candidate meta-elements to be linked. This task is supervised by the expert integrator.

For example, after the expert has chosen to add the *Contribution* relationship, the meta-elements to link are automatically selected, based on the annotations. In this case, the *Operation* meta-element on one side and the *Task* meta-element on the other side are linked.

Figure 13 shows examples of HLCs for CMS. For instance, a

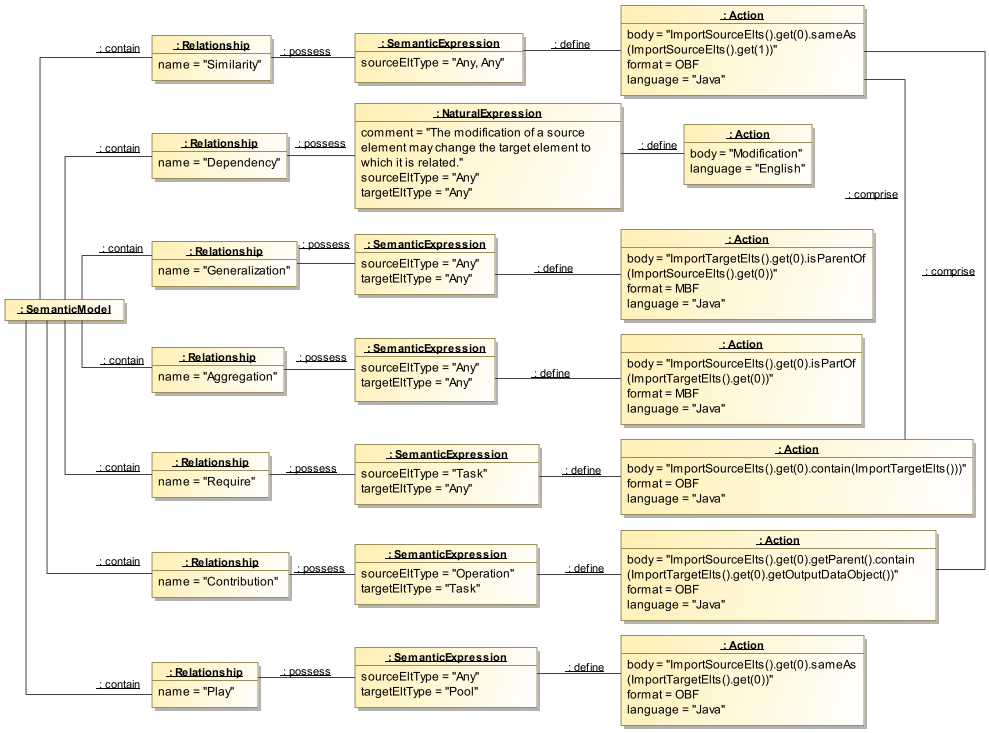

Figure 11 Semantic Expression model (SE model) for CMS

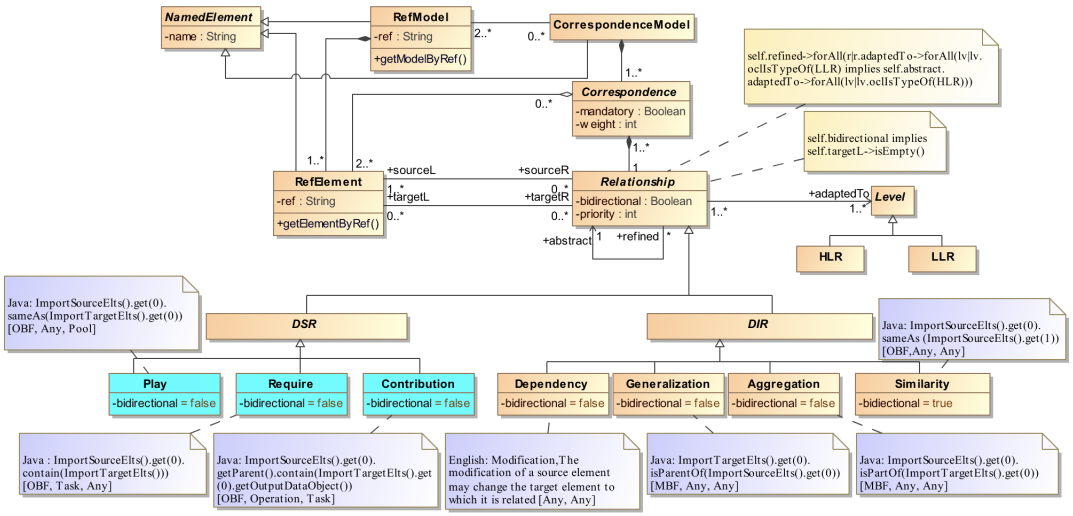

Figure 12 Annotated MMC resulting from a weaving with SE model

correspondence encompasses a link between the meta-element *Task* on one side and the meta-element *Operation* on the other side through a *Contribution* relationship, since an operation can potentially contribute to the achievement of a task. A *Similarity* is established between the meta-elements *Table* and *Entity* and also between the meta-elements *Property* and *Column* on one side and *Column* and *DataObject* on the other side.

4. Refining High Level Correspondences - Step 5

HLCs make it possible to anticipate the complexity of matching by first establishing correspondences between meta-elements. Subsequently, the accuracy of certain meta-model-level details can be managed at model level. Correspondences between model elements, LLCs, are obtained by refinement of HLCs.

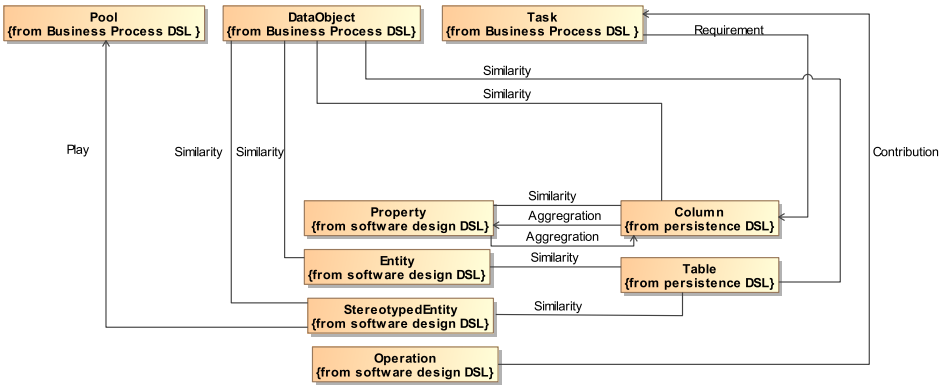

Figure 13 Example of CMS HLCs

4.1. Refining

In software and system engineering, refining is a classical way to reuse artifacts (Wolf 1994), (Wooldridge 1997), (Medvidovic & Taylor 2000). It can be seen as a way of crossing different levels of abstraction with the purpose of adding details when passing from a given level to a more concrete one. According to (Wagelaar 2005), even though refining is a key concept in MDA, it is loosely defined, and open to misinterpretation. The refining notion has also been defined in UML (OMG 2011) as a stereotype for "Abstraction" - a directed relation from an element to another one stating that the first (concrete) depends on the second (abstract).

In this approach, we define a *Refining* relationship as an endogenous transformation of a model to another one. The input model is M2C (Model of correspondences between meta-models, containing HLCs), while the output one is M1C (Model of correspondences between models, containing LLCs). So, HLCs of M2C are projected from meta-model level onto model level to generate the LLCs of M1C.

Refining is thus represented by a relationship, denoted Rf, defined as follows:

Ci *Rf* Cj, (Ci and Cj being two correspondences respectively at model and meta-model level), if and only if Ci connects elements that are instances of meta-elements connected by Cj.

That means that Ci is more accurate than Cj (as it relates instances of the meta-elements being used in Cj) with the possibility of upgrading Ci by adding details to precise and restrict the correspondence.

Figure 14 summarizes the basic concepts of AHM: M2C contains HLCs which are correspondences between meta-elements belonging to different meta-models. By refining this latter, M1C is built respecting its conformance to MMC and contains LLCs, that are the correspondences connecting elements belonging to different models.

We propose two types of refining: **refining by propagation** and **refining by extension**. The first one consists in linking the model elements with the respective relationships defined between their meta-elements. The second one consists in specifying more precisely the relationship at the model level, by adding some constraints for example.

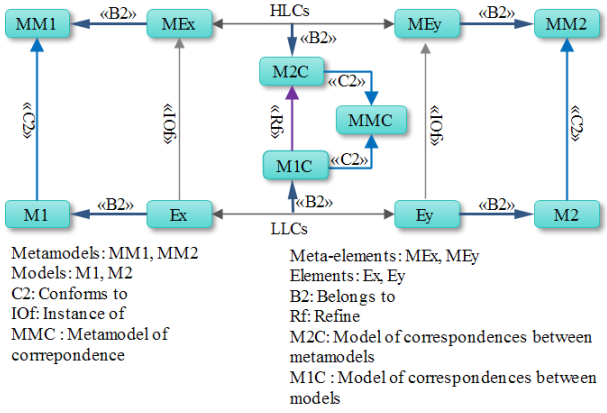

Figure 14 Concepts used in refinement transformation

Both refining types are discussed in the following sections.

4.2. Refining by propagation (From HLC to LLC)

Refining proceeds in two steps to produce the M1C. Primarily, a reproduction operation is initiated on M2C, then a selection operation is performed. In other words, the propagation is a composition of two functions:

$$M1C = = Propagation\ (M2C)$$
$$= Selection \circ Reproduction\ (M2C)$$

4.2.1. Reproduction The Reproduction is a homomorphism - a structural preservation from one algebraic structure to another one - between correspondences in M2C and M1C. Its role is to duplicate all correspondences defined at the meta-model level into the model one. Therefore, there are as many potential LLCs for a given HLC as the Cartesian product of instances of meta-elements involved in this HLC.

According to Figure 14, we are in front of a “problem” where we deal with four categories of models: M1, MM1, M2, and MM2. These models entertain only inter-categories relationships: “InstanceOf” between elements from M1 and elements from MM1 and between elements from M2 and elements from

MM2 and “correspondence” between elements of MM1 and those of MM2. Thus, we are in front of a problem that can, in a natural way, be modelled in the form of a graph. The existence of models of different natures and the relationships that exist exclusively between heterogeneous elements from different categories pleads for the use of multipartite graphs (Jenkinson et al. 2012).

We aim at in finding links between the elements of M1 and those of M2 (i.e. correspondences at the model level). These looked-for correspondences are "the missing and valid" links between the elements which are already bound in an indirect way via M1-MM1, MM1-MM2 and M2-MM2. Thus, they are links between elements for which there is already at least a path of length 3 in the multipartite graph.

So, we formalize the Reproduction operation by using a multipartite graph. Graphs are mathematical structures that provide natural means for complex-data representation. They capture the structure and thus help to model a wide range of complex real-life data in various domains. For this, we firstly define a graph as a set of Nodes N and a set of Edges E, (N, E). Secondly, to embody the above definition, the set of nodes N and the set of edges E, are obtained as follows:

- Given MMi a meta-model, Mi a model, NMMi and NMi the respectively sets of elements of MMi and Mi:
  - "isInstanceOf" is a directed relationship that links a source element of a model to the target element of a meta-model,
    isInstanceOf : NMi $\rightarrow$ NMMi, e $\mapsto$ em
  - "instantiate", is a directed relationship that links the source element of a meta-model to the target element of a model,
    instantiate : NMMi $\rightarrow$ NMi, em $\mapsto$ e
- Given
  - SN = NMM1 U NMM2 U NM1 U NM2, where NMM1, NMM2, NM1, NM2 are respectively sets of elements of MM1, MM2, M1 and M2,
  - (n1,n2) $\in SN \times SN$,
  - $\mathcal{R}$ the set of types of relationships.
- E is defined by the set of couples (n1, n2) such as: (n1, n2) $\in$ E iff
  - (n1,n2) $\in NMM1 \times NMM2$ and n1 R n2, with $R \in \mathcal{R}$,
  - Or (n1,n2) $\in NM1 \times NMM1$ and n1 "isInstanceOf" n2,
  - Or (n1,n2) $\in NMM2 \times NM2$ and n2 "instantiate" n1.

Since the set of nodes is partitioned into four subsets (MM1, MM2, M1, M2) and as there are no edges between elements of the same subset, this definition confirms that it is a multipartite graph.

Figure 15 shows some HLRs (*play*, *contribution*) established among the meta-elements StereotypedEntity, Pool, Operation, and Task. To simplify the detection of these paths and to make

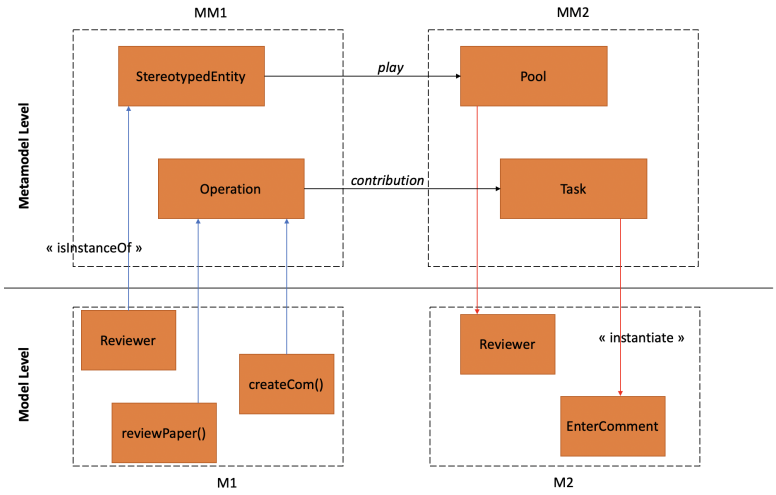

Figure 15 Simple abstract example of HLCs

sure that the only possible paths of length 3 are the ones we look for between elements of M1 and those of M2, we direct the graph ending with a directed multipartite graph.

Reproduction consists firstly in producing automatically the adjacency matrix for each relationship. It includes the meta-elements and their respective elements. The existence of *instanceof*, *instantiate* or inter-model links between (meta-)elements is materialized by the value 1 in the matrix. Intra-model links are not represented, as they are not exploited in the Refining. For *contribution* relationship, the adjacency matrix is the following Matrix $M2C_{contribution}$:

$$MatrixM2C_{contribution} = \begin{array}{c} \\ o \\ r \\ c \\ t \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} o & r & c & t & e \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

(Where we write : o for Operation, r for reviewPaper():Operation, c for createCom():Operation, t for Task and e for EnterComment:Task).

According to previous matrix, the element EnterComment:Task is connected to Task by an *instantiate* link (value 1 in their respective row column intersection). This is also the case for the elements reviewPaper():Operation and createCom():Operation with Operation. Subsequently, the problem of defining the *Reproduction* function boils down to finding the path of length 3, from a model element (e.g, reviewPaper():Operation) to another model element (e.g, EnterComment:Task) passing through the meta-model level (e.g, Operation and Task by the *contribution* relationship) (see Figure 15). Thereafter, by cubing the matrix M2C, we obtain the correspondences that may occur among elements according to the correspondences established between their meta-elements. We can see in $M2C^3_{contribution}$ that reviewPaper():Operation and

createCom():Operation are linked to createCom():Operation by using *contribution*.

$$MatrixM2C^{3}_{contribution} = \begin{array}{c} \\ o \\ r \\ c \\ t \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} o & r & c & t & e \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

By merging the matrix cube of each HLC ($M2C^3_{contribution}$ and $M2C^3_{play}$), we obtain the correspondence Matrix representing the M1C.

$$M1C = \begin{array}{c} \\ se \\ rse \\ o \\ r \\ c \\ p \\ rp \\ t \\ e \end{array} \begin{array}{c} \begin{array}{ccccccccc} se & rse & o & r & c & p & rp & t & e \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

(Where we use *se* for StereotypedEntity, *rse* for Reviewer:StereotypedEntity, *p* for Pool and *rp* for Reviewer:Pool).

The reproduction is also transitive. If we take in addition a third meta-model MM3 (or many), the question of transitivity is essential. If we admit that a model element can be connected to only one meta-element and that there is transitivity at the meta-model level (for 3 meta-elements ME1, ME2, ME3 belonging to three different meta-models) then, if ME1 is linked to ME2 according to a relation R and ME2 is linked with ME3 with R then ME1 is linked to ME3 with also the same relation R. To summarize, the reproduction is deterministic and transitive as it consists of the generation of correspondences between elements whose type participates in a HLC.

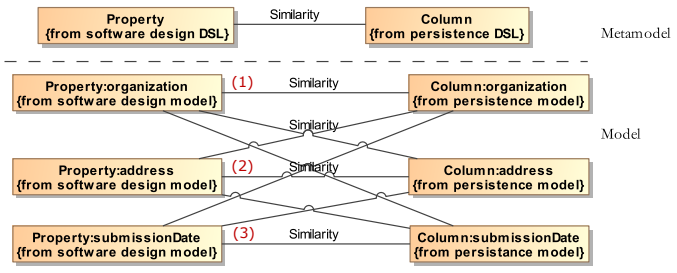

Figure 16 Example of reproduction of a HLC with *Similarity* relationship in CMS

However, even if the contextual information helps avoid the creation of correspondences between elements of types that do not match (e.g., an *Operation* and a *Property*) it does not guarantee that all generated correspondences are semantically correct. Figure 16 shows the nine correspondences obtained by reproduction from an HLC. The unnumbered correspondences are semantically wrong (e.g. *Similarity* between *organization* and *submissionDate*). It is so necessary to operate a selection in the set of correspondences that are generated by reproduction. The following section will discuss this selection function.

4.2.2. Selection This operation consists in filtering out semantically wrong correspondences produced by the reproduction operation in order to keep only those that are valid. We formalize the Selection ($\mathcal{S}$) operation as follows:

$$\begin{aligned} \mathcal{S}: \quad \mathcal{P}(Em) \times Em &\rightarrow \mathcal{P}(Em) \\ (\text{A,e}) &\mapsto \text{A'} \end{aligned}$$

$$\text{A'}=\{elem \in A \mid SE(e) \simeq SE(elem)\}$$

A Semantic Expression (SE) is associated with each relationship. Considering relationships with an informal expression (in natural language), it is the expert's role to decide whether to keep or not the correspondence depending on the expression associated to the relationships. On the contrary, the condition body of relationships with a formal expression, has to be executed. The condition body execution requires a language interpreter of the code in which it is written (a Java Virtual Machine (JVM) in our case). We discuss in the following the formal expression of the condition, depending on whether it is described in a model-based or ontology-based format (value of *format* in the Semantic Expression model - Figure 10). In case relationship's semantics are expressed in an ontology-based format, a transformation to the model-based format is required.

According to MMC presented in Figure 12, the semantic expression associated to the *Generalization* and *Aggregation* relationships are based on models elements (their *format* value is equal to MBF). Their semantic expression is based respectively on *isParentOf* and *isPartOf* methods (cf. Appendix A). These method implementations are based on the generic thesaurus WordNet (Patil & Atique 2013) via its Java API called JAWS.

The following step aims at executing the actions associated with each expression. The relationships *Similarity* and *Play* have in their conditions body, a method called *SameAS*. This function, described in Appendix A.1, retrieves data from a domain ontology using an API called OWL. If there is no domain ontology for the studied domain or if the desired elements are not represented in it, we use a generic thesaurus. Like in MBF format, we chose to use the Java Wordnet API: WS4J (Shima 2013). The similarity computation is performed using a measure based on the information content described by Lin (D. Lin 1998). According to (Gomaa & Fahmy 2013), it is one of the most used techniques. The implementation of other measures are available in (Meoli 2018) depending on the need for example :HSO (Hirst & St-Onge 1998), LCH (Leacock & Chodorow 1998), LESK (Banerjee & Pedersen 2002), etc.

If no result is found, the *SameAs* method uses an implementation of the *levenshteinDistance* method, which measures the effort to transform a string into another one. This method is

defined in an ontology alignment API. Note that we use *levenshteinDistance* metric because it is widely used in several domains and its implementation is already available. However, nothing prevents from using other metrics. A panorama of structured metrics is presented in (Cheatham & Hitzler 2013). Regarding the *Contribution* and *Require* relationships, the body of actions is based on the *Contain* method. It firstly consists in splitting a sentence into words (via the tokenization principle (Cheatham & Hitzler 2013)) and secondly in applying the *SameAs* method. Figure 17, illustrates an extract of the M1C model of CMS obtained at the end of the selection phase. For example, executing the following *sameAs* method:

Author.*sameAs*(AuthorTable)

returns true. The decision is to keep the correspondence involving the two elements. Similarly, the execution of these two code snippets:

fisrtName.*isPartOf*(fullName) and lastNam.*isPartOf*(fullName)

returns true. This leads to keeping the correspondence with the aggregation relationship between the source elements (*firstName*, *lastName*) and the target element (*fullName*).

To sum up, the selection operation is non-deterministic because, unlike the reproduction phase, the validity of LLCs resulting from the selection phase depends not only on the type of the linked meta-elements but also on the semantics associated to the relationships (see section 3.3). Above all the whole selection task lies in the hands of experts which may be highly subjectives.

4.2.3. Refining by extension

This refining can takes place after the refining by propagation. It allows for completing, when necessary, the description of an LLC with required functionalities to accurately specify the relationship used in a correspondence. Indeed, LLCs created by propagation may not totally meet the expert's needs. He or she may have to make choices about some actions to perform (Cariou et al. 2009), to preserve the desired properties or to add information about some correspondences.

Refining by *extension* aims at extending a correspondence by adding domain-specific constraints and treatments to the relationships. This extension implies the creation of a new relationship which inherits from the relationship it extends.

In the CMS example (see Figure 17), we can consider that certain correspondences with *Similarity* or *Aggregation* relationship have a too vague semantics regarding the application domain. They will therefore be refined by other relationships (Figure 18). The first one, *Similarity*, is replaced by an *Equality* relationship to describe the fact that the connected elements represent two different sides of the same entity (two elements are linked by an *Equality* relationship if they contain the same value and if their containers are similar). For example, the properties *organization* and *address* of the element User of the Software Design model are equal to the elements having the same name in AuthorTable and ReviewerTable in the Persistence model.

The second relationship, *aggregation*, is replaced by the *composition* relationship. It express a strict form of aggregation where the life cycles of the elements (components) depend on the aggregate. Thus, in Figure 18, we can see that *fullName* element of the Persistence model is composed of *firstName* and *lastName* elements of the Software Design model.

5. Tool support

So far, we have validated our approach through the development of a prototype, called HMCS, and different case studies. This section presents the support tool HMCS, the enactment of the approach of CMS using the tool.

5.1. Presentation

To assist the expert and put in practice our approach, we have developed HMCS (Heterogeneous Matching and Consistency management Suite). It is a suite of plugins that gives stakeholders two core features. The first one concerns the matching of heterogeneous models. The second one addresses the management of inconsistencies when models evolve. This part is not addressed in this paper. For more information about it, please refer to (El Hamlaoui, Ebersold, Coulette, et al. 2014).

In this section we illustrate the semi-automatic creation of M1C that contains the correspondences between the 3 models of CMS. The architecture of HMCS (Figure 19) is based on a set of modules (AMT, RT, etc.) developed using the following Frameworks: EMF, KOMMA, GMF, Xtext, JET, EMFCollab, TwoUse and CDO.

The Matching Tool (MT) is provided as an Eclipse plugin. As shown in Figure 20, it includes four modules represented by gears, namely: AMT (Assisted Matching Tool), RT (Refining Tool), M2T (Model To Text) and T2M (Text To Model). AMT takes as input the meta-models of the application domain as well as MMC. To be suitable for different uses, AMT module produces as output a correspondence model in textual (sample.m2cT) or graphical (sample.m2cG) form depending on the expert's needs. It must be noted that the two representations are synchronized, meaning that the expert may start by exploiting the graphical representation and continue on the same model of correspondences with the textual representation and vice versa. This is achieved through the M2T and T2M modules. The first module serializes sample.m2cG, through JET technology, in order to produce sample.m2cT. The second module aims at recovering the graphical model by parsing the textual implementation. The implementation of T2M has been done in Java, based on model management libraries. RT takes M2C as input as well as models conform to the previously used meta-models in order to produce the M1C.

5.2. Application to the CMS

Before starting the creation of the correspondence model, MMC may have to be specialized in order to add some relationships specific to the CMS domain. Within this context, the *Contribution* relationship for example is one of the relationships that have been added to MMC's kernel through the *Extend MMC* action (Figure 21). As presented in framed parts of the graphical editor on Figure 22, to create the M2C model, the expert chooses the meta-models and their meta-elements that describe the business domains (1).

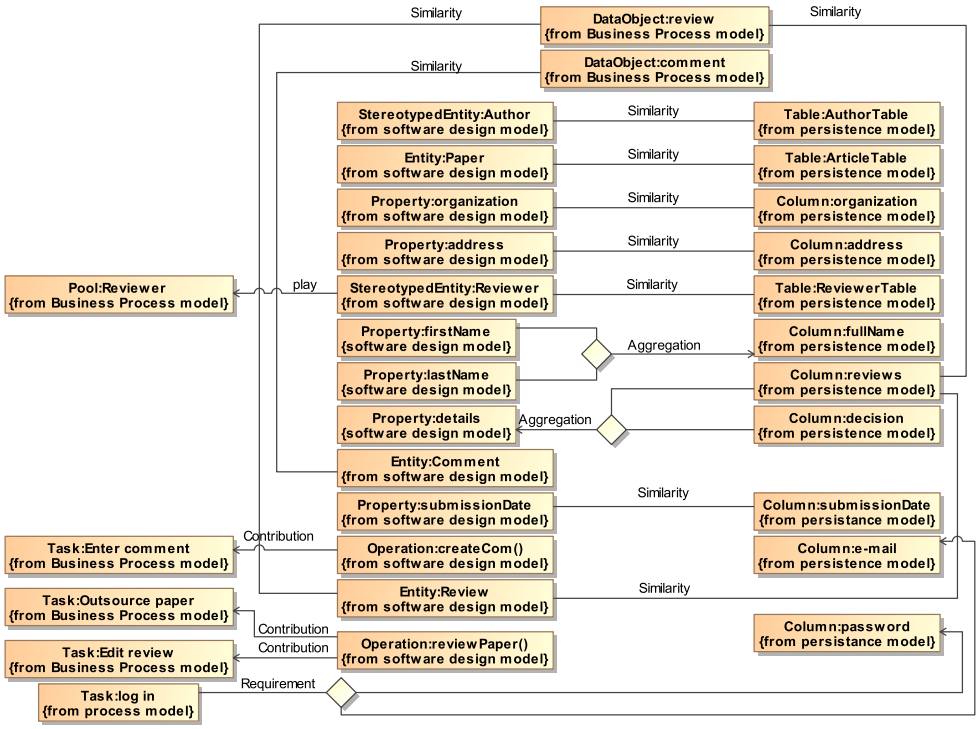

Figure 17 LLCs of CMS obtained by propagation refinement (extract)

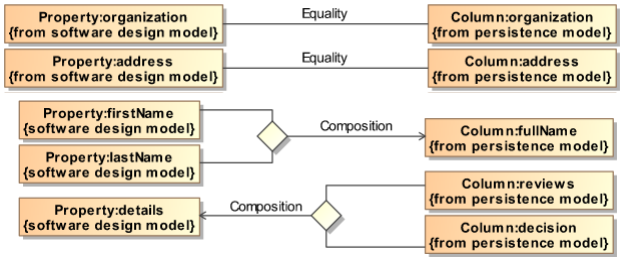

Figure 18 LLCs of CMS obtained by extension refinement

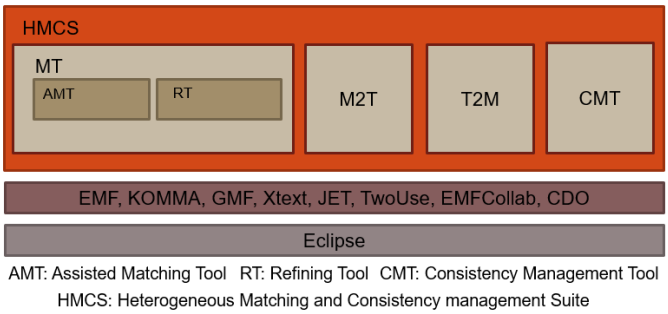

Figure 19 HMCS technical architecture

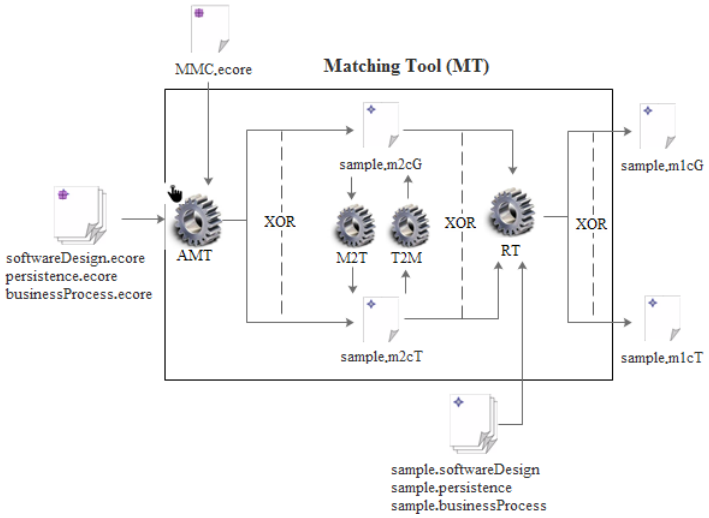

Figure 20 Functional matching tool steps

Once a meta-model is selected (persistence.ecore and software_design.ecore in Figure 22), the list of its meta-elements (instances of Ecore EClass) is loaded. The expert distinguishes source meta-elements (2) from target meta-elements (3) according to the meaning of the relationship that links them and which is specified in (4). When all elements are identified, a correspondence can be created via the *Apply* button. The involved meta-elements and their meta-models respect the expression P="Meta-element $\in$ Meta-model" while the correspondence

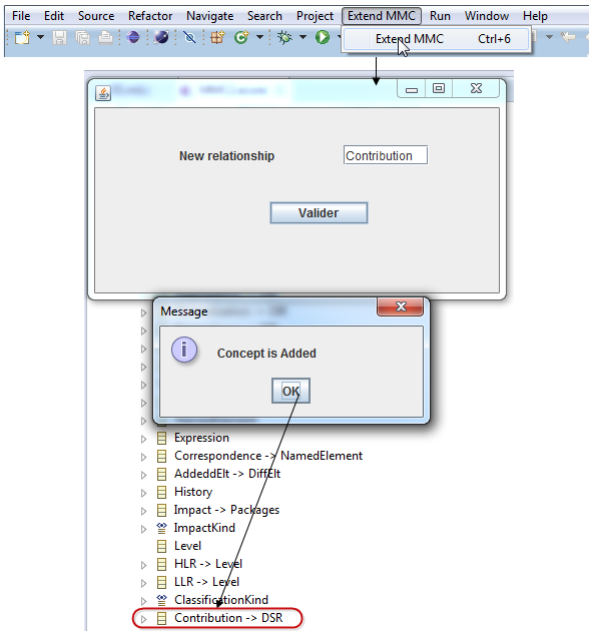

Figure 21 Extension of MMC by a type of relationship

display respects the expression: "Relationship (P(_P)*,P(_P)*)" (5).

Figure 23 shows an example of the M2C in the textual editor.

Once all the HLCs are created, the expert can click on the *save* button in the file menu, which leads to the creation of the correspondence model M2C presented in Figure 24 (two representations: one tabular and the other detailed). Once M2C is built, it goes through a process of refinement (Figure 25) to generate the Low Level Correspondences: firstly, the Reproduction operation (1) generates the cartesian product of the whole HLCs (as described in section 4.2.1), then the operation of Selection (2) permits to keep the correspondences that satisfy the semantics of their relationship's type only. It is possible to get information about the type (meta-element) of each model element, to know the model to which it belongs as well as the meta-model to which this latter conforms to. The expert can save the resulting M1C model if the results of the refinement operation are satisfying, otherwise he or she can start the extension operation (as described in section 4.3).

6. Evaluation

To evaluate HMCS tool, the expert establishes a golden model, i.e., a model of correspondences manually obtained after analyzing the source models, the expert stores in this model LLCs deemed important to have a global and consistent view on the models. After that, a comparison between this golden model and LLCs automatically produced by the tool was performed. In this comparison, we analysed the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) using precision, recall, f-measure and coverage metrics. Table 1 shows the evaluation in terms of these metrics for the CMS system whereas Table 2 summarizes the results of this evaluation for another case study concerning an Emergency department[2] (El Hamlaoui et al. 2016). (Notice that the evaluation is restricted to relationships that have several occurrences in M1C). The golden model of CMS system contains 27 LLCs: 16 using a similarity relationship, 4 using an aggregation relationship and 7 using a Contribution relationship while the model of correspondences produced by the tool for this system contains 14 LLCs using the similarity relationship (12 of these LLCs are true positives), 6 LLCs using the aggregation relationship (3 of them are TP) and 10 LLCs using the contribution relationship (7 are TP).

The precision metric is the ratio of correctly found correspondences (TP) over the total number of returned correspondences (TP + FP).

The recall presents the ratio of correctly found correspondences (TP) over the number of correspondences in the golden model (TP + FN).

The f-measure is the harmonic mean of precision and recall.

The coverage rate is the number of M1C elements involved in a relationship divided by the total number of elements involved in the M1C.

Table 1 Metric evaluation for the CMS case study

Relationship	Precision	Recall	F-measure	Coverage
Similarity	0.86	0.75	0.8	0.52
Aggregation	0.5	0.75	0.6	0.19
Contribution	0.7	1	0.82	0.16

Table 2 Metric evaluation for the ED case study

Relationship	Precision	Recall	F-measure	Coverage
Similarity	0.5	0.42	0.46	0.37
Generalization	1	0.5	0.67	0.21
Induction	0.5	0.33	0.4	0.32
Deduction	0	0	0	0.10

The gap in precision and recall values between the two case studies for the similarity relationship shows that the performances of HMCS tool matching depends on the data dictionaries used in the relationships' semantics. Precision and recall of the similarity relationship are impacted by the structure of the

[2] ED system has a similar complexity as the CMS. Its models have an average size of 45 concepts. It has 7 approved HLCs; the reproduction of these latter at the model level produces approximately 6200 LLCs and the semantic filtering has kept around 30 LLCs.

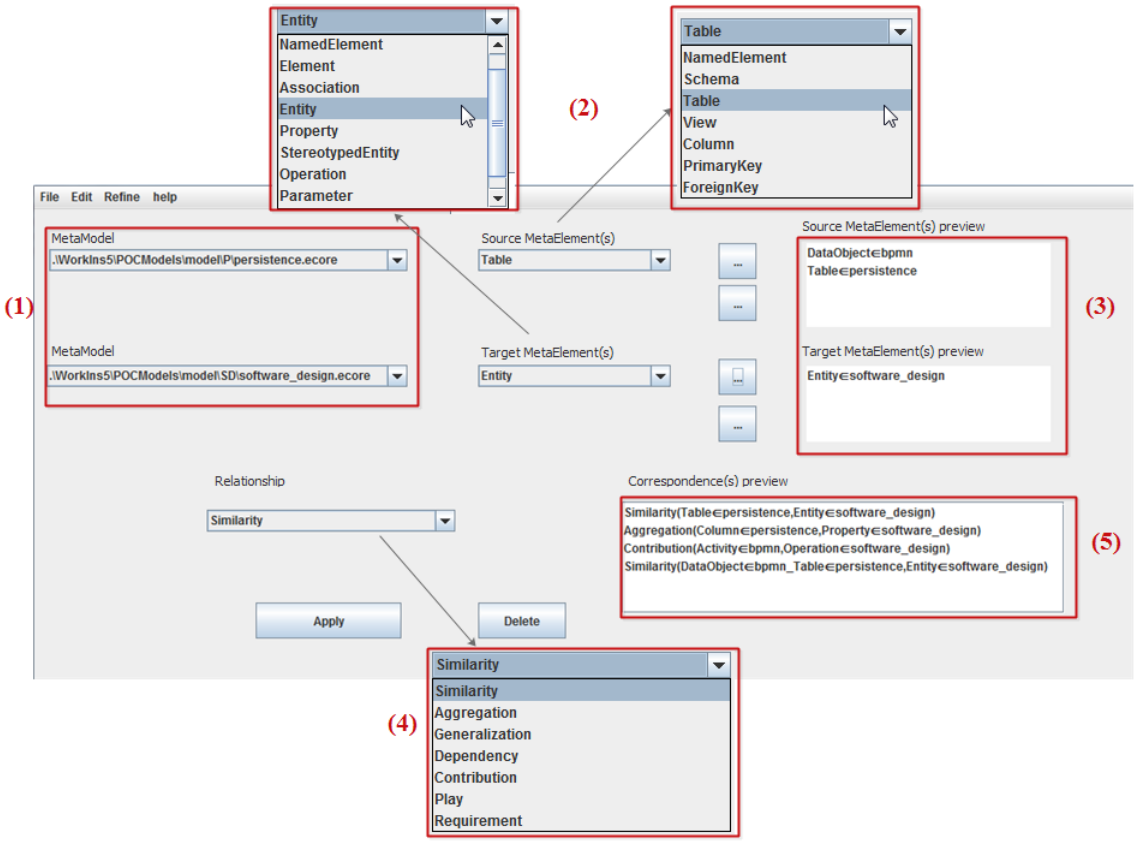

Figure 22 Graphical editor for M2C creation

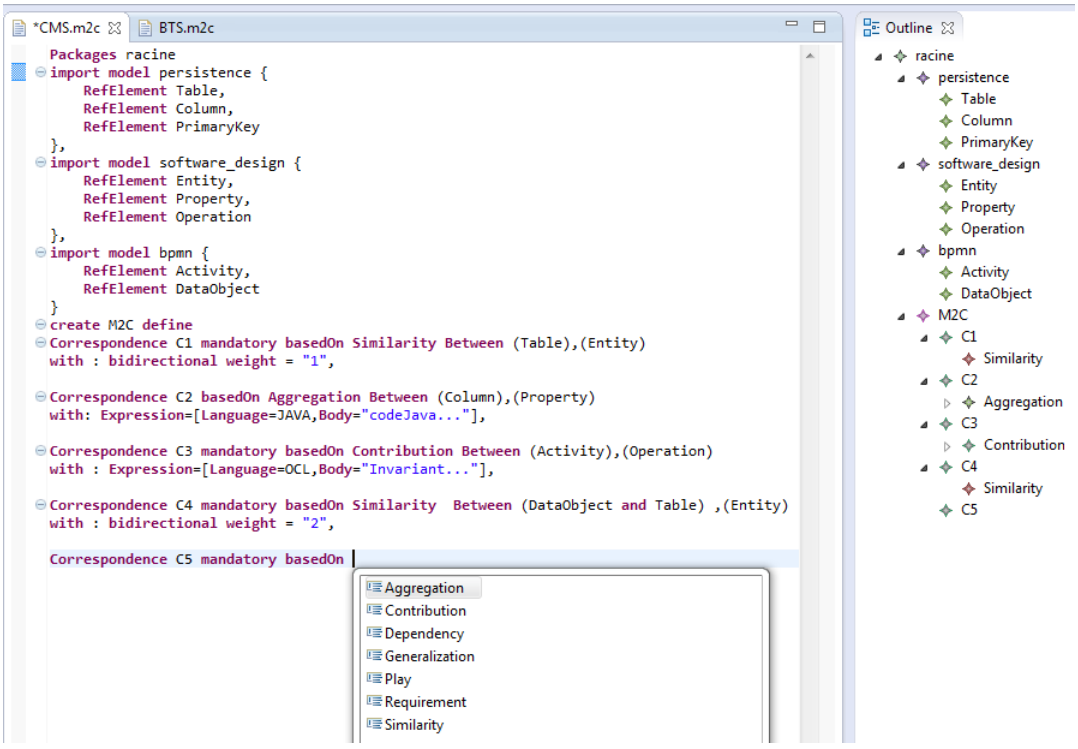

Figure 23 Textual editor for M2C creation

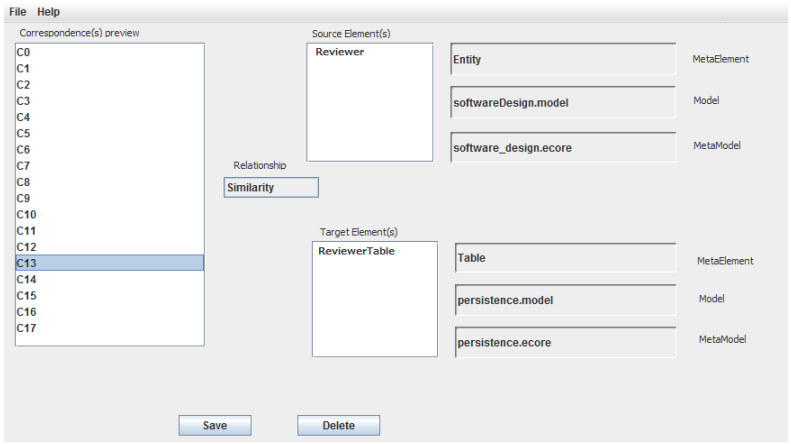

Table view of M1C

Search all fields: Enter keyword

Id	Relationship	Source Elements	Target Elements
	Similarity		
C7	Similarity	BP:DataObject:comment	SD:Entity:Comment
C8	Similarity	BP:DataObject:review	SD:Entity:Review
C9	Similarity	BP:DataObject:review	PS:Column:reviews
C10	Similarity	SD:Property:organization	PS:Column:organization
C11	Similarity	SD:Property:address	PS:Column:address
C12	Similarity	SD:Property:submissionDate	PS:Column:submissionDate
C13	Similarity	SD:StereotypedEntity:Reviewer	PS:Table:ReviewerTable
C14	Similarity	SD:StereotypedEntity:Author	PS:Table:AuthorTable
C15	Similarity	SD:Entity:Paper	PS:Table:ArticleTable
contains	exact	contains	contains

Figure 24 Graphical editor for M1C creation

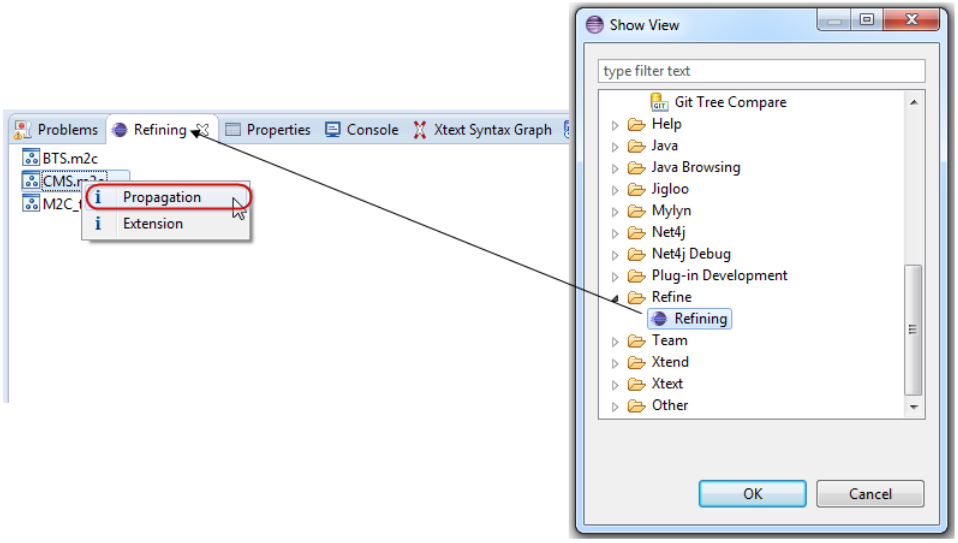

Figure 25 Integration of the refining view

EER model fields which are compound words. In fact, terminological choices and syntactic construction of models' elements may impact the matching. Similarly, operations from the organizational model are written as verbal sentences, therefore it may be difficult to find this structure in a reference ontology This explains the low rate of precision and recall of the induction

relationship.

Hence, we intend to integrate a pre-processing phase to manage compound names and sentences structure to mitigate the impact of syntactical constructs on the performance of the tool during the matching.

7. Related work

Several research works related to models matching have been discussed in the literature. In (Pfeiffer & Wąsowski 2015) and (Cicchetti et al. 2019), authors propose two feature models describing concepts required when dealing with multi-view and intermodel consistency. In this section, we use three characteristics from these feature models to evaluate the related works: *input artefacts*, *proposed relationships* and *produced correspondences*. Input artefacts can be characterized according to (1) their number (two or more) and (2) their heterogeneity. Proposed relationships can be either fixed, free or else domain specific. They are fixed when only a set of relationships is allowed (e.g., similarity, equivalence) and they are domain specific when they are associated to an application domain. Produced correspondences can be characterized according to (1) their arity (binary vs n-airy with n>=3) and (2) the possibility to keep them permanently.

In AMW ((Del Fabro et al. 2005), (Del Fabro & Valduriez 2007)), the authors describe a language that allows using M2M transformations for model comparison. These transformations produce a model of correspondences in the form of a weaving model. However, AMW can be used only when source and target model are very similar and requires that developers add extensions to the meta-model to define relationships, even for the obvious ones (e.g. similarity). Moreover according to (Kolovos 2009) used transformations are generally verbose since M2M languages do not provide customized constructs for model comparison. To optimize the representation of a composed model, the authors of the same team propose a model virtualization technique (VirtualEMF) (Clasen et al. 2011). Such technique may be useful for implementing models tracing and impacts calculation in case of source models evolution. As en extension to AMW, AtlanMod Matching Language (Garcés et al. 2009) proposes a DSL to express matching heuristics defined as transformation rules. It allows combining these heuristics as chains of model transformations in order to produce a weaving model. However, heuristics concern only similarity relationships. Always based on AMW, (Jouault et al. 2010) proposes a framework for model matching. It establishes both links among models and their respective elements by combining the components AMW and AM3 of the AMMA toolkit (AtlanMod Model Management Architecture). This approach considers only two input models simultaneously and produced correspondences do not have precise semantics so it is not easy neither to detect them nor to verify their correctness by the end of the matching. The correspondences are stored into a model of correspondences that may be seen as a megamodel in the sense of (Bézivin et al. 2004) whose elements are source models' elements and links between elements represent relationships between the models.

EMFCompare (Brun & Pierantonio 2008) calculates the correspondences on the basis of the *similarity* relationship. The matching engine is based on heuristics and elements are compared using several metrics including: similarity of name, content type and relationship. The values returned vary from 0 to 1, which will be pooled together to obtain overall similarity values. In the same context, DSMDiff (Y. Lin et al. 2007), extension of the work described in (Xing & Stroulia 2005) that uses name and structural similarities to identify the correspondences between UML models, adopted the same calculation techniques but in a larger context (i.e. different DSLs are supported thanks to GME (Generic Modelling Environment)). Both approaches use only similarity and do not exploit other relationships.

The AgreementMaker system (Cruz et al. 2009) is an extensible framework for matching real world schemas and ontologies, the matching process is divided into two main modules. Firstly, *similarity computation* in which each concept of the source ontology is compared with all the concepts of the target ontology. Secondly, *mappings selection* in which the matrix is scanned to select only the best mappings according to a given threshold and to the cardinality of the correspondences. This approach suits ontologies matching whereas it requires a transformation from the technological space of models to the one of ontologies in the case of model matching. In addition, it is not possible to express relationships other than similarity and equivalence.

In (Iovino et al. 2012), the authors propose a metamegamodel for managing the co-evolution of modeling artifacts. The GMM4EVO meta-model which stands for Global Model Management for managing co-EVOlution offers the possibility to specify relationships between models (and meta-models) and to navigate among them. By combining model weaving and megamodeling, the authors provide a generic infrastructure for managing the coevolution of modeling artifacts and for developing new adaptation techniques that can build on the provided infrastructure. Openflexo (Guychard et al. 2013; Golra et al. 2016) federates heterogeneous models into the same conceptual space through a virtual view that captures relationships and constraints. So far, there is no dedicated language to specify these relationships, so they are almost established manually by an expert, which may be tedious and error-prone. In (Dolques et al. 2011), authors propose a semi-automatic matching approach for discovering links between source and target models. They suppose that the target model results from a transformation of the source model. So, they extend the Anchor-Prompt approach to discover the pairs of elements for which there is a strong assumption of matching. In (Saada et al. 2014), the researchers propose a model matching approach, which adapts the NSGA-II algorithm (Deb et al. 2002) to explore the space of matching possibilities between the source and target model elements: the source model is fragmented using the minimal cardinalities of its meta-model and some defined OCL constraints. Then, for each fragment in the source model, the list of candidate corresponding fragments in the target model is searched. This approach uses the lexical tool TreeTagger (Schmid 1999), (Schmid 2013) to solve the problem of vocabulary between models. In (Atkinson et al. 2013), the authors present the foundations for the construction of a global view called SUM (Single Underlying Model). SUM is responsible for storing all known information

about the system with minimal redundancy, while the views are responsible for editing chosen projections of this information. The approach is similar to ours as it presents three ontological levels O0, O1 and O2. O0 defines types of relation (similar to MMC). O1 and O2 contains respectively High Level Correspondences (similar to M2C) and Low Level Correspondences (similar to M1C). However, so far, the authors present only the foundation for realizing the SUM. There is no explanation neither on how to create the relationships nor on the semantics assigned to them. Also, there is no clarification on how the correspondences in O1 and O2 are established. Authors of (Vanherpen et al. 2016) propose an ontological framework to define interrelationships between different viewpoints of a cyber-physical system (CPS). They introduce the notion of ontological properties (i.e. the domain properties considered critical in the translation of requirements to view-specific properties). Due to overlap in requirements, some ontological properties will be shared and/or will influence each other such that the related view-specific properties will be shared or influenced as well. Therefore, the proposed framework links the ontological properties and defines a satisfactory function for each of them. This function takes into consideration the view-specific properties that affect the ontological property. This approach relies on stakeholders' efforts to first define the ontological properties and then to associate a satisfactory function to them. EMF Views (Bruneliere et al. 2015) allows for the building of a view on a set of interrelated heterogeneous models using various types of link. However,this view is read-only. The authors use NeoEMF (Daniel et al. 2017) and Connected Data Objects (CDO) to provide lazy loading techniques and access and load model elements efficiently. They also use different persistence frameworks to allow for benefiting from different persistence capabilities (e.g graph, relational) (Bruneliere et al. 2018).

Table 3 summarizes the comparison of the presented approaches. In general, the studied matching approaches have shortcomings at two steps of the matching process: before and after the creation of the model of correspondences. Regarding the first step, we can notice the lack of balance between the ability to express correspondences and their reusability. Existing approaches are based mainly on only one of these criteria since reusability comes at the price of less expressiveness and vice versa. Moreover, the presented approaches may be divided in two categories : those that consider only direct mapping (fixed relationships i.e similarity and equivalence between models) and those that seek sound and precise dependencies among models. Work in the context of this second perspective are still undergone. Our approach perfectly meets this need. In fact, it offers a generic and extensible semi automatic solution for model matching. On the other hand, the studied related approaches manage only binary correspondences and therefore cannot establish complex n-ary ones relating a model element to any set of elements belonging to other models. Concerning the second step, we can note that some approaches do not keep the correspondences permanently, which means the matching have to be performed each time. This may be time consuming especially for n-ary correspondences. Also, approaches based on model transformations produce a model of correspondences between each pair of input models; so, for n input models, [n * (n-1)]/2 models of correspondences must be created, which leads to a large number of separate models without any connection between them and makes their management very difficult and almost impossible to automate. In our approach, we produce a unique model of correspondences relating input models.

Note that in this literature review, we dealt with two-level approaches, i.e., approaches that only manage two meta-levels at a time (metamodels and models) while AHM in fact can be applied in multilevel scenarios when it is allowed to work with models at any number of meta-levels simultaneously (Lara et al. 2014; Carvalho & Almeida 2018). Thus, the matching could be advantageous for this type of systems, and the HLCs defined in a meta-level can be propagated to the instances of several meta-levels below (instead of just the ones at the next level).

8. Discussion

With respect to the proposition described in the previous sections, there are some issues to be discussed.

An important issue, common to all multi-modeling approaches, concerns the definition of relationships' semantics used in the correspondences. In our approach, this is performed by the (integrator) expert by using the semantic expression DSL we proposed: SED. It allows for the expression of the meaning of each relationship as well as it serves as a filter during the selection phase of the refining by propagation. However, some semantic expressions are specified in a semi-formal language. One could argue that a semi-formal expression may limit the automation of our matching process. It is true but it can be seen as an option offered to the integrator expert. This latter may first describe a semantic expression in a semi-formal language and then refine it in a more formal language. This (optional) progressive way of describing the semantics of relationships makes our approach flexible and conform to real projects modeling.

Another complementary issue is scalability. How may the number of correspondences increase when the system gets bigger? Is it manageable with our approach? The HLC correspondences are defined at the meta-model level by the integrator expert. Even if a tool assists the expert, the task is mainly manual. The maximum number of correspondences is given by the cartesian product of the sets of meta-elements of all meta-models, multiplied by the number of types of relationships. It may be large but if the separation of concerns is well done, the number of inter-models relationships should be not so big. It is still manageable by an expert since this task is performed once for a given application domain. Actually, in real case studies we led, we noticed that most of HLC are binary and few of them are meaningful compared to the theoretical maximum. Concerning the LLC correspondences established at the model level, their number may be much higher since models of a complex system may be big. Furthermore, this task must be done for each new system to model in a given application domain. To make this task scalable and reduce the expert's work, we have proposed to automate several sub-tasks. Thus, the generation of a first set of LLC is produced via a propagation function composed of reproduction and selection sub-functions. The reproduction

Table 3 Summary comparison of matching approaches

Approach	Artefacts		Relationships		Correspondences		
	>2	Heter.	Free	Specific	binary	n-ary	persistent
AMW	-	+	∼	-	+	+	+
VirtualEMF	+	+	∼	-	+	-	+
AgreementMaker	-	+	-	+	+	+	+
GMM4EVO	+	+	∼	-	+	+	+
Openflexo	+	+	+	-	+	+	-
EMF Views	+	+	-	-	+	+	-
Dolques (Dolques et al. 2011)	-	+	-	+	+	-	-
Saada (Saada et al. 2014)	-	+	-	-	+	-	-
Atkinson (Atkinson et al. 2013)	+	∼	+	+	+	+	-
Vanherpen (Vanherpen et al. 2016)	+	+	+	+	+	+	-
AHM	+	+	+	-	+	+	+

sub-function is completely automated whereas the selection sub-function is semi-automated. In this latter case, it is up to the expert to take the right decisions such as keeping or removing a correspondence.

More generally we assume in our approach that semi-automatic tasks are performed by an expert having a global understanding of the various models and hence of the underlying business domains. This assumption makes the process dependent on the domain's expert and therefore relatively centralized. For instance, the expert is in charge of checking whether the MMC contains all needed relationships for a new studied domain. Throughout the HMCS tool, he or she is responsible for adding appropriate semantics, defining correspondences between meta-elements and removing some invalid correspondences generated at model level. These tasks may be tedious and difficult to perform by a single person. A more realistic approach in the context of complex systems would be to consider that the expert might ask models' designers to clarify the scope or meaning of an element and to help deciding whether to keep or not a correspondence, particularly when the semantics is expressed in natural language. In other words we can affirm that non automatable tasks of the matching process are intrinsically collaborative ones. For example, a given correspondence should be identified by the expert and model designers concerned by model elements involved in the correspondence; removing an LLC may require a decision (including a vote) taken by several stakeholders. In this purpose, we have conducted a work to support collaboration, especially collective decision making, in the matching process (Bennani et al. 2018).

9. Conclusion and perspectives

Our main research topic addresses the matching of interrelated heterogeneous models in the context of complex system development. Thereby, we are interested in establishing correspondences between heterogeneous models described through different DSLs used in a given application domain. In this paper, we first have proposed an heterogeneous matching process that relies on a correspondences meta-model (MMC) providing several advantages listed below:

- Commonality: MMC provides "generic" concepts, common to all application domains. It defines the most conventional relationships. Examples giving *similarity*, present in the majority of studied approaches, and the relationships that all UML users are familiar with,
- Variability: MMC can be extended depending on the peculiarities of the studied domain in order to support specific relationships. This is accomplished through specialization of the DSR meta-class,
- Flexibility: Thanks to flexibility at the conceptual level, MMC can relate several models (through their model elements) and express n-ary cardinality for each possible correspondence,
- Lightweight: Two models of correspondences are instanciated from MMC. M1C at model level and M2C at meta-model level. Both M1C and M2C are built in a virtual manner (as introduced in (Clasen et al. 2011)) as they only contain elements accessible through references.

Secondly, we have described the refining mechanism used to

produce the needed model of correspondences.

Finally, to minimize the integrator expert's work, we show how AHM provides the possibility to describe relationships semantics through a specific language (SED).

The model of correspondences can be used for different purposes. Firstly, it allows for having a global view on the system (Figure 1). Each stakeholder has not only a view on his own model but also a light view on the other models through the model of correspondences. This model uses the virtualization mechanism for accessing the elements used in the various correspondences. Secondly, it can be used for interoperability allowing different stakeholders to exchange information and exploit them especially since it reinforces the semantics associated to the system via the dedicated DSL. Thirdly, through the model of correspondences, that we can consider as a PIM (Platform Independent Model) and by using the M2M and M2T Frameworks, it is possible to generate code (depending on the associated PDM (Platform Dependent Model)), for instance, database schema of the system, whole domain application code in Java, etc. Last, one of the important features of the model of correspondences, presented in (El Hamlaoui, Ebersold, Coulette, et al. 2014), is to facilitate the management of models consistency when one or several models evolve.

To tackle issues mentioned in the discussion section and enhance AHM, several works are in progress. Firstly, we are working on the improvement of the refining formalization to represent the global matrix bypassing the fusion of each HLC matrices. For this, we are exploiting the use of edge-colored multigraphs (Águeda et al. 2011), to take into account the fact that a couple of nodes (elements) can be connected with several edges (relationships) of different natures. Secondly, to facilitate the choice in case of large quantity of correspondences, we are also planning to add a measure (to be built) indicating the probability of keeping a correspondence.

Besides, the model of correspondences can be used to manage model consistency (synchronization) when source models evolve. Changes in one or several source models are automatically detected and a semi-automated process is performed to update the related correspondence model (M1C), calculate impacts of changes in the other models, and take decisions about effective changes to do. We are exploring this track whose principles are described in (El Hamlaoui, Ebersold, Coulette, et al. 2014),

Another track of improvement is to lead more experiments on real industrial projects. We have already applied our approach on significant case studies but it would worth going further. The idea is to get more feedbacks based on experiments conducted using evaluation metrics. This work will be conducted with users having skills in system modeling, once the alpha tests of our tool performed and its collaborative version finalized.

Acknowledgments

We thank all the participants to the Emergency Department case study: Jean-Christophe Bach, Antoine Beugnard, Yassine Jamoussi, Lee Osterweil, Seung Yeob Shin, Hanh-Nhi Tran. We are also grateful to the reviewers for their suggestions that helped to improve the article.

References

Águeda, R., Borozan, V., Groshaus, M., Manoussakis, Y., Mendy, G., & Montero, L. (2011). Proper hamiltonian paths in edge-colored multigraphs. *Electronic Notes in Discrete Mathematics*, *38*, 5–10.

Anwar, A., Ebersold, S., Coulette, B., Nassar, M., & Kriouile, A. (2010). A rule-driven approach for composing viewpoint-oriented models. *Journal of Object Technology*, *9*(2), 89–114.

Atkinson, C., Gerbig, R., & Tunjic, C. (2013). A multi-level modeling environment for sum-based software engineering. In *Proceedings of the 1st workshop on view-based, aspect-oriented and orthographic software modelling* (pp. 2:1–2:9). New York, USA.

Banerjee, S., & Pedersen, T. (2002). An adapted lesk algorithm for word sense disambiguation using wordnet. In *International conference on intelligent text processing and computational linguistics* (pp. 136–145).

Baudry, B., Combemale, B., DeAntoni, J., Mallet, F., AOSTE, E., & Antipolis, S. (2012). *Action spécifique 2011 gemoc du gdr gpl ingénierie du logiciel pour les systèmes hétérogènes.*

Bennani, S., El Hamlaoui, M., Nassar, M., Ebersold, S., & Coulette, B. (2018). Collaborative model-based matching of heterogeneous models. In *2018 ieee 22nd international conference on computer supported cooperative work in design ((cscwd))* (pp. 443–448).

Bézivin, J., Jouault, F., & Valduriez, P. (2004). On the need for megamodels. In *Proceedings of the oopsla/gpce: Best practices for model-driven software development workshop, 19th annual acm conference on object-oriented programming, systems, languages, and applications.* Vancouver, Canada.

Boronat, A. (2007). Moment: a formal framework for model management. *PhD in Computer Science, Universitat Politenica de Valencia (UPV), Spain*.

Boronat, A., Knapp, A., Meseguer, J., & Wirsing, M. (2009). What is a multi-modeling language? In *International workshop on algebraic development techniques: Recent trends in algebraic development techniques.*

Boulanger, F., Jacquet, C., Hardebolle, C., & Rouis, E. (2010). Modeling heterogeneous points of view with modhel'x. In S. Ghosh (Ed.), *Models in software engineering* (pp. 310–324). Berlin, Heidelberg: Springer Berlin Heidelberg.

Brun, C., & Pierantonio, A. (2008). Model differences in the eclipse modeling framework. *UPGRADE, The European Journal for the Informatics Professional*, *9*(2), 29–34.

Bruneliere, H., de Kerchove, F. M., Daniel, G., & Cabot, J. (2018). Towards scalable model views on heterogeneous model resources. In *Proceedings of the 21th acm/ieee international conference on model driven engineering languages and systems* (pp. 334–344).

Bruneliere, H., Perez, J. G., Wimmer, M., & Cabot, J. (2015). Emf views: A view mechanism for integrating heterogeneous models. In *International conference on conceptual modeling* (pp. 317–325).

Brunelière, H., Perez, J. G., Wimmer, M., & Cabot, J. (2015). Emf views: A view mechanism for integrating heterogeneous

models. In *International conference on conceptual modeling* (pp. 317–325).

Cariou, E., Belloir, N., & Barbier, F. (2009). Contrats de transformations pour la validation de raffinement de modèles. *5èmes journées sur l'Ingénierie Dirigée par les Modèles*, *1501*(09), 1–16.

Carvalho, V. A., & Almeida, J. P. A. (2018). Toward a well-founded theory for multi-level conceptual modeling. *Software & Systems Modeling*, *17*(1), 205–231.

Cheatham, M., & Hitzler, P. (2013). *The role of string similarity metrics in ontology alignment.* (Technical report, Kno.e.sis Center, Wright State University)

Cicchetti, A., Ciccozzi, F., & Pierantonio, A. (2019). Multi-view approaches for software and system modelling: a systematic literature review. *Software & Systems Modeling*, 1–27.

Clasen, C., Jouault, F., & Cabot, J. (2011). Virtualemf: a model virtualization tool. In *International conference on conceptual modeling* (pp. 332–335).

Cruz, I. F., Antonelli, F. P., & Stroe, C. (2009). Efficient selection of mappings and automatic quality-driven combination of matching methods. In *Proceedings of the 4th international conference on ontology matching (iwom@iswc) ceur workshop proceedings* (pp. 49–60).

Daniel, G., Sunyé, G., Benelallam, A., Tisi, M., Vernageau, Y., Gómez, A., & Cabot, J. (2017). Neoemf: a multi-database model persistence framework for very large models. *Science of Computer Programming*, *149*, 9–14.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, *6*(2), 182–197.

Del Fabro, M. D., Bézivin, J., Jouault, F., Breton, E., & Gueltas, G. (2005). Amw: a generic model weaver. In *1 ere journées sur l'ingénierie dirigée par les modèles (idm05)* (pp. 105–114).

Del Fabro, M. D., & Valduriez, P. (2007). Semi-automatic model integration using matching transformations and weaving models. In *Proceedings of the 2007 acm symposium on applied computing* (pp. 963–970). Seoul, Korea.

Dolques, X., Dogui, A., Falleri, J.-R., Huchard, M., Nebut, C., & Pfister, F. (2011). Easing model transformation learning with automatically aligned examples. In *European conference on modelling foundations and applications* (pp. 189–204).

Drey, Z., Faucher, C., Fleurey, F., Mahé, V., & Vojtisek, D. (2009). Kermeta language reference manual. *Manuscript available online http://www. kermeta. org*.

El Hamlaoui, M., Bennani, S., Ebersold, S., Nassar, M., & Coulette, B. (2018). Ahm: Handling heterogeneous models matching and consistency via mde. In *International conference on evaluation of novel approaches to software engineering* (pp. 288–313).

El Hamlaoui, M., Bennani, S., Nassar, M., Ebersold, S., & Coulette, B. (2018). A MDE approach for heterogeneous models consistency. In *Proceedings of the 13th international conference on evaluation of novel approaches to software engineering, ENASE 2018, funchal, madeira, portugal, march 23-24, 2018.* (pp. 180–191).

El Hamlaoui, M., Coulette, B., Ebersold, S., Bennani, S., Nassar, M., Anwar, A., ... Tran, H. N. (2016). Alignment of viewpoint heterogeneous design models: Emergency department case study. In *4th international workshop on the globalization of modeling languages (gemoc 2016) co-located with acm/ieee models 2016.*

El Hamlaoui, M., Ebersold, S., Anwar, A., Coulette, B., & Nassar, M. (2014). Towards a framework for heterogeneous models matching. *Journal of Software Engineering*, *8*(3), 132–151.

El Hamlaoui, M., Ebersold, S., Coulette, B., Anwar, A., & Nassar, M. (2013). A process for defining a unique correspondence model to relate heterogeneous models. In *International conference on evaluation of novel approaches to software engineering (enase).* SciTePress.

El Hamlaoui, M., Ebersold, S., Coulette, B., Nassar, M., & Anwar, A. (2014). Heterogeneous models matching for consistency management. In *Research challenges in information science (rcis), 2014 ieee eighth international conference on* (pp. 1–12).

El Hamlaoui, M., Trojahn, C., Ebersold, S., & Coulette, B. (2014). Towards an ontology-based approach for heterogeneous model matching. In *2nd international workshop on the globalization of modeling languages (gemoc 2014) co-located with with acm/ieee international conference on model driven engineering languages and systems (models).* CEUR Workshop Proceedings.

Filman, R. E., Elrad, T., Clarke, S., & Aksit, M. (2004). *Aspect-oriented software development*. Addison Wesley.

France, R., & Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In *2007 future of software engineering* (pp. 37–54).

Garcés, K., Jouault, F., Cointe, P., & Bézivin, J. (2009). A domain specific language for expressing model matching. In *Proceedings of the 5ère journée sur l'ingénierie dirigée par les modèles (idm09)* (pp. 33–48). Nancy, France.

Golra, F. R., Beugnard, A., Dagnat, F., Guerin, S., & Guychard, C. (2016). Addressing modularity for heterogeneous multi-model systems using model federation. In *Companion proceedings of the 15th international conference on modularity* (pp. 206–211).

Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. *International Journal of Computer Applications*, *68*(13), 13–18.

Gosling, J. (2000). *The java language specification*. Addison-Wesley Professional.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, *43*(5-6), 907–928.

Guychard, C., Guerin, S., Koudri, A., Beugnard, A., & Dagnat, F. (2013). Conceptual interoperability through models federation. In *Semantic information federation community workshop.*

Hilliard, R. (2001). Viewpoint modeling. In *Proceedings of 1st icse workshop on describing software architecture with uml (dsau@icse)* (Vol. 7). Toronto (Canada): ACM.

Hirst, G., & St-Onge, D. (1998). Lexical chains as representations of context for the detection and correction of

malapropisms. *WordNet: An electronic lexical database*, *305*, 305–332.

Iovino, L., Pierantonio, A., & Malavolta, I. (2012). On the impact significance of metamodel evolution in mde. *Journal of Object Technology*, *11*(3), 3–1.

Jackson, D. (2002). Alloy: a lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *11*(2), 256–290.

Jenkinson, T., Truss, J., & Seidel, D. (2012). Countable homogeneous multipartite graphs. *European Journal of Combinatorics*, *33*(1), 82–109.

Jouault, F., Vanhooff, B., Bruneliere, H., Doux, G., Berbers, Y., & Bézivin, J. (2010). Inter-dsl coordination support by combining megamodeling and model weaving. In *Proceedings of the 2010 acm symposium on applied computing, track "coordination models, languages and applications"* (pp. 2011–2018). Sierre, Switzerland.

Kedji, K. A., Lbath, R., Coulette, B., Nassar, M., Baresse, L., & Racaru, F. (2014). Supporting collaborative development using process models: a tooled integration-focused approach. *Journal of Software: Evolution and Process*, *26*(10), 890–909.

Kolovos, D. S. (2009). Establishing correspondences between models with the epsilon comparison language. In *European conference on model driven architecture-foundations and applications* (pp. 146–157).

Kolovos, D. S., Paige, R. F., & Polack, F. A. (2006a). Merging models with the epsilon merging language (eml). In *International conference on model driven engineering languages and systems* (pp. 215–229).

Kolovos, D. S., Paige, R. F., & Polack, F. A. (2006b). Model comparison: a foundation for model composition and model transformation testing. In *Proceedings of the 2006 international workshop on global integrated model management* (pp. 13–20). Shanghai, China.

Koning, H., & van Vliet, H. (2006). A method for defining ieee std 1471 viewpoints. *Journal of Systems and Software*, *79*(1), 120–131.

Lano, K. (1996). *The b language and method: a guide to practical formal development*. Springer-Verlag New York, Inc.

Lara, J. D., Guerra, E., & Cuadrado, J. S. (2014). When and how to use multilevel modelling. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *24*(2), 1–46.

Leacock, C., & Chodorow, M. (1998). Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, *49*(2), 265–283.

Lin, D. (1998). An information-theoretic definition of similarity. In *Icml* (Vol. 98, pp. 296–304).

Lin, Y., Gray, J., & Jouault, F. (2007). Dsmdiff: a differentiation tool for domain-specific models. *European Journal of Information Systems*, *16*(4), 349–361.

Liu, H., & Singh, P. (2004). Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, *22*(4), 211–226.

Medvidovic, N., & Taylor, R. N. (2000). A classification and comparison framework for software architecture description languages. *IEEE Transactions on software engineering*, *26*(1), 70–93.

Meoli, D. (2018). *Wordnet similarity for java provides an api for several semantic relatedness/similarity algorithms.* Retrieved from https://github.com/dmeoli/WS4J

Mernik, M., Heering, J., & Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, *37*(4), 316–344.

Mosterman, P. J., & Zander, J. (2016). Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems. *Software & Systems Modeling*, *15*(1), 5–16.

Motik, B., Patel-Schneider, P. F., Parsia, B., Bock, C., Fokoue, A., Haase, P., … Sattler, U. (2009). Owl 2 web ontology language: Structural specification and functional-style syntax. *W3C recommendation*, *27*(65), 159.

Nassar, M. (2003). Vuml: a viewpoint oriented uml extension. In *Automated software engineering, 2003. proceedings. 18th ieee international conference on* (pp. 373–376). Montreal, Canada.

Ober, I., Coulette, B., & Lakhrissi, Y. (2008). Behavioral modelling and composition of object slices using event observation. In *Acm/ieee international conference on model driven engineering languages and systems(models)* (pp. 219–233).

OMG. (2011). *Unified modeling language (uml).* Retrieved 2014-11-20, from http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF (version 2.4.1)

OMG. (2013). *Business process model and notation (bpmn).* Retrieved 2014-02-10, from http://www.omg.org/spec/BPMN/2.0.2/PDF (version 2.0.2)

OMG. (2014). *Object constraint language (ocl).* Retrieved 2014-11-10, from http://www.omg.org/spec/OCL/2.4/PDF (version 2.4)

Patil, L. H., & Atique, M. (2013). A novel feature selection based on information gain using wordnet. In *Science and information conference (sai), 2013* (pp. 625–629).

Pfeiffer, R.-H., & Wąsowski, A. (2015). The design space of multi-language development environments. *Software & Systems Modeling*, *14*(1), 383–411.

Saada, H., Huchard, M., Nebut, C., & Sahraoui, H. (2014). Model matching for model transformation a meta-heuristic approach. In *Model-driven engineering and software development (modelsward), 2014 2nd international conference on* (pp. 174–181).

Salay, R., Kokaly, S., Chechik, M., & Maibaum, T. (2016). Heterogeneous megamodel slicing for model evolution. In *Me@ models* (pp. 50–59).

Schmid, H. (1999). Improvements in part-of-speech tagging with an application to german. In *Natural language processing using very large corpora* (pp. 13–25). Springer.

Schmid, H. (2013). Probabilistic part-ofispeech tagging using decision trees. In *New methods in language processing* (p. 154). Routledge.

Shima, H. (2013). *Ws4j wordnet similarity for java.* Retrieved from https://code.google.com/p/ws4j/

Silva Parreiras, F., Staab, S., & Winter, A. (2007). *Twouse: Integrating uml models and owl ontologies.*

Vanherpen, K., Denil, J., Dávid, I., De Meulenaere, P., Mosterman, P. J., Torngren, M., ... Vangheluwe, H. (2016). Ontological reasoning for consistency in the design of cyber-physical systems. In *Cyber-physical production systems (cpps), 2016 1st international workshop on* (pp. 1–8). Vienna.

Vogel, T., Seibel, A., & Giese, H. (2010). The role of models and megamodels at runtime. In *International conference on model driven engineering languages and systems* (pp. 224–238).

Voigt, K., Ivanov, P., & Rummler, A. (2010). Matchbox: combined meta-model matching for semi-automatic mapping generation. In *Proceedings of the 2010 acm symposium on applied computing* (pp. 2281–2288). New York, USA.

Wagelaar, D. (2005). Context-driven model refinement. In *Model driven architecture* (pp. 189–203). Springer.

Wolf, W. H. (1994). Hardware-software co-design of embedded systems. *Proceedings of the IEEE*, *82*(7), 967–989.

Wooldridge, M. (1997). Agent-based software engineering. *IEE Proceedings-software*, *144*(1), 26–37.

Xing, Z., & Stroulia, E. (2005). Umldiff: an algorithm for object-oriented design differencing. In *Proceedings of the 20th ieee/acm international conference on automated software engineering* (pp. 54–65). Long Beach, USA.

Yang, G., Sangiovanni-Vincentelli, A., Watanabe, Y., & Balarin, F. (2004). Separation of concerns: overhead in modeling and efficient simulation techniques. In *Proceedings of the 4th acm international conference on embedded software* (pp. 44–53).

Yousafzai, A., Chang, V., Gani, A., & Noor, R. M. (2016). Multimedia augmented m-learning: Issues, trends and open challenges. *International Journal of Information Management*, *36*(5), 784–792.

Zito, A., Diskin, Z., & Dingel, J. (2006). Package merge in uml 2: Practice vs. theory? In *International conference on model driven engineering languages and systems* (pp. 185–199).

Appendices

A. Semantic expressions

In the following sections, we present the proposed implementation of the different methods used for the CMS system.

A.1. sameAs

```
public boolean sameAs (String element)
    {
    OWLOntologyManager manager =
    OWLManager.createOWLOntologyManager ();
    // load the ontology to be imported
    OWLOntology kb = manager.loadOntologyFromOntologyDocument
    (new StringDocumentSource(kb));
    boolean state = false;
    for (OWLClass c : kb.getClassesInSignature ())     {
        if (c.getName ().equals (this)     || c.getName ().equals (
      element))
            if (c.getName ().isEquivalentTo (this) || c.getName ()
            .isEquivalentTo (element))
            state = true
        }

    if (state) return true;
        /* if data is not found in the ontology or if there is no
      ontology */
        else{
      ILexicalDatabase db = new NictWordNet ();
      RelatednessCalculator rc = new Lin (db);
      double score = rc.calcRelatednessOfWords(this, element);
          if (rc > 0.7) return true;
            // if the data is not found lexicographically, use
        structural comparison
            else {
                uri1 = new URI ( file:ontology1.owl );
                uri2 = new URI ( file:ontology2.owl );
              AlignmentProcess al = new StringDistAlignment();
              al.init (uri1, uri2);
                  //Apply levenshteinDistance method
              params.setProperty ("stringFunction", "levenshtein Distance"
        );
                al.align ( (Alignment)null, params );
                  for (Enumeration e = al.getElements (); e.
        hasMoreElements ();)
                      {
                        Cell cell = (Cell) e.nextElement ();
                      String ent1 = cell.getEntity1 ();
                      String ent2 = cell.getEntity2 ();
                      if (ent1.equals (this) && ent2.equals (element)
                    &&
                                cell.getMeasure > 0,9) return true;
                      }
              }
      return false;
          }
    }
```

Listing 1 code of sameAs method

A.2. isParentOf

```
public boolean isParentOf (String element)
    {
    WordNetDatabase database = WordNetDatabase.getFileInstance ();
    ArrayList <String> Hyponyms = new ArrayList <String> ();
    NounSynset nounS = (NounSynset) this;
    Hyponyms.addAll (nounS.getHyponyms);
    if (Hyponyms.contains (element)) return true;
    return false;
    }
```

Listing 2 code of isParent method

A.3. isPartOf

```
public boolean isPartOf (String element)
    {
    WordNetDatabase database = WordNetDatabase.getFileInstance ();
    ArrayList <String> Holonyms = new ArrayList<String> ();
    NounSynset nounS = (NounSynset) this;
    // retrieve holonyms associated with nounS
    Holonyms.addAll (nounS.getHolonyms ());
    // check if the element in parameter exists in the list
    if (Holonyms.contains (element)) return true;
    return false;
    }
```

Listing 3 code of isPartOf method

A.4. contain

```
public boolean contain (String sentence)
    {
    // split the sentence parameter in several words
    String [ ] tokens = tokenizer.tokenize (sentence);
    //Look up for similarity between the tokens and the current object
    for (int i = 0; i < tokens.length; ++i)
        if (this.SameAS (tokens[i])) return true
    return false;
    }
```

Listing 4 code of contain method

About the authors

Mahmoud El Hamlaoui is a Professor of Software Engineering at ENSIAS@UM5R where he teaches different topics related to Software Engineering. He is specialized in Model Driven (Software) Engineering, Reverse-Engineering, Internet of Things and Cloud Computing. He is a research scientist within IMS@ADMIR team at Rabat It Center of the Mohammed V University in Rabat and also within SM@RT team at the CNRS Research Institute of Computer Science in Toulouse (IRIT). Co-organizer of several events around technology watch and multidisciplinary coach of teams in national and international hackathons with several awards to his credit. He participated in the 2019 and 2020 NASA Space Apps challenge as a mentor and the NASA Space Apps COVID-19 challenge as a coach. Contact him at mahmoud.elhamlaoui@um5.ac.ma.

Sophie Ebersold is associate professor at the University of Toulouse Jean Jaures (UT2J) and is member of the SM@RT Research Team of IRIT Laboratory. She was notably co-chair of international workshops MSE@STAF18, the third last editions of FormReq@RE, and co-chief editor of a special magazine issue. Her topics of research are in Sotware Engineering, especially focused on multi-views complex systems. She co-leads the Model driven Engineering working group of GDR GPL (http://gdr-gpl.cnrs.fr). Since three years, she is co-supervising a PhD thesis in collaboration with Bertrand Meyer and Jean-Michel Bruel in the fields of Requirements Engineering and is currently working on requirements from eliciation to formalization and verification still in the scope of complex systems. Together, they recently published a paper in the ACM Computing Surveys. Contact her at sophie.ebersold@irit.fr.

Saloua Bennani is a research and teaching assistant at the University of Toulouse Jean Jaures (UT2J). She received her PhD from the University of Toulouse and the University of Mohammed V in Rabat. Her research interests include model driven engineering, collaborative engineering and human group decision-making. Contact her at saloua.bennani@irit.fr.

Adil Anwar works as a full Professor in Computer Science at Mohammadia School of engineers, at the University Mohammed V in Rabat. In 2009, he received a Ph.D degree in Computer Science at the University of Toulouse. His area of interest includes software engineering, Model Driven engineering, Requirement engineering as well as Domain Specific Modeling Languages. Contact him at adil.anwar@um5.ac.ma.

Taoufiq Dkaki is an Assistant Professor of Software Engineering at the University of Toulouse Jean Jaures (UT2J). He received an engineer degree in mathematics and computer science from INP-ENSEEIHT a state-funded French school of engineering and a PhD from the University Paul Sabatier of Toulouse. His research activities include Information Retrieval models, graph based information representation and analysis and information visualization. Contact him at taoufiq.dkaki@irit.fr.

Mahmoud Nassar is a full Professor at National Higher School for Computer Science and Systems Analysis (ENSIAS), Mohammed V University in Rabat, Morocco. He is a Head of IMS (IT architecture and Model Driven Systems development) Team / ADMIR Laboratory of Rabat IT Center. He received his PhD in Computer Science from the Toulouse-INP, France. His research interests are Context-Aware Service-Oriented Computing, Component based Engineering, Model-Driven Engineering, Cloud computing, and Cloud Migration. He leads numerous R&D projects related to the application of these domains in Smart Cities, Embedded Systems, e-Health, and e-Tourism. Contact him at mahmoud.nassar@um5.ac.ma.

Bernard Coulette is an emeritus professor at the University of Toulouse Jean Jaures (UT2J), and member of the IRIT laboratory of Toulouse. His research field of interest is mainly Software Engineering, and more particularly: integration of viewpoints in Analysis/Design, Model Driven Engineering, Composition of heterogeneous models, Collaborative strategies for models alignment, Modeling and Enactment of Collaborative Software Processes, Collaborative patterns for processes execution. He has directed numerous PHD students and has been strongly involved in international collaborations. Contact him at bernard.coulette@irit.fr.