Is my model right? Let me ask the expert

Antonia Bertolino, Guglielmo De Angelis,
Alessio Di Sandro, Antonino Sabetta

CNR-ISTI
Pisa, Italy
{antonia.bertolino, guglielmo.deangelis, alessio.disandro, antonino.sabetta}@isti.cnr.it

Abstract
Defining a domain model is a costly and error-prone process. It requires that the knowledge possessed by domain experts be suitably captured by modeling experts. Eliciting what is in the domain expert’s mind and expressing it using a modeling language involve substantial human effort. In the process, conceptual errors may be introduced that are hard to detect without a suitable validation methodology. This paper proposes an approach to support such validation, by reducing the knowledge gap that separates modeling experts and domain experts. While our methodology still requires the domain expert’s judgement, it partially automates the validation process by generating a set of yes/no questions from the model. The answers from the domain experts are compared with the model in order to identify elements that may require further consideration. Our methodology was implemented as a tool and was applied to a real case study, within the IPERMOB project.

1. Introduction

Model Driven Engineering (MDE) is a very active research topic [24, 7]. It focuses on creating an abstract representation of a system that is close to some relevant domain concepts rather than describing the system in terms of the underlying and general-purpose technologies by which it is implemented. The principle behind is that everything is a model [1].

The definition of models (e.g. domain models) is an effort-intensive activity that requires the synergy between different backgrounds and skills: from modeling experts (MEs) to master MDE environments and resources; and from the domain experts (DEs) to delve into the application domain.

The definition of models is also a critical activity. Since models usually are the starting point for many subsequent transformations, faults that are possibly introduced in this stage can have deleterious impacts. Therefore, ensuring that the model is right becomes of paramount importance. The term “right” in the previous sentence may mean different things: paraphrasing Boehm’s informal definitions for distinguishing between verification and validation [2], we have to ask ourselves:
• are we building the model right?
• are we building the right model?

Our work addresses the second question, which is the thorniest of the two. More precisely, in their highly referenced framework for conceptual model quality, Lindland and coauthors [15] distinguish syntactic, semantic and pragmatic quality of a model. Syntactic quality targets syntactic correctness, i.e., adherence to the language rules, which can be done, relatively easily, in automated way. In contrast, achieving semantic quality is extremely difficult and costly [15]. Lindland and coauthors reduce it to two main goals: validity and completeness\(^1\), and we agree with their consideration that semantic quality mostly entails manual inspection of the model. The goal is to validate that the model rightly captures the intended domain knowledge, which is a matter of mutual understanding between MEs and DEs (who usually are not the same people). Here, the third quality dimension of a model comes into play: pragmatic quality, which targets model comprehension, i.e., that all involved stakeholders understand the parts of the model that are relevant to them.

The issue is that a gap in knowledge, background and skills exists between MEs and DEs. In general we cannot assume that the DEs yield the expertise required to directly inspect and navigate models. Visual aids and animation can be of help, but ultimately the usual approach remains that MEs explain models to DEs in natural language (NL) and collect and process their feedback. Tool support in this step can be extremely valuable, especially when models are large and span over many different conceptual domains. Indeed, the MEs themselves without guidance may lack systematicity or introduce errors when translating from the model to NL.

We recently experienced the challenge of building and validating the model of a complex embedded Information System for urban mobility in the scope of the IPERMOB project (presented in Section 4.1). IPERMOB brings together experts from a variety of domains, spanning over software engineering, realtime embedded systems, wireless sensor networks, transport engineering, image processing, and data mining. This is the situation we depicted above: on one side we acted as the MEs and built in UML a common domain model through several interactions with the project partners, that on the other side acted as the DEs. This initial model needed to undergo semantic validation to check that we had rightly captured the DEs intents, but the knowledge of the IPERMOB domain was distributed among several partners, not all of which possessed a sufficient expertise to directly inspect the UML model we built.

To support the task of interviewing the IPERMOB experts, we developed a novel approach that uses state-of-art model-driven techniques to navigate the current version of a model and to automatically formulate questions in NL that the DEs can understand, so to stimulate their feedback and possibly identify

\(^1\)Model consistency checking in this framework is seen as a means to achieve semantic quality, and not a goal.
problematic parts of the model. This approach has been implemented into a practical tool and has been already usefully applied to the IPERMOB project.

Our approach stands between semantic and pragmatic model quality, and the scientific challenges that it poses stay at the convergence of different research directions: (a) approaches for manipulating and querying formal models, such as [26, 27, 21]; (b) approaches and techniques for UML model verification [17, 10] (c) approaches for expressing model properties in NL [12]; (d) cognitive approaches for eliciting requirements and extracting human knowledge [9, 5].

We are aware that our results are far from having answered all the challenges posed in the above research fields. Rather, we see our contribution as a humble start which from our experience looks promising and certainly deserves further investigation.

The paper is organized as follows: after an overview of the approach in the next section, we introduce the developed tool in Section 3. Section 4 describes the settings of the case study in the IPERMOB project, and the data we collected. Section 5 reports the results of the case study, while Section 6 discusses their validity. Finally, relevant related work is dealt with in Section 7, while Section 8 wraps-up the conclusions of the paper and outlines future work.

2. Overview

In this section we first overview a generic process of semantic model validation, in which we highlight the steps related to the communication between DEs and MEs. Then, we introduce our approach and tool within this process, to facilitate mutual comprehension during validation.

2.1. Model building and validation process

Through the UML Activity Diagram in Fig. 1 we conceptualize the steps involved in a domain modeling process, with an emphasis on the communications occurring between DEs and MEs.

DEs and MEs start interacting to establish a shared vocabulary and elicit a common high-level view of the input domain. After this phase, the MEs can
propose an initial model of the domain, which, as anticipated in the Introduction, has to be validated by the DEs. To this purpose, an iterative process starts. Along these iterations the MEs try to make the model comprehensible to DEs by generating and explaining appropriate model views. Based on such views, MEs request, collect, and analyze the feedbacks from the DEs. If the feedbacks reveal any mismatch between the domain model and the DEs intents, the MEs should understand the nature of the conflict and address it (e.g. proposing a new version of the domain model). This process can be iterated until all stakeholders are satisfied with the domain model.

2.2. Tool-supported querying

Although the process described above can never be fully automated, we can use model-driven technology to make it more systematic and more effective. In particular, the approach we propose here helps MEs in bridging the knowledge gap existing between them and the DEs.

With reference to Fig. 1, we support the automatic transformation of the model into views comprehensible by DEs (the dark-shaded activity in the figure), by automatically generating a tunable list of simple Yes/No questions that span over all model elements.

The idea is that at each iteration of the model validation process, DEs can interview the MEs by means of such automatically generated questionnaires, thus they are freed from the difficult and error-prone task of translating the model into NL descriptions.

We also generate for each question the expected answer based on the current version of the model. Therefore in the subsequent step of feedback analysis, MEs can also immediately identify possible conflicts and efficaciously focus the costly task of discussing feedbacks on those parts of the model to which they belong.

How we obtain the questionnaires is described in the next section.

3. Tool Architecture and Implementation

Our tool is called Mothia (MOdel Testing by Human Interrogations & Answers)\(^2\). Section 3.1 describes the overall architecture of Mothia, and Section 3.2 proposes the reference implementation we developed.

3.1. Mothia: the architecture

As described in Section 2, Mothia deals with the knowledge gap that separates MEs and DEs. Mothia takes in input a model representative of the domain and produces in output a questionnaire expressed in a language that is closer to the domain experts.

Fig. 2 (left side) depicts Mothia structure and how it interacts with the model (right side). The KnowledgeBaseGenerator is responsible to convert the

\(^2\)http://labse.isti.cnr.it/tools/mothia
input domain model into an internal representation describing both the domain entities, and their relations as facts. The InferenceEngine loads such facts, and the set of rules representing the semantics of the input modeling language. By querying the InferenceEngine, MOTHIA checks whether a given property is valid in the input domain model. Also, the InferenceEngine can return all the entities in the domain model satisfying a given property.

The QuestionnaireGenerator loads the configurations that drive the queries on the InferenceEngine, and the creation of the questionnaire. The configurations of the QuestionnaireGenerator are expressed in term of patterns and criteria.

A pattern represents a syntactical combination of elements in the input domain model that is considered relevant. In other words, MOTHIA uses patterns as a strategy to match a portion of the specific input domain model driven by the syntax. A pattern is defined in terms of an id, a signature (i.e. the name of the pattern with the list of its formal parameters), and a textual description. Also, a pattern composes one or more queries: each query implements a clause that select syntactical structures on the input domain models. A priority attribute defines the sorting on how the different clauses should be applied.

The criteria define the properties that MOTHIA uses in order to explore and query the input domain model. In other words, each criterion abstracts a type of question in the questionnaire. A criterion implements a question by composing patterns. Deductive questions are generated from criteria that infer true predicates from the input domain model. Alternatively, a question that corresponds to a false predicate in the input domain model is called, in this paper, a distractor\(^3\). Examples of the two types of questions can be found in

\(^3\)This is a slight abuse of language; in the literature, distractors are incorrect alternative answers to a multiple-choice question [23].
Section 3.2 (deductive questions are marked with $\checkmark$, distractors with $\times$).

Thus, MOTHIA expects that DEs answer “true” to the deductive questions and “false” to the distractors.

In general, querying the input domain model from the set of criteria leads to a questionnaire with an intractably large number of questions. To reduce the number of questions, the QuestionnaireGenerator uses the Filters component. For example, the Filters component can do a random selection of questions, or can target the questionnaire only towards some specific domain.

Finally, the EmittersContainer component deals with the output format of the questionnaire. It abstracts the functionalities of a configurable transformation engine and composes a set of specific emitters.

3.2. MOTHIA: the reference implementation

This section describes a reference implementation for MOTHIA. Specifically, we will present the tool by means of the illustrative scenario depicted in Fig. 3. Such scenario models the entities of an imaginary municipality and their relations: among the candidates of SomeParty, a City votes its Mayor, who in turn appoints the Councillors (up to 10).

MOTHIA is developed as an Eclipse plugin, using JAVA and EMF technologies, such as ECore and Acceleo. In particular, the current implementation can generate questions formulated in NL out from models defined by means of either ECore, or UML Class Diagram. SWI-Prolog\(^4\) is the InferenceEngine we chose (see Section 7); it is accessed by our plugin via the JPL library. The internal representation of the input model is therefore a collection of facts. Without loss of generality, the reference implementation considers only the most important modeling elements (additional ones can be easily added to the KnowledgeBaseGenerator if needed). Four types of facts are used:

1. class, to identify a class;
2. classAttribute, to identify an attribute of a class, and its properties;
3. inheritance, to identify a generalization;
4. association, to identify an association between two classes, and its properties.

In addition to the facts, we defined a set of Prolog rules implementing the semantics of the input modeling language (i.e. UML Class Diagrams, ECore Diagrams). For example, the InferenceEngine can load a semantic rule implementing the application of the Liskov substitution principle [16]: in all the questions where a super-class is used, it is possible to substitute it with any of its sub-classes. Thus, with respect to Fig. 3, any question concerning the relation citizen from City to Person can be also rephrased in terms of either Councillor or Mayor, since they are persons too. Note that the semantic rules we implemented within MOTHIA can be extended or overloaded with other semantic interpretations (e.g. domain specific semantic).

\(^4\)http://www.swi-prolog.org/
The previous four facts are the basic blocks that allow patterns to be built. As previously stated, questionnaires are derived by using patterns and criteria. A pattern represents a syntactical structure in the model. The InferenceEngine implements a pattern as a SWI-Prolog rule where the signature is the head, while a query is the body. It is interesting to note that a pattern itself can be used as a building block for other patterns, to allow an arbitrarily complex structure. The most relevant patterns defined in the reference implementation are described in Table 1.

For example, three noteworthy patterns can be found in Fig. 3:

- the *marriage* pattern identifies the inheritance structure composed by classes *Person*, *Authority* (i.e. father and mother respectively), *Councillor* (i.e. a step-son) and *Mayor* (i.e. a son);
- the *subclassesAssociation* pattern identifies the inheritance structure composed by classes *Authority*, *Party* (i.e. super-classes), *Mayor* and *SomeParty* (i.e. sub-classes);
- the *indirectAssociation* pattern identifies the chain of associations composed by classes *Councillor*, *Mayor* and *SomeParty*, and another chain composed by classes *Councillor*, *Mayor* and *City*.

As described in Section 3.1, a criterion combines patterns together in order to generate specific kind of questions. In our reference implementation, we defined some criteria from common modeling slips while others were inspired by the literature. In particular, in [6] and [22] the respective authors discussed some relevant syntactic structures to look for into a domain model. They also presented some strategies in order to generate questions. For example, we took inspiration from [6] for questions about both the direction and the type of a relation (e.g. “What is the relation between *EntityA* and *EntityB*?”), “Can an *EntityA* have exactly one and only one *EntityB*?”). Also, we referred to [22] to generate questions about the generalization (e.g. “Is *EntityA* an *EntityB*?”). Finally, we defined complex criteria combining some of these generation strategies.
<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributeNotAssociation</td>
<td>matches class attributes that are not used as association member ends</td>
</tr>
<tr>
<td>chain</td>
<td>matches all sort of information about a class, to rank its relevance (attributes, associations from and to the class, multiplicities, parent and child classes)</td>
</tr>
<tr>
<td>family</td>
<td>matches a grandfather super-class with sub-classes, some of which (fathers) have other sub-classes (sons)</td>
</tr>
<tr>
<td>indirectAssociation</td>
<td>matches all classes that are reachable from a certain class by navigating associations, with a configurable amount of intermediate classes</td>
</tr>
<tr>
<td>marriage</td>
<td>matches a class (son) that inherits from two super-classes (father and mother), one of which has other sub-classes (step-sons)</td>
</tr>
<tr>
<td>multiplicityRandom</td>
<td>generates a positive random integer between lower and upper bounds of an association multiplicity</td>
</tr>
<tr>
<td>siblings</td>
<td>matches sibling classes that inherit from the same super-class</td>
</tr>
<tr>
<td>subclassesAssociation</td>
<td>matches an association between two classes that are each derived from a distinct super-class</td>
</tr>
<tr>
<td>superclassesAssociation</td>
<td>matches an association between two classes that have each a distinct sub-class</td>
</tr>
<tr>
<td>unrelatedClasses</td>
<td>matches two classes in a super/sub-class relationship and a third class, with no associations to and/or from the previous two classes</td>
</tr>
</tbody>
</table>

Table 1: Summary of patterns
The main differences between our approach and such works are discussed in Section 7. In the following, some definitions of criteria are shown.

Let us consider a simple criterion crit1 that uses only the marriage pattern. MOThiA generates a simple deductive question considering the inheritance relationship between a class (e.g., son) and its parents, while a distractor can fake such relationship matching a class (e.g., step-son) with the wrong parent class.

For example, with respect to the illustrative scenario depicted in Fig. 3, among the others crit1 produces:

✓ Is Councillor a kind of Person?

✗ Is Councillor a kind of Authority?

A complex criterion can be formulated by combining more than one pattern by means of a join operation (i.e. combining together the overlapping model elements). For example, let us define crit2 as the criterion that looks for a class A that is acting: as a sub-class in the pattern subclassesAssociation; as son in the pattern marriage; and also as an intermediate class in the pattern indirectAssociation. By querying the model in Fig. 3 with crit2, class A is instantiated with the entity Mayor. In the following an example about the type of questions that can be generated using crit2:

✓ Can the relation from Mayor to Councillor exist, where Councillor is appointed by Mayor?

✓ Can the relation from City to Mayor exist, where Mayor is a citizen of City?

✓ Is Mayor a kind of Person?

✗ Can the relation from SomeParty to Councillor exist, where Councillor is appointed by SomeParty?

✗ Is Mayor a kind of Party?

✗ Can the relation from SomeParty to an unlimited number of Mayors exist, where a Mayor is a candidate of SomeParty?

To summarize, MOThiA allows its user (i.e., the MEs) to add arbitrary complexity when looking for elements in the input model, just by combining together elementary building blocks. Even more, it allows MEs to define what such building blocks are.

Both patterns and criteria conform to their respective metamodels, which we defined. MOThiA stores them as XMI encoded files. Note that in this paper our goal is to present the MOThiA framework and investigate its effectiveness in practice through its application to a real case study. This is the reason why we do not further devote specific attention to investigate which are the most common modeling mistakes and which patterns and criteria should be defined to identify them; although we believe that this is an important topic, this is
outside the scope of this paper. Nonetheless, we put an effort in creating a reference implementation that supports extensible patterns and criteria. MEs using Mothia can include our default set of patterns and criteria. Besides, they can simply enhance and customize the QuestionnaireGenerator by extending the existing patterns and criteria or by defining their owns. Such customization can be easily implemented by MEs as Mothia refers to many popular technologies running under the Eclipse Modeling Framework Project.

Filters are Java classes that implement a specific filtering strategy, in order to reduce the size of the questionnaire after all questions have been generated. Each filter must define three filtering functions working at different levels of data aggregation, precisely: a reasoning category (deductions or distractors), a criterion, a whole questionnaire. This allows for a fine grained control over which questions make their way to the output. Two types of filter have been implemented: a random filter and a score-based filter that exploits class relevance into the model. For example, only the $x$ most important questions (by applying the score-based filter) are kept from each criterion, then the questionnaire has a $y$ overall questions cap and a random filter cuts the exceeding ones.

Finally, two output emitters have been implemented within the EmittersContainer: an ECore emitter, to visualize the raw composition of the questionnaire, and a XHTML emitter, to publish it as a web form.

4. Case Study

This section presents an experience report in applying the approach described to a real case study.

Specifically, the main goal of our experience was to stimulate further feedback from DEs with respect to the specification of a domain model. For this purpose, we applied Mothia as a means for DEs and MEs to communicate with each other without using the modeling notation, but only through NL. We also wanted to estimate the performance of such approach in revealing errors in the IPERMOB domain model.

In the following, we define first the reference scenario within IPERMOB. Then, we describe the settings and the activities that we conducted. The results of these activities are discussed in the next section.

4.1. Scenario

Our approach has been developed and validated in the context of the IPERMOB project.

The goal of IPERMOB is to realize an integrated multi-tier information system to improve urban mobility. From a technical point of view, the project aims at combining low-cost wireless technology and efficient image processing techniques to collect, analyze, and elaborate information related to traffic flow.

\footnote{http://www.eclipse.org/modeling/emf/}
in order to offer real-time advanced information to citizens and management functionalities to local governing authorities.

As anticipated in the Introduction, a distinctive characteristic of the project is its interdisciplinary nature, which brings together experts from a variety of domains (software engineering, real-time embedded systems, wireless sensor networks, transportation engineering, image processing, and data mining). Since the early stages of the project, the project partners decided that a common domain model should be constructed. Because of their heterogeneous backgrounds, this was deemed necessary to establish a shared understanding of the key concepts coming from each of the sub-domains.

The resulting domain model is structured in five packages, each containing concepts from a particular sub-domain or area of concern, and includes 47 classes, 54 associations (of which 14 compositions), 16 inheritance relationships.

4.2. The Experts’ Answers

In this study we used MOTHIA to generate 14 questionnaires. This corresponds to the number of DEs from different partners, who were available to carry out this model validation exercise on behalf of the PERMOb project. Each questionnaire covered all the areas of the domain model and included a mix of both deductive and distractor questions.

For each question produced by MOTHIA, we proposed to the interviewee a multiple-choice answer based on four possible, and exclusive, items: (1) Yes, (2) No, (3) “I’m not an expert of this particular subject” (NE), and (4) “I’m an expert of the subject but I’m not sure what to answer” (DK).

Of course we did not know in advance which and how many errors were actually present in the domain model as it was originally formulated. In other words, we had no reference baseline to measure how effective MOTHIA was in discovering faults in the model.

As is done for instance in mutation testing, we hence decided to mutate the input domain model and create a new slightly different version by injecting some artificial faults. This ensures that even if errors were not present in the model, we could still evaluate MOTHIA effectiveness on such injected errors; and, even if some real errors were detected (as it happened), we could not estimate which percentage they were of all those present. Table 4.2 reports the faults we used in order to mutate the domain model; some of them were taken from the literature on mutation testing for both models and model transformations [20].

In order to avoid bias, for each fault we randomly selected the portion of the domain model to mutate. Note that differently than mutation testing of code (were one mutant only includes a mutation), all the five faults were simultaneously inserted in the same model.

7The patterns and the criteria implemented are available at http://labse.isti.cnr.it/tools/mothia.
### Table 2: Injected Faults

<table>
<thead>
<tr>
<th>Fault Id</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Create association between two (previously unassociated) classes (referred to as CACA in [20]).</td>
</tr>
<tr>
<td>F2</td>
<td>Replace element with parent or sibling (CCCR in [20]).</td>
</tr>
<tr>
<td>F3</td>
<td>Flip direction of monodirectional association.</td>
</tr>
<tr>
<td>F4</td>
<td>Turn simple association into composition and viceversa.</td>
</tr>
<tr>
<td>F5</td>
<td>Change cardinality (set lower bound to 0 or 1, set upper bound to 1 or (\infty)).</td>
</tr>
</tbody>
</table>

From the mutated domain model, MOTHIA generated the questionnaires made of 30 questions each. Each questionnaire mixed randomly questions affected by the artificial faults (in the following faulty questions), and questions not affected (in the following genuine questions). The two types of questions could not be distinguished by the DEs, who were unaware that faults has been ad hoc introduced.

Finally, each questionnaire was administered to DEs of one or more subdomains of IPERMOB. The exercises have been carried out independently in small separate groups (just depending on the availability of the DEs. Answers to genuine questions are summarized in Table 3(a), while results for the faulty questions are given in Table 3(b). For the purposes of reporting, a question can be regarded as a test on the input model: the test is **Passed** if the answer of the DE matches the answer expected by MOTHIA, **Failed** otherwise. Considering faulty questions, they are considered “killed”\(^8\) (K column in Table 3(b)) if the DEs provide a **Yes** or **No** answer different from the expected (UN-expected column), or a **DK** answer. MEs should always pay attention when DEs assert their expertise on a question but answer **DK**. It is something worth to investigate deeper, because could be due either to a model error that disorients the DE or to a malformed question. When the answer is **DK**, then a faulty question is considered killed for this very reason, as the consequent analysis will lead to the discovery of the (injected) fault.

After collecting the answers, we performed an ex-post analysis on the questionnaires, during which we discussed the wrong answers with the interviewees. Of course, in this analysis, we only considered the questions that were not related to the artificial faults we injected, which were analysed and interpreted separately, as discussed in Section 5.

#### 4.3. The Experts’ Feedbacks

In the case study we also aimed at collecting feedback from the interviewees about the questionnaires generated by MOTHIA. To this end, after they filled

---

\(^8\)In mutation testing, a mutant is said to be killed when its outcome on a test case is different from the outcome of the original program on the same test case.
(a) Results for the genuine questions.

<table>
<thead>
<tr>
<th>Questionnaire</th>
<th>#Pass</th>
<th>#Fail</th>
<th>#NE</th>
<th>#DK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>10</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>11</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>5</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>8</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>13</td>
<td>8</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>9</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>6</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>6</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td><strong>176</strong></td>
<td><strong>103</strong></td>
<td><strong>53</strong></td>
<td><strong>10</strong></td>
</tr>
</tbody>
</table>

(b) Results for the faulty questions.

<table>
<thead>
<tr>
<th>Questionnaire</th>
<th># Faulty Questions</th>
<th>#UN</th>
<th>#DK</th>
<th>(UN+DK)</th>
<th>#NK</th>
<th>#NE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td><strong>78</strong></td>
<td><strong>26</strong></td>
<td><strong>12</strong></td>
<td><strong>38</strong></td>
<td><strong>27</strong></td>
<td><strong>13</strong></td>
</tr>
</tbody>
</table>

Table 3: The Experts’ Answers
their questionnaire, we asked DEs to fill out a form, which was structured in two main parts. The first part aimed at evaluating the extent to which the generated questionnaires covered the domain (evaluation questions 1–4); the second part aimed at evaluating the usefulness of the questionnaire as perceived by the DE (evaluation questions 5–7). The questions of the evaluation form are presented in Table 4(a).

To evaluate how the questionnaires generated by Mothia were perceived by the domain experts, in question 5 we proposed to express a forced-choice preference by means of a four-point Likert scale [14]. Specifically, the domain experts evaluated our approach by ranking the questionnaire with a score from \(A\) (e.g. Useful, effective, etc.), to \(D\) (e.g. Useless, waste of time, superficial, etc.). According to Table 4(a), most of the domain experts had a positive impression about the questions and the way they supported the cooperation and the sharing of knowledge among the different stakeholders within IPERMOB.

5. Discussion

This section discusses the results of the case study presented in the previous section, and attempts an interpretation of those results.

Note that, in this paper we report our experience in modeling and validating a domain model for a real case study, in the interaction with real DEs that were not MEs and the vice-versa. The difficulty and costs of experimenting with professionals are a well known issue in empirical software engineering [28]. Moreover, as a warning, we notice that the interpretation of the results we give in the following is valid for the case study IPERMOB. Thus, even though the results of this experience look promising, it is not in the scope of the paper the formulation of generally valid considerations (these are left to future work).

As described in Section 4, our evaluation was conducted by means of two-phase questionnaires: questions for the first phase were generated through Mothia (see Section 4.2), while questions for the second phase (see Section 4.3) were formulated by us, and aimed at collecting feedback about the questionnaire of the first phase.

5.1. Was Mothia effective in revealing modeling errors in IPERMOB?

Section 4.2 describes how we measured the performance of Mothia in terms of its ability to detect model faults.

Although we do not have extensive empirical results, we believe the outcome of our experience could be positively interpreted with respect to the effectiveness of Mothia in helping to reveal modeling errors. Concerning the faults that we deliberately introduced, the questionnaires were able to detect all types of faults (in at least one questionnaire), and to kill almost half of the faulty questions (58% if we exclude questions in which the MEs declared themselves not expert).

More than this, we were encouraged by the results in the ex-post analysis of wrong answers not caused by the artificial faults. The discussion stimulated by such conflicts allowed us to detect several other faults that had slipped into
(a) Responses to the feedback-collecting questions.

<table>
<thead>
<tr>
<th>Evaluation Question</th>
<th>Responses</th>
</tr>
</thead>
</table>
| 1. Did any question stimulate you to think about aspects of the domain that you had not considered before? | Yes: 9  
No: 5 |
| 2. If yes, which ones? | Feedbacks: 9  
No Feedback: 0 |
| 3. In your opinion, were the questions heterogeneous enough? | Yes: 12  
No: 2 |
| 4. In your opinion, were the questions covering most of the concepts within the domain? | Yes: 14  
No: 0 |
| 5. Which is your overall evaluation about the previous questionnaire? | A: 6  B: 6  C: 2  D: 0 |
| 6. Please, explain your answer | Feedbacks: 13  
No Feedback: 1 |
| 7. Please, suggest us how we can improve the questionnaire | Feedbacks: 9  
No Feedback: 5 |

(b) Evaluation Metrics for RQ2

<table>
<thead>
<tr>
<th>Questionnaire</th>
<th>% Efficacy</th>
<th>% Adequacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>88.89</td>
<td>75</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>96.15</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>82.61</td>
</tr>
<tr>
<td>4</td>
<td>86.36</td>
<td>91.67</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>95.65</td>
<td>95.83</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>76</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>64</td>
</tr>
<tr>
<td>9</td>
<td>92</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>84</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>88</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>84</td>
</tr>
<tr>
<td>13</td>
<td>94.12</td>
<td>70.83</td>
</tr>
<tr>
<td>14</td>
<td>94.74</td>
<td>76</td>
</tr>
<tr>
<td>Mean</td>
<td>96.55</td>
<td>84.58</td>
</tr>
</tbody>
</table>

Table 4: The Experts’ Feedbacks
the model. Due to the heterogeneity of IPERMOB experts, the interconnections among concepts of different sub-domains (i.e. the five model packages) in fact proved to be particularly critical. The faults we discovered amount at 16 major faults (i.e. associations, and generalizations either deleted or introduced) plus several other minor issues, and included mostly ambiguities in relating a same entity from different domains, as well as errors that were introduced in interpreting DEs exigencies. Notably, some wrong answers highlighted the need to introduce new model elements (classes and associations).

The last result seems to hint at the promising perspective that, with reference to semantic quality [15], not only MOTHIA can be a means to check validity, that is what we had expected, but could also help to check model completeness. We plan to devote specific experiments to this purpose.

5.2. Was MOTHIA helpful in reducing the gap in IPERMOB?

The possible answers to questionnaire described in Section 4.2, can be partitioned into two groups: the answers that classify the responder as an “expert” of the topic treated by the question (i.e. Yes, No, DK); and the answers that classify the responder as “not expert” with respect to that specific topic (i.e. NE). To assess whether MOTHIA is effective in formulating questions we refer to the number of responders indicating themselves as “experts”.

Hence, we define the efficacy of a questionnaire as a metric that compares the number of the questions on which the responder was actually able to express an opinion (i.e., answering Yes, or No), over the total number of questions in the questionnaire for which the interviewees classified themselves as “experts” of the topic:

\[
\frac{\#\text{Pass} + \#\text{Fail}}{\#\text{Total} - \#\text{NE}} \times 100
\]

Since in this case study the questionnaires covered several sub-domains, we had to accept that a questionnaire might not be perfectly tailored to the expertise of a particular interviewee. This explains why we admit a NE answer. Thus, when considering the efficacy of a questionnaire, we should also take into account a measure of the adequacy of that questionnaire w.r.t. the specific interviewee. We define the adequacy metric for a questionnaire and a responder as the number of questions where the responder identified him/herself as an “expert”, over the total number of questions contained in the questionnaire:

\[
\frac{\#\text{Pass} + \#\text{Fail} + \#\text{DK}}{\#\text{Total}} \times 100
\]

According to Table 4(b), most DEs were able to express an opinion on the modeled elements starting from the questions generated using MOTHIA. Thus, considering also the mean adequacy level we got from the analysis of the questionnaires, we believe that MOTHIA can actually contribute in reducing the gap between domain experts and modeling experts.
5.3. Was Mothia perceived as useful in IPERMOB?

To evaluate the opinions of the DEs with respect to the perceived usefulness of Mothia, we examine the responses to the questions described in Section 4.3. 9 DEs out of 14 answered that the questionnaire generated by Mothia made them think about aspects of the domain that they had not considered before (see Table 4(a), evaluation question 1). In addition, some of the experts spontaneously declared to be interested in further investigating those aspects. These positive results were also clearly confirmed by the positive answers to the evaluation questions 3 and 4.

Furthermore, in one case, even before our ex-post interviews, a DE realized that she had introduced an ambiguity in the domain description that she had previously given to us.

Overall, we conclude that IPERMOB DEs perceived Mothia as useful.

6. From the experience to the experiment

Our studies must be considered in light of some potential threats to validity by which our case study must be considered closer to an experience rather than to an experiment.

Concerning the effectiveness of Mothia in generating questionnaires that reveal modeling errors, although we derived them from a study of literature and attentive consideration, the artificial faults we introduced might not be representative of real faults. Nevertheless, our analysis was also backed by the detection of real faults. In addition, to avoid biasing DEs towards negative answers, responders were not aware that we had deliberately introduced some faulty questions.

Tackling the gap between DEs and MEs, we are not deep experts of question lexicalization, thus Mothia questions might not be always meaningful to DEs. It was to cope with this issue that we introduced the answer of type DK. We also fear that the positive feedback collected in Section 4.3 might be caused by the existing work relationship. To minimize such bias, we did not tell the DEs that the questionnaires were aimed at evaluating our approach, but only at validating the IPERMOB model.

Finally, even though we believe that the results of this first application of Mothia look promising, the conclusions of our experience are far from being generalizable, as our validation was based on one model only and a relatively small number of interviewees. Indeed, we plan to make more extensive validation studies in the future. In addition, all the experiment settings, such as: the number of questions in each questionnaire, the distribution of questions over the criteria, the number and types of artificial faults, and so on, were decided ad hoc, although of course good sense and reasonings were applied. Next studies will be adjusted based on these first results.

7. Related Work

The research in this paper is related to work in several fields.
Quality of models: Bolloju and Leung [3] tried to identify the most typical set of errors frequently committed in UML modeling, and discussed how they affect the quality (syntactic, semantic and pragmatic) of models developed. With regard to class diagrams, the errors they listed are analyzed by our criteria.

Shanks et al. [25] focused on the importance of early, effective and efficient validation to avoid the propagation of defects to subsequent system design. They argumented that ontologies could help choosing the domain grammar, understanding the phenomena represented in conceptual models and make sense of ambiguous semantics.

Syntactic Realization: The problem of generating plausible NL questions has not been tackled in our paper: we used a naive approach, as this is not in the scope of our research. There exist tools that address this problem, referred to as the problem of “syntactic realization” 9.

Kroha and Rink [13] described a text generation method for requirements validation, in which they automatically generate a textual description of a UML diagram (limited to use case, sequence or state diagrams). Similarly to our approach, their purpose is to bridge the gap between DEs and MEs, but they only translate the model to NL rather than testing also wrong assertions on it.

Cabot et al. [4] implemented a transformation from UML class diagrams (with possible OCL constraints) to SBVR, a OMG metamodel for documenting the semantics of business vocabulary, business facts and business rules. They also paraphrased the result in Structured English and RuleSpeak English notations to facilitate the discussion of the diagrams with the stakeholders. Again, they only provide a NL translation of the model.

Dalianis [6] proposed several categories of NL discourses on a conceptual model, in order to ease its representation and validation. The input model is initially converted to a Prolog knowledge base to be queried. The DE then actively investigates the knowledge base by asking questions (chosen from a limited set) that involve model entities, and gets a NL reply with a selection of relevant information. The main difference from our approach is the proactivity of the DE, that chooses which areas of the model to inspect and may miss some problems.

Validation using multiple-choice questions: Mitkov et al. [19] proposed a technique to generate multiple-choice tests starting from electronic questions. The approach of Mitkov differs from ours in several aspects: their goal is to produce tests that are useful to test the expertise of the responders in the area covered by the questionnaire; they use NL analysis to extract knowledge from a corpus and to formulate questions related to that corpus. Our goal is to test the model, not the responder’s knowledge.

Hoshino and Nakagawa [11] used machine learning techniques to generate questions of fill-in-the-blank type, in order to allow the testing of any kind of knowledge, rather then for a specific purpose like our approach.

---

Papasalouros et al. [22] proposed the automatic creation of NL multiple choice questions from domain ontologies. The goal is to create a questionnaire for educational purposes, rather than validation, since the input knowledge base is considered as correct. While we focused on simpler Yes/No questions, strategies to create distractors can be seen as the equivalent of our criteria.

**Model querying:** During the last years, Prolog has been proposed and studied in literature as an analysis engine for software models. Specifically, in [26] the author discussed why Prolog should be considered a valid alternative to other widespread and more common querying languages such as SQL or OCL. In [21] the authors measured and compared the model querying performance of Prolog and OCL; and more recently, Prolog has been also proposed as a platform for a high level programming interface to query UML models [27].

Such discussions are stimulating several debates into the model-driven communities; we here maintain a neutral position on which is the best approach for model querying. What we consider important in this work, is having an inference engine able to deal with the semantics associated to the input modeling language. Given the well established tradition of the Prolog inference engine, we adopted Prolog in implementing Mothia, but the overall architecture proposed in Fig. 2 does not prevent from developing other implementations based on other querying languages.

8. Conclusions and future work

Domain models are an essential asset in MDE; their definition involves on the one side MEs who master the modeling technology but might be agnostic of specific domains concepts, and on the other the DEs who know all tiny details of the domain but do not understand the model representation. The knowledge gap between MEs and DEs is a well known challenge of MDE and different research directions are taken to fill it.

In this work we contributed to such effort with an approach, and its supporting tool Mothia, to interview DEs via NL questionnaires automatically generated from the domain model. The approach has been applied to a real–life model validation problem in the context of the IPERMOB research project. The results of the case study within IPERMOB are by no means conclusive and certainly a more extensive evaluation is necessary. Nonetheless, our initial studies provide some evidence to confirm that our approach is viable, applicable to real modeling problems, and deserves further investigation.

Future research work will include experimentation on other models with more interviewees and expanding the set of artificial faults, focusing on understanding how patterns of common modeling slips affect the modeling activity. We also plan to further automate the process illustrated in Fig. 1, by automating the analysis of DEs responses so to map wrong answers on the model elements they refer to, and by also exploring more systematic ways to deal with inconsistent answers from different interviewees, which we currently handled manually.

As technical future work, we plan to improve the lexicalization module, so to reduce the effect of poor syntactic realization on the understandability of
questions. In order to simplify the definition of custom patterns and criteria, we are considering the integration of more sophisticated Prolog libraries (such as [27]). We are also planning to widen the range of accepted UML input models, to include Use Case Diagram, Activity Diagram, and Sequence Diagram. Due to the modularity of the tool architecture, only a few modifications are required: (a) extension of the KnowledgeBaseGenerator to deal with such diagrams and create appropriate facts; (b) definition of new semantic rules, if any; (c) creation of patterns and criteria that use the new facts and rules.

An interesting direction for future research concerns extending MOTHIA in order to support the validation of model completeness. In other words, we are investigating how to generate questions that stimulate discussions about elements that do not exist in the model yet. To this end, the information contained in the input model itself is not enough, and it should be supported by additional knowledge bases (e.g. domain ontologies). Specifically, we are planning to exploit semantic relations among sets of cognitive synonyms (i.e. synsets) [8] that include model elements. Unfortunately, such ontologies are seldom available, and definitely they do not cover every possible domain. However, in a preliminary study under development, we are planning to implement a domain-independent approach integrating MOTHIA with a lexical database such as WordNet [18]. For example, by means of such integration MOTHIA would generate questions using entities that are not strictly defined within the domain model, but that WordNet defines as synonym for another entity contained in the domain model.

Acknowledgements

This paper has been supported by the Regione Toscana Project POR CReO FESR: IPERMOB, and partially by the European Project FP7 IP 216287: TAS3.

References


