Towards a Decision Making Framework for Model Transformation Languages

Soroosh Nalchigar
Department of Computer Science
University of Toronto
Toronto, Canada
soroosh@cs.toronto.edu

ABSTRACT

Model transformations play increasingly important role in Model Driven Engineering (MDE). A wide variety of tools and languages are generated and released for model transformations. A challenging task of software engineer is to choose a particular model transformation language, given a set of alternatives, that is best suited for non-functional requirements of the context, which are mostly intangible and difficult to measure, if not impossible. This paper proposes a decision making framework that aids software engineer to select most proper model transformation language, given non-functional requirements (of which some are conflicting). The proposed framework is based on Fuzzy-Analytic Hierarchy Process (AHP) technique, and uses relative degree of importance of non-functional requirements for the given context in terms of fuzzy linguistic terms. The output is ranking of model transformation languages. The approach is implemented and its applicability is illustrated in three different cases.

Categories and Subject Descriptors

D.2 [**Software**]: Software Engineering

General Terms

DESIGN, LANGUAGES

Keywords

Model Transformation, Non-Functional Requirement

1. INTRODUCTION

Model Driven Engineering (MDE) is a software engineering discipline in which models are the prime artifacts and play a central role throughout the entire development process [35, 37]. MDE combines domain specific modeling languages for modeling (software) systems, and model transformations for, among others, synthesizing them [4]. A Domain Specific Language (DSL) is a language designed to provide a notation tailored toward a problem domain, and is based on the relevant concepts and features of that domain. By providing tailored notations, a DSL offers substantial gains in productivity and even enables end-user programming [29, 49]. Model transformations are used to automatically transform models expressed in these DSLs into different models [4], where source code could be considered as a model too [35]. Performing a model transformation, which is process of automatic generation of one or more target models from one or more source models, require a clear understanding of the abstract syntax and the semantics of both the source and target [40]. Model transformations could be used for different reasons and intents, eg. to extract different views from a model (query), add or remove detail (refinement or abstraction), generate code from model (synthesis) [3].

Model transformations play an important role in MDE since they automate complex, tedious, error-prone, and recurrent software development tasks [7]. They are core mechanism of MDE for building software from design to code, and hence have a significant impact on software development process [9]. Since model transformations play a critical role in this process, their non-functional requirements and evaluation of them are of great importance. These requirements could be the reason for accepting or rejecting a tool or language, while deciding about the most suitable one. Therefore, selecting the most suitable model transformation language based on non-functional requirements is a critical and challenging task for software developer [35].

Previous works have addressed the problem of non-functional requirements in model transformations. Some have proposed a set of non-functional requirements that a transformation language or tool should satisfy (e.g. [5, 35, 40]). They argue that these requirements are of particular interest to industrial users and are useful to be considered as decision making criteria. On the other hand, some works have tried to tackle the problem by defining quality attributes and metrics for model transformation artifact itself (e.g. [4, 37, 47]). They believe that these attributes and metrics could be used for developing quality assurance techniques for model transformations. While these two groups of works provide a proper base for understanding various non-functional requirements of model transformations, there remain many challenges for decision making, and in particular selection of most suitable model transformation language given a set of non-functional requirements.

The goal of this research is to propose a decision making framework for selecting most suitable model transformation language (among a set of alternatives) given non-functional

requirements (of which some are conflicting). The proposed framework is based on Fuzzy-Analytic Hierarchy Process (AHP) technique, and uses relative degree of importance of non-functional requirements for the given context in terms of fuzzy linguistic terms. The output is ranking of model transformation languages. The proposed framework uses fuzzy triangular numbers for pairwise comparison of non-functional requirements, since these requirements are difficult, if not impossible, to measure in term of precise, crisp numbers. The results of this research could be used to choose, for a given context and set of non-functional requirements, the language or tool that is best suited for those requirements. Given the lack of studies on decision making of model transformation languages, we believe that this work is of interest to people doing research in related areas as well as industrial users.

The rest of this paper is organized as follows: Section 2 reviews the related works on non-functional requirements of model transformations and proposes a comprehensive list of them. Section 3 performs a pairwise comparison of two model transformation languages with regarding to non-functional requirements of Section 2. Section 4 describes the steps of proposed approach. Section 5 illustrates applicability of the approach for three different cases. The paper ends at Section 6 with some concluding remarks.

2. RELATED WORKS

In this section we review the previous works with a focus on non-functional requirements of model transformation. The goal in this section is to build a comprehensive list of those requirements by reviewing previous published works.

Amstel et al. (2008) proposed a set of eight quality attribute for model transformation (in ASF+SDF term rewriting system) and a set of metrics for assessing them. Quality attributes were understandability, modifiability, reusability, reuse, modularity, completeness, consistency, and conciseness. Metrics also were presented within four categories: size (e.g. number of functions), function (e.g. number of equations and conditions per function), module (e.g. number of library modules), and consistency (e.g. number of variables per type). They discussed relationships between metrics and quality attributes. It should be mentioned that some of the quality attributes they presented were adopted from previous studies which proposed them for assessing quality of software designs [47]. Similar metrics have been proposed by Vignaga (2009) to assess quality of ATL model transformations [48]. In another work, Amstel (2010) proposed two definitions for two different views on quality of model transformations (internal and external) and presented some examples of quality assessment techniques for model transformations (direct and indirect) [4].

Mens and Gorp (2006) differentiated functional and non-functional (or quality) requirements of model transformation languages or tools and proposed usability, usefulness, verbosity, conciseness, scalability, extensibility, interoperability, acceptability and standardization as main non-functional requirements [35]. Sendall and Kozaczynski (2003) classified different approaches to model transformations and proposed a set of desirable characteristics for a model transformation language, among others easy to understand, precise, unambiguous, concise, easy-to-modify, complete, and graphical representation [40].

Aziz (2011) evaluated quality of four model transformation technologies (ATL, IBM transformations, Acceleo, and Java APIs) for model to model and model to text transformation in a case study in Ericsson AB. His quality model included usability (understandability, learnability, and operability), maintainability (analyzability and changeability), functionality (suitability and accuracy), and scalability as quality attributes (and their sub characteristics) [5]. Mohagheghi and Aagedal (2007) presented quality goals in MDE and argued that the quality of models is affected by quality of modeling languages, tools, modeling processes, the knowledge and experience of modelers, and quality assurance techniques applied. They presented related works on these factors and mentioned some quality attributes including understandability, modifiability, reusability, reuse, extensibility, interoperability [37].

Table 1 presents a list of non-functional requirements of model transformations mentioned in previous studies. Although these works have contributed significantly and defined the non-functional requirements, however there is a lack of systematic way of comparing and choosing the transformation language among a set of possible options. This paper fills the gap by proposing a decision making framework for selecting the most suitable model transformation language, given non-functional requirements.

3. COMPARISION OF TWO LANGUAGES

In this section, we compare two famous model transformation languages with regarding to non-functional requirements described in previous section. In particular, we are interested to see how each language affects non-functional requirements. The first language is ATLAS Transformation Language (ATL) [27] and the second one is Attributed Graph Grammar (AGG) [44]. Here the comparison is done by reviewing the previous related papers, and comparing capability of the language to satisfy the non-functional requirements. The result of this comparison will be part of decision making mechanism proposed in this paper.

ATL includes mix on imperative and declarative constructs. Although the mix make the language powerful and compact, but it is non- intuitive from understandability point of view and hence it negatively affect understandability of the language [5].Göknil et al. (2008) stated that ATL reduces modifiability of model transformations because it implements the source pattern in multiple rules and helpers (i.e. due to the decomposition in multiple constructs). All rules and helpers should be updated even for a minor constraint change in source pattern definition [22].

ATL is textual [5] and supports modularity and allows packaging rules into modules. A module can import another module to access its content [14,23]. ATL is capable of managing complex models because of its imperative language constructs and use of helper functions [41].

According to [46], ATL supports Higher-Order Transformations (HOT), whcih are model transformations that analyze, produce or manipulate other model transformations. Moreover, it is compliant to relevant standards such as UML, MOF (Meta Object Facility), and XML. An example of MOF to UML transformation is given in [24]. Also, [25] shows how to bridge XML, Grafcet, Petri net, and PNML by using ATL.

Randak et al (2011) extended ATL for natively supporting UML profiles in transformations. They provided an extended ATL syntax comprising keywords for handling UML

Non-functional Requirement	Definition	Author(s)
Understandability (UN)	The amount of effort required to understand a model transformation.	[3–5, 37, 40, 47]
Modifiability (MF)	The extent to which a model transformation can be adapted to provide different or additional functionality.	[4, 5, 37, 40, 43, 47]
Reusability (RY)	The extent to which (a part of) a model transformation can be reused by other model transformations (as-is reuse).	[4, 37, 40, 43, 47]
Reuse (RE)	The extent to which a model transformation reuses parts of other model transformations. It is considered as a quality attribute since it is good practice to reuse tested units.	[5, 37, 40, 47]
Modularity (MD)	The extent to which a model transformation is systematically structured (every model in a model transformation has its own purpose).	[4, 47]
Conciseness (CS)	The extent to which a model transformation does not include superfluous information.	[35, 47, 47]
Verbosity (VB)	The transformation to introduce extra syntactic sugar for frequently used syntactic constructs.	[35]
Performance and Scalability (PS)	Ability of language or tool to cope with large and complex transformations or transformation of large and complex software models without sacrificing performance.	[5, 35, 43]
Extensibility (EX)	The ease with which the tool can be extended with new functionality.	[35, 37]
Interoperability (IN)	The ease with which the tool can be integrated with other tools used within the (model-driven) software engineering process.	[35, 37, 43]
Standardization (ST)	The transformation tool should be compliant to all relevant standards (e.g. XML, UML, MOF).	[35, 37]
Visualization (VS)	Whether the transformation technology provides visual specifications of transformation.	[5, 40]

Table 1: Summary of non-functional requirements of model transformations in previous studies

profiles which is reduced by a preprocessor based on a HOT again to the standard ATL syntax [38]. Moreover, Mens et al. (2006) argue that the AGG tool is extensible in the sense that its internal graph transformation engine, which is implemented in Java, can be extended freely to cover a wide variety of different applications [36].

AGG is better than ATL with regarding to amount of code or models that a programmer or modeler needs to specify in order to make an executable solution [23]. Usability of both languages to support model transformation is also shown by various authors (e.g. [23, 36]).

Languages based on the graph transformation paradigm (e.g. AGG) employ graph patterns and it is not clear how OCL-based queries are translated to graph patterns and vice versa. On the other hand, there is not any major obstacle for translating from ATL to QVT Operational Mappings and from Operational Mappings to imperative ATL [28]. Moreover, interoperability of ATL and QVT is discussed and shown in [26, 30].

Graph transformations are sometimes accused of generating inefficient programs or having inefficient algorithms [36]. AGG provides graphical languages to define model-to-model transformations. Graph transformations are defined upon metamodel elements and visualized with a generic layout, called abstract syntax, where nodes are visualized as rectangles and edges as directed arrows [23]. AGG shows simulation runs in the visual editor panel of the host graph, where not only the start graph can be drawn but also rule application effects are shown at runtime (based on the abstract visual syntax). Various graph layouting options are offered for tuning the visualization, makes it appealing and understandable [8]. It should be noted that abstract syntax is less familiar to developers working with a given modeling language than the concrete syntax [14, 23].

There are two standards for graph transformation languages: GXL which is an exchange format for graphs and GTXL which is an exchange format for graph transformations. Both standards are supported by AGG. Moreover, AGG supports XML [36] and could be used for refactoring of UML models [20]. Reuse mechanisms such as inheritance between rules (e.g. rule inheritance, derivation, extension, and specialization) is not supported in AGG [14, 36].

Figure 2 summarizes the comparison between ATL and AGG by visualizing the non-functional requirements and contribution links of the languages. In this figure, through contribution link types of $-, --, +$ and $++$ [13], the qualitative effect of alternative languages are propagated to the non-functional goals. We use qualitative contribution links since non-functional requirements are intangible and the effects of languages on them are difficult, if not impossible, to measure. This model can aid software engineer/modeler to see the alternative languages and the criteria of decision making; However, it is not enough for decision making since it lacks a systematic way of choosing an specific alternative. The next section of this paper proposes a decision making mechanism which uses this model and aids decision maker to rank the model transformation languages with regarding to non-functional requirements and select the most suitable one.

4. PROPOSED APPROACH

4.1 The Fuzzy Set theory

Fuzzy set theory proposed by Zadeh in [50] to deal with vagueness of human thought and was oriented to the rationality of uncertainty due to imprecision or vagueness. It resembles human reasoning in its use of approximate infor-

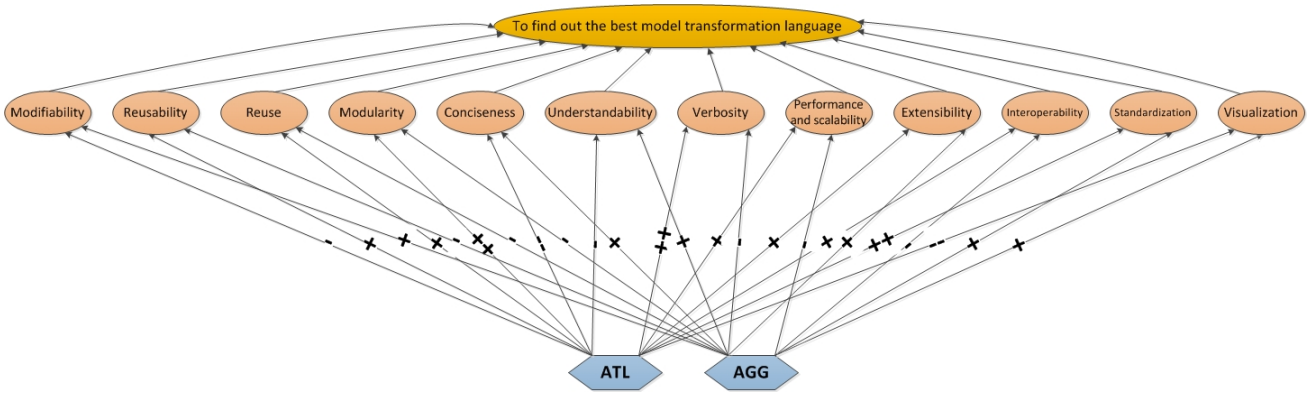

Figure 1: Alternatives, goals and contribution links

mation and uncertainty to generate decisions. It is applied in a variety of ways and in many disciplines such as artificial intelligence, computer science, medicine, control engineering, decision theory, expert systems, logic, management science, operations research, pattern recognition, and robotics [51].
A Fuzzy set is a collection of objects with unsharp boundaries in which the transition from membership to nonmembership in a subset of a reference set is gradual rather than abrupt [34]. A fuzzy number is a special fuzzy set $M = \{(x, \mu_M(x)), x \in R\}$, where the value of x lies on the real line R i.e. $-\infty < x < +\infty$ and $\mu_M(x)$ is a continuous mapping from R to the close interval $[0, 1]$. A triangular fuzzy number M denoted as a triplet (l, m, u) where $l \leq m \leq u$, has the following triangular-type membership function:

$$\mu_M(x) = \begin{cases} \frac{x-l}{m-l} & l \leq x \leq m \\ \frac{u-x}{u-m} & m \leq x \leq u \\ 0 & \text{otherwise} \end{cases}$$

where l and u stand for the lower and upper value of the support for M, and m for the modal value.
Consider two fuzzy triangular numbers $M_1 = (l_1, m_1, u_1)$ and $M_2 = (l_2, m_2, u_2)$. Main fuzzy operations are illustrated here:
Fuzzy number addition $\oplus$:

$$M_1 \oplus M_2 = (l_1, m_1, u_1) \oplus (l_2, m_2, u_2) = (l_1 + l_2, m_1 + m_2, u_1 + u_2).$$

Fuzzy number subtraction $\ominus$:

$$M_1 \ominus M_2 = (l_1, m_1, u_1) \ominus (l_2, m_2, u_2) = (l_1 - l_2, m_1 - m_2, u_1 - u_2).$$

Fuzzy number multiplication $\odot$:

$$M_1 \odot M_2 = (l_1, m_1, u_1) \odot (l_2, m_2, u_2) = (l_1 \times l_2, m_1 \times m_2, u_1 \times u_2).$$

Fuzzy number division $\oslash$:

$$M_1 \oslash M_2 = (l_1, m_1, u_1) \oslash (l_2, m_2, u_2) = (l_1/l_2, m_1/m_2, u_1/u_2).$$

Fuzzy number reciprocal:

$$(M)^{-1} = (l, m, u)^{-1} \approx (1/u, 1/m, 1/l).$$

In this paper addition, multiplication and reciprocal operations are used.

4.2 Fuzzy-AHP

Analytic Hierarchy Process (AHP), one of the Multi-Criteria Decision-Making (MCDM) methods proposed in [39], is aimed at facilitating decision making in problems which involve multiple criteria. It enables domain experts to structure a complex decision making problem in the form of hierarchy, where alternatives and factors are identified and evaluated with respect to other related factors [18]. It provides a way to rank the alternatives of a decision making problem through a four step process: development of problem hierarchy, development of judgment matrices by pairwise comparisons, calculation of local priorities based on judgment matrices, and finally calculation of global priorities. AHP has been applied in a huge variety of application fields, e.g. software development [19,31], project management [2], medical and health care decision making [33], marketing [16], and supplier selection [10]. According to [42] there is a vast literature on the applications of AHP with more than 1300 papers and 100 doctoral dissertations.
Although AHP is popular and simple, it is often criticized for its inability to adequately handle the uncertainty and fuzziness of decision maker's perception. In traditional AHP, human's judgments are represented as exact values [21]. However, many practical evaluation and decision making contexts are too complex to be understood and measured quantitatively. In other words, linguistic assessment of human feelings and judgments are vague and it is not reasonable to represent it in terms of precise numbers [11]. For instance, non-functional requirements in software engineering are intangible concepts and difficult, if not impossible, to evaluate.
Fuzzy-AHP combines the traditional AHP with the fuzzy set theory. In this paper, Chang's fuzzy-AHP approach [12] is applied to evaluate mode transformation languages given a set of non-functional requirements. In this approach, tri-

Intensity of importance	Membership function
Extremely more importance (EMI)	(7,9,9)
Very strong importance (VSI)	(5,7,9)
Strong importance (SI)	(3,5,7)
Moderate importance (MI)	(1,3,5)
Equal importance (EI)	(1,1,3)

Table 2: Definition and membership function of fuzzy scale.

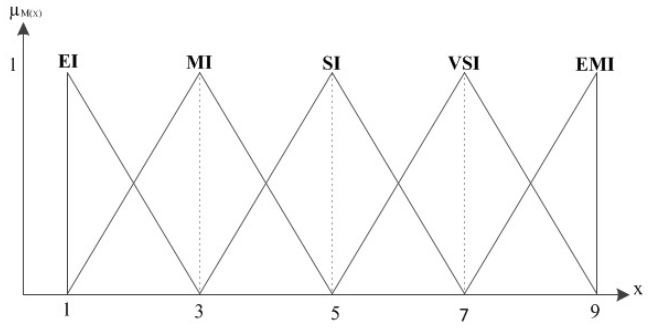

Figure 2: The membership function of linguistic variables. See Table 2 for abbreviations.

angular fuzzy numbers are used for a pairwise comparison scale of non-functional requirements. Table 2 shows how linguistic scales are converted into fuzzy scales and Figure 2 shows the membership function of linguistic variables.
Then by using extent analysis and the principle of the comparison of fuzzy numbers the weight vectors and finally the global priorities of languages are calculated.

Let $X = \{x_1, x_2, \ldots, x_n\}$ be an object set, and $U = \{u_1, u_2, \ldots, u_m\}$ be a goal set. According to concept of extent analysis [15] each object is taken and extent analysis for each objective U_j is performed, respectively. Therefore the m extent analysis values for each object are obtained with the following signs:

$$M_{g_i}^1, M_{g_i}^2, \ldots, M_{g_i}^m, \quad i = 1, 2, \ldots, n.$$

where all the $M_{gi}^j (j = 1, 2, \ldots, m)$ are triangular fuzzy numbers. The computational procedure of fuzzy-AHP are as follows:
Step 1: In this step, the decision problem is modeled as a hierarchy including the decision goal, the alternatives for reaching it, and the criteria for evaluating the alternatives.
Step 2: Then, the fuzzy comparison matrix is constructed. First, relative importance/strength of each pair of elements in the same hierarchy is judged based on linguistic terms (via pair-wise comparison). Then, by converting the linguistic terms to triangular fuzzy numbers, the fuzzy comparison matrix is constructed:

$$M = \begin{bmatrix} M_{g_1}^1 & M_{g_1}^2 & \cdots & M_{g_1}^m \\ M_{g_2}^1 & M_{g_2}^2 & \cdots & M_{g_2}^m \\ \vdots & \vdots & \ddots & \vdots \\ M_{g_n}^1 & M_{g_2}^2 & \cdots & M_{g_n}^m \end{bmatrix}$$

where all the $M_{g_i}^j (i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m)$ are fuzzy triangular numbers and $M_{g_i}^j = (1, 1, 1)$ iff $i = j$.
Step 3: In this step, for each object the value of fuzzy synthetic extent is calculated:

$$S_i = \sum_{j=1}^{m} M_{gi}^j \odot \left[\sum_{i=1}^{n} \sum_{j=1}^{m} M_{g_i}^j \right]^{-1} \quad (1)$$

where the value of $\sum_{j=1}^{m} M_{gi}^j$ and $\sum_{i=1}^{n} \sum_{j=1}^{m} M_{g_i}^j$ can be found by performing the fuzzy addition operation. Moreover, $\odot$ is fuzzy multiplication operator (See Section 4.1 for further explanation).
Step 4: Then, the fuzzy synthetic extents are compared.The degree of possibility of $S_i \geq S_j$ is calculated by the following equation:

$$V(S_i \geq S_j) = hgt(S_j \cap S_i)$$
$$= \begin{cases} 1 & \text{if } m_i \geq m_j \\ 0 & \text{if } l_j \geq u_i \\ \frac{l_j - u_i}{(m_i - u_i) - (m_j - lj)} & \text{otherwise} \end{cases} \quad (2)$$

where $S_i = (l_i, m_i, u_i)$ and $S_j = (l_j, m_j, u_j)$ are fuzzy triangular numbers calculated in *Step 3*. To compare S_i and S_j, we need both values of $V(S_i \geq S_j)$ and $V(S_j \geq S_i)$.
Step 5: In this step, for each object the minimum degree of possibilities (calculated in previous step) is calculated. According to [12], the degree possibility for a convex fuzzy number to be greater than k convex fuzzy numbers can be defined as:

$$\begin{aligned} V(S \geq S_1, S_2, \ldots, S_n) = & V[(S \geq S_1) \text{ and } (S \geq S_2) \\ & \text{and } \ldots \text{ and } (S \geq S_n)] = \\ & \min V(S \geq S_i),\ i = 1, 2, \ldots, n. \end{aligned} \quad (3)$$

Using this equation, the weight (or priority) vector is defined as $w_i' = Min\ V(S_i \geq S_k)$ for $k = 1, 2, \ldots, n$ and $k \neq i$. Therefore:

$$W' = (w_1', w_2', \ldots, w_n')^T$$

Step 6: In the last step, the normalized weight vector $W = (w_1, w_2, \ldots, w_n)^T$ is calculated as follows:

$$w_i = \frac{w_i'}{\sum_{i=1}^{n} w_i'} \quad (4)$$

This weight vector is a non-fuzzy (crisp) value.
In the next section of the paper, we show the application of fuzzy-AHP method fro selecting model transformation languages.

5. APPLICATION

In this section, we describe three scenarios of model transformations and then apply the proposed decision making mechanism to select the proper model transformation language for each case.
Case 1. In this case, the goal is to choose a language for defining a model transformation between two widely recognized standards for business process modeling: the Business Process Modeling Notation (BPMN) and Business Process Execution Language (BPEL). BPMN is a standard for business process modeling that provides a graphical notation for the purposes of business analysis. The target users of this notation are usually business analysts. BPEL is a standard executable language for specifying actions within business processes with web services. Transformations between these standards are complex because of inherent differences between them: BPMN process models are graph-oriented (with only minor topological restrictions), while BPEL process definitions are block-structured. The transformation has been used as a case study in [17]. In this case, since we are dealing with business analyst, we assume that understandability, conciseness, modularity, and visualization are more important than other non-functional requirements.
Case 2. In this case, the transformation generates a relational database model as output by receiving a class schema

	UN	MF	RY	RE	MD	CS	VB	PS	EX	IN	ST	VS	Overal Weight
UN	(1,1,1)	(3,5,7)	(5,7,9)	(3,5,7)	(1,1,3)	(1,1,3)	(5,7,9)	(1,1,3)	(5,7,9)	(5,7,9)	(1,3,5)	(1,3,5)	**0.15**
MF	(0.14,0.20,0.33)	(1,1,1)	(3,5,7)	(1,3,5)	(0.20,0.33,1)	(0.20,0.33,1)	(3,5,7)	(0.20,0.33,1)	(3,5,7)	(3,5,7)	(1,1,3)	(1,1,3)	**0.11**
RY	(0.11,0.14,0.20)	(0.14,0.20,0.33)	(1,1,1)	(0.20,0.33,1)	(0.14,0.20,0.33)	(0.20,0.33,1)	(1,1,3)	(0.11,0.14,0.20)	(1,1,3)	(1,1,3)	(0.14,0.20,0.33)	(0.11,0.14,0.20)	**0**
RE	(0.14,0.20,0.33)	(0.20,0.33,1.00)	(1,3,5)	(1,1,1)	(0.14,0.20,0.33)	(0.14,0.20,0.33)	(1,3,5)	(0.14,0.20,0.33)	(1,3,5)	(1,3,5)	(0.20,0.33,1)	(0.20,0.33,1)	**0.05**
MD	(0.33,1,1)	(1,3,5)	(3,5,7)	(3,5,7)	(1,1,1)	(1,1,3)	(5,7, 9)	(1,1,3)	(5,7,9)	(5,7,9)	(1,3,5)	(1,3,5)	**0.15**
CS	(0.33,1,1)	(1,3,5)	(1,3,5)	(3,5,7)	(0.33,1,1)	(1,1,1)	(5,7,9)	(1,1,3)	(5,7,9)	(5,7,9)	(1,3,5)	(1,3,5)	**0.14**
VB	(0.11,0.14,0.20)	(0.14,0.20,0.33)	(0.33,1,1)	(0.20,0.33,1)	(0.11,0.14,0.20)	(0.11,0.14,0.20)	(1,1,1)	(0.11,0.14,0.20)	(1,1, 3)	(1,1,3)	(0.14,0.20,0.33)	(0.14,0.20,0.33)	**0**
PS	(0.33,1,1)	(1,3,5)	(5,7,9)	(3,5,7)	(0.33,1,1)	(0.33,1,1)	(5,7,9)	(1,1,1)	(5,7,9)	(5,7,9)	(1,3,5)	(1,3,5)	**0.15**
EX	(0.11,0.14,0.20)	(0.14,0.20,0.33)	(0.33,1,1)	(0.20,0.33,1)	(0.11,0.14,0.20)	(0.11,0.14,0.20)	(0.33,1,1)	(0.11,0.14,0.20)	(1,1,1)	(1,1,3)	(0.14,0.20,0.33)	(0.14,0.20,0.33)	**0**
IN	(0.11,0.14,0.20)	(0.14,0.20,0.33)	(0.33,1,1)	(0.20,0.33,1)	(0.11,0.14,0.20)	(0.11,0.14,0.20)	(0.33,1,1)	(0.11,0.14,0.20)	(0.33,1,1)	(1,1,1)	(0.14,0.20,0.33)	(0.14,0.20,0.33)	**0**
ST	(0.20,0.33,1)	(0.33,1,1)	(3,5,7)	(1,3,5)	(0.20,0.33,1)	(0.20,0.33,1)	(3,5,7)	(0.20,0.33,1)	(3,5,7)	(3,5,7)	(1,1,1)	(1,1,3)	**0.10**
VS	(0.20,0.33,1)	(0.33,1,1)	(5,7,9)	(1,3,5)	(0.20,0.33,1)	(0.20,0.33,1)	(3,5,7)	(0.2,0.33,1)	(3,5,7)	(3,5,7)	(0.33,1,1)	(1,1,1)	**0.11**

Table 3: Fuzzy triangular numbers representing pairwise comparison of importance of non-functional requirements for *Case 1*. See Table 1 for abbreviations. Moreover, see Table 2 for linguistic terms equivalent to these numbers. The last column are overall weight (importance) scores of non-functional requirements resulted from calculations explained in Section 5.

model as input. Such a transformation needs to realize three main mappings: package-to-schema, class-to-table, and attribute to-column. This transformation is implemented by various authors in different languages, e.g. ATL [1], QVT Relations [14], AGG [45], and Tefkat [32]. In this case we assume that all the non-functional requirements are of same level of importance.

Case 3. In the last case, adopted from [6], decision maker is going to decide a transformation language in a large industrial context. The intent of model transformation is automatic code generation from models. In this case, using standardized and non-proprietary modeling languages is of great importance. Moreover, the projects in this context are large, development efforts are multi-site and collaborative and the models are huge. Hence, we assume that performance and scalability, interoperability, standardization and reusability are more important than other non-functional requirements.

In all of the above cases the question is how to select most suitable model transformation language among a set of alternatives, i.e. given a specific context, the intention of transformation, and a set of criteria, which language is the best choose? In following, we show applicability of proposed framework for these cases. In this project, the assumption is that there are only two alternatives: ATL and AGG. It should be mentioned that the proposed method can handle more number of alternatives, however, for the purpose of this paper we assume there are only two of them.

Figure 2 is used as hierarchy of the decision problem. To solve the problem for *Case 1*, first the fuzzy comparison matrix of the non-functional requirements is constructed. To do that, we performed a pairwise comparison of the different non-functional requirements using triangular fuzzy numbers, which is shown in Table 3. In this project, these values are defined by author based on the context and the non-functional requirements that are important in that specific case. However, in future versions of this work, we can use opinions of a set of domain experts. Then, the value of fuzzy synthetic extent with respect to each non-functional requirement is calculated by using Eq. 1. We have:

$$\begin{aligned}
S_{UN} &= (32.00, 48.00, 70.00) \odot (184.90, 300.88, 448.67)^{-1} \simeq \\
&\quad (0.07, 0.16, 0.37), \\
S_{MF} &= (16.74, 27.20, 43.33) \odot (184.90, 300.88, 448.67)^{-1} \simeq \\
&\quad (0.03, 0.09, 0.23), \\
S_{RY} &= (5.16, 5.70, 13.60) \odot (184.90, 300.88, 448.67)^{-1} \simeq \\
&\quad (0.01, 0.01, 0.07), \\
&\vdots \\
S_{VS} &= (17.47, 29.33, 42.00) \odot (184.90, 300.88, 448.67)^{-1} \simeq \\
&\quad (0.03, 0.09, 0.22).
\end{aligned}$$

Now, by using Eq. 2, we have:

$$\begin{aligned}
&V(S_{UN} \geq S_{MF}) = 1 \quad \text{(because } 0.16 \geq 0.09\text{)} \\
&V(S_{UN} \geq S_{RY}) = 1 \quad \text{(because } 0.16 \geq 0.01\text{)} \\
&\vdots \\
&V(S_{UN} \geq S_{VS}) = 1 \quad \text{(because } 0.16 \geq 0.09\text{)} \\
&V(S_{MF} \geq S_{UN}) = \frac{0.07 - 0.23}{(0.09 - 0.23) - (0.16 - 0.07)} = 0.7 \\
&\vdots \\
&V(S_{MF} \geq S_{VS}) = \frac{0.03 - 0.23}{(0.09 - 0.23) - (0.09 - 0.03)} = 0.96 \\
&\vdots \\
&V(S_{VS} \geq S_{SD}) = 1 \quad \text{(because } 0.097 \geq 0.091\text{)}
\end{aligned}$$

Then, by using Eq. 3 we obtain:

$$\begin{aligned}
w'_{UN} &= \text{Min}(1, 1, \ldots, 1) = 1 \\
w'_{MF} &= \text{Min}(0.7, 1, \ldots, 0.96) = 0.7 \\
w'_{RY} &= \text{Min}(0.02, 0.34, \ldots, 0.31) = 0.02 \\
&\vdots \\
w'_{VS} &= \text{Min}(0.72, 1, \ldots, 0.31) = 0.72
\end{aligned}$$

Then, using Eq. 4, the normalized weight vector of non-functional requirements for *Case 1* is obtained. The last column in Table 3 shows the weights.

After that, by applying the same procedure for the alternatives (which are in the other layer of the problem hierarchy, see Figure 2), the relative strength of each language with regarding to each non-functional requirement is calculated. For example, Table 4 shows the pairwise comparison of languages with regarding to understandability and standardization. The values in this table are defined based on the literature and explanations in Section 3. The calculations of weights of languages with respect to each of non-functional requirements will not be given in this paper since they are similar to calculations above.

Finally, the overall priority weight of each language is calculated by multiplying its local weights with non-functional requirements weights:

		ATL	**AGG**	**Weight**
UN	ATL	(1,1,1)	(0.14,0.20.0.33)	**0**
	AGG	(3,5,7)	(1,1,1)	**1**
SD	ATL	(1,1,1)	(1,3,5)	**0.75**
	AGG	(0.20,0.33,1)	(1,1,1)	**0.25**

Table 4: Pairwise comparison of languages, in terms of fuzzy triangular numbers, with regarding to understandability and standardization (for *Case 1*).

$$\begin{aligned}
\text{ATL} &= (0 \times 0.15) + (0 \times 0.11) + \cdots + (0.75 \times 0.10) \\
&\quad + (0 \times 0.11) = 0.45 \\
\text{AGG} &= (1 \times 0.15) + (1 \times 0.11) + \cdots + (0.25 \times 0.10) \\
&\quad + (1 \times 0.11) = 0.55
\end{aligned}$$

Table 5 summarizes the final results of fuzzy-AHP scores for each of the cases. This table indicates that for the first case,

AGG language is more suitable. Also, for the second and third cases, the ATL language has a better rank.

It should be noted that because of the space limitation, the

	Languages	Score
Case 1	ATL	0.45
	AGG	**0.55**
Case 2	ATL	**0.59**
	AGG	0.41
Case 3	ATL	**0.67**
	AGG	0.33

Table 5: Results of Fuzzy-AHP for all cases

step by step computations and fuzzy comparison matrices of all the cases are not shown in this paper. In this research, the fuzzy-AHP method was implemented in Excel and the codes are submitted with this paper.

6. CONCLUSION AND FUTURE WORKS

Model transformation is primary activity in MDE. Various languages and tools have been proposed, coming from different approaches and programming paradigms. This paper proposed a decision making framework which helps software engineer/developer to decide what model transformation language is the right choice, given the context and non-functional requirements. This approach is implemented and its application is shown in this paper for three different scenarios. One of the main advantages of the proposed approach is the relative ease with which it handles multiple, sometimes conflicting, non-functional requirements. Moreover, the proposed approach introduces concepts of fuzzy set theory for measuring and evaluating of alternatives. This allows decision-maker(s) to have freedom of estimation regarding the non-functional requirements. The proposed approach could be extended to consider a broader set of alternative languages. This basically need to answer how well, relatively, the new added languages do with regarding to each requirement. In this paper we did it based on literature and only for two languages, but the future works can easily extend it to new languages. Besides, future works can do a more broad implementation involving a set of domain expert to compare degree of importance of non-functional requirements for each context. Finally, performing a sensitivity analysis to the approach and also a comparative study of this approach with other MCDM techniques could be other interesting venues.

Acknowledgement. I would like to thank Rick Salay for the weekly meetings that we had and also his useful comments.

7. REFERENCES

[1] Atl transformations. http://www.eclipse.org/m2m/atl/atlTransformations.

[2] K. M.-S. Al-Harbi. Application of the ahp in project management. *International Journal of Project Management*, 19(1):19 – 27, 2001.

[3] M. Amrani, J. Dingel, L. Lambers, L. LÂt'ucio, R. Salay, G. Selim, E. Syriani, and M. Wimmer. Towards a model transformation intent catalog. 1st Workshop on the Analysis of Model Transformations (AMT), 2012.

[4] M. F. v. Amstel. The right tool for the right job: Assessing model transformation quality. In *Proceedings of the 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops*, COMPSACW '10, pages 69–74, Washington, DC, USA, 2010. IEEE Computer Society.

[5] K. M. A. Aziz. *Evaluating Model Transformation Technologies: An exploratory case study.* Bachelor of Science in software engineering and management, Chalmers University of Technology, Goteborg, Sweden, 2011.

[6] P. Baker, S. Loh, and F. Weil. Model-driven engineering in a large industrial context-motorola case study. In L. Briand and C. Williams, editors, *Model Driven Engineering Languages and Systems*, volume 3713 of *Lecture Notes in Computer Science*, pages 476–491. Springer Berlin Heidelberg, 2005.

[7] B. Baudry, S. Ghosh, F. Fleurey, R. France, Y. Le Traon, and J.-M. Mottu. Barriers to systematic model transformation testing. *Commun. ACM*, 53(6):139–143, June 2010.

[8] E. Biermann, C. Ermel, L. Lambers, U. Prange, O. Runge, and G. Taentzer. Introduction to agg and emf tiger by modeling a conference scheduling system. *Int. J. Softw. Tools Technol. Transf.*, 12(3-4):245–261, July 2010.

[9] E. Brottier, F. Fleurey, J. Steel, B. Baudry, and Y. Le Traon. Metamodel-based test generation for model transformations: an algorithm and a tool. In *Software Reliability Engineering, 2006. ISSRE '06. 17th International Symposium on*, pages 85 –94, nov. 2006.

[10] G. Bruno, E. Esposito, A. Genovese, and R. Passaro. Ahp-based approaches for supplier evaluation: Problems and perspectives. *Journal of Purchasing and Supply Management*, (0):–, 2012.

[11] F. T. Chan and N. Kumar. Global supplier development considering risk factors using fuzzy extended ahp-based approach. *Omega*, 35(4):417 – 431, 2007.

[12] D.-Y. Chang. Applications of the extent analysis method on fuzzy ahp. *European Journal of Operational Research*, 95(3):649 – 655, 1996.

[13] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering.* 2000.

[14] K. Czarnecki and S. Helsen. Feature-based survey of model transformation approaches. *IBM Syst. J.*, 45(3):621–645, July 2006.

[15] C. D.-Y. Extent analysis and synthetic decision, optimization techniques and applications. *World Scientific*, 1:352, 1992.

[16] M. Davies. Adaptive ahp: a review of marketing applications with extensions. *European Journal of Marketing*, 35:872 – 894, 2001.

[17] M. Dumas. Case study: Bpmn to bpel model transformation. http://is.ieis.tue.nl/staff/pvgorp/events/grabats2009 /cases/grabats2009synthesis.pdf, 2009.

[18] R. F. Dyer and E. H. Forman. Group decision support with the analytic hierarchy process. *Decision Support Systems*, 8(2):99 – 124, 1992.

[19] G. R. Finnie, G. E. Wittig, and D. I. Petkov. Prioritizing software development productivity factors using the analytic hierarchy process. *Journal of Systems and Software*, 22(2):129 – 139, 1993.

[20] A. Folli and T. Mens. Refactoring of UML models using AGG. *ECEASST*, 8, 2007.

[21] G˙Strategic analysis of healthcare service quality using fuzzy ahp methodology. *Expert Systems with Applications*, 38(8):9407 – 9424, 2011.

[22] A. Göknil, N. Topaloglu, and K. van den Berg. Operation composition in model transformations with complex source patterns, 2008.

[23] R. Gronmo, B. Moller-Pedersen, and G. Olsen. Comparison of three model transformation languages. In R. Paige, A. Hartman, and A. Rensink, editors, *Model Driven Architecture - Foundations and Applications*, volume 5562 of *Lecture Notes in Computer Science*, pages 2–17. Springer Berlin Heidelberg, 2009.

[24] A. group. The mof to uml atl transformation. Technical report, LINA & INRIA, Nantes, September 2005.

[25] P. Guyard. Atl transformation example: Bridging grafcet, petri net, pnml and xml. Technical report, INRIA, August 2005.

[26] F. Jouault and I. Kurtev. On the architectural alignment of atl and qvt. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 1188–1195, New York, NY, USA, 2006. ACM.

[27] F. Jouault and I. Kurtev. Transforming models with atl. In *Proceedings of the 2005 international conference on Satellite Events at the MoDELS*, MoDELS'05, pages 128–138, Berlin, Heidelberg, 2006. Springer-Verlag.

[28] F. Jouault and I. Kurtev. On the interoperability of model-to-model transformation languages. *Science of Computer Programming*, 68(3):114 – 137, 2007.

[29] T. Kosar, P. E. M. Lopez, P. A. Barrientos, and M. Mernik. A preliminary study on various implementation approaches of domain-specific language. *Information and Software Technology*, 50(5):390 – 405, 2008.

[30] A. Laarman. Achieving qvto & atl interoperability: An experience report on the realization of a qvto to atl computer. In F. Jouault, editor, *1st International Workshop on Model Transformation with ATL, MtATL 2009*, CEUR Workshop Proceedings, pages 119–133, Aachen, October 2009. Sun SITE Central Europe.

[31] V. S. Lai, B. K. Wong, and W. Cheung. Group decision making in a multiple criteria environment: A case using the ahp in software selection. *European Journal of Operational Research*, 137(1):134 – 144, 2002.

[32] M. Lawley, K. Duddy, A. Gerber, and K. Raymond. Language features for re-use and maintainability of mda transformations. In *In OOPSLA Workshop on Best Practices for Model-Driven Software Development*, 2004.

[33] M. J. Liberatore and R. L. Nydick. The analytic hierarchy process in medical and health care decision making: A literature review. *European Journal of Operational Research*, 189(1):194 – 207, 2008.

[34] J. Maiers and Y. Sherif. Applications of fuzzy set theory. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-15(1):175 –189, jan.-feb. 1985.

[35] T. Mens and P. Van Gorp. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, 152:125–142, Mar. 2006.

[36] T. Mens, P. Van Gorp, D. Varró, and G. Karsai. Applying a model transformation taxonomy to graph transformation technology. *Electron. Notes Theor. Comput. Sci.*, 152:143–159, Mar. 2006.

[37] P. Mohagheghi and J. Aagedal. Evaluating quality in model-driven engineering. In *Proceedings of the International Workshop on Modeling in Software Engineering*, MISE '07, pages 6–, Washington, DC, USA, 2007. IEEE Computer Society.

[38] A. Randak, S. Martínez, and M. Wimmer. Extending atl for native uml profle support: An experience report. CEUR Workshop Proceedings, 2011.

[39] T. L. Saaty. *The Analytic Hierarchy Process*. McGraw-Hill, New York, 1980.

[40] S. Sendall and W. Kozaczynski. Model transformation: the heart and soul of model-driven software development. *Software, IEEE*, 20(5):42–45, sept.-oct. 2003.

[41] M. Stephan and A. Stevenson. A comparative look at model transformation languages. Software Technology Laboratory at Queen's University, 2009.

[42] N. Subramanian and R. Ramanathan. A review of applications of analytic hierarchy process in operations management. *International Journal of Production Economics*, 138(2):215 – 241, 2012.

[43] E. Syriani and J. Gray. Challenges for addressing quality factors in model transformation. In *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*, pages 929 –937, april 2012.

[44] G. Taentzer. Agg: A graph transformation environment for modeling and validation of software. In J. Pfaltz, M. Nagl, and B. BÃűhlen, editors, *Applications of Graph Transformations with Industrial Relevance*, volume 3062 of *Lecture Notes in Computer Science*, pages 446–453. Springer Berlin Heidelberg, 2004.

[45] G. Taentzer, K. Ehrig, E. Guerra, J. D. Lara, T. Levendovszky, U. Prange, D. Varro, and et al. Model transformations by graph transformations: A comparative study. In *MODEL TRANSFORMATIONS IN PRACTICE WORKSHOP AT MODELS 2005, MONTEGO*, page 5, 2005.

[46] M. Tisi, J. Cabot, and F. Jouault. Improving higher-order transformations support in atl. In *Proceedings of the Third international conference on Theory and practice of model transformations*, ICMT'10, pages 215–229, Berlin, Heidelberg, 2010. Springer-Verlag.

[47] M. F. van Amstel, C. F. J. Lange, and M. G. J. van den Brand. Metrics for Analyzing the Quality of Model Transformations. 2008.

[48] A. Vignaga. Metrics for measuring atl model transformations. Technical report, Universidad de Chile, 2009.

[49] D. S. Wile. Supporting the dsl spectrum. *Journal of Computing and Information Technology*, 9(4):263 – 287, 2001.

[50] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338âĂŞ353, 1965.

[51] H.-J. Zimmermann. Fuzzy set theory. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2:317–332, 2010.