Embracing Imperfection in Enterprise Architecture Models

Hector Florez, Mario Sánchez, and Jorge Villalobos

Department of Systems and Computing Engineering,
Universidad de los Andes,
Bogotá, Colombia
{ha.florez39,mar-san1,jvillalo}@uniandes.edu.co

Abstract. In Enterprise Architecture (EA) projects, models are built to represent business and Information Technologies (IT) elements, and to abstract the relation between them in one enterprise under study. The construction of EA models depends on information provided by different kinds of sources, but sources could be insufficient or information could be incomplete or incorrect regarding aspects of the enterprise. As a result, modelers must often create models based on low quality information that could not properly represent the enterprise. Since EA models are used as the base for the analysis of the enterprise, using models that do not represent the enterprise correctly, there is a risk that the analysis produces unreliable conclusions. This paper presents a proposal for managing imperfect models in the EA context. An imperfect model is capable of representing information that can be imprecise, inconsistent, vague, uncertain, or incomplete; thus, when analyses are performed, they can use this information to produce more reliable results. In this proposal, imperfect models also include information about modeler decisions.

Keywords: Enterprise Architecture, Model Driven Engineering, Models Imperfection

1 Introduction

Enterprises increasingly depend on Information Technologies (IT) and require support in order to achieve their business goals. Moreover, the enterprise complexity and constant evolution generate difficulties in the relation between the business and IT. EA projects require the construction of one model that we call *enterprise architecture model (m_{EA})*, which is an abstract representation of one simplification [1, 2] of the enterprise that includes enterprise elements and their relations. The m_{EA} is usually big because enterprises have a large number of elements and is complex because they have a large number of typed relations between their elements. One m_{EA} is built by *modelers* through direct and indirect observation. Direct observation is the action in which modelers obtain enterprise information without consulting sources. Indirect observation requires the participation of sources (e.g., persons, documents, and meetings). In an EA project, the m_{EA} is used to analyze the enterprise. These analyses are performed

by domain experts called *analysts*, who manipulate the m_{EA} in order to extract useful information for evaluating the current state of the enterprise.

The model construction process is an observation of human process, sources consulting, interpreting, and expression [3], so the construction of one m_{EA} that properly represents the enterprise has a high level of difficulty. The difficulty is based on two main reasons: (1) the enterprise size and complexity and (2) several uncontrolled factors that affect the modeling process [4] such as sources quality and lack of information. Quality refers to the knowledge level of the sources on specific aspects of the enterprise. Lack of information refers to the incapability of the source for providing information on some aspects of the enterprise. Because of this, in some cases it is impossible to build an m_{EA} that correctly represents the enterprise. For instance, consider the case where a model will be used to document the business applications of an enterprise, and analyze their availability. Then, one employee may assert that the availability (that must be a number) of the *BusinessApp_X* is between 92% and 95%, and the availavility of the *BusinessApp_Y* is *High*. The modeler must choose one numeric value for the *BusinessApp_X* between 92% and 95%, and a numeric value for the *BusinessApp_Y* that must be high (up to 100 in this case). In any cases, the values selected by the modeler do not represent the enterprise correctly because he/she cannot know the appropiate value. Later on, the analyst will use the model to perform an analysis of the enterprise; however, since he/she will be using a m_{EA} that might not represent some enterprise elements or relations correctly; in such case, the results should not be considered very reliable.

Model imperfection is inevitable in EA projects. Then, it is better to include information in the model to represent those problems than to ignore them and to assume that the model accurately represents the enterprise. Modeling the imperfection implies creating another model that we call *imperfect enterprise architecture model (m_ζ)*, which is an approximation of the m_{EA} with information that can assess the imperfection. Thus, there is a distance between m_ζ and m_{EA} and the modeler must build the best possible approximation given the quality of the available information. To establish the level of the information validity, the imperfect information should be related with sources that provide the information. In the construction process of the m_ζ, modelers consult sources through observations (e.g., interviews, reviews, meeting acts, etc.) where each observation provides facts. Through these facts, modelers make decisions to assign values for imperfect attributes and relations. Finally, based on the m_ζ, it is possible to create analysis methods while taking into account the imperfection.

The rest of the paper is structured as follows. Section 2 describes the modeling process in the EA context. Section 3 presents our proposed taxonomy of imperfect sources, where we classify sources based on the quality of the information provided. In section 4, we present our proposal for modeling the imperfection in EA projects based on the sources classification. In section 5, we present our tool for modeling imperfection and creating the m_ζ. In section 6 we present modeling imperfection involving traces about decision making by modelers, for imperfect attributes and relations. Finally, section 7 presents the conclusions.

2 Construction of Enterprise Architecture Models

In EA projects, it is necessary to construct one domain metamodel that we call *enterprise architecture metamodel (MM_{EA})* representing the concepts of the enterprise and the relations between them. Those concepts are typed elements which contain structural features. The MM_{EA} is different for each specific project, since it reflects the horizontal and vertical scope for the project. In this paper, we assume that MM_{EA} is correct and never changes during the project.

Normally, the m_{EA} must conform to the MM_{EA}. The m_{EA} is made up of several instances ($\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_p\}$) that conform to the correspondent typed elements of MM_{EA}. Each instance in Γ contains several structural features ($\Phi_{\gamma_i} = \{\varphi_{\gamma_i,1}, \varphi_{\gamma_i,2}, \ldots, \varphi_{\gamma_i,q}\}$), which can be attributes ($A_{\gamma_i} = \{\alpha_{\gamma_i,1}, \alpha_{\gamma_i,2}, \ldots, \alpha_{\gamma_i,r}\}$) or typed relations ($B_{\gamma_i} = \{\beta_{\gamma_i,1}, \beta_{\gamma_i,2}, \ldots, \beta_{\gamma_i,s}\}$). The conformance relation between m_{EA} and MM_{EA} establishes that Φ_{γ_i} must respect the types and structures defined for each metatype in MM_{EA}. However, if modelers build one m_ζ instead of one m_{EA}, this m_ζ does not conform to MM_{EA}.

The m_ζ can include imperfect information, where the modeler can decide which attributes or relations are imperfect. To build the m_ζ, we use the distinction between *ontological conformance* that is based on the relation between the model and metamodel in terms of their meaning, and *linguistic conformance* that is based on the relation between the model and metamodel in terms of their structure [5]. In addition, we achieve the linguistic conformance by the construction of a generic intermediate metamodel that serves to represent any type, attribute and relation; and the ontological conformance by the definition of semantic rules [6]. Then, the m_ζ allows the creation of instances of a type with regular or imperfect structural features. One structural feature is imperfect regarding the MM_{EA}, when its value does not adjust with the characteristics established in the MM_{EA}.

Figure 1 illustrates the proposed strategy. The m_ζ *conforms linguistically* to one generic metamodel called *Extended Metamodel of Imperfection (EiMM)* that serves to represent any type, attribute and relation, where attributes and relations can be imperfect. The m_ζ does not conform ontologically to the MM_{EA} because attributes and relations of instances of the same metatype can have different characteristics. We say that the m_ζ *semi-conforms ontologically* to the MM_{EA}. We introduce the *ontological semi-conformance* concept as the relation between the m_ζ and the MM_{EA} in which the m_ζ can include instances of the metatypes in the MM_{EA}, but the instances' structural features can be imperfect and may change some of their features. Given this, we can say that the m_ζ is an approximation of the m_{EA}. Furthermore, m_ζ also includes decision trace information about the decisions that led to the construction of the imperfect model. Decision trace information is related with sources that provide information about aspects of the enterprise, observations that are the activities in which the modeler can collect enterprise information, and decision details. This decision trace information is structured through the *EiMM* that has types to represent all elements involved in the decision making process. As a result, the m_ζ conforms linguistically to *EiMM* and *semi-conforms ontologically* to the MM_{EA}.

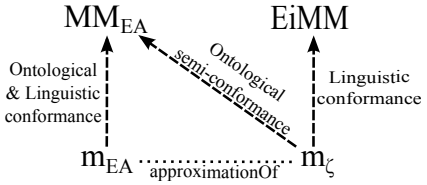

Fig. 1. Strategy for Modeling imperfection.

3 Sources of Enterprise Information

In the construction process of one model, modelers must consult enterprise sources *($\Psi = \{\psi_1, \psi_2, \ldots, \psi_r\}$)*. One source ψ_i provides information to modelers through *Observations ($\Theta_{\psi_i} = \{\theta_{\psi_i,1}, \theta_{\psi_i,2}, \ldots, \theta_{\psi_i,s}\}$)*. One observation $\theta_{\psi_i,j}$ produces some knowledge about enterprise elements that we call *Facts ($\Omega_{\theta_{\psi_i,j}} = \{\omega_{\theta_{\psi_i,j},1}, \omega_{\theta_{\psi_i,j},2}, \ldots, \omega_{\theta_{\psi_i,j},t}\}$)*. Each fact $\omega_{\theta_{\psi_i,j},k}$ provides information about instances, attributes, or relations, for building the model.

Enterprise sources could be imperfect because they can provide information that does not properly represent some elements of the enterprise. We classify sources based on the properties of the information provided as incorrect [4], imprecise [4, 7, 8, 9], inconsistent [10, 11], vague, uncertain [7, 12, 13] or a combination of some of them. In addition, it is possible that one source, which should provide information, cannot provide information about one structural feature. Then, the corresponding observation will not produce any fact. In this case, we classify the observation as an incomplete observation.

Incorrect source. One source ψ_i is incorrect when one observation $\theta_{\psi_i,j}$ over one structural feature $\varphi_{\gamma_k,l}$ establishes a fact $\omega_{\theta_{\psi_i,j},k}$ that provides a value that is not true or is not usable.

Imprecise source. One correct source ψ_i is imprecise when one observation $\theta_{\psi_i,j}$ over one attribute $\alpha_{\gamma_k,l}$ that requires a specific numeric value, provides a range of numeric values with a minimum numeric value *(v_{min})* and a maximum numeric value *(v_{max})* (e.g., the CTO (ψ_i) in one interview ($\theta_{\psi_i,j}$) asserts that the availability ($\varphi_{\gamma_k,l}$) of the *BusinessApp_X* (γ_k) is between 90% and 95%).

Inconsistent source. There are the following cases in which one source ψ_i or several sources $\{\psi_1, \psi_2, \ldots, \psi_r\}$ can be inconsistent.

Case A. Several sources $\{\psi_1, \psi_2, \ldots, \psi_r\}$ are inconsistent when one observation of each source $\{\theta_{\psi_1,i}, \theta_{\psi_2,j}, \ldots, \theta_{\psi_r,k}\}$ about the same aspect of the enterprise determines that one instance γ_l exists; however, some other observations determine that the instance γ_l does not exist.

Case B. Several sources $\{\psi_1, \psi_2, \ldots, \psi_r\}$ are inconsistent when one observation of each source $\{\theta_{\psi_1,i}, \theta_{\psi_2,j}, \ldots, \theta_{\psi_r,k}\}$ provides different values for one structural feature $\varphi_{\gamma_k,l}$ (e.g., the CIO (ψ_1) in one interview ($\theta_{\psi_1,i}$) asserts that the availability ($\varphi_{\gamma_k,l}$) of the *BusinessApp_X* (γ_k) is 92%, and the CTO (ψ_2) in one interview ($\theta_{\psi_2,j}$) asserts that the availability is 95%).

Case C. One source ψ_i is inconsistent when several observations $\{\theta_{\psi_i,1}, \theta_{\psi_i,2}, \ldots, \theta_{\psi_i,s}\}$ provide different values for one structural feature $\varphi_{\gamma_k,l}$. (e.g., the

CTO (ψ_i) in one interview ($\theta_{\psi_i,1}$) asserts that the availability ($\varphi_{\gamma_k,l}$) of the *BusinessApp_X* (γ_k) is 90%, but in other interview ($\theta_{\psi_i,2}$) asserts that is 92%).

Vague source. Vague sources are those that provide one linguistic value among a set of linguistic values $\Xi_{\varphi_{\gamma_k,l}} = \{\xi_{\varphi_{\gamma_k,l},1}, \xi_{\varphi_{\gamma_k,l},2}, \ldots, \xi_{\varphi_{\gamma_k,l},n}\}$ for one specific attribute $\alpha_{\gamma_k,l}$. One correct source ψ_i is vague when one observation $\theta_{\psi_i,j}$ provides a linguistic value to one attribute $\alpha_{\gamma_k,l}$ that requires one specific numeric value (e.g., the CTO (ψ_i) in one interview ($\theta_{\psi_i,1}$) asserts that the availability ($\varphi_{\gamma_k,l}$) of the *BusinessApp_X* (γ_k) is "High" ($\xi_{\varphi_{\gamma_k,l},r}$)).

Uncertain source. One source ψ_i is uncertain when one observation $\theta_{\psi_i,j}$ over one structural feature $\varphi_{\gamma_k,l}$ that requires a specific value, determines that it can have a value with a certainty degree. The value with a certainty degree must be a combination of a value with a probabilistic value *([v, P(v)])*. (e.g., the CTO (ψ_i) in one interview ($\theta_{\psi_i,j}$) asserts that the *BusinessApp_X* (γ_k) supports ($\varphi_{\gamma_k,l}$) the *BusinessProcess_A* with a certainty degree of 80%).

Incomplete observation. One observation $\theta_{\psi_i,j}$ of one source ψ_i that should provide information is incomplete when the observation's facts do not have any value for one structural feature γ_l that requires a specific value. Consequently, the value of the structural feature is indeterminated $\ominus$.

4 Modeling the Imperfection

Modeling the imperfection implies allowing the construction of approximate models instead of exact models regarding MM_{EA}. Then, the modeler does not build a m_{EA}, but builds a m_{ζ}. Given the facts obtained from one or several sources $\{\Omega_{\Theta_{\psi_1}}, \Omega_{\Theta_{\psi_2}}, \ldots, \Omega_{\Theta_{\psi_r}}\}$, modelers have to make decisions in order to assign values to the model's structural features. There is a set of possible decisions *(Λ=$\{\lambda_1, \lambda_2, \ldots \lambda_u\}$)* based on the information provided by the sources. Some examples of the decisions that the modeler can make are the following. λ_1: Select all the values; λ_2: Select a subset of the values; λ_3: Select one value; λ_4: Select one value with a certainty degree; λ_5: Select several values with a certainty degree; λ_6: Select a range of numeric values; λ_7: Select one linguistic value; λ_8: Calculate one value; λ_9: Do not select a value.

Decisions λ_3 or λ_8 can be used to remove the imperfection, but this means losing information and potentially creating an incorrect model with respect to the enterprise (e.g., the CTO (ψ_i) in one interview ($\theta_{\psi_i,j}$) asserts that the availability ($\varphi_{\gamma_k,l}$) of the *BusinessApp_X* (γ_k) is between 90% and 94%, but the modeler decides assigning the value 92%). To be able to model the imperfection, it is necessary to model the imprecision, inconsistency, vagueness, uncertainty, and incompleteness.

Imprecise model. One model is imprecise if at least one attribute of one instance has a range of numeric values instead of a single value. The range is given by a minimum value and a maximum value instead of just one value. The modeler can build an imprecise model due to the sources are *imprecise* (decision λ_6), or *inconsistent cases B or C* (decision λ_6).

Inconsistent model. One model is inconsistent if at least one structural feature of one instance has several values instead of just one value. The modeler can build an inconsistent model due to the sources are *imprecise* (decision λ_2), or *inconsistent cases B or C* (decisions λ_1 or λ_2).

Uncertain model. One model is uncertain if at least one structural feature $\varphi_{\gamma_k,l}$ of one instance γ_k has a value, a set of values, or a range of values with a certainty degree. The modeler can build an uncertain model due to the sources are *imprecise* (decisions λ_4 or λ_5), *inconsistent cases B or C* (decisions λ_4 or λ_5), or *uncertain* (decision λ_4).

Vague model. One model is vague if the modeler decides to asign a linguistic value instead of a numeric value to one attribute of an instance. The modeler can build a vague model due to the sources are *imprecise* (decision λ_7), *inconsistent cases B or C* (decision λ_7), *uncertain* (decision λ_7), or *vague* (decision λ_7).

Incomplete model. One model is incomplete if at least one instance's structural feature does not have any value. It might happen because the modeler decided not to assess any value because he/she does not have enough information.

5 A Tool for Modeling Imperfection

In order to achieve modeling imperfection, this proposal includes a tool to build imperfect models *(m_ζ)*. This tool is a graphical editor named *iModeler* based on *GraCoT (Graphical Co-Creation Tool)* presented in Gomez et al. [6]. The editor is based on one metamodel named *iMM (Metamodel of Imperfection)*.

5.1 Metamodel of Imperfection iMM

iMM provides a basic linguistic framework for the definition of imperfect models *(m_ζ)*. Figure 2 presents *iMM*. This metamodel has the type called `Model`, which serves as container for all other elements. The type `Element` serves to represent

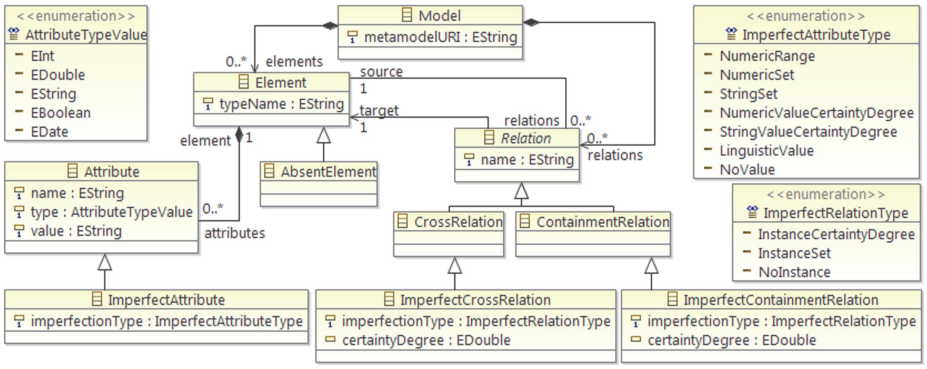

Fig. 2. Metamodel of Imperfection iMM.

the element instances of the imperfect model. Each element in a model has an attribute called `typeName` that serves to relate the element to a type in the MM_{EA}. The types `CrossRelation` and `ContaintmentRelation` serve to represent the relations between elements. The type `Attribute` serves to represent the actual values of attributes contained in elements of the m_ζ. Attribute values may only be integers, doubles, strings, booleans, or dates. The types `ImperfectAttribute`, `ImperfectCrossRelation`, and `ImperfectContainmentRelation` serve to represent, respectively, imperfect attributes and relations. Imperfect attributes may contain a range of numeric values, a set of numeric values, a set of strings, a set of numeric values with a certainty degree, a set of strings with a certainty degree, a linguistic value, or no value. Imperfect cross and containment relations may contain an instance relation with certainty degree or a set of instance relations.

5.2 iModeler Editor

The strategy described in section 2 has been implemented in a graphical editor based on Eclipse Modeling Framework Project (EMF) and Graphical Modeling Project (GMP) named *iModeler*. In addition, for the creation of *iModeler*, the project EuGENia was used in order to create the required GMP's components. *iModeler* serves to create imperfect models *(m_ζ)* that conform to *iMM*. This editor is also capable of validating the *ontological semi-conformance* of the m_ζ with respect to a MM_{EA} providing assistance to the user. The m_ζ has the appearance of an object diagram from UML (see Figure 3b), where each instance displays its metatype from the MM_{EA}, its attributes, and its relations. Imperfect attributes are decorated with a black rounded square. Imperfect containment and cross relations have respectively a filled and unfilled square in the side of the source. Another important characteristic of *iModeler* is the way it handles problems. This is achieved using EMF's validation elements, and our own validation engine based on Epsilon Validation Language (EVL)

For example, the MM_{EA} presented in Figure 3a has the types `Enterprise`, `BusinessApplication`, and `BusinessProcess`. In the construction of the m_ζ,

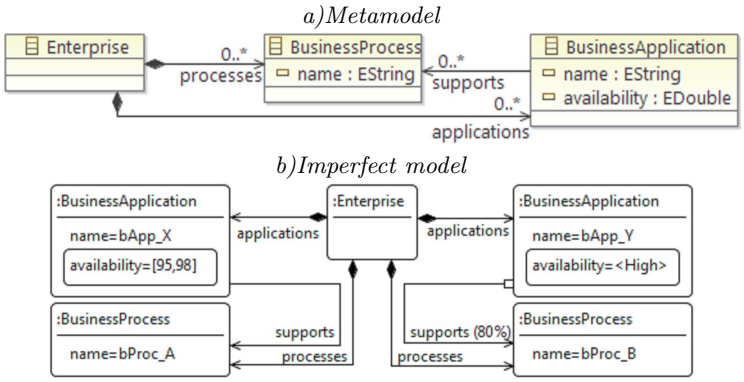

Fig. 3. Enterprise modeling example.

the modeler decides to include imperfection in some attributes and relations. The m_ζ (see Figure 3b) *conforms linguistically* to *iMM* and *semi-conforms ontologically* to MM_{EA}. It has the following instances: 1) `Enterprise` as instance of `Element` that contains all elements in the m_ζ; 2) `BusinessApplication` that represents the application *bApp_X*, created by the containment relation `applications` with the imperfect attribute `availability` with the range of values *[95,98]*; 3) `BusinessApplication` that represents the application *bApp_Y* with the imperfect attribute and relation: a) `availability` with the linguistic value <*High*>, and b) `supports` with the instance that represents the process *bProc_B*; 4) `BusinessProcess` that represents the process *bProc_A*, created by the containment relation `processes`; and 5) `BusinessProcess` that represents the process *bProc_B*. This example demonstrates several kinds of imperfection. In *bApp_X*, the attribute `availability` is imprecise; in *bApp_Y*, the attribute `availability` is vague, and the relation `supports` is uncertain.

6 Decision Trace on Imperfect Models

Modelers can include decision trace information into the m_ζ in order to represent the way in which the information was gathered, and the decisions to construct the model were made. The trace includes the enterprise sources, source consulting through observations, facts produced by observations, and decisions related with imperfect attributes or imperfect relations of the m_ζ.

In order to include all elements described above, we complemented the *iMM* creating one metamodel named *Extended Metamodel of Imperfection (EiMM)* (see Figure 4) for our tool *iModeler*; thus, the m_ζ now *conforms linguistically* to *EiMM*, The *EiMM* includes all elements presented in *iMM* and the following additional types: 1) `ModelExtension` serves as container for all other additional elements; 2) `Source` and its specializations `DirectObservation`, `Document`,

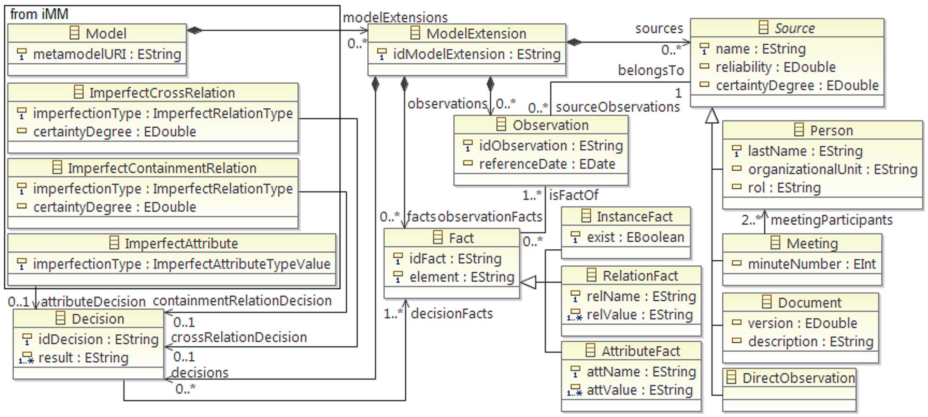

Fig. 4. Extended Metamodel of Imperfection EiMM.

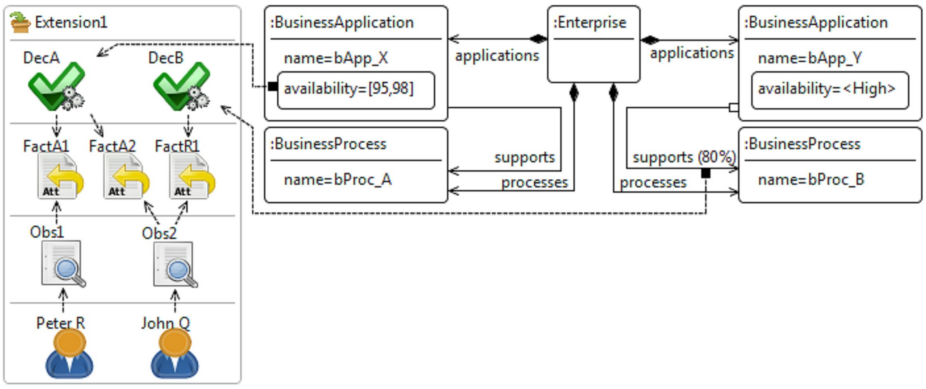

Fig. 5. Extended Imperfect Model Example.

`Meeting`, and `Person` serve to represent sources; 3) `Observation` serves to represent interviews, document revisions, and meeting acts; 4) `Fact` and its specializations `InstanceFact`, `AttributeFact`, and `RelationFact` serve to create registers of information obtained about instances, attributes, or relations of the enterprise; and 5) `Decision` serves to specify the decision made by modelers for imperfect attributes or relations. By means of the *EiMM*, *iModeler* allows modelers to include all necessary decision trace information into the m_ς.

Continuing the example presented in the previous section, the modeler decides to include decision traces for the attribute `availability` of the *bApp_X* and for the relation `supports` of the *bApp_Y*. Then, the modeler does an interview to the CTO named "Peter R", who asserts that the availability of *bApp_X* is 95%. However, in an interview to the CIO named "John Q", the information obtained is that *bApp_Y* supports *bProc_B* with a certainty degree of 80% and the availability of *bApp_X* is 98%. Consequently, the m_ς includes the following trace elements: 1) one instance of the type `ModelExtension` to include all decision trace elements; 2) two instances of the type `Person` with information about the CTO and the CIO; 3) two instances of the type `Observation` with the information of the interviews; 4) two instances of the type `AttributeFact` that specifies the availavility of *bApp_X*; 5) one instance of the type `RelationFact` that relates the *bApp_Y* with the *bProc_B*; and 6) two instances of the type `Decision` with the results of the decisions made by the modeler for the imperfect attribute `availability` and the imperfect relation `supports`. Figure 5 represents the imperfect model with decision trace of the example.

7 Conclusions

In the Enterprise Architecture context, models are built using information provided by various and heterogeneous sources. These sources usually do not have perfect information, so enterprise models might not represent the enterprise correctly; thus, analysis results based on the mentioned models can be incorrect.

Imperfect models represent and structure imperfect information while enabling modelers to keep all the information provided by sources about the elements' attributes and relations. Modelers can also include decision trace information. Based on the imperfect information and decision trace information, analysts can perform better analysis processes with a higher level of reliability.

The solution strategy presented in this paper represents the imperfection in EA models based on the creation of models that *conform linguistically* to *EiMM* and *semi-conform ontologically* with the MM_{EA}. The m_ζ is created using the tool *iModeler* that supports imperfect attributes and relations, and also allows the inclusion of decision traces.

In this work, we have focused on providing mechanics to properly manage and keep the information about imperfection. It was necessary to conceptualize these kinds of imperfection, while understanding the impact from the sources in the quality of the models. Based on these results, we will start to study how to analyze the enterprise, taking into account the mentioned imprecision while providing better conclusions.

References

1. Seidewitz, E.: What models mean. Software, IEEE **20**(5) (2003) 26–32
2. Ludewig, J.: Models in software engineering–an introduction. Software and Systems Modeling **2**(1) (2003) 5–14
3. Bézivin, J.: On the unification power of models. Software and Systems Modeling **4**(2) (2005) 171–188
4. Henricksen, K., Indulska, J.: Modelling and using imperfect context information. In: Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on, IEEE (2004) 33–37
5. Kuhne, T.: Matters of (meta-) modeling. Software and Systems Modeling **5**(4) (2006) 369–385
6. Gómez, P., Sánchez, M., Florez, H., Villalobos, J.: Co-creation of models and metamodels for enterprise architecture projects. In: Proceedings of the 2012 Extreme Modeling Workshop. XM '12, ACM (2012) 21–26
7. Smets, P.: Imperfect information: Imprecision, and uncertainty. Uncertainty Management in Information Systems **1996** (1996) 225–254
8. Hayashi, T., Wada, R.: Choice with imprecise information: an experimental approach. Theory and Decision **69**(3) (2010) 355–373
9. Li, X., Dai, X., Dezert, J., Smarandache, F.: Fusion of imprecise qualitative information. Applied Intelligence **33**(3) (2010) 340–351
10. Hunter, A., Nuseibeh, B.: Managing inconsistent specifications: reasoning, analysis, and action. ACM Transactions on Software Engineering and Methodology (TOSEM) **7**(4) (1998) 335–367
11. Hunter, A., Konieczny, S.: Approaches to measuring inconsistent information. In: Inconsistency Tolerance. Springer (2005) 191–236
12. Afshani, J., Harounabadi, A., Dezfouli, M.A.: A new model for designing uncertain enterprise architecture. Management Science **2** (2012)
13. Dai, J., Xu, Q.: Approximations and uncertainty measures in incomplete information systems. Information Sciences (2012)