Supporting Drafts for Enterprise Modeling

Hector Florez, Mario Sánchez, Jorge Villalobos
Department of Systems and Computing Engineering
Universidad de los Andes
Bogotá, Colombia
Email: {ha.florez39, mar-san1, jvillalo}@uniandes.edu.co

***Abstract*—Enterprise models are built for representing one enterprise under study. These models require a lot of information, which frequently is not totally available before starting the construction of the model. Modelers create enterprise models based on information provided by different kinds of sources through observations. However, these sources could be insufficient or the information could be incomplete or incorrect regarding specific aspects of the enterprise. As a result, the construction process of enterprise models requires the creation of drafts of models allowing modelers to create an initial draft and to refine it when new and correct enterprise information is obtained. One draft of one enterprise model is a temporal model, which can include incomplete or imperfect elements. Drafts can include supporting data, which is additional data regarding the sources observed. In this paper we present a proposal for creating and managing drafts of enterprise models using ArchiMate as modeling language.**

Keywords*—*Enterprise Architecture, Model Driven Engineering, Drafts of Models

I. Introduction

Enterprises increasingly depend on Information Technologies (IT) and require support in order to achieve their business goals. Enterprise Architecture (EA) is used as a guide to the design of enterprise organizational structure, business processes, information systems, and infrastructure [1], [2]. EA projects rely on the construction of models [3] that abstract the enterprise for understanding its organizational and technological aspects [1], [4]; typically, these models focus on structural aspects of the enterprise [5], and serve for documentation, communication, analysis, discussion, and design purposes [6]. EA models are usually big, complex and its construction has a high level of difficulty.

In the construction process of one enterprise model, a team of modelers identifies and classifies several available sources of enterprise information (e.g., persons, documents, meetings). Modelers obtain the required enterprise information consulting aforementioned sources through observations (e.g., interviews, reviews, meeting acts). They start the modeling process using one modeling language such as ArchiMate [7], which is one of the most extended enterprise modeling language, while being a graphical language defined with one metamodel that incorporates active structure elements, behavior elements, and passive structure elements.

Sources of enterprise information have different levels of precision and reliability. Usually, it is possible to find several sources with contradictory information, imprecision, opposite points of view, conflicts of interest, lack of information, and other kind of situations that obstruct the EA model construction. In addition, it is possible to find information that was true in the past, but sources cannot confirm whether it remains valid. Next, modelers must extract, consolidate, and interpret information from several sources in order to express the information in the model. This task requires the creation of the EA model through drafts that allow creating the model including additional information regarding the modeling process (e.g., which sources provided the information, when the information was obtained, which information is not obtained yet). As a result, drafts do not conform to the metamodel of the modeling language (e.g., ArchiMate); then, the modeling tool used to create these drafts should be able to support this condition.

One draft of one enterprise model is a temporal model, which can include incomplete elements or imperfect information [8] (i.e., imperfect elements, imperfect attributes, and imperfect relations). In the modeling process, modelers create drafts of the enterprise model based on the information obtained. Then, they validate and refine these drafts based on the necessary sources in order to complete the model. In addition, drafts can include supporting data, presenting information regarding the sources consulted by the modeler, when he/she decides to create one incomplete or imperfect element. The supporting data helps modelers to understand how the elements in the model can be refined and can present 1) an interpretation of the information obtained by sources; 2) the time stamp for informing when the information was obtained; 3) the source reliability; and 4) the certainty level of a source. Once the elements in the draft are refined, the correspondent supporting data can be deleted in order to produce the final EA model.

In this paper we present a proposal for creating and managing drafts of enterprise models using ArchiMate as modeling language. The proposal has been implemented in *iArchiMate*, which is a modeling tool that allows creating drafts using one graphical editor, while including supporting data for all elements using another graphical editor. The supporting data includes specific components for describing 1) sources of enterprise information, 2) observations of sources made by the modeler, and 3) facts provided by observations.

The rest of the paper is structured as follows. Section II describes enterprise modeling process in the EA context and presents the kinds of sources of enterprise information. Section III describes drafts of enterprise models and presents our solution strategy for creating and managing drafts. In section IV, we present our tool for creating and managing drafts using ArchiMate as modeling language. Finally, section V concludes the paper.

978-1-4799-6717-9/14/$31.00

II. Enterprise Modeling

EA projects require the construction of one model to represent the relation between business and IT [5]. The EA model must conform to one metamodel, which abstracts the enterprise concepts through typed elements, which contain attributes and relations. One EA model is usually big because enterprises have a large number of elements and it is complex because they have a large number of typed relations between their elements. The EA model 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, meetings). In the construction process of the EA model, modelers consult sources through observations (e.g., interviews, reviews, meeting acts), where each observation provides facts, which provide the enterprise information to create elements, relations, and attributes in the model. EA models require a modeling language such as ArchiMate, which has become the standard language for describing and visualizing EA models involving different domains. It has been designed to provide a complete graphical representation of EAs over time and is closely linked to TOGAF standard [9].

In most cases, it is impossible to obtain all required information before creating the correspondent EA model. This condition is presented for several different reasons. Some of these reasons are: 1) enterprise information is provided by several different sources (e.g., executive employees, technical employees, contracts); 2) several specific information must be provided by more than one source and those sources provide contradictory information (e.g., the CIO asserts that one new server is already installed, but one technical employee asserts that the server has not been delivered by the provider); 3) sources that should provide specific information, are not able to provide it (e.g., the CTO does not know which device is used by the CRM); 4) sources provide information that does not properly represent some elements of the enterprise (e.g., the CTO asserts that the availability of one device, which must be a number, is "High"); 5) sources are not reliable for providing specific information (e.g., the CEO provides information related with the technology department). In Florez et al. [8], sources, which provide information that cannot be used to create the model correctly have been called *imperfect sources* and have been classified as *incorrect*, which are sources that provide false information; *imprecise*, which are sources that provide ranges of values instead of a unique value for one specific numeric attribute; *inconsistent*, which are sources that provide more than one value for one attribute or relation; *vague*, which are sources that provide linguistic values instead of a numeric value for one specific numeric attribute (e.g., availability="High"); and *uncertain*, which are sources that provide one value with a certainty degree. In addition, observations have been classified as *incomplete*, when the source does not provide any information.

III. Drafts of Enterprise Models

The construction of one EA model is a complex task and could require a long time to be completed. When modelers decide to build an EA model, they obtain enterprise information from aforementioned sources (which can be imperfect) through observations. However, due to the size of an enterprise, usually modelers start their model construction before obtaining all required information. Then, modelers start creating a draft of the enterprise model, which is temporal and could include additional data regarding its incomplete elements or its imperfect information. We call the additional information *supporting data* as documentation in the model that contains details about the sources, observations, and facts involved in the incomplete or imperfect component. In this work, all drafts conform to ArchiMate metamodel. Thus, each draft contains elements that represent ArchiMate concepts (e.g., BusinessProcess, ApplicationComponent), relations that represent ArchiMate relations (e.g., UsedBy, Flows), and attributes that can belong to elements or relations.

Based on the supporting data, modelers can understand the reasons why such a draft has been created; thus, he/she can identify which new source can be observed or which new observation of one already observed source can be made in order to obtain the required information for refining the draft. The supporting data can contain 1) sources such as *Persons*, *Meetings* (which have associated the correspondent persons), *Documents*, and *Direct Observations* (forming information regarding the enterprise provided by the modeler); 2) *Observations*, which determines the time stamp where the source was consulted; and 3) facts such as *Instance Fact*, *Attribute Fact*, and *Relation Fact*, which allow to document the values obtained from observations. Then, based on the supporting data, modelers can have specific information for refining the draft. Once the modeler refines one element in the draft by completing it or by removing its imperfection, the correspondent supporting data can be removed as well. At the end of the modeling process, when the draft does not have incomplete or imperfect elements and does not have supporting data, the draft becomes the final enterprise model.

A. Related Work

Some approaches have been based on the use of drafts in order to properly obtain their results. For instance, Erol et al., [10] present a proposal for collaborative drafting of business processes models. In this work, the authors assert that modeling in a general sense is a process that could consist the following activities: 1) elicitation, which refers to the act of collecting information from domain experts; 2) modeling, which is the transformation of the informal specification into a formal specification; and 3) validation, which refers to the act of evaluating the congruence of the formal specification regarding the informal specification. Based on these activities, this work presents a case study of a wiki supported drafting of process models, which includes a wiki engine developed to support collaboration in processes for building models.

Some other approaches allow the creation of enterprise models with different levels of abstraction. For instance, the work of Frank [11] proposes a multi-perspective enterprise modeling *(MEMO)*, which is an approach able to represent different perspectives of the enterprise. MEMO offers a framework that includes common enterprise abstractions. The enterprise perspectives are represented by the following languages: 1) *Strategy Modeling Language (MEMO-SML)* which includes concepts from strategic planning; 2) *Organization Modeling*

Language (MEMO-OrgML) which serves to model organizational concepts; and 3) *Object-Oriented Modeling Language (MEMO-OML)* which allows the specification of information. MEMO architecture contains the specification and integration of modeling languages. The architecture is extensible allowing the specification of additional languages. Although MEMO does not take the creation of drafts nor the representation of imperfect information into consideration, its multi-level extensibility allows to include this imperfect information by updating the metamodels. Also, metamodels can be updated in order to include information for documenting the construction process of the enterprise model.

B. Solution Strategy

In order to build drafts using ArchiMate as modeling language, we use the distinction between *ontological conformance* (based on the relation between the model and metamodel in terms of their meaning) and *linguistic conformance* (based on the relation between the model and metamodel in terms of their structure) [12]. 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 [8]. Figure 1 illustrates the proposed strategy. Modeling process starts with the creation of one draft μ_0. This draft is refined by the iteration γ_1, which produces the μ_1. The iteration γ_n produces the μ_n. All drafts $M=\{\mu_0, \mu_1 \ldots \mu_n\}$ *conforms linguistically* to one generic metamodel called *Intermediate Metamodel (iMM)* (See Figure 2). Each of μ_i does not conform ontologically to the ArchiMate metamodel, but it *semi-conforms ontologically* to the ArchiMate metamodel. *Ontological semi-conformance* is the relation between μ_i and the ArchiMate metamodel in which instances of the μ_i can include imperfect information [8]. Moreover, in order to include supporting data, *iMM* has been extended in one metamodel called *Extended Intermediate Metamodel (EiMM)* (See Figure 3), which serves to represent sources, observations, and facts.

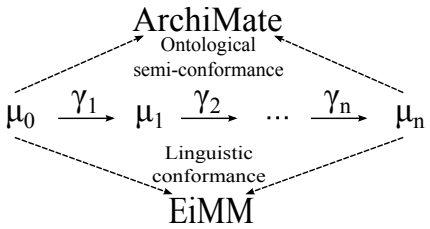

Fig. 1. Strategy for Creating and Managing Drafts of Models.

IV. Tool for Creating and Managing Drafts of ArchiMate Models

This proposal includes a tool to build drafts of ArchiMate models. This tool named *iArchiMate* is based on the Eclipse Modeling Framework Project (EMF)[1] and the Graphical Modeling Framework Project (GMF)[2]. In addition, for the creation of the required GMF components, the project EuGENia[3] was used. *iArchiMate* serves to create drafts of ArchiMate models (μ_i) that conforms to *EiMM*. This editor is also capable of validating the *ontological semi-conformance* of the μ_i regarding the ArchiMate metamodel providing assistance to the user. *iArchiMate* also allows importing and exporting models compatible with the tool Archi[4]. This function has been implemented to take advantage of the great amount of EA models already created using this tool.

A. Intermediate Metamodel iMM

iMM, which is presented in Figure 2, provides a basic linguistic framework for the definition of drafts of ArchiMate models (μ_i). *iMM* has the type called `Model` which contains all other elements. The abstract type `Component` is specialized by the types `Group`, which serves to create groups in the draft and `Element`, in order to represent element instances of the draft. In the type `Element`, there is an attribute named `typeName` with the attribute type `ElementTypeName` that is an enumeration, which contains the name of all possible elements in the ArchiMate metamodel (e.g., BusinessProcess, ApplicationComponent). The type `Relation` serves to represent relations between elements. In the type `Relation`, there is an attribute named `typeName` with the attribute type `RelationTypeName` that is an enumeration, which contains the name of all possible relations in the ArchiMate metamodel (e.g., UsedBy, Flows). The type `Attribute` serves to represent the actual values of attributes contained in elements and relations of the draft. The types `AbsentElement`, `ImperfectAttribute` and `ImperfectRelation` serve to represent, respectively, imperfect elements, imperfect attributes, and imperfect relations of the draft. The imperfection type of imperfect attributes and relations (e.g., range of values, linguistic value, instance set) are specified by the attribute `imperfectionType`.

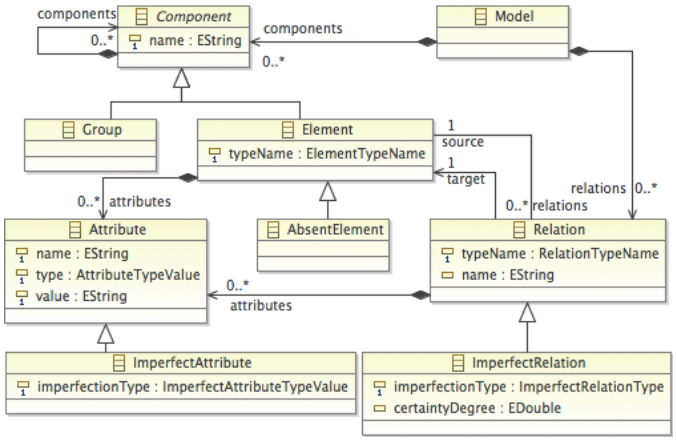

Fig. 2. Intermediate Metamodel iMM.

B. Supporting Data on Drafts of ArchiMate Models

Modelers can include supporting data on drafts in order to represent the way in which the information to construct the draft was gathered. Supporting data includes the enterprise sources, observations that refer to source consulting, and facts produced by observations. In order to include all elements described above, we complemented the *iMM* creating one

[1] http://www.eclipse.org/modeling/emf/
[2] http://www.eclipse.org/modeling/gmp/
[3] http://www.eclipse.org/epsilon/doc/eugenia/
[4] http://archi.cetis.ac.uk/

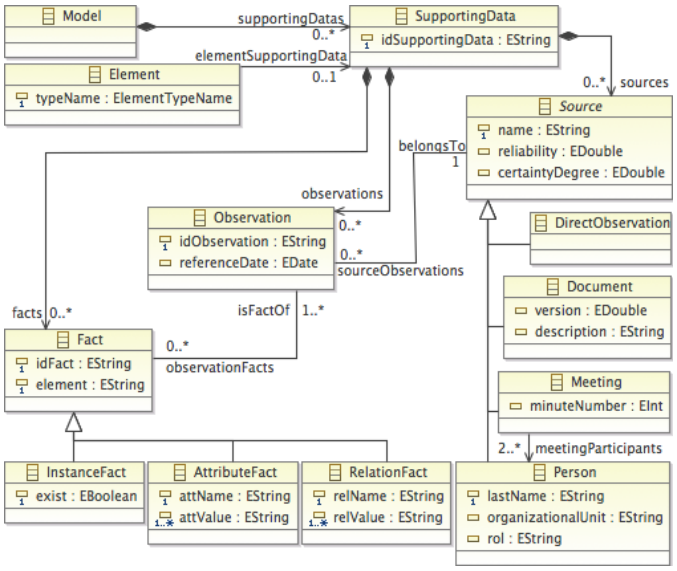

Fig. 3. Extended Intermediate Metamodel EiMM.

metamodel named *Extended Intermediate Metamodel (EiMM)* (see Figure 3) for the tool *iArchiMate*; thus, the draft (μ_i) now *conforms linguistically* to *EiMM*. The *EiMM* includes all elements presented in *iMM* and the following additional types: 1) `SupportingData` serves as container for all other additional elements; 2) `Source` and its specializations `DirectObservation`, `Document`, `Meeting`, and `Person` serve to represent sources; 3) `Observation` serves to represent interviews, document revisions, and meeting acts; and 4) `Fact` and its specializations `InstanceFact`, `AttributeFact`, and `RelationFact` serve to create registers with information obtained about elements, attributes, or relations of the enterprise. By means of the *EiMM*, *iArchiMate* allows modelers to include all necessary supporting data into the draft. Thus, each element can have one supporting data component, for specifying all data related with the enterprise information obtained, when the modeler decides to include a) the correspondent element as imperfect element or b) imperfect attributes, or imperfect relations in the correspondent element, or c) imperfect attributes in one relation of the correspondent element.

C. iArchiMate Editors

iArchiMate has been designed and developed in order to provide two editors: one main editor (See Figure 4) for modeling drafts of ArchiMate models; and one secondary editor (See Figure 6) for modeling the supporting data of the draft. In the main editor, modelers can include several instances of one specific component representing the supporting data, which is used to open the secondary editor (See Figure 5).

The editor for modeling drafts has the following characteristics: 1) all elements are drawn with rounded squares that include the correspondent icon and color established in the ArchiMate specification; 2) all relations have the graphical representation established in the ArchiMate specification; 3) attributes, which can belong to elements or relations, can be displayed through an additional tab in the properties view; 4) imperfect elements are drawn with blue squares that include the correspondent ArchiMate icon; 5) imperfect relations are drawn with a blue color and the correspondent ArchiMate representation; 6) imperfect attributes are displayed through an additional tab in the properties view; 7) components for opening the supporting data editor are drawn with gray rounded squares; and 8) relations from elements to supporting data components are drawn with a blue dashed arrow that contains a filled square in the source.

The editor for modeling supporting data can be opened through a double click on the supporting data component placed in the draft. This editor allows the inclusion of all elements related with supporting data, where each element is differentiated by icons and contains the required attributes for documenting all possible information regarding the enterprise information obtained by the enterprise sources. In addition, relations are drawn by dashed arrows and can only be placed from sources to observations and from observations to facts.

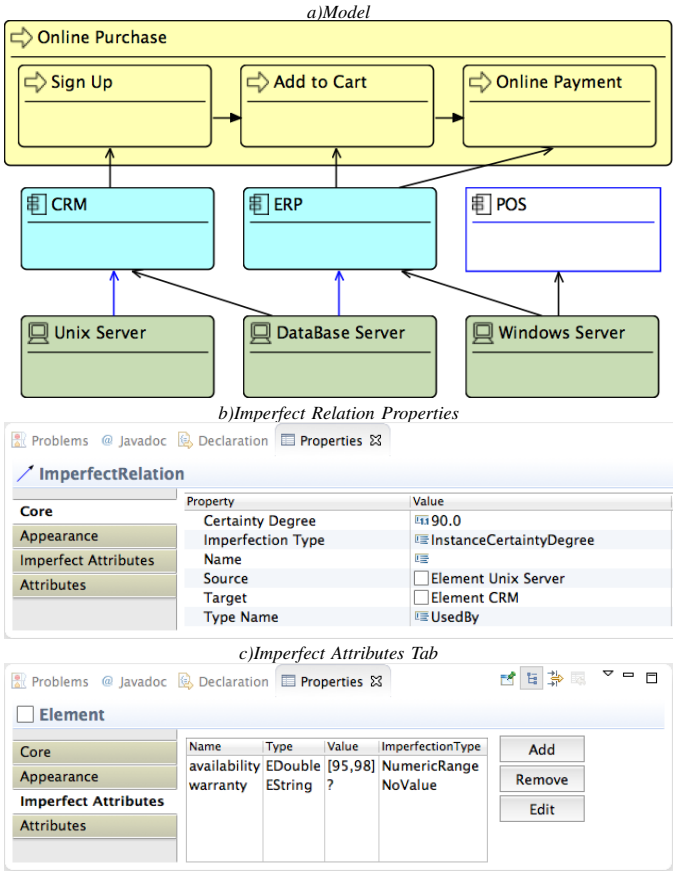

Fig. 4. Example of Draft of Enterprise Model.

Figure 4 presents one example of one draft of ArchiMate model. In this example, the modeler wants to create one enterprise model for documenting the characteristics of the business processes, application components, and devices. The modeler received the following information. 1) The *CTO*, whose name is *"John Q"* asserts that the `Device` *DataBase Server* has one year of `warranty` and its `availability` is *95%*. 2) The *CIO*, whose name is *"Paul R"* asserts that the `Device` *DataBase Server* has two years of `warranty` and it is only `UsedBy` the `ApplicationComponent` *CRM*, which is `UsedBy` the `BusinessProcess` *Sign Up*.

3) The *Purchase Contract* of the *DataBase Server* denotes that its `availability` is *98%* and that it is configured to be `UsedBy` the `ApplicationComponent` *ERP*. 4) In one *Technology Committee*, its members determined that the `Device` *Windows Server* is `UsedBy` the *ERP*, which is `UsedBy` the `BusinessProcess` *Add to Cart* and *Online Payment*. Also, the *Windows Server* is `UsedBy` the `ApplicationComponent` *POS*; however, they are not sure whether the *POS* has already been used. 5) The *Server Administrator*, whose name is *"Petter S"* asserts that the `Device` *Unix Server* is `UsedBy` the *CRM* with *90%* of certainty degree. 6) In one *Business Committee*, its members determined that the `BusinessProcess` *Online Purchase* is composed by the processes *Sign Up*, *Add to Cart*, and *Online Payment*. Then, the modeler based on the enterprise information obtained through those enterprise sources, creates a draft presented in Figure 4a, where the relations `UsedBy` from the `Device` *Unix Server* to the `ApplicationComponent` *CRM* and from the `Device` *DataBase Server* to the `ApplicationComponent` *ERP* are imperfect because they have associated a certainty degree (See Figure 4b); the `Device` *DataBase Server* has the imperfect attributes `availability` with a numeric range value, and `warranty` with no value (See Figure 4c); and the `ApplicationComponent` *POS* is an imperfect element because its existence is not confirmed.

Continuing with the example, the modeler wants to include supporting data for those elements that contain any imperfect information. Then, the modeler includes in the draft three supporting data components. The first supporting data is related with the *Unix Server* because it contains the imperfect relation `UsedBy` to the *CRM*. The second supporting data is related with the *DataBase Server* because it contains the imperfect attributes `availability` and `warranty`, and the imperfect relation `UsedBy` to the *ERP*. The third supporting data is related with *POS* because it was placed in the draft as imperfect element (See Figure 5).

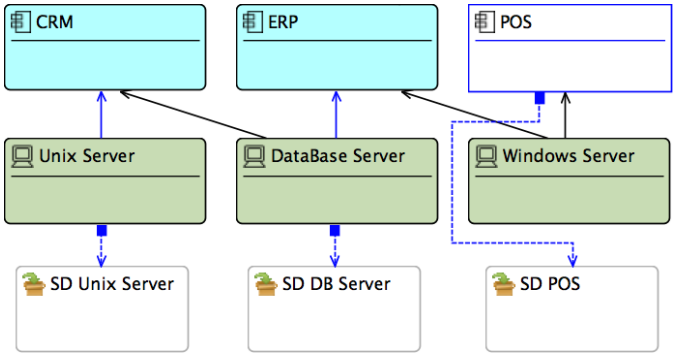

Fig. 5. Example of Draft with Supporting Data

Figure 6 presents the complete supporting data for the `Device` *DataBase Server* presented in Figure 5. In this case, the modeler obtained the enterprise information related with the *DataBase Server* through the following sources: the *CTO "John Q"*, the *CIO "Paul R"*, and the *Purchase Contract* of the *DataBase Server*. The `Person` *"John Q"* provided in the `Observation` *Interview1* the `AttributeFact` *AttFact1* that includes information about the `availability` and the `AttributeFact` *AttFact2* that includes information about the `warranty`. The `Person` *"Paul R"* provided in the `Observation` *Interview2* the `AttributeFact` *AttFact3* that includes information about the `warranty` and the `RelationFact` *RelFact1* that includes information about the relations `UsedBy` to the *CRM* and *ERP*. The `Observation` *Review1* of the `Document` *Contract* provided the `AttributeFact` *AttFact4* that includes information about the `availability` and the `RelationFact` *RelFact2* that that includes information about the relation `UsedBy` to the *ERP*. Based on all facts, the modeler included in the draft the following information: 1) the imperfect attribute `availability` with the range of values *[95,98]*; 2) the imperfect attribute `warranty` with no value represented with the symbol *?*; 3) the imperfect relation `UsedBy` to the *ERP*; and 4) the relation `UsedBy` to the *CRM*.

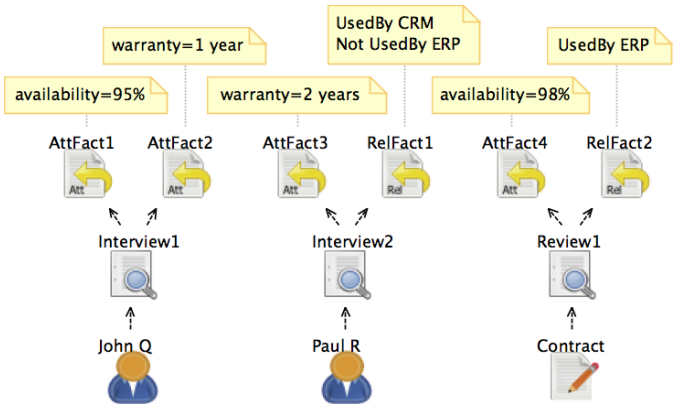

Fig. 6. Example of Supporting Data

D. Conformance Validation

iArchiMate handles *linguistic conformance* using EMF's validation engine, and *ontological semi-conformance* using *iArchiMate*'s validation engine, which is based on Epsilon Validation Language (EVL)[5]. Also, *iArchiMate*'s validation engine includes several rules for validating the correct format for imperfect elements, attributes and relations.

Listing 1 presents a fragment of the validation script for validating *ontological semi-conformance* with the ArchiMate metamodel. In this validation, we verify the attribute `typeName` in elements and relations, and sources and targets for relations. It has the following lines: 1) Lines 1 and 10 establish the EClass `Element` and `Relation` respectively from the *EiMM* as context. 2) Lines 2, 11, and 18 set the names of the rules. 3) Lines 3 to 6 and 12 to 15 check the correct value for the attribute `typeName`. 4) Lines 7 and 16 present messages for informing the wrong selection of the `typeName`. 5) Lines 19 to 33 validate the correct source and target of `Composition` relations. In the listing we include only two conditions and we only present the rule for the relation `Composition`; nevertheless, there are several conditions and there is one rule for each possible ArchiMate relation (e.g., UsedBy, Triggering). 6) Line 34 presents a message for informing that the relation `Composition` cannot be created with the source and target assigned.

[5]http://www.eclipse.org/epsilon/doc/evl/

Listing 1. Fragment of Validation Script

```
context Element {
  constraint hasRightElementTypeName {
    check{
      if (self.typeName.name == 'NotSelected')
        {return false;} else {return true;}
    }
    message: 'The required feature \'typeName\' of ↙
        ↳ \'<Model>::<Element>\' cannot have the value ↙
        ↳ \'NotSelected\''
  }
}
context Relation {
  constraint hasRightRelationTypeName {
    check{
      if (self.typeName.name == 'NotSelected')
        {return false;} else {return true;}
    }
    message: 'The required feature \'typeName\' of ↙
        ↳ \'<Model>::<Relation>\' cannot have the value ↙
        ↳ \'NotSelected\''
  }
  critique validateComposition {
    check {
      if(self.typeName.name == 'Composition') {
        if (self.source.typeName.name == ↙
            ↳ self.target.typeName.name) {
          return true;
        }else if(self.source.typeName.name == ↙
            ↳ 'BusinessRole') {
          var validTargetNames = Collection {
          'BusinessCollaboration',
          'BusinessInterface'};
          if (validTargetNames.includes( ↙
              ↳ self.target.typeName.name))
            return true;
        }
        return false;
      }
      return true;
    }
    message : 'Cannot create Composition from ' + ↙
        ↳ self.source.typeName.name + ' to ' + ↙
        ↳ self.target.typeName.name
  }
}
```

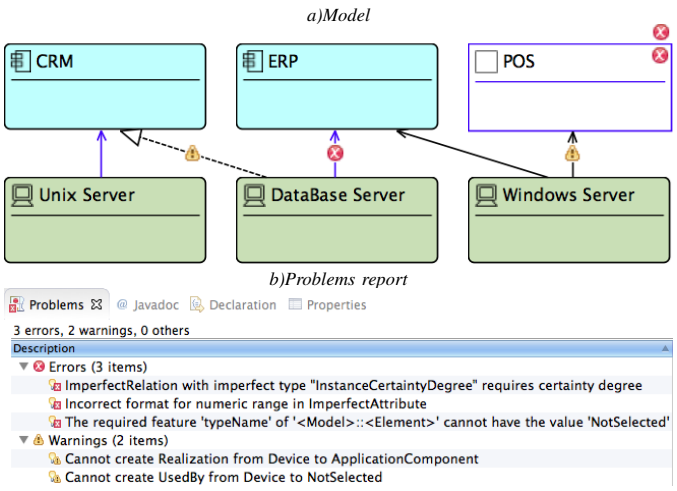

Fig. 7. Example of iArchiMate validation.

Figure 7a presents an example of the validation result. In this example, there are the following problems: 1) the imperfect relation `UsedBy` from the *DataBase Server* to the *ERP* does not have assigned certainty degree; 2) the imperfect attribute `availability` in the *DataBase Server* has format error. 3) the imperfect element named *POS* does not have selected any ArchiMate concept; 4) the `Realization` from the *DataBase Server* to the *CRM* is not allowed in the ArchiMate specification; and 5) the relation from the *Windows Server* cannot target to the element named *POS* while the target element does not have one ArchiMate concept assigned. Figure 7b shows the *Problems view* with details about the correspondent errors and warnings.

V. Conclusions

In the EA context, the ArchiMate specification allows creating models that satisfy the characteristics of the enterprises. However, models are built using information provided by various and heterogeneous sources. These sources usually do not have all required or completely reliable information, so enterprise models might not represent the enterprise correctly.

Drafts are temporal models that can represent and structure incomplete and imperfect information while enabling modelers to keep all the information provided by sources about elements, attributes, and relations. In addition, modelers can also include supporting data in order to provide further documentation that is useful to understand how the draft has been created and how it can be refined.

The solution strategy presented in this paper allows creating drafts of ArchiMate models that *conform linguistically* to *EiMM* and *semi-conform ontologically* with the ArchiMate metamodel. Drafts are created using the tool *iArchiMate*, which is composed with one editor that supports imperfect elements, attributes and relations, and one editor that supports the creation of supporting data.

References

[1] M. Lankhorst, *Enterprise architecture at work: Modelling, communication and analysis*. Springer, 2013.

[2] M.-E. Iacob and H. Jonkers, "Quantitative analysis of enterprise architectures," *Interoperability of Enterprise Software and Applications*, 2006.

[3] M. Buschle, J. Ullberg, U. Franke, R. Lagerström, and T. Sommestad, "A tool for enterprise architecture analysis using the prm formalism," in *Information Systems Evolution*. Springer, 2011, pp. 108–121.

[4] R. Lagerström, U. Franke, P. Johnson, and J. Ullberg, "A method for creating enterprise architecture metamodels–applied to systems modifiability analysis," *International Journal of Computer Science and Applications*, vol. 6, no. 5, pp. 89–120, 2009.

[5] S. Buckl, M. Buschle, P. Johnson, F. Matthes, and C. M. Schweda, "A meta-language for Enterprise Architecture analysis," *Enterprise, Business-Process and Information Systems Modeling*, pp. 511–525, 2011.

[6] S. Kurpjuweit and R. Winter, "Viewpoint-based Meta Model Engineering." in *Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures*, 2007, pp. 143–161.

[7] The Open Group, *ArchiMate 2.0 Specification*. Van Haren Publishing, 2012.

[8] H. Florez, M. Sánchez, and J. Villalobos, "Embracing Imperfection in Enterprise Architecture Models," in *Proceedings of the 6th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling (PoEM 2013)*. ACM, 2013, pp. 8–17.

[9] The Open Group, *TOGAF Version 9.1*. Van Haren Publishing, 2011.

[10] S. Erol and G. Neumann, "A Case-Study of Wiki-Supported Collaborative Drafting of Business Processes Models," in *Business Informatics (CBI), 2013 IEEE 15th Conference on*, 2013, pp. 382–390.

[11] U. Frank, "Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges," *Software & Systems Modeling*, pp. 1–22, 2012.

[12] T. Kuhne, "Matters of (meta-) modeling," *Software and Systems Modeling*, vol. 5, no. 4, pp. 369–385, 2006.