Modeling and Analyzing Imperfection in Enterprise Models

Hector Florez, Mario Sánchez and Jorge Villalobos

Abstract—**Nowadays, Enterprise Models are very important assets because they allow documentation, communication, and analysis about the current state and desirable future states of an enterprise. These models are built supported on certain information offered by enterprise sources with different characteristics. Nevertheless, this information can be insufficient, incomplete, or incorrect, as sources might fail to accurately describe certain facts of the enterprise. For this reason, we argue that Enterprise Models carry an inherent imperfection that affects the conclusions that we can draw from these models and their analysis. In this article, we propose an approach to allow supporting the understanding and correction of possible imperfection points in an enterprise model through measurements with quantitative analysis techniques, based on domain-specific and topological properties of the model. For this purpose, domain and topological analysis techniques have been developed on top of the tool *iArchiMate*, in order to support the identification and resolution of model imperfection.**

Index Terms—**Enterprise Models, Enterprise Modeling, Enterprise Analysis.**

I. Introduction

HUMAN brains interpret reality by taking information from sensory organs and making a model of the world based on such information [1]. Modeling is an essential activity for humans because every action in common life is preceded by the explicit or implicit construction of a model. Thus, if the model is incorrect, the action may be inappropriate [2]. Humans create mental models observing the reality in different ways. Most humans look directly at the reality, but blind humans can make their observations by listening or touching the reality; however, humans can also make models using information provided by other humans, documents, and other kinds of sources of information.

In the last decades, computer systems have gained great importance around the world. In this area, it is possible to represent the concepts of a system through a conceptual model. Moreover, conceptual models can be specialized in order to provide a representation of a specific context such as the enterprise context. Thus, enterprise models are used to abstract and represent various domains of enterprises; nevertheless, these models get special importance, when they are used to analyze the enterprise.

Enterprise Models allow representing Information Technologies (IT) and business elements as well as the relation between those elements in an enterprise under study. These models are useful for understanding enterprise systems through proper abstractions of the business and technological perspectives of an enterprise [3], [4], [5]. They also serve for multiple purposes such as documenting, communicating, diagnosing, analyzing, and designing [6] the architecture of the enterprise. Their construction is based on human system observation [2], supported by different information sources such as interviews, diagrams, and reports. In this construction process, modelers must identify and classify these sources. Later on, modelers capture the source knowledge using the syntax and semantics of a modeling language (e.g., ArchiMate).

The difficulty of constructing a model that properly represents the enterprise lies in certain factors that might affect the modeling process, such as the lack of sufficient information or the quality of the information [7]. For example, if the modeler asks an infrastructure technician about the availability of a device, it is possible that he is uncertain about the up-time of such a device, e.g., by claiming it is between 95% and 99%. In this case, the modeler might require one specific value for the attribute; he could then decide to approximate this availability as the average of both values (e.g., 97%), which in some cases can be an incorrect assumption.

Thus, despite the modelers' best intentions, enterprise models are inevitably imperfect abstractions of the enterprise, as the accumulated information could be inaccurate or missing. Modelers start the construction process of an enterprise model by creating drafts, which are incomplete models based on the initial information obtained [8]. These drafts are gradually refined and enriched based on the available sources; thus, imperfect information might be introduced at any moment of the modeling process.

In this work, we focus on models that describe the current state of the enterprise and we argue that it is better to create models that explicitly describe this imperfection, instead of assuming that the model accurately represents a desired state of the enterprise. Following the example above, it is desirable that the modeler can state the uncertainty of the source, (e.g., with a range of values *[95% - 99%]*), instead of being forced to provide just one value, in order to avoid incorrect assumptions.

Identifying and measuring the imperfection of a model can determine whether the model is useful for further purposes such as documentation, communication, or analysis. Thus, imperfection analyses allow finding the impact of an imperfect element or relationship to desired business analyses. For instance, if one model built based on ArchiMate has a high imperfection level in the elements that belong to the application layer, it is very likely to have a high impact on the accuracy of business analyses that are based on these elements. In addition, imperfection analyses might provide information for assessing how sensitive one specific business analysis is regarding the imperfection level in the model.

Detecting imperfection while developing enterprise mod-

Manuscript received June 24, 2020; revised January 21, 2021.

Hector Florez is Full Professor at the Universidad Distrital Francisco Jose de Caldas, Bogota, Colombia. E-mail: haflorezf@udistrital.edu.co

Mario Sánchez is Associate Professor at the Universidad de los Andes, Bogota, Colombia. E-mail: mar-san1@uniandes.edu.co

Jorge Villalobos is Full Professor at the Universidad de los Andes, Bogota, Colombia. E-mail:jvillalo@uniandes.edu.co

els can be an onerous task, given the size and complexity of these models, as well as the different types of imperfection that we can encounter: imprecision, inconsistency, vagueness, uncertainty, or incompleteness [9]. Nevertheless, it is desirable to discover flaws in the model as early as possible. In order to correct them, it is necessary to make them visible. Then, we propose to use quantitative analyses, which is composed of domain analysis and topological analysis to measure *how much* imperfection a model has. Then, the results provided by these analysis techniques allow us to determine the model's requirements for refinement and to identify new necessary information.

The rest of the article has the following structure. Section II briefly describes the concepts and processes of enterprise modeling. Section III presents some fundamental concepts about enterprise analysis. Section IV describes imperfection in enterprise models, while our approach for modeling the imperfection is presented in Section V. In section VI, we present our analysis techniques for analyzing the imperfection. Section VIII presents a discussion of the presented work, followed by Section IX, which discusses related work, and Section X, which presents the conclusions.

II. Enterprise Modeling

Enterprise modeling focuses on the use of multiple modeling languages to describe and specify all desired components of an organization in a coherent manner, as well as the relationships between these components [10]. Models produced with these languages are useful for understanding the business and technological domains of an enterprise [11]. Their development is a cooperative process that involves domain experts, modelers, and analysts [12].

The enterprise modeling process starts when a **domain expert** understands the enterprise requirements and based on them produces some concerns about the enterprise. These enterprise requirements allow guiding experts to identify insights regarding business issues and goals. Then, domain experts select the relevant modeling languages needed to address these concerns [13]. Usually, these languages are public specifications (e.g., Business Process Modeling and Notation BPMN, ArchiMate, Business Motivation Model BMM) defined by committees and expert groups that produce a language specification. However, when existing languages are not satisfactory to address these concerns, the domain expert can create enterprise-specific languages.

Once a modeling language with the corresponding metamodel is chosen, a **modeler** identifies and consults the **enterprise sources** that provide relevant data about these concerns. The information provided by each of these sources is extracted, consolidated, and interpreted, using the entities and relationships specified in the metamodel. Then, the modeler starts creating the enterprise model that implies creating elements, relationships, and attributes in elements or relationships.

An enterprise model can go through several rounds of refinement. As soon as the model is finished (i.e., the model includes the desired elements, relationships, and attributes of the enterprise), an **analyst** starts analyzing the enterprise by interacting with the model, usually by view generation, by the formulation of queries, or by running specialized functions while calculating a metric or while enriching the model. Finally, the analyst draws some conclusions, based on the observations and insights that come from his interaction with the model [14].

The quality of the analysis depends on the quality of the information that is present in the model. If this information is not accurate for a set of concerns, the analyses made with this model might be unreliable [15]. Then, we introduce the notion of *imperfect models* to express inaccurate or missing attributes, elements, and relationships. Creating an imperfect model, for representing certain inaccurate information avoids wrong interpretations regarding the correctness of the model i.e., one model might be considered accurate despite it is not, but one imperfect model is not an inaccurate model because it represents certain problems through imperfect information that is included with a specific notation. In this way, modelers can identify imperfection points, and either correct them, or take this imperfection into account in further uses.

A. Enterprise Modeling Requirements

Enterprise modeling demands a minimum set of requirements that DSMLs should accomplish to provide the necessary characteristics in order to be able to properly create, update, and manage large enterprise models.

- **Extensibility**. DSMLs must be extensible in order to warrant that enterprise models may include further elements, which are not defined in the DSML after the model has been created.
- **Flexibility**. Enterprise models might be upgraded; then, the DSML needs to be flexible in order to provide the mechanisms to manipulate properly the conforming models.
- **Modularity**. Since models are big and complex, they used to be represented in smaller and simpler fragments. It means that one model can have different kinds of interconnected modules, which each one represents a particular aspect of the enterprise.
- **Points of view**. Enterprise models may offer stakeholders access to specific elements that may come from different modules, in order to provide desirable portions of the model that are useful to achieve intended goals.
- **Separation of Concerns**. Different stakeholders have different backgrounds; thus, an enterprise might have various models, where each model satisfies specific concerns. Thus, models should also include concepts to facilitate clear communication for diverse groups of stakeholders [16].

B. Enterprise Modeling Domains

Enterprise modeling includes four domains, where each domain might be represented by multiple enterprise models [17]. Models from different domains can be used for different purposes. An enterprise model of a specific domain can include various modules to fragment and facilitate its comprehension as well as points of view to observe desired elements of the model. These domains are:

- *Business Domain* presents a description of the enterprise in order to provide an understanding of the organization. The goal of this domain is to relate all business elements such as business goals, business processes, business

functions, business capabilities, business actors among others.

- *Information Domain* provides a common business language to enable consistent communication between enterprise systems by defining kinds of data to provide a basic vocabulary and assigning critical attributes of the enterprise [18]. This domain includes data models, which provide a well defined data structure used by the organizational information systems [19]. There are three different types of data models: conceptual data models, which are used to identify domain concepts; logical data models, which defined the specific entities, attributes and relations that belongs to a system under study; and physical data models, which are used to design the internal schema of a database enacting its corresponding data tables, columns (placed on the tables), and relations between tables. Data models aid to assist business and IT staff to understand and use the information of an enterprise and its information systems as well as to to manage data as a resource in order to foster the integration of information systems. Data models are usually created during the analysis and design phases of a project in order to warranty that the business requirements regarding enterprise information are included in the project design.
- *Application Domain* identifies the systems required to support the business, defining their structure and behavior making emphasis on the interaction between them and with users. This domain defines sets of capabilities to manage enterprise information and facilitates identifying business improvements [18]. This domain includes enterprise application and integration components, custom application development models, services definitions, processes alignment specifications, and services architectures. This domain includes models for describing the structure of the application systems of the organization such as components models, architecture models, and development models.
- *Technology Domain* enables capabilities required by the enterprise. These capabilities are offered through technology infrastructure systems such as hardware devices, infrastructure services, software, networks among others. In this domain, models are created to describe infrastructure elements such as networking models, which allows representing communication devices (e.g., routers, switches) with corresponding protocols as well as their physical location; infrastructure models, which represents hardware (e.g., servers) and software (application servers, databases) systems.

There are five relations between the aforementioned domains. The relation from *Business* to *Information* relates business processes and indicators to business entities. The relation from *Business* to *Application* relates business processes and business services to application services. The relation from *Information* to *Application* relates business entities to application services. The relation from *Information* to *Infrastructure* relates information structure to technological support. Finally, the relation from *Application* to *Infrastructure* relates application components to infrastructures services and software.

III. Enterprise Analysis

In a general context, analysis consists in the identification, separation, and examination of the parts of a component in order to study its function and meaning. An analysis performs different actions that allow obtaining conclusions regarding the component that is being analyzed. When the component is represented by a model, the analysis of the component can be performed by analyzing the corresponding model. Nevertheless, analyzing a model is challenging because of its size and complexity. In this chapter we cope with enterprise analysis challenges through automated analysis methods.

In the enterprise context, enterprise models are mainly used to support enterprise analysis. The analysis of an enterprise model is a complex human activity that involves formulating hypotheses, discovering insights, and interpreting results in order to communicate assessments. Thus, enterprise analysis is the process of applying and evaluating certain business criteria on enterprise models in order to obtain results that might enrich the model and be used to make assessments of the actual (AS-IS) or desired (TO-BE) state of the enterprise.

Various enterprise analyses are currently used in the enterprise context. These analyses can be classified as internal, external, or both. Internal analyses are those that are focused on elements that belongs to the enterprise, while external analyses are those that take into account elements that are not part of the enterprise.

Some internal analyses are:

- *Capabilities analysis* is the study of enterprise processes based on given specifications in order to determine whether or not enterprise processes comply to desired enterprise objectives.
- *Risk analysis* is the study of dangers to business and people provoked by human events. Its results can be quantitative or qualitative and are used to align business and IT objectives.
- *Business process analysis* evaluates business processes performance by measuring processes elements (e.g., events, activities, resources) in order to reduce overall costs, increase efficiency of resources, and provide better customer service and support.
- *Change impact analysis* allows identifying the consequences on resources, effort, and schedule of a business or IT change as well as estimating the required adjustments in order to perform a desired change.
- *Financial analysis* evaluates the income statement, balance sheet, and cash flow statement of the enterprise in order to determine its profitability, solvency, liquidity, and stability, among others.

Some external analyses are:

- *Market analysis* studies the commercial viability and dynamics of the market that belongs to a specific industry, based on some desired characteristics such as market segments, competitors, customers behavior, potential demand, market trends, consumption patterns, etc.
- *Product life cycle analysis* evaluates the evolution of a product through its life span. This evaluation is done along five stages: development, introduction, growth, maturity and decline. This analysis allows assessing the

actual state of the product and the actions that should be taken in order to increase the products profit.
- *Competitors analysis* is used to identify competitors of an enterprise and to evaluate their business strategies in order to predict their behavior. This analysis involves the key aspects: objectives, assumptions, strategy and capabilities.

Some internal and external analyses are:

- *SWOT (Strengths, Weaknesses, Opportunities, and Threats) analysis* identifies internal and external characteristics of an enterprise. Internal characteristics are: strengths that give the business a competitive advantage and weaknesses that place the business competitive disadvantage provoking a reduction of progress. External characteristics are: opportunities that provide the business a favorable position and threats that impacts negatively the business.
- *Stakeholders analysis* determines the power and interest of stakeholders, who are those individuals or groups that have concerns in the enterprise. In this analysis, stakeholders with high power and high interest are key players. Stake holders can be internal (e.g., managers, developers, finance department) or external (e.g., customers, suppliers).

When the analyst performs a business analysis on an enterprise model, according to the objectives of the analysis, the analyst needs to browse the enterprise model supported by a modeling tool to get the required information from the model and obtain results that might enrich the model. The analyst must interpret those to provide assessments. For example, if an enterprise model is used to perform a financial analysis in order to assess the annual product profit, the analyst must verify that the attributes *price*, *fixed cost*, *variable cost*, and *sales* are included in the model. Using a modeling tool, the analyst collects the values of said attributes and make calculations in order to provide the products' profit. Nevertheless, if the enterprise sells a great amount of different products, which are classified by different categories, the enterprise model might become huge; thus, browsing the model might produce errors.

Since a business analysis might include several tasks (e.g., browsing the model, calculating numeric values), it might be decomposed in various individual analyses in charge to perform some procedures in order to solve specific tasks on the model. These individual analyses are analysis methods that provide specific results useful to business analyses. For instance, continuing with the previous example, a full financial analysis may require the calculation of partial information such as *income per product*. Thus, an analysis method created for this task provides such information. However, to perform this method, it is necessary that the model includes values to the attributes *price* and *sales* in every product. Then, the metamodel of the modeling language needs to include these attributes. In addition, if the results need to be stored in the model, the metamodel also needs to include the attribute *income* in the corresponding type.

IV. Imperfection in Enterprise Models

The construction process of an enterprise model is complex and normally requires certain tough activities such as consulting heterogeneous sources, and interpreting unstructured information [2]. Thus, building an enterprise model that represents an enterprise accurately is very difficult because enterprise models have a lot of elements and a lot of relations between those elements.

Enterprise models represent the state of an enterprise in a specific moment; however, enterprises continuously have changes, so they remain imprecise. Thus an enterprise model usually lacks of complete information and even contains imprecise or inconsistent information [20].

An enterprise model can be imperfect for two reasons. On the one hand, modelers start the modeling process before they obtain all the required information. Then, modelers create drafts of the enterprise model, which are temporary models and could include incomplete elements or imperfect information [8]. On the other hand, modelers decide to include imperfect information because the collect information is insufficient; nonetheless, it is not possible to obtain further information from additional sources; as a result, the final version of the model remains imperfect.

When modelers create imperfect models, it is possible to represent inaccurate information of the enterprise instead of incorrect information of the enterprise. However, it demands the characterization of the imperfection, which implies understanding the sources of enterprise information.

A. Sources of Enterprise Information

Enterprise knowledge is usually fragmented in several sources of information. These sources provide information to modelers for building the enterprise model and they can be *observational* (e.g., meetings and interviews with employees), *factual* (e.g., surveys, documents, reports, presentations, diagrams), or *reflective* (i.e., based on the experience and insights of the modeler). Thus, each source has a different reliability and precision level. Consequently, it is usual to consult various sources to get information about certain aspects of the enterprise.

When approaching these sources, it is very likely to find different issues such as lack of information, imprecise statements, contradictions, conflicts, and other kinds of issues that hinder the construction of the model. Moreover, it is also likely to find certain information that was true in previous states of the enterprise, but sources are not able to confirm if this information remains valid in the present. Thus, based on the quality of the information provided by sources, they can fall into one or various of the following categories:

- A source is **incorrect** [7], when it provides false, irrelevant, or unusable information. This might happen when the source has obsolete information about certain aspects of the enterprise.
- A source is **imprecise** [21], [22], [23], [24], when it provides a range of values to a numeric attribute that requires an individual value (e.g., based on various experiments to measure the availability of the application elements, the *CTO* informs that the availability of the application component *CRM* is between *0.92* and *0.95*).
- A source is **inconsistent** [25], when it provides different values to an attribute or relationship (e.g., the *CTO* makes an experiment to measure availability of the application component *CRM* providing the value *0.92*,

but later the same *CTO* based on a different experiments asserts that the availability is *0.95*). In addition, various sources are inconsistent if they provide different values for the same attribute or relation (e.g., the CEO asserts that the expected increment of sales for the next year is 30%, but the sales committee asserts that the expected increment of sales is 20%).

- A source is **vague** [26], when it provides a linguistic value to one attribute that requires a numeric value. (e.g., based on an experiment made to measure the availability of application elements, the *CTO* asserts that the availability of the application component *CRM* is *"High"*)
- A source is **uncertain** [21], [27], [24], when it provides a value with certainty degree (e.g., the *CTO* asserts that the application component *CRM* is used by the business process *Customer Profile Analysis* with a certainty degree of *80%*).

B. Imperfect Models

Modelers are in charge of assigning values to attributes, as well as connecting model elements with relationships. Then, once modelers receive information from imperfect enterprise sources, they do not have a unique course of action when dealing with various values for an attribute, or various targets for a relationship. Then, they need to make decisions in the construction of the enterprise model, which might cause a loss of information. For instance, consider the following situation: a *CTO* of an enterprise declares that the *CRM* (an Application Component, which manages the customer data) has availability of 96%; however, an IT employee asserts that the availability of the *CRM* is 99%. Then, in this case, the modeler has at least the following options to assign a value to the attribute availability:

1) Assign one numeric value (e.g., the higher values or the average).
2) Assign a range of values (e.g., [96% - 99%]), where each value has the same occurrence probability. This also requires modifications in the metamodel, e.g., by creating the attributes *minimumAvailability* and *maximumAvailability*.
3) Assign a set of values (e.g., (96%, 99%)), where more than one value with the same occurrence probability of are provided by several sources.
4) Include a certainty degree to each conflicting value based on the reliability level of the source, e.g., 96% with a 60% of certainty, and 99% with 40% certainty.
5) Modify the attribute type to support qualitative values, e.g., to describe availability as *"High"*.
6) Do not assign any value.

Taking the first option implies that the model is complete; however, it is incorrect. Options two, three, and four are not desirable because these options imply including information that does not conform to the metamodel. The last option is also undesired because it produces an incomplete model.

In addition, it is also possible to find imperfection in *one to one* relationships with multiple elements i.e., relationships with various possible targets or sources. For example, continuing with the previous case, the *CTO* asserts that the *MySQL Database Service*, which is an Application Service, is realized by the *Windows Server*, which is an Application Component, or the *Linux Server*, which is also an Application Component, but being unsure by which one specifically. In this case, modelers have four options:

1) Make the realization relationship with one component realized by the Application Component.
2) Make two realization relationships, where each relationship has the same probability.
3) Make two realization relationships assigning to each relationship a desired certainty degree.
4) Do not assign any relationship.

Taking the first option implies that the model is complete; nevertheless, it might be incorrect. Options two and three demands updating the metamodel of the modeling language. Finally, option four produces an incomplete model.

In order to model imperfection, it is mandatory to adjust the metamodel of the modeling language to allows describing incompleteness, uncertainty, vagueness, inconsistency, and imprecision of enterprise sources, while modeling the business information they report to modelers [9], [28]. Then, based on the information provided by sources, modelers might create five types of models or a combination of them:

Imprecise model: Includes at least one attribute that has a range of numeric values instead of a single numeric value, when sources of information are *inconsistent* or *imprecise*.

Inconsistent model: Includes at least one attribute or relationship that has multiple values. Attributes can have a set of values when sources are *inconsistent* or *imprecise*. For relationships, modelers can create various relationships with the same name that share the same source or target element, when sources of information are *uncertain* or *inconsistent*.

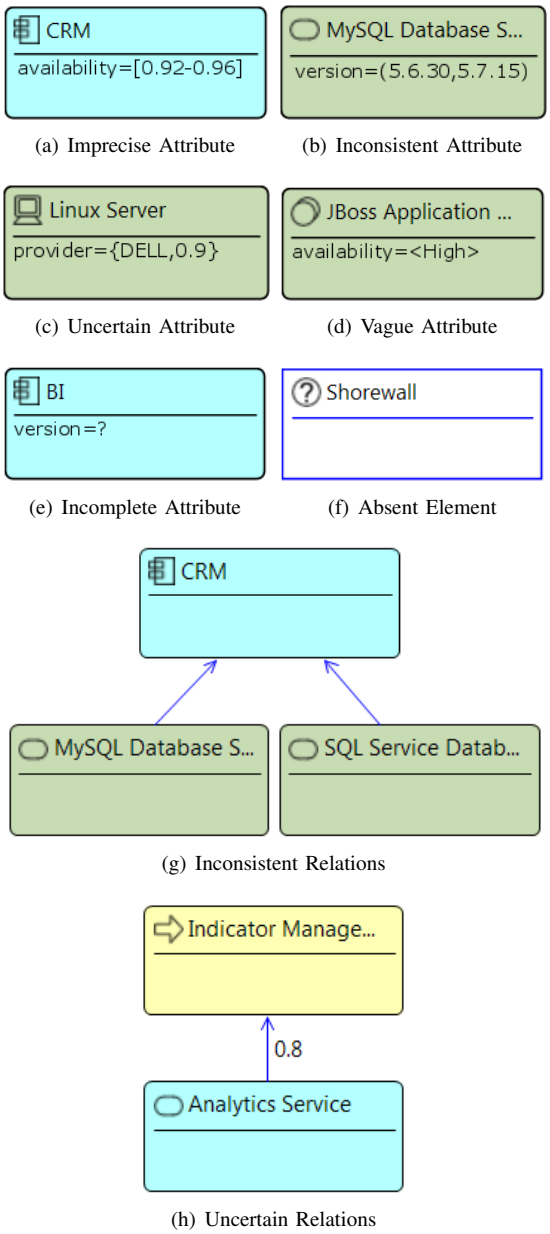
Uncertain model: Includes at least one attribute or relationship that has a certainty degree, when sources of information are *uncertain*, *inconsistent* or *imprecise*.

Vague model: Includes at least one attribute that has a linguistic value. This happens when sources are *vague*, *uncertain*, *inconsistent*, or *imprecise*.

Incomplete model: Includes at least one attribute with empty value, or an *absent* element, when modelers do not have enough information.

Then, with the purpose of making these issues explicit, we use a notation for each kind of imperfection:

- Imprecise attributes include a minimum and a maximum numeric values separated by a dash inside square braces (e.g., *[0.8-0.9]*). Fig. 1a presents the application component *CRM* with the `attribute` availability that contains the imprecise value *[0.92-0.96]*.
- Inconsistent attributes include a list of values in parentheses separated by commas (e.g., *(X, Y, Z)*). Fig. 1b presents the application service *MySQL Database Service* with the attribute `version` that contains the inconsistent value *(5.6.30,5.7.15)*.
- Uncertain attributes include two elements: the actual value of the attribute and a numeric value that represents a certain degree of the actual value. These elements must be placed with braces and separated by comma (e.g., {*Z, 0.8*}). Fig. 1c presents the device *Linux Server* with the attribute `provider` that contains the uncertain value {*DELL, 0.9*}, which implies that the provider is *DELL* with a certainty degree of *90%*.



(a) Imprecise Attribute (b) Inconsistent Attribute

(c) Uncertain Attribute (d) Vague Attribute

(e) Incomplete Attribute (f) Absent Element

(g) Inconsistent Relations

(h) Uncertain Relations

Fig. 1. Notation for imperfection

- Vague attributes include a linguistic value inside of angle brackets (e.g. *<High>*). Fig. 1d presents the system software *JBoss Application Server* with the attribute `availability` that contains the vague value *<High>*.
- Empty attributes of incomplete elements include a question mark (*?*) as value. Fig. 1e presents the application component *BI* with the attribute `version` that contains as value the symbol *?*.
- *Absent* elements are placed with uncolored squares. Fig. 1f presents an element called *Shorewall*, which is supposed to be system software, but its existence is not proved.
- Inconsistent relationships are drawn in blue and include a set of relationships with the same type (e.g., `UsedBy`) from one source to various targets. Fig. 1g presents the application component *CRM* supported by the infrastructure services *MySQL Database Service* and *SQL Server Database Service*; nevertheless, the *CRM* is supposed to be supported by just one database service.
- Uncertain relationships are drawn in blue and have an attribute that specifies its certain degree. Fig. 1h presents the UsedBy relation from the application service *Analytics Service* to the business process *Indicator Management* with a certainty degree of 80%.

V. Modeling Imperfection

Although imperfect enterprise models can be developed using an arbitrary modeling language [29], we will focus on the ArchiMate [30] modeling language. In order to address modeling imperfection, we have used the distinction and separation of linguistic and ontological conformance.

To achieve this, we developed a modeling tool called *iArchiMate*[1], which has been created on top of the Eclipse Modeling Framework (EMF)[2]. Models created using the tool *iArchiMate* conform to one metamodel called *Imperfection MetaModel (iMM)*, which is presented in Fig. 2 and includes the required concepts and relationships for modeling imperfect ArchiMate models.

Thus, the imperfect model conforms linguistically to the *iMM* that allows representing any element, attribute, and relationship, where attributes and relationships can be imperfect. The imperfect model conforms ontologically to the domain metamodel (e.g., ArchiMate); nevertheless, attributes and relations of instances of the same metatype can have different characteristics i.e., different kinds of imperfection; then, imperfect information must follow the notation provided by the modeling language.

The central element of *iMM* is the `Model` concept, with a containment relationship to the other concepts. The abstract type `Component` is specialized by concepts `Element` and `Group`. `Group` allows defining collections of elements, while `Element` serves to represent instances of a given concept of a model. Each `Element` includes one attribute named `typeName`, which must match the name of an ArchiMate concept (e.g., `BusinessProcess`). The concept `Relation` allows representing relationships between two elements. Each `Relation` includes an attribute named `typeName`, which has to match one of the relationship types of ArchiMate (e.g., `UsedBy`). The type `Attribute` allows representing the actual values of attributes included in elements or relationships of the model.

The concept `ImperfectAttribute` is created to represent imperfect attributes of the original enterprise model, and contains the attribute `imperfectionType`, which determines its kind of imperfection. This attribute can have different values. `NumericRange` value implies that the attribute is imprecise. The values `NumericSet` and `NumericString` define an inconsistent attribute. Also, the values `NumberCertaintyDegree` and `StringCertaintyDegree` are used to describe an uncertain attribute. When the attribute is vague, the type `LinguisticValue` must be assigned. Incomplete attributes have the literal `NoValue`.

The concept `ImperfectRelation` represents imperfect relationships. This concept has the attribute `imperfectionType`, which determines its imperfection.

[1]http://iarchimate.virtual.uniandes.edu.co
[2]https://www.eclipse.org/modeling/emf

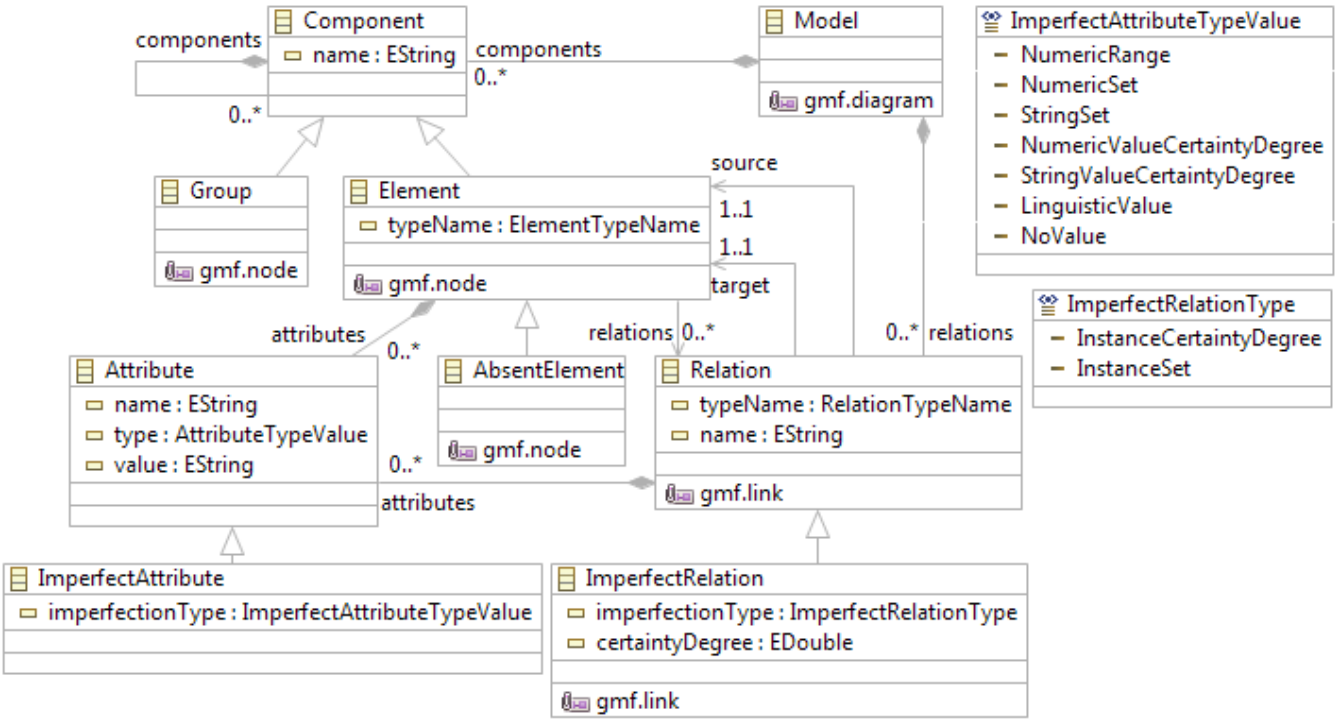


Fig. 2. Metamodel of Imperfection iMM.

Finally, we use the concept `AbsentElement` for describing incomplete elements.

A. Running Example

This example uses the ArchiMate [30] modeling language just for illustration purposes. We selected this language because it allows creating enterprise models from a holistic perspective, facilitating the execution of different business and IT analyses. Furthermore, its widespread use and tool support facilitated the implementation of our approach, and its metamodel contains a well-defined collection of concepts specific to enterprise modeling.

In order to illustrate the notion of imperfect models and their imperfection analyses, we use one scenario of a publisher scenario. The enterprise model of this scenario was built using the *iArchiMate* modeling tool, and contains 184 elements in the Business Layer, 13 elements in the Application Layer, 13 elements in the Infrastructure Layer, 28 elements in the Motivation Layer, and 432 relationships arranged in 12 views. Having several elements that present different kinds of imperfection, this scenario is appropriate for our purposes in this work.

Fig. 3 illustrates a fragment of the layered view created for the publisher scenario. This view contains elements of the application and infrastructure layers, which includes the following imperfect information:

1) Imprecise attribute `availability` and uncertain attribute `storageUsed` in *Windows Server*.
2) Vague attribute `availability` and inconsistent attribute `provider` in *Linux Server*.
3) Inconsistent attribute `version`, vague attribute `availability`, and uncertain *Realization* relationship to *Analytics Service* in *BI*.
4) Incomplete attribute `availability`, vague attribute `storageUsed`, and uncertain *Realization* relationship to *Version Control Service* in *DMS*.
5) Incomplete attribute `version` and uncertain *Realization* relationship to *Customer Management Service* in *CRM*.
6) Absent element *Profile Service*.

Imperfect relationships are depicted in blue, and absent elements are filled uncolored but outlined with a blue square. When selecting an imperfect relationship, the Properties view of *iArchiMate* displays relevant information about the relationship such as certainty degree, imperfection type, name, source, target, and type name.

By using our approach for modeling imperfection, it is then possible to introduce in the model all the information provided by sources even if such information is imperfect. Nevertheless, we have identified some limitations:

- In some cases, sources might not provide the information they are supposed to provide, and thus the modeler is forced to leave the model incomplete.
- When several sources provide inconsistent information, the modeler might decide to include them as a set of values; nevertheless, if some of those values are already imperfect, the modeler would need to include a set of values in which some of them might be imprecise, vague, or uncertain. So far, our approach is not able to support this situation.

VI. Analyzing Imperfection

When an enterprise model is imperfect, it is important to identify and measure its imperfection level in order to understand how the model might be refined to reduce or even remove certain imperfect information. Identifying and measuring the imperfection of an enterprise model can

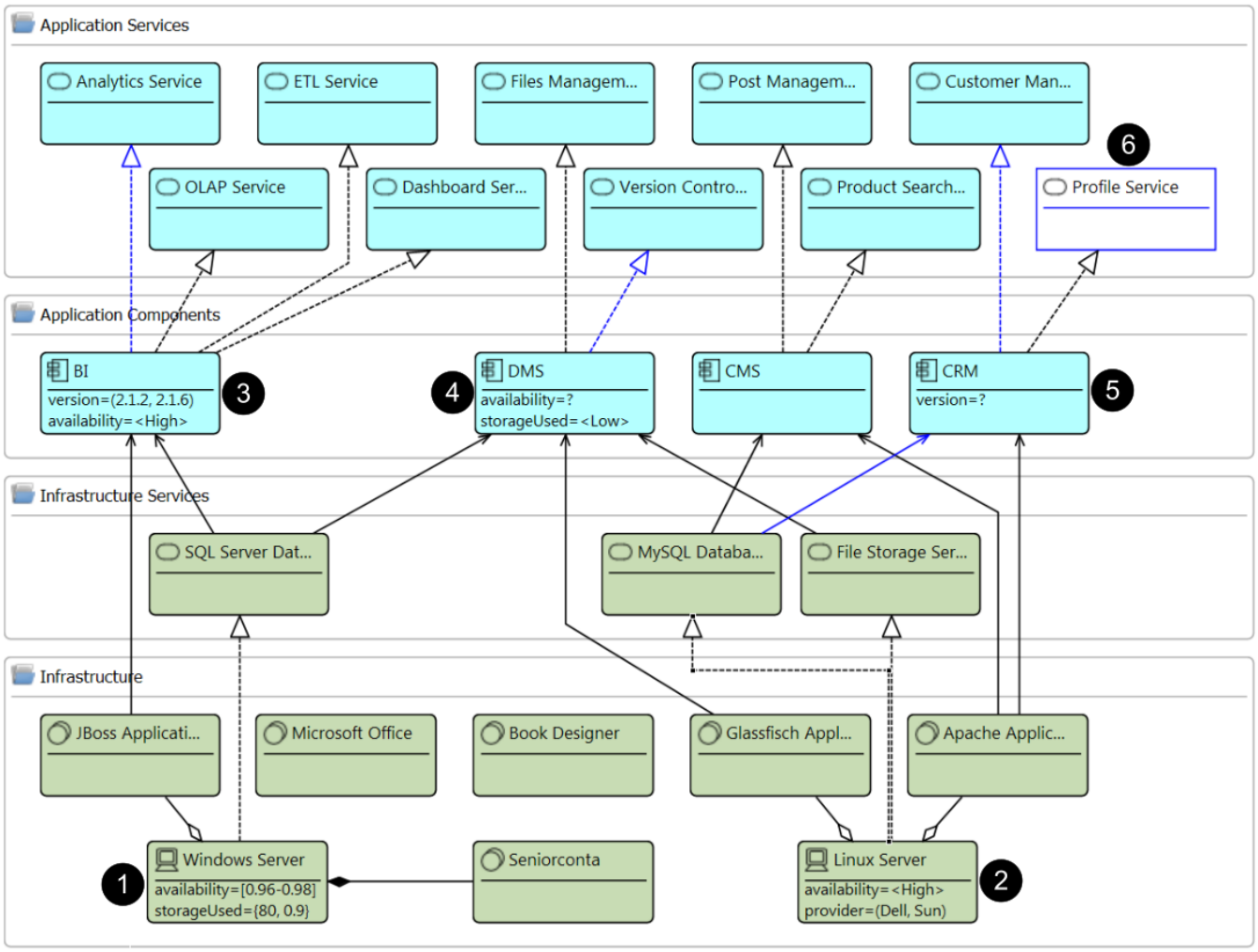


Fig. 3. Layered view of publisher enterprise scenario using *iArchiMate*.

determine whether the model is useful for further purposes such as documentation, communication, or business analysis. Detecting imperfection while developing enterprise models can be an onerous task, given the size and complexity of these models, as well as the different types of imperfection that we can encounter i.e., imprecision, inconsistency, vagueness, uncertainty, or incompleteness. Then, it is desirable to discover flaws in the model as early as possible. In order to correct them, it is necessary to make them visible, something that can be done by different means.

Imperfection analyses offer modelers and analysts valuable information, for studying and measuring imperfection problems in the models in order to understand and, if possible, correct such problems. For instance, a modeler can realize that a model is not finished because the imperfection level of the model suggests that a) there is a lack of elements in the whole model or in one specific ArchiMate layer, b) there is a lack of relationships between elements placed in different ArchiMate layers, or c) there are missing attributes in elements or relations. The analyst can also conclude that the enterprise model is not useful to perform a specific business analysis because there are different kinds of imperfections in elements, attributes, or relationships required for the analysis; as a result, the analysis results might not be accurate or reliable enough.

In this work, we propose two analysis techniques: a) domain analysis, which analyses imperfection based on certain desired enterprise elements and b) topological analysis, which analyses imperfection based on the structure of the model. By using these techniques, an analyst can obtain relevant insights regarding the imperfection in the model, as well as a global perspective that facilitates the discovery of additional imperfections.

A. *Domain Analysis of Imperfection*

In certain circumstances, an analyst needs to know the imperfection level of a particular meta-type in order to focus on specific issues on the model. In this case, the purpose of domain-specific analysis of imperfection is to know the imperfection level of the model by focusing on business rules and patterns, as well as considering specific characteristics of the modeling language used to create the enterprise model. For instance, our model has a large number of business elements, but a small number of technology elements.

Domain analysis of imperfection might suggest that 1) the model does not contain the necessary technological elements to support the business elements of the enterprise; as a result, the model is incomplete, or 2) the enterprise is not working properly. Thus, there might be a large number of domain analyses; however, in this section, we just illustrate two domain-specific analyses: incompleteness and uncertainty.

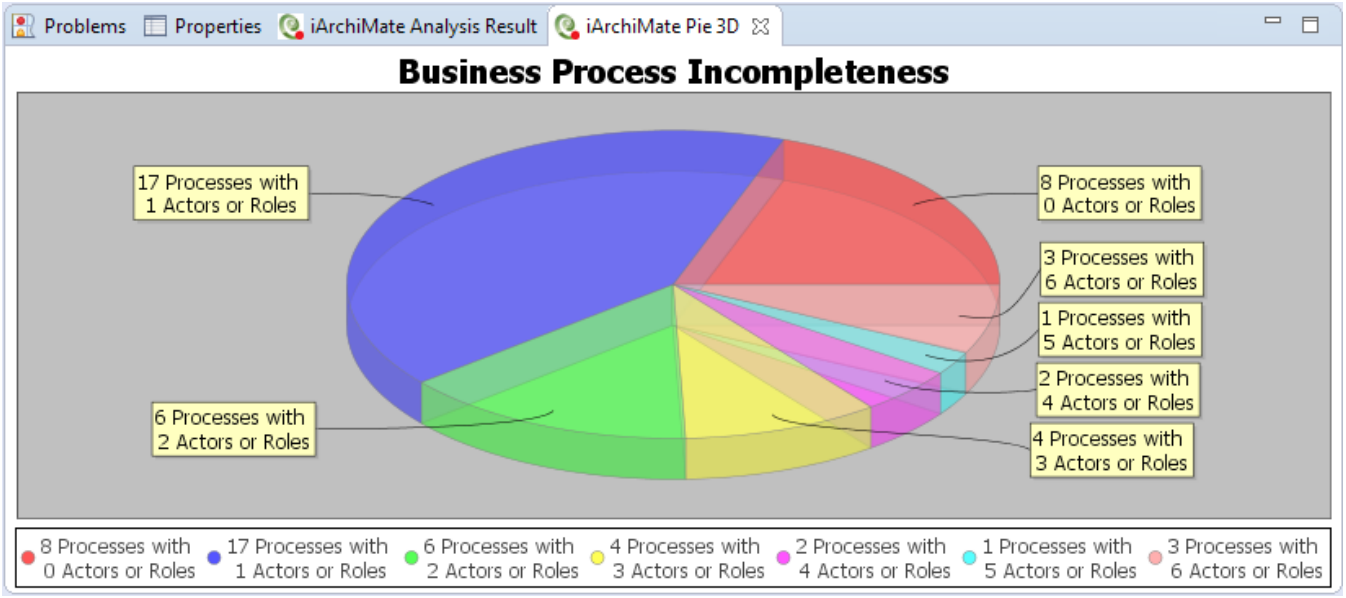


Fig. 4. Results of Incompleteness Domain Analysis.

1) Incompleteness domain analysis: This analysis aims to identify the possible absence of certain elements, relationships, or attributes in the enterprise model, based on the specific characteristics of the modeling language. Thus, the metamodel of the modeling language specifies structural characteristics; however, in some cases, the metamodel cannot define semantic characteristics. Consequently, modeling languages might contain certain rules that this analysis must take into account. In addition, this analysis must allow the inclusion of parameters, for determining the values of required variables and analysts might be able to include certain rules in order to customize the analysis when it is required.

For instance, based on the ArchiMate metamodel, each Business Process in the model should be related to at least one Business Role or Business Actor. This analysis determines whether there are Business Processes that are not performed by any role or actor. By performing this analysis on our publisher scenario, we have obtained the results shown in Fig. 4, which presents the number of business processes that have been related to the number of business actors or roles.

We can observe that 8 processes do not have an assigned business actor or role, and 17 processes just have one business actor or role. On the one hand, these results indicate that the model is incomplete because there is a lack of business actors or roles and a lack of corresponding relationships with business processes. On the other hand, the results also suggest that there is a possible additional level of incompleteness corresponding to the business processes with just one business actor or role; thus, the lack of relationships between business processes and business actors or roles is very likely. The rest of the processes seem to be complete, but analysts can also use these results to know how many processes have been correctly modeled. This analysis evidences incompleteness; nevertheless, this incompleteness might be present because the reality is in that way; thus, albeit the analysis reflects incompleteness, the model is not necessarily representing correctly the enterprise.

Incompleteness might also be studied in other elements or relations of the imperfect enterprise model. Thus, analysts can focus their attention on some specific characteristics that the model should accomplish regarding the modeling language, in order to identify whether the imperfect model lacks necessary elements or relations in such focused parts of the model. Then, this incompleteness domain analysis becomes a useful tool to measure the lack of certain elements or relations needed for further uses such as performing business analysis on the enterprise model.

2) Uncertainty domain analysis: This analysis provides information about uncertain relationships of the model, based on a given certainty degree. The analysis evaluates one desired relationship type, of those available in the modeling language. However, in several modeling languages one relationship type can have as a source and target elements, different meta-types. Thus, the analysis might also be performed, for one specific source meta-type and one specific target meta-type, in order to obtain filtered results. The algorithm of this analysis method browses the corresponding relationships and compares their assigned certainty degree with the given certainty degree.

For example, the enterprise model of our publisher scenario has some uncertainty degree in the `UsedBy` relationships that goes from the elements `ApplicationService` to the elements `BusinessProcess`. In order to measure the uncertainty of these relationships, the analyst considers those with a certainty degree greater than *0.9* as valid. Fig. 5 presents the results of the analysis method through a report, which informs the amount of imperfect `UsedBy` relationships with certainty degree greater than *0.9* that go from the elements `ApplicationService` to the elements `BusinessProcess`.

Based on these results, the analyst can not only identify in which elements imperfect relations are located but also the number of imperfect relations filtered by a criteria given by the analyst. For example, the presented results alert the presence of several imperfect relations concentrated in just two elements: *Customer Management Services* and *File*

Problems | Properties | iArchiMate Analysis Result

Element	# Relations	# Imperfect	% Imperfect	# Imperfect over 0.9	% Imperfect over 0.9
Analytics Service	3	2	66.67	1	33.33
Customer Management Service	7	3	42.86	0	0.0
Product Searching Service	5	1	20.0	1	20.0
Dashboard Service	1	1	100.0	1	100.0
Files Management Service	7	3	42.86	2	28.57

Fig. 5. Results of Uncertainty Domain Analysis.

Problems | Properties | iArchiMate Analysis Result

Overall Model Imperfection

Component	Total	Imperfect	Percentage
Arttribute	137	84	61.31
Relation	435	4	0.92
Element	241	50	20.75

(a) *TCF01* Overall Model Imperfection

Problems | Properties | iArchiMate Analysis Result

Imperfect Attributes

Imperfect Attribute Type	Imperfect Attributes	Percentage
Imprecise	45	32.85
Inconsistent	12	8.76
Uncertain	10	7.3
Vague	11	8.03
Incomplete	6	4.38

(b) *TCF02* Imperfect Attributes

Problems | Properties | iArchiMate Analysis Result

Imperfect Relations

Imperfect Relation Type	Imperfect Relations	Percentage
Inconsistent	0	0.0
Uncertain	4	0.92

(c) *TCF03* Imperfect Relations

Problems | Properties | iArchiMate Analysis Result

Imperfect Elements

Imperfect Element Type	Imperfect Elements	Percentage
Absent Elements	1	0.41
Elements with Imperfect Attributes	49	20.33
Elements with Imperfect Relations	4	1.66

(d) *TCF04* Imperfect Elements

Fig. 6. Topological Context-Free Analysis Methods Results

Management Service and more than the half of the imperfect relations have a certain degree over 0.9% which demands their refinement.

B. Topological Analysis of Imperfection

The last phase of our approach to analyze imperfection is the topological analysis. This kind of analysis measures the imperfection level of the model, based on the structural properties of the model. It provides quantitative results and can be done from different perspectives by running topological analysis methods. We identified two classes of topological analysis methods for measuring the imperfection in enterprise models: *Context-Free* and *Context-Dependent* analysis methods.

Context-Free analysis methods provide the number of imperfect elements, attributes and relationships in the model, while *Context-Dependent* analysis methods provide information regarding the imperfection level of the model based on the modeling language concepts. This work has been instantiated in ArchiMate; consequently, *Context-Dependent* analysis methods are based on ArchiMate elements (e.g., `BusinessProcess`), relationships (e.g., `UsedBy`) and layers (e.g., business, application). Table I presents the set of topological analysis methods for analyzing the imperfection that we have identified. For instance, the *Context-Free* analysis method *TCF02* presents the amount of imperfect attributes classified by imperfection type (e.g., *imprecise*, *inconsistent*, *uncertain*, *vague*, or *incomplete*), while the *Context-Dependent* analysis method *TCD02* presents imperfect elements classified by the ArchiMate layers *Business*, *Application*, *Infrastructure*, *Motivation*, and *Implementation*.

TABLE I
TOPOLOGICAL ANALYSIS METHOD FOR ANALYZING IMPERFECTION

Type	Id	Name
Context-Free	TCF01	Overall Model Imperfection
	TCF02	Imperfect Attributes
	TCF03	Imperfect Relationships
	TCF04	Imperfect Elements
Context-Dependent	TCD01	Imperfect Elements by Element Type
	TCD02	Imperfect Elements by Layer
	TCD03	Imperfect Attributes by Element Type
	TCD04	Imperfect Attributes by Element Type and Attribute Name
	TCD05	Imperfect Relationships by Element Type
	TCD06	Imperfect Relationships by Element Type and relationship Type

1) Context-Free: Fig. 6 reports the results of the topological *Context-Free* analysis methods for the complete model of the publisher scenario, where sub-figures (a), (b), (c), and (d) present the results of the analysis methods *TCF01*, *TCF02*, *TCF03*, and *TCF04* respectively.

The method *TCF01* presents the percentage of imperfect attributes, relationships, and elements, where imperfect elements are those that have been instantiated as `AbsentElement`, or those that include imperfect attributes or relationships. The method *TCF02* presents the percentage of imperfect attributes classified by their imperfection type (e.g., *imprecise*, *inconsistent*, *uncertain*, *vague*, or *incom-*

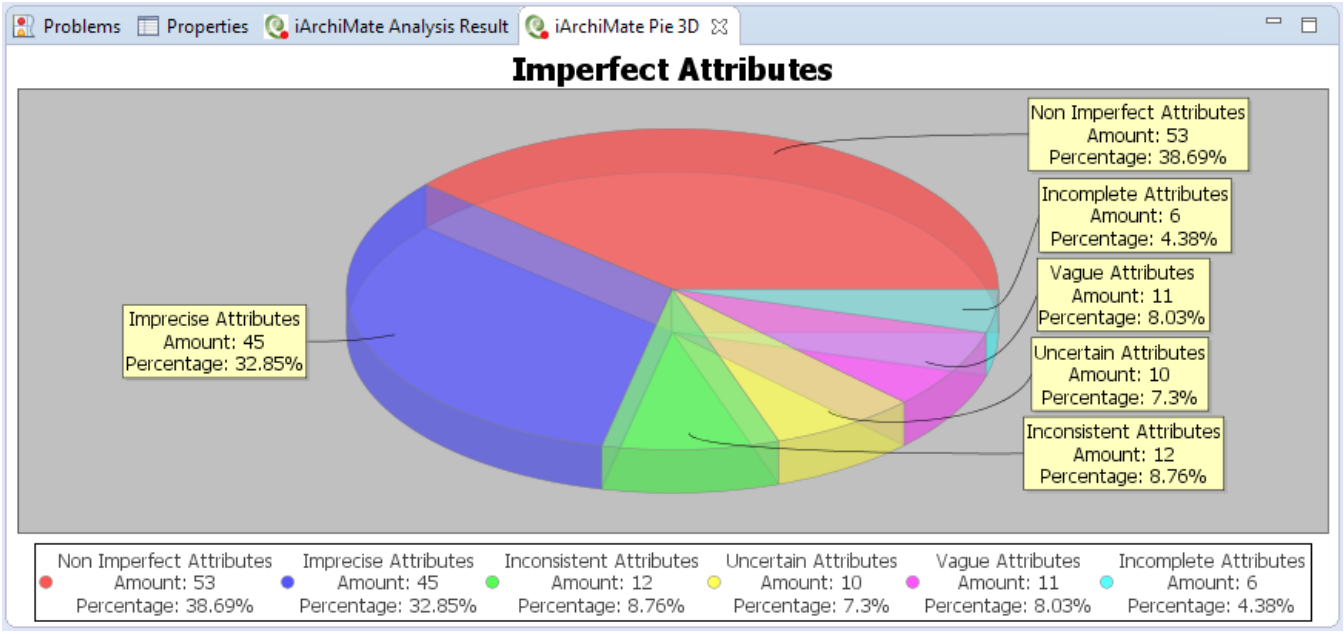


Fig. 7. Chart of Imperfect Attributes.

plete). The method *TCF03* presents the percentage of imperfect relationships based on their imperfection type (e.g., *inconsistent* or *uncertain*). Finally, the method *TCF04* presents the percentage of imperfect elements grouping them in absent elements, elements that contains imperfect attributes, and elements that contain imperfect relationships.

The *iArchiMate* environment also displays the topological analysis results through different charts. Fig. 7 is a pie 3D chart that contains the results of the analysis method *TCF02*. Based on this chart, it is possible to observe the proportion of imperfect attributes by imperfection type in the model, as well as the proportion of the non-imperfect attributes.

Using the results of the *Context-Free* analysis methods, an analyst can make different conclusions such as:

- 61.31% of the attributes are imperfect which seems to be a high amount of imperfect attributes; however, just 20.75% of the elements are imperfect, where 20.33% of those elements are imperfect because they contain imperfect attributes. This implies that most of the imperfect attributes are concentrated in a small set of the model's elements.
- Just 4 relations that correspond to the 0.92% are imperfect. This amount is very low and does not have a strong impact.
- From the imperfect attributes (i.e., 61.31% of the attributes), 32.85% are imprecise attributes. This suggests that refining the model might be focused on reducing or removing attributes imprecision.

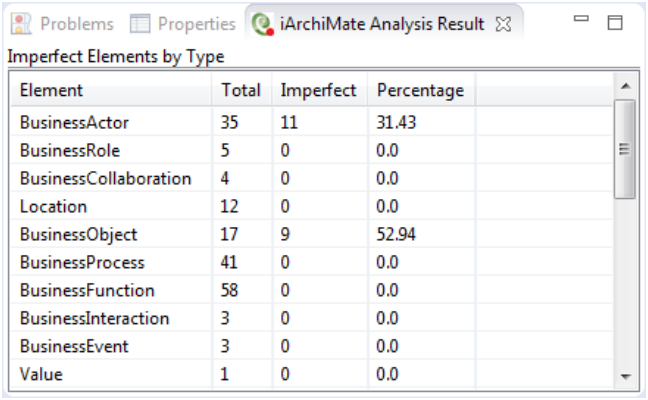
2) Context-Dependent: Fig. 8 presents some reports with the results of the topological *Context-Dependent* analysis methods for the model of the publisher scenario, where subfigures (a), (b), (c), (d), (e), and (f) present the results of the analysis methods *TCD01*, *TCD02*, *TCD03*, *TCD04*, *TCD05*, and *TCD06* respectively.

The results of the methods: i) *TCD01* presents the percentage of imperfect elements grouped by the ArchiMate element type, where imperfect elements are those that have been instantiated as `AbsentElement` or those that include imperfect attributes or relationships; ii) *TCD02* presents the imperfection level for each layer of ArchiMate such as *Business*, *Application*, *Implementation*, and *Motivation*; iii) *TCD03* calculates the percentage of imperfect attributes grouped by the element type; iv) *TCD04* calculates the percentage of imperfect attributes grouped by the name of the attribute (e.g., `availability`) and the element type; v) *TCD05* calculates the percentage of imperfect relationships grouped by the element type; and vi) *TCD06* calculates the percentage of imperfect relationships grouped by the element type and the relationship type.

Fig. 9 presents a bar chart with the results of the analysis method *TCD01*. Based on the chart, it is possible to observe which ArchiMate elements include attributes and imperfect attributes. In addition, the imperfection level (regarding the number of imperfect attributes) for each ArchiMate element is observable. The chart presents in red bars, the total amount of attributes for each element type (e.g., 24 attributes for business actors) and in blue bars, the total amount of imperfect attributes for each element type (e.g., 15 imperfect attributes for business actors).

The results of the *Context-Dependent* analysis methods run on the model of the publisher scenario enable analysts to make several additional conclusions such as:

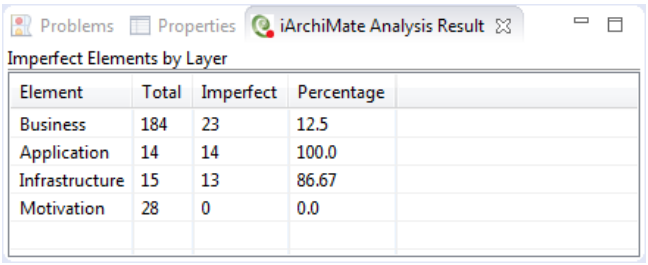
- Imperfect elements in the model are concentrated in just three element types which are `BusinessActor`, `BusinessObject`, and `ApplicationService`. Albeit the greater amount of imperfect elements are placed in the *Business* layer, those imperfect elements correspond to the 12.5% of *Business* elements. Moreover, 100% of the *Application* elements and 86.67% of the *Technology* elements are imperfect. This suggests that the *Application* and *Technology* layers need to be refined.
- Imperfect elements are concentrated in few element types of the model and some element types contain only imperfect attributes. However, the greater amount of imperfect elements are placed in just


Problems Properties iArchiMate Analysis Result

Imperfect Elements by Type

Element	Total	Imperfect	Percentage
BusinessActor	35	11	31.43
BusinessRole	5	0	0.0
BusinessCollaboration	4	0	0.0
Location	12	0	0.0
BusinessObject	17	9	52.94
BusinessProcess	41	0	0.0
BusinessFunction	58	0	0.0
BusinessInteraction	3	0	0.0
BusinessEvent	3	0	0.0
Value	1	0	0.0

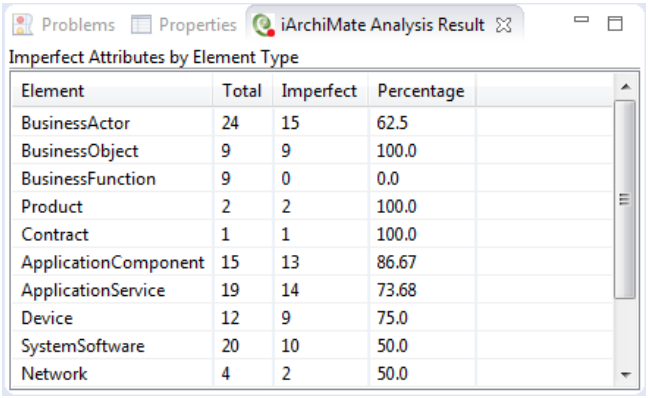
(a) Imperfect Elements by Element Type


Problems Properties iArchiMate Analysis Result

Imperfect Elements by Layer

Element	Total	Imperfect	Percentage
Business	184	23	12.5
Application	14	14	100.0
Infrastructure	15	13	86.67
Motivation	28	0	0.0

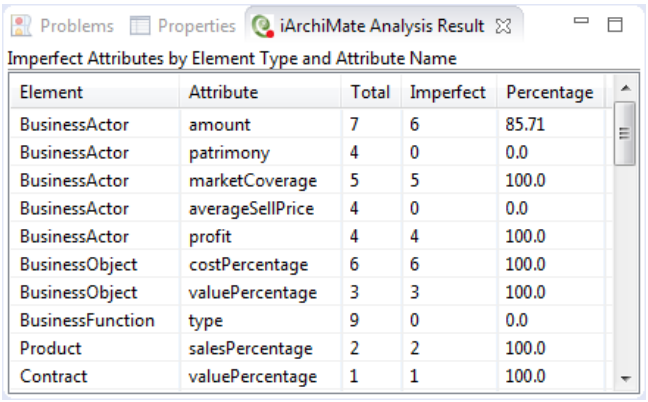
(b) Imperfect Elements by Layer


Problems Properties iArchiMate Analysis Result

Imperfect Attributes by Element Type

Element	Total	Imperfect	Percentage
BusinessActor	24	15	62.5
BusinessObject	9	9	100.0
BusinessFunction	9	0	0.0
Product	2	2	100.0
Contract	1	1	100.0
ApplicationComponent	15	13	86.67
ApplicationService	19	14	73.68
Device	12	9	75.0
SystemSoftware	20	10	50.0
Network	4	2	50.0

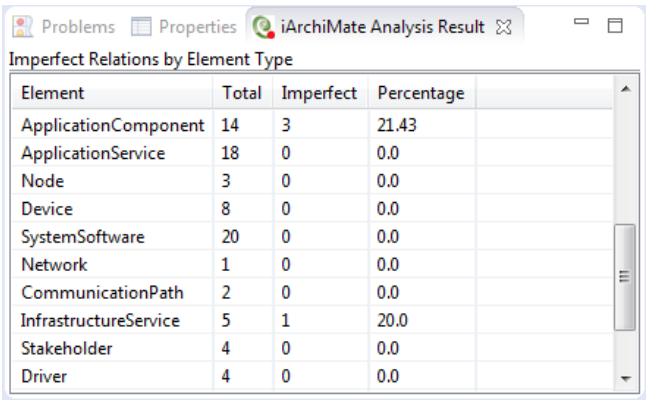
(c) Imperfect Attributes by Element Type


Problems Properties iArchiMate Analysis Result

Imperfect Attributes by Element Type and Attribute Name

Element	Attribute	Total	Imperfect	Percentage
BusinessActor	amount	7	6	85.71
BusinessActor	patrimony	4	0	0.0
BusinessActor	marketCoverage	5	5	100.0
BusinessActor	averageSellPrice	4	0	0.0
BusinessActor	profit	4	4	100.0
BusinessObject	costPercentage	6	6	100.0
BusinessObject	valuePercentage	3	3	100.0
BusinessFunction	type	9	0	0.0
Product	salesPercentage	2	2	100.0
Contract	valuePercentage	1	1	100.0

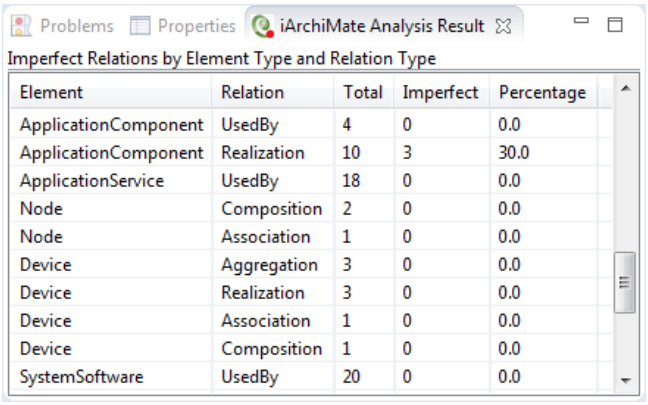
(d) Imperfect Attributes by Element Type and Attribute Name


Problems Properties iArchiMate Analysis Result

Imperfect Relations by Element Type

Element	Total	Imperfect	Percentage
ApplicationComponent	14	3	21.43
ApplicationService	18	0	0.0
Node	3	0	0.0
Device	8	0	0.0
SystemSoftware	20	0	0.0
Network	1	0	0.0
CommunicationPath	2	0	0.0
InfrastructureService	5	1	20.0
Stakeholder	4	0	0.0
Driver	4	0	0.0

(e) Imperfect Relationships by Element Type


Problems Properties iArchiMate Analysis Result

Imperfect Relations by Element Type and Relation Type

Element	Relation	Total	Imperfect	Percentage
ApplicationComponent	UsedBy	4	0	0.0
ApplicationComponent	Realization	10	3	30.0
ApplicationService	UsedBy	18	0	0.0
Node	Composition	2	0	0.0
Node	Association	1	0	0.0
Device	Aggregation	3	0	0.0
Device	Realization	3	0	0.0
Device	Association	1	0	0.0
Device	Composition	1	0	0.0
SystemSoftware	UsedBy	20	0	0.0

(f) Imperfect Relationships by Element Type and relationship Type

Fig. 8. Topological Context-Dependent Analysis Methods Results

four element types. In addition, some attributes such as *availability* in `ApplicationService` elements are always imperfect. These results suggest that imperfection in the model must be refined in the element types `ApplicationService`, `BusinessActor`, `SystemSoftware`, and `ApplicationComponent`.

- Few relations in the model are imperfect; however, all of them are `UsedBy` relations, which are placed in `ApplicationService` elements. Thus, it is important to pay special attention to the relations placed in the *Application* layer.

VII. Approach to Analyze Imperfect Enterprise Models

When the enterprise model is imperfect, automated analysis methods might not be performed because these methods follow algorithms that make numeric calculations; as a result, analyses must be done manually by analysts. Nevertheless, manual analysis is a though task because imperfect enterprise models are big, complex, and include all types of imperfection (i.e., imprecision, inconsistency, vagueness, uncertainty, and incompleteness).

Thus, once the level of imperfection has been evaluated, there are two courses of action. On the one hand, the modeler tries to refine the imperfect model, if new useful information can be gathered. On the other hand, the analyst tries to perform analysis through automated analysis methods using

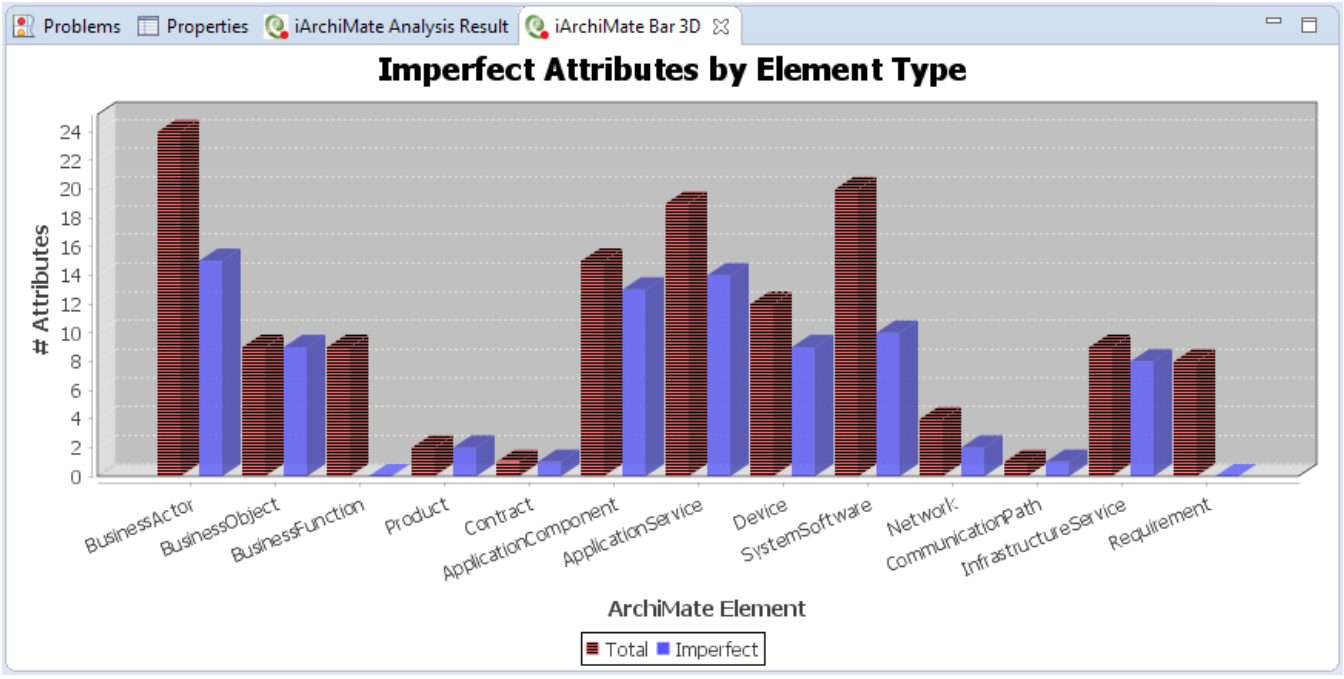


Fig. 9. Chart of Imperfect Attributes by Element Type.

the imperfect model. In this section, we focus on a solution for the second course of action.

Due to automated analysis methods need to make numeric calculations, some imperfection types cannot be used. For instance, vague attributes because they contain linguistic values (e.g., attribute `availability` with the value <*High*>); inconsistent attributes that contain a set of textual values (e.g., attribute `provider` with the value *(Dell, SUN)*); inconsistent attributes that contain a set of numeric values (e.g., attribute `memory` with the value *(4,8,16)*) because it would be necessary to make the calculations using all values of the set; uncertainty attributes because they contain a values and a certainty degree (e.g., attribute `storage` with the value {*1000,90%*}).

However, imprecise attributes serve to make numeric calculation because any calculation can be done using the minimum and maximum values and thus the result is represented with a range of values, which corresponds to a new imprecise attribute. Then, if some automated analysis methods can be upgraded in order to be performed on imprecise enterprise models, it is necessary to prepare the imperfect information involved in a desired analysis method to be able to make the corresponding calculations.

We propose an approach that consist in transforming imperfect information into imprecise information. However, it is not possible in every case. Imperfect information that can be transformed is the information assigned to numeric attributes. Then, following information can be transformed into imprecise information:

1) **Inconsistency to imprecision**. An inconsistent numeric attribute contains a set of numeric values inside of parenthesis and separated by commas (e.g., *(0.9, 0.97, 0.93, 0.95)*). This transformation puts a range of values to the corresponding attribute, where the minimum and maximum values are selected from the set of values.
2) **Vagueness to imprecision**. Vague attributes include a linguistic value inside of angle brackets (e.g., <*Good*>). The linguistic value must be selected from a set of linguistic values defined for the corresponding vague attribute (e.g., *Outstanding*, *Good*, *Acceptable*, *Bad*). This transformation puts a range of values for each linguistic value, where the range is equitable for the set of linguistic values. This range is then assigned to the numeric attribute.

Any transformation generates an alternative imprecise enterprise model, which is an approximation of the original imperfect model. It means that the original imperfect enterprise model is not altered. An imperfect enterprise model can be transformed in three different ways:

- **Automatic**. *iArchiMate* is able to make every transformation from inconsistent and vague values to imprecise values. There are the following alternatives:
  - AI1: For inconsistent values, the range of values includes as minimum and maximum values the minimum and maximum values of the set of values. For instance, if the inconsistent value is *(0.9, 0.97, 0.93, 0.95)* the transformed imprecise value is *[0.9 - 0.97]*.
  - AV1: For vague values, a default range of values is assigned to each linguistic value. The default scale is from 0.1 to 1. For instance, if the vague attribute can have one of the following linguistic values: *Outstanding*, *Good*, *Acceptable*, *Bad*; then, the equivalent ranges for each linguistic value are: *Outstanding*: *[0.76 - 1]*, *Good*: *[0.54 - 0.75]*, *Acceptable*: *[0.26 - 0.5]*, *Bad*: *[0.1 - 0.25]*; thus, if the linguistic value is *Good*, the transformed imprecise value is *[0.54 - 0.75]*.
- **Semiautomatic**. *iArchiMate* offers analysts services to customize the transformation. This customization has different alternatives for each imperfection type.

TABLE II
ADVANTAGES AND DISADVANTAGES OF TRANSFORMATION ALTERNATIVES

Id	Advantage	Disadvantage
AI1	Every inconsistent attribute is transformed into a imprecise attribute with just one process	Transformations cannot be done for just specific elements (e.g., *BusinessProcess*) or specific elements that belong to a specific layer (e.g., *business layer*).
AV1	Vague attributes are transformed based on the amount of linguistic values	The resultant ranges of values might have a big difference between the maximum and minimum value
SI1	Analysts can decide the imprecision to be introduced	In the model has a lot of inconsistent attributes, analysts have to make lot of selections
SI2	Analyst have to make just one operation for transforming every inconsistent attribute of the model	This alternative might introduce incorrect information
SV1	Analysts can decide the scale of imprecision	Analysts need to make one action for each vague attribute in the model
SV2	Analyst can provide non equitable ranges of values for linguistic values	Analyst has to provide one range of values for each linguistic value for each vague attribute

  - For inconsistent values, there are the following alternatives:
    - SI1: The analyst can select for each imprecise attribute, the minimum and maximum values from the set of values. *iArchiMate* validates that the minimum value is less than the maximum value.
    - SI2: When the inconsistent value is a set of four or more values, *iArchiMate* offers the analyst the option to remove the lower and greater value of the set, and proceeds to create a range of values using the automatic way AI1.
  - For vague values, there are the following alternatives:
    - SV1: The analyst can assign a scale for the set of linguistic values and *iArchiMate* assigns the corresponding range of values based on such scale. For instance, if the linguistic values are: *High*, *Medium*, *Low* and the analyst sets the scale 71-100, the equivalent ranges for each linguistic value are: *High*: *[91 - 100]*, *Medium*: *[81 - 90]*, *Low*: *[71 - 80]*; thus, if the linguistic value is *Medium*, the transformed imprecise value is *[81 - 90]*.
    - SV2: The analyst can assign a range of values for each linguistic value that belongs to a set of linguistic values and *iArchiMate* assigns the corresponding range of values. For instance, the linguistic values are: *High*, *Medium*, *Low* and the analysts assigns the following ranges: *High*: *[95 - 100]*, *Medium*: *[81 - 95]*, *Low*: *[61 - 80]*;
- **Manual**. The analyst can decide to introduce imprecision, for desired imperfect attributes. However, it might produce an incorrect model and thus automated analysis results might be incorrect as well.

These alternatives facilitates the transformation; however, it is important to take into account their advantages and disadvantages that are presented in Table II.

VIII. DISCUSSION

The value of an enterprise model is proportional to the knowledge extracted by its analysis. However, different types of imperfection can appear:

- Inaccurate, erroneous, contradictory, ambiguous, or unreliable information can be placed in attributes or relationships.
- Incorrect sources or targets of relations can be assigned.
- Missing information can emerge in attributes or relationships.

These imperfections affect the analysis stage of enterprise modeling, as they imply inaccurate analysis results, as well as additional costs each time the model is rejected, refined, or more information is recollected.

Analyzing imperfection can be expensive, but it is better to invest effort and time identifying the level of imperfection in the model than using the model to perform business analysis that might provide inaccurate, or even incorrect results. Then, the initial cost of analyzing imperfection can be high, but such cost is compensated when the model is ready to be used for performing business analysis.

To minimize these risks, assessing the impact of imperfection to enterprise analysis, and discover new imperfection, two imperfection analysis techniques have been presented, each one focusing on different characteristics of the model. On the one hand, when the analyst is interested in objectively measuring the imperfection of a model based on the modeling language characteristics, domain analysis techniques provide quantitative results taking into account those characteristics. On the other hand, when the model is considered as a graph and the analyst requires knowing the number of imperfect elements, attributes, or relationships, the topological analysis provides these kinds of results. Consequently, each technique does not exclude the other one; then, they should be used together in order to find the desired information regarding imperfection.

IX. RELATED WORK

Some approach in the literature address quality metrics in Enterprise Models, and provide methods and tools for their modeling and analysis. Frank [3] presents an approach called Multi-perspective Enterprise Modeling *(MEMO)* to represent different perspectives of the enterprise. It offers a framework that includes common enterprise abstractions, represented by the languages *Strategy Modeling Language (MEMO-SML)*, *Organization Modeling Language (MEMO-OrgML)*, and *Object-Oriented Modeling Language (MEMO-OML)*. The MEMO architecture, which describes the specification and integration of these languages, provides extension mechanisms that allow the inclusion of additional languages. MEMO does not consider imperfect information; nevertheless, its extensibility allows including this kind of information by updating each metamodel, or even the MEMO

meta-metamodel. However, after modifying the metamodels, and modeling imperfect elements, there is no explicit way to know the degree of imperfection of the model, i.e. to analyze the imperfection as such, in order to decide whether the model is appropriate for a concrete purpose, such as the analysis of the model, or the delivery of a diagram of a part of the model to a stakeholder.

Narman *et al.* [31] propose a metamodel to support the creation of enterprise architecture models, which are suitable to quality attributes analyses. The quality attributes proposed by the authors are *Accuracy*, *Efficiency*, *Interoperability*, *Maintainability*, *Reliability*, *Security*, *Suitability*, and *Usability*. This approach uses extended influence diagrams to describe the enterprise architecture analysis, and allows the inclusion of information related to the proposed quality attributes. Our approach supports not only imprecise information [32], which allows ranges of values, but also inconsistent, vague, uncertain, and incomplete information, so modelers do not need to transform any information taken from the sources.

Johnson *et al.* [33] present a tool for creating scenarios to perform enterprise architecture analysis. Also, the tool generates quantitative assessments of these scenarios around various quality attributes such as: *accuracy*, *availability*, *functional suitability*, *information security*, *interoperability*, *maintainability*, *performance*, and *usability*. This tool allows the decoration of model elements, based on the relevant quality attributes. In our approach, we support the detailed assessment of the imperfection of the model by allowing the analyst to first navigate through the imperfect elements and relationships of the model and then obtaining quantitative results that facilitate this assessment.

Regarding *Imperfection*, some authors have proposed multiple classifications, as well as methods to identify and mitigate specific kinds of imperfection. For instance, Smets [21] proposes a classification for imperfect information, emphasizing on imprecision, inconsistency, and uncertainty. The author also presents methods for modeling imprecision and uncertainty, grouped into symbolic and quantitative models.

Henricksen *et al.* [7] depict four different kinds of imperfection in entity attributes: *unknown*, *ambiguous*, *imprecise*, and *erroneous* attributes. Based on this classification, they propose a modeling approach based on Object Role Modeling (ORM), designed to support a variety of tasks across the software life cycle. Xiao *et al.* [24] propose a method to represent *uncertainty* and *imprecision* in product development processes. The authors formalize and integrate these attributes into the modeling process through a workflow based on an extension to UML (Unified Modeling Language). In addition, they present a metamodel for uncertainty and imprecision, which is the basis for their UML profile.

Even though these proposals [21], [7], [24] identify some kinds of imperfections, our proposal allows to model imperfection in a more detailed manner, as we consider other imperfection attributes such as inconsistency, uncertainty, and incompleteness, and apply these imperfection types to elements, attributes, and relationships of the model.

The **elicitation process** concerns with the recollection of information for constructing a model through interviews, documents, and observation. In the discipline of Requirements Engineering, authors have been interested in imperfection properties such as vagueness, ambiguity, and uncertainty during the elicitation of requirements.

For instance, Bhatia *et al.* [34] propose a theory of vagueness and privacy risk perception, based on the empirical analysis of domain-specific documents. This theory consists of a description of vagueness through exclusive semantic categories, a prediction of the variation of this vagueness, semantic functions to determine how semantic categories predict vagueness, and remarks on people's behavior based on vagueness variations. This theory is useful in the elicitation process, as it helps to identify natural language modifiers that suggest vagueness in the information provided by a source. Furthermore, Ferrari *et al.* [35] present an approach to identify ambiguity in interviews. In their work, the authors classify the types of ambiguity based on term cues in natural language that might reveal the presence of tacit knowledge: under-specified, vague, quantifiers, pronoun, and domain-specific terms. In order to illustrate this classification, the authors present fifteen situations where these categories of term cues arise.

Given that in enterprise modeling there is also an elicitation process where information is gathered from imperfect sources of information, we consider that approaches such as [34], [35] can be useful for starting with less ambiguous and less vague enterprise models. In this paper, we present a classification of enterprise sources based on the information they provide and contemplate additional imperfection types. In addition, we argue that the imperfection of the model is relative to the purposes at hand, and therefore, analysts might identify these imperfections too late in the analysis process, thus raising the cost of obtaining insights from the model.

Famelis and Chechik [36] propose the methodology: "Design-Time Uncertainty Management" (DeTUM), in which modelers develop partial models that contain several possible situations, which collapse to a single one when enough information is available. This methodology describes the life cycle of design-time uncertainty, divided into three stages: Articulation, Deferral, and Resolution. Articulation pertains to the identification and modeling of uncertain elements. The second stage, Deferral, consists of circumventing uncertain aspects of the model, which allows working on fragments of the model that are not impacted by this uncertainty. Finally, Resolution pertains to the refinement of the uncertain points of the model as soon as additional information is available. In this paper, we argue that this lifecycle can be adapted to manage the imperfection of enterprise models.

In the realm of Cyber-Physical Systems Testing, Zhang *et al.* [37], [38] present an uncertainty-wise Modeling Framework called *UncerTum*, with the purpose of applying model-based testing techniques that take into account *environmental uncertainty*. The core of *UncerTum* is the UML Uncertainty Profile (UUP), which allows the modeling of uncertain and vague aspects of UML model elements, and is based on the *U-Model*. Furthermore, the authors propose a modeling methodology that fosters completeness of the uncertainty description, as well as a validation process for eliminating errors introduced by a test modeler. While the authors underscore that "*UncerTum only supports test modeling for enabling the generation of executable test cases*", and thus is applicable to test models –much less detailed than enterprise models– this approach reflects the importance of

providing tools that support the identification and analysis of imperfection with effort.

The practice of using visualization techniques to discover new facts has been examined since the widespread use of enterprise models. Lagerström *et al.* [39] use a visibility matrix in order to uncover the internal structure of enterprise applications. On another note, Fill [40] describes different visualization techniques commonly used for ontology analysis, and proposes a framework for ontology visualization. However, we consider that interactivity is necessary for navigation and exploration of the model, something that is out of the scope of this framework.

X. Conclusion

Enterprise models are created based on information provided by different and heterogeneous enterprise sources. Usually, these sources do not provide perfect information. In addition, it is likely that modelers consider that the information gathered is not enough to be included in the enterprise model. Consequently, enterprise models might not represent the enterprise accurately. Thus, imperfect models represent and structure imperfect information regarding the metamodel of the modeling language. We have identified five types of imperfection; however, further types of imperfection might be found.

When models are imperfect, it is necessary to study and measure the imperfection level of the model in order to 1) determine whether the model is usable for further purposes such as business analysis; 2) take into account the imperfection during the business analysis process; and 3) discover which elements of the model need to be fixed or complemented. Thus, analysis methods for analyzing the imperfection can inform the imperfection level of the model in different ways.

Thus, we have proposed mechanisms to properly manage and keep the information about imperfection. In addition, we provided analysis methods for analyzing the imperfection focusing on two techniques: domain analysis of imperfection, which provides results regarding the concepts and standard characteristics of the modeling language and topological analysis of imperfection, which provides results regarding the entire model.

Using the presented techniques together, analysts can identify and navigate the imperfect elements, attributes, and relationships of the model, for obtaining visual and quantitative results, which are useful for determining how imperfect the model is, where the imperfection is located, and whether or not the model can be used for further purposes.

In addition, these techniques can also help analysts on discovering possible incompleteness points that were ignored in the modeling phase.

References

[1] S. Hawking and L. Mlodinow, *The grand design*. Random House LLC, 2010.

[2] J. Bézivin, "On the unification power of models," *Software and Systems Modeling*, vol. 4, no. 2, pp. 171–188, 2005.

[3] U. Frank, "Multi-perspective enterprise modeling: foundational concepts, prospects and future research challenges," *Software and Systems Modeling*, vol. 13, no. 3, pp. 941–962, 2014.

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

[5] 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.

[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] K. Henricksen and J. Indulska, "Modelling and using imperfect context information," in *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*. IEEE, 2004, pp. 33–37.

[8] H. Florez, M. Sanchez, and J. Villalobos, "Supporting drafts for enterprise modeling," in *2014 IEEE 9th Computing Colombian Conference (9CCC)*. IEEE, 2014, pp. 200–206.

[9] H. Florez, M. Sánchez, and J. Villalobos, "Embracing Imperfection in Enterprise Architecture Models," *CEUR Workshop Proceedings*, vol. 1023, pp. 8–17, 2013.

[10] H. Jonkers, M. Lankhorst, R. Van Buuren, S. Hoppenbrouwers, M. Bonsangue, and L. Van Der Torre, "Concepts for modeling enterprise architectures," *International Journal of Cooperative Information Systems*, vol. 13, no. 03, pp. 257–287, 2004.

[11] M. Y. Haouam and D. Meslati, "Towards automated traceability maintenance in model driven engineering," *IAENG International Journal of Computer Science*, vol. 43, no. 2, pp. 147–155, 2016.

[12] Y. Rhazali, Y. Hadi, I. Chana, M. Lahmer, and A. Rhattoy, "A model transformation in model driven architecture from business model to web model," *IAENG International Journal of Computer Science*, vol. 45, no. 1, pp. 104–117, 2018.

[13] A. Becker and D. Görlich, "What is game balancing?-an examination of concepts," *ParadigmPlus*, vol. 1, no. 1, pp. 22–41, 2020.

[14] C. Balsa, C. V. Rodrigues, I. Lopes, and J. Rufino, "Using analog ensembles with alternative metrics for hindcasting with multistations," *ParadigmPlus*, vol. 1, no. 2, pp. 1–17, 2020. [Online]. Available: https://journals.itiud.org/index.php/paradigmplus/article/view/11

[15] D. Sanchez and H. Florez, "Model driven engineering approach to manage peripherals in mobile devices," in *International Conference on Computational Science and Its Applications*. Springer, 2018, pp. 353–364.

[16] S. M. Sutton Jr and I. Rouvellou, "Modeling of software concerns in cosmos," in *Proceedings of the 1st international conference on Aspect-oriented software development*. ACM, 2002, pp. 127–133.

[17] S. H. Spewak and S. C. Hill, *Enterprise architecture planning: developing a blueprint for data, applications and technology*. QED Information Sciences, Inc., 1993.

[18] S. H. Spewak and M. Tiemann, "Updating the enterprise architecture planning model," *Journal of Enterprise Architecture*, vol. 2, no. 2, pp. 11–19, 2006.

[19] M. West, *Developing high quality data models*. Elsevier, 2011.

[20] H. Astudillo, J. Pereira, and C. López, "Identifying "interesting" component assemblies for NFRs using imperfect information," in *European Workshop on Software Architecture*. Springer, 2006, pp. 204–211.

[21] P. Smets, "Imperfect information: Imprecision, and uncertainty," *Uncertainty Management in Information Systems*, vol. 1996, pp. 225–254, 1996.

[22] T. Hayashi and R. Wada, "Choice with imprecise information: an experimental approach," *Theory and Decision*, vol. 69, no. 3, pp. 355–373, 2010.

[23] X. Li, X. Dai, J. Dezert, and F. Smarandache, "Fusion of imprecise qualitative information," *Applied Intelligence*, vol. 33, no. 3, pp. 340–351, 2010.

[24] J. Xiao, P. Pinel, L. Pi, V. Aranega, and C. Baron, "Modeling uncertain and imprecise information in process modeling with uml," in *Fourteenth International Conference on Management of Data (COMAD), Mumbai*, 2008.

[25] A. Hunter and S. Konieczny, "Approaches to measuring inconsistent information," in *Inconsistency Tolerance*. Springer, 2005, pp. 191–236.

[26] C. B. Anagnostopoulos, Y. Ntarladimas, and S. Hadjiefthymiades, "Situation awareness: Dealing with vague context," in *ACS/IEEE International Conference on Pervasive Services*, 2006, pp. 131–140.

[27] J. Dai and Q. Xu, "Approximations and uncertainty measures in incomplete information systems," *Information Sciences*, 2012.

[28] H. Florez, M. Sanchez, and J. Villalobos, "iArchiMate: A Tool for Managing Imperfection in Enterprise Models," in *18th IEEE International Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW)*. IEEE, 2014, pp. 201–210.

[29] H. Florez, M. Sánchez, and J. Villalobos, "Drafting enterprise models," in *Modeling Methods for Business Information Systems Analysis and Design*. IGI Global, 2019, pp. 183–214.

[30] The Open Group, *ArchiMate 2.0 Specification*. Van Haren Publishing, 2012.
[31] P. Närman, P. Johnson, and L. Nordstrom, "Enterprise Architecture: A Framework Supporting System Quality Analysis," in *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*. IEEE, oct 2007, pp. 130–141.
[32] H. Florez, M. Sánchez, and J. Villalobos, "Analysis of Imprecise Enterprise Models," in *International Workshop on Business Process Modeling, Development and Support*. Springer International Publishing, 2016, pp. 349–364.
[33] P. Johnson, E. Johansson, T. Sommestad, and J. Ullberg, "A Tool for Enterprise Architecture Analysis," in *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*. IEEE, oct 2007, pp. 142–153.
[34] J. Bhatia, T. D. Breaux, J. R. Reidenberg, and T. B. Norton, "A theory of vagueness and privacy risk perception," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, Sep. 2016, pp. 26–35.
[35] A. Ferrari, P. Spoletini, and S. Gnesi, "Ambiguity Cues in Requirements Elicitation Interviews," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, Sep. 2016, pp. 56–65.
[36] M. Famelis and M. Chechik, "Managing design-time uncertainty," *Software & Systems Modeling*, pp. 1–36, Mar. 2017.
[37] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren, "Understanding Uncertainty in Cyber-Physical Systems: A Conceptual Model," in *Modelling Foundations and Applications*. Springer, Cham, Jul. 2016, pp. 247–264.
[38] M. Zhang, S. Ali, T. Yue, R. Norgren, and O. Okariz, "Uncertainty-Wise Cyber-Physical System test modeling," *Software & Systems Modeling*, pp. 1–40, Jul. 2017.
[39] R. Lagerstrom, C. Baldwin, A. MacCormack, and S. Aier, "Visualizing and measuring enterprise application architecture: an exploratory telecom case," in *System Sciences (HICSS), 2014 47th Hawaii International Conference on*. IEEE, 2014, pp. 3847–3856.
[40] H.-G. Fill, *Visualisation for Semantic Information Systems*. Gabler Verlag, 2009.

Hector Florez is Full Professor at the Universidad Distrital Francisco Jose de Caldas, Bogota, Colombia. He is Ph.D. in Engineering at the Universidad de los Andes. His research interests are: enterprise modeling, model driven engineering, and enterprise analysis.

Mario Sanchez is Associate Professor at the Universidad de los Andes, Bogota, Colombia. He is Ph.D. in Engineering at the Universidad de los Andes. His research interests are: enterprise architectures, and executable models.

Jorge Villalobos is Full Professor at the Universidad de los Andes, Bogota, Colombia. He is Ph.D. in Informatics at the Université Joseph Fourier, Grenoble, France. His research interests are: enterprise modeling and software design.