An iStar 2.0 Syntax Validation Formal Rules and Its Implementation on a New Translator

Francisco Kenandi Cahyono, Bayu Hendradjaya, Hari Purnama
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
franciscokenandi@gmail.com , bayu@std.stei.itb.ac.id, purnama@informatika.org

***Abstract*— i * framework is a socio-technical goal-based modeling framework and models the actors in the project/system environment. In 2016 iStar 2.0 was proposed to further evolve i* basic concepts to be more acceptable for wider users. Therefore, it motivates us to propose a formal rule for validating iStar 2.0 in XML-based modelling standard similar to iStarML, called iStarML 2.0. In addition to validation process, this paper proposes formal methods for translating i* to iStar 2.0 model and iStar 2.0 to class diagram. iStarService is a tool developed for iStar 2.0 modelling based on iStarML 2.0 with functionalities such as iStar 2.0 model validation, i* to iStar 2.0 model translation, and iStar 2.0 model to class diagram translation. It is implemented in form of web API using Java and had been tested with various models from multiple iStar proceedings.**

Keywords—i*;iStar 2.0;class diagram; iStarML; validation; translation;tool

I. INTRODUCTION

i* framework is socio-technical modelling framework proposed in mid-90 for early requirements and it models intentional actors. i* models elements such as actors, their links, dependencies with other actors, and their intentions (e.g. goals to fulfill, resources available to them, qualities to be achieved, and tasks needed to be performed). Over the years, the academic community has evaluated the framework and found that the framework itself is relatively challenging to adopt for newcomers[1]. There are some challenges present, one of them is the complexity of an i* model itself compared to other more common modelling framework such as UML. Other problem lies in the diverse set of extensions and variations to pick up and learn based on their functions and specialties.

The academic community responded with further refining the framework and widely agreed on a set of core i* concepts in 2016. The new core concepts is called iStar 2.0 to distinguish itself from its predecessor [2]. With the new framework released, the community needs modelling tools to learn and evaluate iStar 2.0. Few (two) iStar 2.0 tools has been made since the iStar 2.0 framework first released with basic functionalities such as creating and validating an iStar 2.0 diagram. Unfortunately, both of the tools use their own standard to represent the iStar 2.0 model.

There had been works that tried to bridge early requirements modelled in earlier and later requirements by proposing a translation formal rule from i* model to UML's class diagram (as a modelling framework used in model driven development). The formal rule was proposed by [3] which further extended in [4] and finally [5] formalized the translation method. Until now there has not been any work that adapts the formal method for iStar 2.0 framework therefore, a formal rule for translating iStar 2.0 model to class diagram needs to be proposed.

II. RELATED WORKS

Few works had been done regarding i* & iStar 2.0 model validation and translation. piStar [6] and leaf[1] is an open source web based iStar 2.0 modelling tool. It validates their internal representation of a visual element. Before iStar 2.0, iStarML[7] was proposed in order to be a standard of interoperable i* framework model. A formal rule for translating i* model to class diagram was also proposed and explained in prior section.

III. PROPOSED SOLUTION

In order to solve the aforementioned problems, iStarService tool is proposed with four main aspects: XML-based iStar 2.0 model standard, validation of the model, translation from i* to iStar 2.0 model, as well as translation from iStar 2.0 model to class diagram. The tool runs on back-end side of the web platform in form of APIs. iStarService only processes XML models and does not carry out diagram visualization or diagram creation tasks. Therefore, another visualization tool is needed. Presently only one visualization tool communicates with iStarService which is cistar, developed by Putu Arya Pradipta. By visualizing the iStar, users can manipulate and explore the diagram.

A. iStarML 2.0

iStarService uses its own standard for representing the iStar 2.0 model. The iStarML 2.0 standard is adopted from iStarML, a XML based standard for representing i* model [7] which is adjusted according to iStar 2.0 rule with recommendations

[1] http://www.cs.toronto.edu/~amgrubb/leaf-2.0/Tool.html (last accessed June 26th 2019)

978-1-7281-4992-9/19/$31.00 ©2019 IEEE

from [8]. The standard schema is defined in an XSD (XML Schema Definition).

B. Validation

Fig. 1 shows the proposed validation schema in iStarService. In general, there are two validation phases: (1) structural validation and (2) complementary validation. Structural validation utilizes XSD to restrict element definition in an XML. Structural validation process validates an iStarML 2.0 model towards its XSD and it restricts factors such as the element arrangement, enumerating attributes, and number of each element.

There are some limitations an XSD can restrict. For instance, in a dependency, the depender element might refer to a nonexistent element or to an element that is not inside the depender. An additional rule checking process is required thus called the complementary validation process.

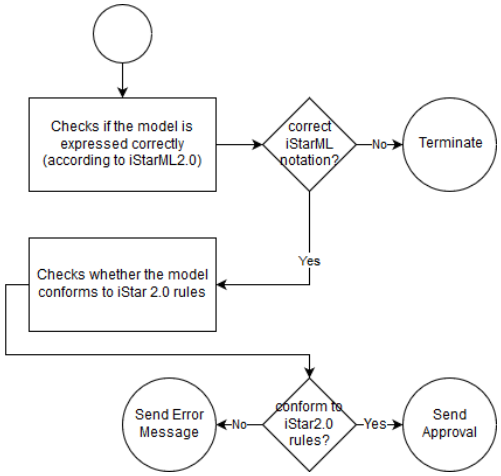

Fig. 1 Proposed validation process

C. i* to iStar 2.0 Translation

A set of translation rules is proposed in table 1 largely based on the i* to iStar 2.0 comparison table in the iStar 2.0 official site[2]. The i* rules used is the i* Wiki[3] variant. Apparently there are some cases not completely anticipated by the referenced table, for instance task decomposition link (a link between task and goal/task/resource/softgoal) is translated into refinement link (a link between task and goal) resulting in a potential missing translation for task decomposition link between task and resource/softgoal. Therefore, the existing translation rule needs to be further detailed and expanded.

Expanding the rule is done by comparing the definition between i* and iStar 2.0 elements and find the closest semantic translation between the two models. The expanded translation rules cover all of the intentional element links. Means end link translation is further detailed into an OR refinement link (multiple alternatives to reach an end) while task decomposition link between task and goal/task element is translated into an AND refinement link (since all of the decomposed element needs to be fulfilled). Task decomposition link between tasks and softgoals/resources are translated into needed by link and qualification link in iStar 2.0. It is noteworthy that there are minor semantic differences between the new translation, for instance the softgoal in task decomposition needs to be fulfilled for the task to be completed while a qualification link associates a task and a quality that it should fulfill.

There are some issues to be addressed such as the semantic of AND & OR in refinement link as a result of goal-task task decomposition translation, and iStar 2.0 rule which stated a depender element should not be refined or contributed.

After proposing the formal rule, a formal method that applies the formal rule is also proposed and shown in Fig. 2. iStarService implemented the formal method in its i* to iStar 2.0 translation feature. After implementing the rules for each i* element, the previously addressed issues should also be checked.

Table 1. Proposed I And Istar 2.0 Mapping Rules*

	i* Element	iStar 2.0 Element
Actor	actor	actor
	role	role
	agent	agent
	position	actor
Actor link	is a	is a
	is part of, plays, occupies, covers	participates in
	instance of	-
Intentional element	goal	goal
	task	task
	resource	resource
	softgoal	quality
Intentional element link	means end	refinement or
	task decomposition [goal, task]	refinement and
	task decomposition [resource]	needed by
	task decomposition [softgoal]	qualification
	contribution [make, help, hurt, break]	contribution [make, help, hurt, break]
	contribution [some+]	contribution [help]
	contribution [some-]	contribution [hurt]
	contribution [and,or,unknown]	-

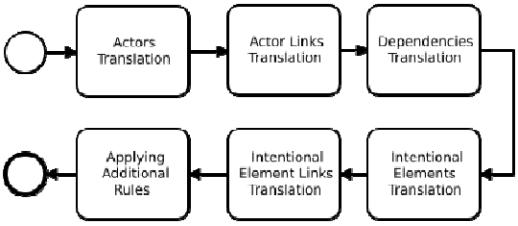

Fig. 2 i* to iStar 2.0 translation formal method

[2] https://sites.google.com/site/istarlanguage/diff
[3] http://istar.rwth-aachen.de/tiki-view_articles.php (last accessed June 26th 2019)

D. iStar 2.0 to Class Diagram Translation

Since i* to class diagram rule was proposed and refined in [3], [4], [9], translation rules for iStar 2.0 has not been proposed. Taking into account that there are some similarities between i* framework and iStar 2.0, the translation rule should also be similar. The new translation rule can be seen in table 2 adapted from i* translation rules and similar with the i* translation rule, there are two major results which is the class diagram itself and an OCL template. According to [10], there were some problems following the translation rules. The iStar 2.0 translation tool does not solve these problems although some problems are being avoided. Since avoiding the problem does not equal to solving it, the problems originated in i* translation rules are also carried into the iStar 2.0 translation rules.

One such example comes from the first problem of the original i* translation rule regarding resource representation. [4] proposed that a resource can be translated into class or attribute based on its properties and behavior. Due to the limitation of iStarService's implementation, it is not possible to automatically classify an instance of resource. Therefore, regarding the translation of resource, iStarService reverts back to [3] where every resource are translated into classes.

Fig. 3 shows the formal method in translating an iStar 2.0 model to class diagram. Each process implements the formal rule of every iStar 2.0 element respectively. Generate Class Diagram and Generate OCL activities sets a boundary between class diagram generation OCL generation activities.

Table 2. Proposed Istar 2.0 And Class Diagram Mapping Rule

iStar 2.0 Element		Class Diagram
Actor	Actor, Role, Agent	Class
Actor Link	IS-A	Aggregation/Specialization link
	Participates In	Association link named 'participates in'
Intentional Element	Task	Private method
	Resource	Class with public boolean attribute named 'availability'
	Goal and quality	Private boolean attribute.
Intentional Element Link	Needed By	OCL template: precondition of task and availability of resource followed by postcondition of task.
	Refinement	OCL template: [Task&Goal] Disjunction/ conjunction of task precondition and goal fulfilment followed by task post condition. [Goal&Goal] Goal fulfilment implies goal fulfilment.
	Contribution	OCL template: conjunction of intentional element fulfilment implies fulfilment/failure of quality.
	Qualification	OCL template: conjunction of intentional element fulfilment and quality fulfilment.
Dependency	Task dependency	Public method in dependee
	Resource dependency	Association link named 'dependency' between dependee, class with public boolean attribute named 'availability', and dependum
	Goal & quality dependency	Public boolean attribute in dependee

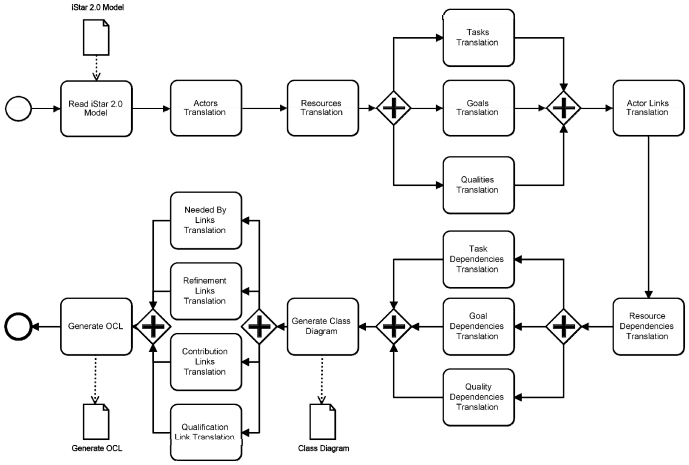

Fig. 3 iStar 2.0 to class diagram translation formal method

IV. DESIGN & IMPLEMENTATION

iStarService was implemented in Java and used Spring framework to support the REST request handling. It utilized various libraries which will be described in the next section.

Fig. 4 shows the general interaction between functionalities while Fig. 5 shows the components of iStarService. Syntax validator received iStarML 2.0 file, applied the defined rule, and returned the validity status of the model: true if the model is correct, false and its error message if the model does not conform to the iStar 2.0 rule. iStar 2.0 to class diagram translator received validated model from the syntax validator and applied the translation rule. It returned the class diagram image and OCL file. The i* to iStar 2.0 translator received the iStarML file generated from a specific i* modelling tool called OpenOME and converts it into iStarML 2.0 file.

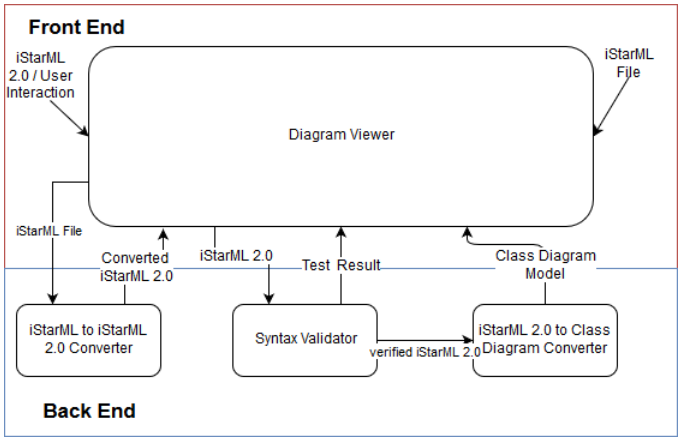

Fig. 4. General architecture and interaction of iStarService (back end part)

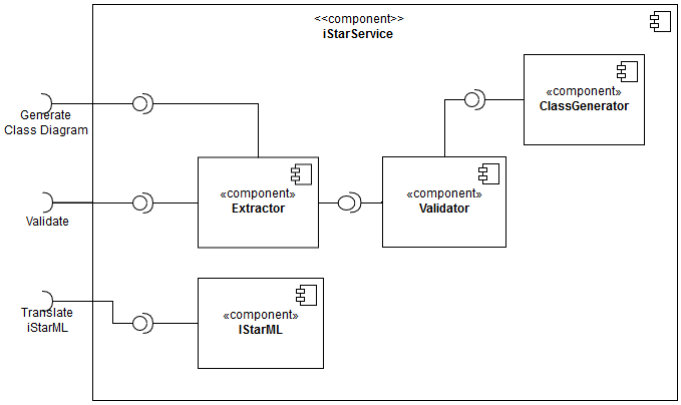

Fig. 5. Component diagram of iStarService

A. *Extractor Component*

This component extracts the iStarML 2.0 file and saves it into an intermediary class. First, after the initial call and the iStarML 2.0 model is received, it will validate the structure of the XML using the defined XSD with the help of javax XML libraries. Second, using Java's DOM parser library the component will generate the DOM model to be iterated and saved into the class representation model. This allows the tool to reject the model if it is structurally incorrect before further process the model.

B. *Validator Component*

Validator component implements the validation aspect of the tool, the complementary validation phase to be precise. After undergoing the structural validation phase in the extractor component, this component receives the class model and applies the complementary validation rules.

C. *Class Generator Component*

As the name suggests, Class Generator component translates the validated iStar 2.0 model into class diagram image and OCL file using the defined formal translation method. Class diagram image is generated using a library called PlantUML[4]. PlantUML is a library for generating UML diagrams based on DSL and is available in various languages and plugins. While applying the translation rule, a DSL specification is generated. After reaching the last process regarding the generation of class diagram image, the component calls PlantUML library to generate the image based on the generated DSL.

D. *iStarML Component*

iStarML component deals with i* and iStar 2.0 functionality and it does not communicate with other components since the validation and extraction of the i* model is not done by iStarService. Each iStarML model received is validated using an open source iStarML validation library called ccistarml followed by the implementation of the translation method. There are some differences in iStarML standard in models generated by OpenOME that needs to be adjusted. OpenOME's standard of an actor in SD view is to model it as an actor element whereas iStarService's standard needs the boundary element as well.

V. VALIDATION

A. *Case Study*

The buyer drives ecommerce[5] and smart CD system [3] case study are used in this case study for i* to iStar 2.0 translation and iStar 2.0 to class diagram respectively.

After the model shown in Fig. 6 being applied by the i* translation method, resulting in Fig. 7, the following aspects will be changed: changing softgoals into qualities (although the graphical notation will still be the same) and the elimination of loyalty, Pay for purchasing [Service] and good quality [Service] dependums since their depender elements are refined / contributed.

The application of iStar 2.0 formal method for class diagram translation of model shown in Fig. 8 will result in Fig. 9 and OCL templates below. Notice that seven classes are generated from six actors and one resource. Some of the actor's classes receive additional public methods from their quality/goal/task dependencies. The store class receives two additional public attributes from its quality dependencies while the SmartCD class receives an additional public method and attribute from its goal and task dependency. Internet Sales class receives 3 additional public attributes and 4 additional public methods from quality, goal, and task dependencies. The Inventory class receives 2 public methods from 2 task dependencies and the Financial class receives 1 public method from its task dependency.

```
Internet sales:: Register client
pre: Interact by site.preCondition="Value"Register
client.preCondition="Value"
post:  and Interact by site.postCondition="Value"Register
```

[4] http://plantuml.com/ (last accessed November 4th 2019)
[5] http://istar.rwth-aachen.de/tiki-index.php?page=Strategic+Rationale+Example+Model%3A+Buyer+Drive+E-Commerce+from+Yu01&structure=i%2A+Guide (last accessed June 28th 2019)

```
client.postCondition="Value"

Internet sales:: Assistance to client
pre: Interact by site.preCondition="Value"Assistance to
client.preCondition="Value"
post:  and Interact by site.postCondition="Value"Assistance to
client.postCondition="Value"

Internet sales:: Search CD
pre: Interact by site.preCondition="Value"Search
CD.preCondition="Value"
post:  and Interact by site.postCondition="Value"Search
CD.postCondition="Value"

Internet sales:: Super search
pre: Search CD.preCondition="Value"Super search.preCondition="Value"
post:  and Search CD.postCondition="Value"Super
search.postCondition="Value"

Internet sales:: Fast search
pre: Search CD.preCondition="Value"Fast search.preCondition="Value"
post:  and Search CD.postCondition="Value"Fast
search.postCondition="Value"

Store:: To Attract Customers
pre: To Attract Customers.preCondition="Value" and Good Price=true
post: To Attract Customers.postCondition="Value"

Store:: To Attract Customers
pre: To Attract Customers.preCondition="Value" and Friendly
staff=true
post: To Attract Customers.postCondition="Value"

Store:: Volume purchase [High]
Volume purchase [High]=true implies Good Price=true

Store:: Discount price
Discount price=true implies Good Price=true

Store:: Delivery
pre: To Attract
Customers.preCondition="Value"Delivery.preCondition="Value"
post:  and To Attract
Customers.postCondition="Value"Delivery.postCondition="Value"

Store:: Sales by web
pre: To Attract Customers.preCondition="Value"Sales by
web.preCondition="Value"
post:  and To Attract Customers.postCondition="Value"Sales by
web.postCondition="Value"

Store:: Maintain stock
pre: To Attract Customers.preCondition="Value" and Have all
titles=true and Replenish.preCondition="Value"Maintain
stock.preCondition="Value"
post:  and To Attract Customers.postCondition="Value" and
Replenish.postCondition="Value"Maintain stock.postCondition="Value"

Store:: Buy new releases
pre: Maintain stock.preCondition="Value"Buy new
releases.preCondition="Value"
post:  and Maintain stock.postCondition="Value"Buy new
releases.postCondition="Value"
```

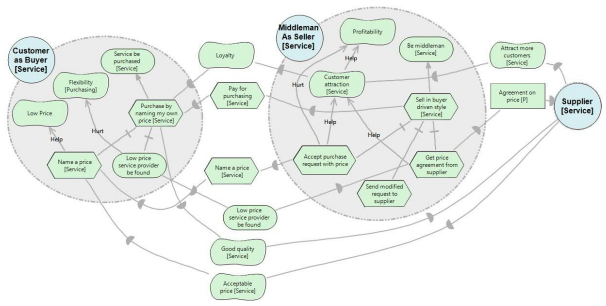

Fig. 6 Buyer drives ecommerce case

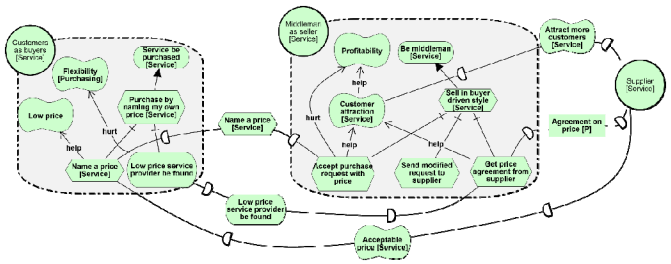

Fig. 7 iStar 2.0 version of buyer drives ecommerce case

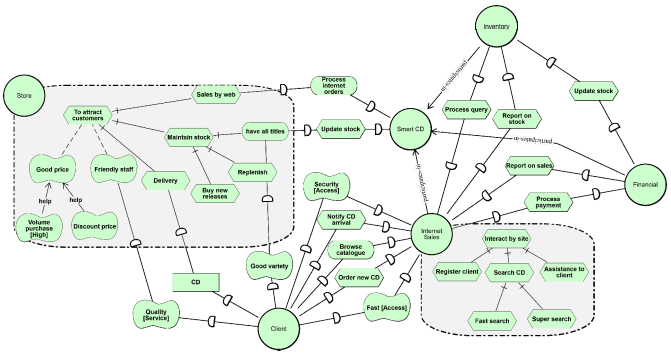

Fig. 8 Smart CD case

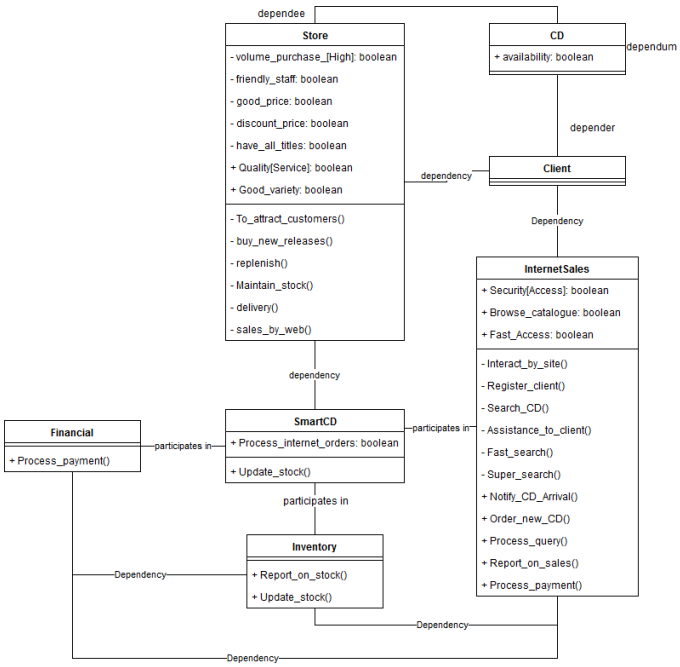

Fig. 9 Smart CD class diagram

B. Testing

The tool had been tested with test cases of 33 i* diagrams and 4 iStar 2.0 diagrams from iStar proceedings ranging from 2011 until 2018. The validation feature is able to catch any mistakes from current test cases meanwhile the translation features are able to handle all of the correct cases. There are

some cases expressed incorrectly and the tool are not able to translate incorrect models since incorrect models are rejected by the tool.

VI. Evaluation

The proposed validation rules and implementation are able to accommodate every types and rules of element in iStar 2.0 framework. Using iStarML 2.0 as a standard for representing the iStar 2.0 model is beneficial for people who prioritize integrity of the model since XML has strict structure definition. However, the rigid nature of XML schema may prove to be a disadvantage if the iStar 2.0 framework is being extended or changed.

Not all i* elements can be syntactically translated into iStar 2.0 and still retain its semantic correctness. The proposed method, although able to translate most of the elements, ignores some of the untranslatable elements. As a result, some information will be lost in the translation process. Since the proposed method produces syntactically valid model, the semantic of each instance of the element is not guaranteed to be correct, therefore users are heavily advised to reevaluate the translated model in order to retain semantical correctness and recomplete the information missing from the translation process.

The proposed iStar 2.0 to class diagram translation method does not solve the problems of the old method defined in [10]. In the future a more refined method and tool that solves those problems should be proposed and implemented. Lastly, sometimes images generate by PlantUML from complex diagram may appear disorganized thus reducing readability.

VII. Conclusion

The formal rules for validating iStar 2.0 model represented in XML format has been proposed and implemented in a iStarService.

A formal method for translating i* model to iStar 2.0 model represented in XML structure has been proposed and implemented in iStarService. Automation of translation process is implemented in hope of lowering the barrier of entry/adoption of iStar 2.0 framework for people who are familiar with i* framework.

A formal method for translating iStar 2.0 model (represented in XML) to class diagram has been proposed and implemented in iStarService in hope of further bridging GORE model and MDD model.

Future work will concentrate on completing the missing informations during the translation process and rephrase error messages to be more insightful.

References

[1] J. P. Carvallo and X. Franch, “Lessons learned on the use of i* by non-technical users,” *CEUR Workshop Proc.*, vol. 1157, no. Cm, 2014.

[2] F. Dalpiaz, X. Franch, and J. Horkoff, “iStar 2.0 Language Guide,” pp. 1–15, 2016.

[3] J. F. Castro, F. M. R. Alencar, G. A. C. Filhol, and J. Mylopoulos, “Integrating organizational requirements and object oriented modeling,” pp. 146–153, 2002.

[4] F. Alencar, F. Pedroza, J. Castro, and R. Amorim, “New Mechanisms for the Integration of Organizational Requirements and Object Oriented Modeling,” *Proc. 6th Work. Requir. Eng. - WER 2003*, pp. 109 – 123, 2003.

[5] J. Melo, A. Sousa, C. Agra, and F. Alencar, “Formalization of the i * Mapping Rules for Class Diagram,” no. istar, pp. 97–101, 2015.

[6] J. Pimentel and J. Castro, “PiStar tool - A pluggable online tool for goal modeling,” in *Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018*, 2018, pp. 498–499.

[7] C. Cares, X. Franch, A. Perini, and A. Susi, “IStarML: An XML-based interchange format for i* models,” in *CEUR Workshop Proceedings*, 2008, vol. 322, pp. 13–16.

[8] C. Cares and L. López, “Towards iStarML 2 . 0 : Closing Gaps from Evolved Requirements,” pp. 0–5.

[9] J. Melo, A. Sousa, C. Agra, J. Júnior, J. Castro, and F. Alencar, “Formalization of Mapping Rules from iStar to Class Diagram in UML,” *Proc. - 29th Brazilian Symp. Softw. Eng. SBES 2015*, pp. 71–79, 2015.

[10] F. Alencar *et al.*, “From i * to OO-Method : Problems and Solutions,” pp. 9–14, 2010.