Verifying Goal-Oriented Specifications Used in Model-Driven Development Processes

Giovanni Giachetti[1], Beatriz Marín[2], Lidia López[3], Xavier Franch[3], and Oscar Pastor[4]

[1] Universidad Andres Bello, Facultad de Ingeniería, Santiago, Chile. Phone number: (+56) 2 2770 3514, email: giovanni.giachetti@unab.cl (corresponding author)
[2] Universidad Diego Portales, Facultad de Ingeniería, Escuela de Informática y Telecomunicaciones, Santiago, Chile
[3] Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
[4] Centro de Investigación en Métodos de Producción de Software, Universidad Politécnica de Valencia, Valencia, Spain

SUMMARY

Goal-oriented requirements engineering promotes the use of goals to elicit, elaborate, structure, specify, analyze, negotiate, document, and modify requirements. Thus, goal-oriented specifications are essential for capturing the objectives that the system to be developed should achieve. However, the application of goal-oriented specifications into model-driven development (MDD) processes is still handcrafted, not aligned in the automated flow from models to code. In other words, the experience of analysts and designers is necessary to manually transform the input goal-oriented models into system models for code generation (models compilation). Some authors have proposed guidelines to facilitate and partially automate this translation, but there is a lack of techniques to assess the adequacy of goal-oriented models as starting point of MDD processes. In this paper, we present and evaluate a verification approach that guarantees the automatic, correct, and complete transformation of goal-oriented models into design models used by specific MDD solutions. In particular, this approach has been put into practice by adopting a well-known goal-oriented modeling approach, the *i** framework, and an industrial MDD solution called Integranova.

KEYWORDS: Model-driven development; verification approach; goal-oriented requirements; *i** framework

1 Introduction

The software development context is rapidly moving towards the model-driven development (MDD) paradigm [1], which has motivated the emergence of multiple approaches oriented to automating the final software product generation by means of model compilation processes. MDD is a generic term used interchangeably with MDE (model-driven engineering) [2], which describes an approach where systems are represented as models defined to conform to a metamodel, and model transformations are used to manipulate the representation of systems [3]. Just as any software development process does, MDD also requires an appropriate requirement engineering activity to obtain software products that fit to the customers' needs [4].

There are several works that intend to demonstrate the relevance of bridging the gap between requirement specifications and system modeling, such as [5] [6] and the systematic review about requirement engineering approaches and their MDD application presented in [7]. In this direction, several approaches – [4] [8] [9] [10] [6] [11] – have fostered the use of high-level analysis models (*i.e.*, requirement models) as part of a sound software MDD process. A representative example is the MDA approach [12], which proposes the definition of a *computation-independent model* as the starting point of the software development process [13]. However, most of the current requirement approaches are not automatically applied [14] or are not based on modeling standards [15]. Thus, an effective solution that includes requirement models as part of a complete, standardized, and automatic MDD process [16] is still an unsolved challenge [7].

Over the last two decades, goal-oriented modeling has been widely considered in requirements and software engineering. Horkoff *et al.* provides evidence that there is active research on transforming goal-oriented models to UML artifacts, which are mostly used at early requirement stages [17]. Goals have also been recognized as essential components of requirements elicitation [18]. Moreover, there are some approaches that use goal models as the starting point of MDD processes [19] [20] [21]**.** In [22], it is demonstrated how the integration of goal-oriented modeling into an MDD process leads to the fulfillment of the requirements of the CMMi software process maturity model. Thus, goal-oriented modeling is a good choice for requirement specification in MDD processes. However, there is still a gap to bridge between goal-oriented modeling and MDD processes, since goal-oriented models are focused on early development (analysis) stages, not centered on automatic software generation.

Certain approaches have defined guidelines to perform the integration of goal-oriented models into MDD processes. In general terms, this integration involves the addition of particular modeling information (modeling extensions) into the reference goal-oriented specification, making it possible to transform the defined analysis models into design models used by concrete MDD tools. However, these guidelines consider the input goal-oriented models and the additional modeling information to already be perfectly defined for the model-to-model transformation process [23] (from analysis models to design models). In real application contexts, this desired scenario is not feasible – models are defined by humans, and humans commit errors. Hence, verification mechanisms are necessary to assure the proper use of goal-oriented models in MDD processes. At this point, it is important to mention that the verification process corresponds to the confirmation with objective evidence as to whether the software and its associated products and processes conform to the requirements regarding the completeness, correctness, consistency, and accuracy [24] [25]. Taking into account that the manual verification of the transformation of analysis models to design models is not a trivial task, since it is necessary to have a deep understanding of the target MDD technology and the transformation rules involved, the automation of verification mechanisms is of paramount importance.

This paper presents the definition and validation of a verification approach called VeMI (verification for model integration). The VeMI approach is applied to assess the transformations of enhanced goal-oriented models into the system models in MDD processes. In particular, we demonstrate how the VeMI approach can be used to fix and improve the input goal-oriented models to assure the completeness of the system models that are specific for MDD solutions. Thus, VeMI reduces the effort related to completing and refining these system models at design time.

For the specification of goal-oriented models, we selected the *i** framework [26] as reference goal-oriented modeling approach, since it is used in several activities and contexts of software engineering at the early phases of requirements engineering [27]. Moreover, the versatility and expressive power

of *i** is extensively documented [28], which facilitates the adaptation of the proposed approach to different development domains.

As a target MDD solution, we have considered the Integranova technology [20], since it has more than 10 years of application in different development projects, it is certified by Gartner, and it provides complete and automatic software code generation from a model-driven perspective. The core of the Integranova conceptual model is a UML-like class model definition. Thus, the results obtained can be easily adapted to other object-oriented MDD approaches.

To achieve the objective of this work, the following activities are performed:

(i) Definition: The definition and application of the VeMI approach is driven by a systematic process. The VeMI process is centered on the definition of a metamodel that includes specific rules to verify the transformation of the input *i** models into models used by the Integranova technology; *i.e.*, the VeMI approach indicates those issues present in the input goal-oriented model that need to be fixed to assure the automatic generation of the models for the target MDD solution.

(ii) Evaluation: The VeMI approach is empirically validated through a laboratory experiment, which demonstrates that its application provides support to achieve the completeness of the Integranova model generated from the input *i** model. The execution of this experiment is also used to show how the VeMI approach is applied to improve *i** models in a specific MDD context.

The rest of the paper is organized as follows. Section 2 discusses the background and the related work. Section 3 presents the problem statement and further motivation. Section 4 covers the VeMI Approach and details the process for its application. Section 5 presents the evaluation of the VeMI Approach in a concrete development scenario. Section 6 discusses an overall analysis of the proposal. Finally, Section 7 presents our conclusions and further work.

2 Background and Related Work

In this section, the *i** framework and the Integranova MDD approach that are used to explain and evaluate our proposal are briefly introduced. Afterwards, a discussion about the work related to the VeMI approach is presented.

2.1 *The i* Goal-Oriented Requirements Framework*

Goal-oriented requirement approaches are oriented to obtain the "what" of the intended systems through the analysis of organizational scenarios [18] [29]. Among several existing goal-oriented approaches, *i** [30] is one of the most widespread modeling and reasoning frameworks. It emphasizes the analysis of strategic relationships among organizational actors, capturing the intentional requirements. The term *actor* is used to generically refer to any unit for which intentional dependencies can be ascribed. Actors are intentional in the sense that they do not simply carry out activities and produce entities; they also have desires and needs. The *i** framework offers two types of models: the *strategic dependency* (SD) model and the *strategic rationale* (SR) model.

The SD model is focused on external relationships among actors called dependencies. Actors can be related by *is-a* and *is-part-of* links representing specialization and aggregation, respectively. A *dependency* is a relationship between two actors: one of them, called the *depender*, depends on a second actor, called the *dependee*, for the accomplishment of some internal intention. The dependency is characterized by an intentional element (*dependum*), which represents the dependency's element. The main intentional elements include *resource*, *task*, *goal*, and *softgoal*. A softgoal represents a goal that can be partially satisfied or a goal that requires additional agreement regarding how it is satisfied.

The SR model provides the internal decomposition of SD actors' intentions. The separation between the external and internal actor's worlds is represented by the actor's *boundary*. Inside this boundary, the rationality of each actor is represented using the same types of intentional elements described above. Additionally, these intentional elements can be interrelated by using one of the following relationships: *means-end* (*e.g.*, a task can be a means to achieve a goal), *contributions* (*e.g.*, some resource could contribute to reach a quality concern or softgoal), and *decompositions* (*e.g.*, a task can be divided into subtasks).

We have considered the *i** SR model to perform the integration of *i** models into MDD approaches, since it offers a detailed representation of the scenario analyzed, which provides extra information that is relevant to generating appropriate inputs for MDD processes.

2.2 *The Integranova MDD Approach*

Integranova is an industrial MDD technology that supports automatic code generation from the conceptual representation of software systems (see Figure 1). The modeling representation used by Integranova is defined from the OO-Method modeling approach [31], which captures the static and dynamic properties of the system in a *class model*, a *dynamic model*, and a *functional model*. This conceptual model also allows the specification of the user interfaces in an abstract way through a *presentation model*.

The *class model* of the OO-Method modeling approach is similar to the class model defined by UML [32]. The main conceptual construct is a class that represents the objects of the solution. Each class has attributes and services related to the management of its instances (creation, update, delete, etc.) and its relationships with other classes. Each service can have preconditions for its execution. Moreover, each class can have invariants that every instantiated object must fulfill.

The *dynamic model* allows the specification of the valid states of the objects of a class; *i.e.*, it represents all the possible states that an object can reach, the valid state transitions, and the execution restrictions that these transitions have. Each transition between states must be controlled by one or more services of the class involved (defined in the class model).

The *functional model* specifies the behavior related to the change of values for class attributes, which is always performed through a class service execution. To do this, the services of a class must be previously specified, indicating the inbound and outbound arguments of each service.

The *presentation model* allows the specification of the interaction units and the presentation patterns used to define the graphical user interfaces, such as instance interaction units, population interaction units, service interaction units, master-detail interaction units, display patterns, filters, order patterns, etc. An instance interaction unit corresponds to the visualization of each instance of a class such that its definition depends of the class model specification. The same occurs with the other interaction units.

From the previous definition, it is possible to observe that the class model is the core of the Integranova conceptual model. The other models are defined (or derived) from this central model. For this reason, the class model has been considered to evaluate the approach proposed in this paper. More details about the Integranova technology and its industrial application can be found in [20].

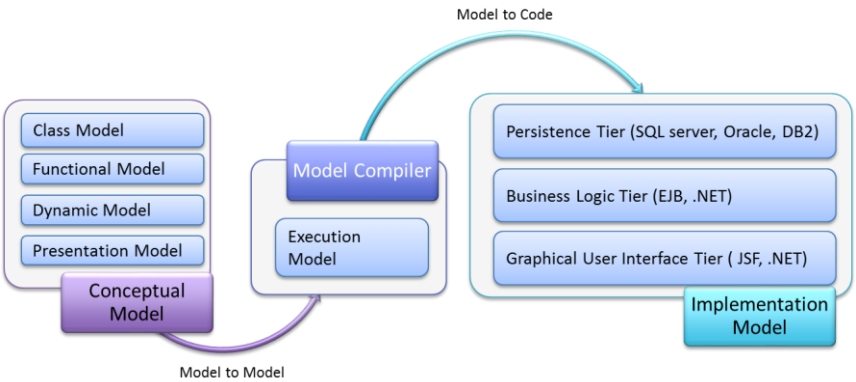

Figure 1. The Integranova software production process

2.3 *Related Work*

One of the main issues in regard to linking goal-oriented modeling and MDD processes is the proper definition of the requirement models for the generation of system models related to concrete MDD approaches (such as modeling tools or model compilers) [33]. Most of the proposals are oriented to translate requirement models into modeling specification for MDD approaches [34] [35] [36]. Laguna and González-Baixauli [34] propose a transformation process based on metamodels for manually transforming goal-oriented models (*i** models) into feature models, and later, from feature models

into UML class diagrams. Lapouchnian *et al.* [35] propose the use of goal-oriented models to generate feature models and statecharts in the context of autonomic application software. Li *et al.* [36] propose automatic transformation from goal models to business process models.

To assure the automatic requirement transformation, certain proposals suggest the manual translation of the defined requirement documents to a specific computable format [37] [9]. These approaches restrict the flexibility of the original specification, which together with the manual translation of the requirements may cause loss of information. Letier and van Lamsweerde [37] define a process based on formal derivation rules for mapping goal models to software operation specifications. Lu *et al.* [9] present the MOR Editor, which supports requirement document modeling and model-driven document editing.

Other approaches suggest adding quantitative information to existing requirement modeling approaches [38] [39] [40], which allows the automatic measure and analysis of the defined models without restricting their original specification. Amyot *et al.* [38] add numeric weights to the intentional elements links of the goal-oriented requirements language (GRL) to support the evaluation of actors and intentional elements satisfaction. Giorgini *et al.* [39] propose a goal-oriented technique for requirements analysis for data warehouse design. This proposal includes numerical measures attached to goals. Pardillo *et al.* [40] propose a measurable requirements metamodel that connects goals, requirements, and measures.

Despite the presence of several proposals using goal-oriented modeling as starting point in the MDD process, there is a lack of approaches to support the verification of requirement models related to MDD processes [33]. To fill this gap, we have defined the VeMI approach, which considers the principles related to object-oriented models verification [41] [42] and the definition of measures to verify the correct generation and compilation of domain-specific models [43] [44].

The VeMI approach uses transformation rules as the starting point of the verification process and the measures definition to verify the effectiveness of the transformation execution. Thus, the software models will be complete in relation to the input requirement specification and properly defined for the final software code generation process.

The implementation of the VeMI approach is based on current modeling standards, such as MOF [45], UML [32], and XMI [46]. It has been developed by considering our previous experience related to linking requirements and MDD processes [47] [21], the definition of modeling measures and model verification mechanisms [48] [49], and the industrial application of MDD approaches [48].

3 Problem Statement and Motivation

The success of computer applications increasingly depends on a good understanding of the system requirements. A proper requirement specification must describe the context in which the intended system will operate. During the early stages of the requirement engineering process, it is necessary to identify and specify how the intended system meets the organizational goals, why the system is needed, what alternatives were considered, what the implications of the alternatives are for the stakeholders, and how the interests and concerns of the stakeholders might be addressed.

Hence, goal-oriented requirements engineering stood out, because it is mainly concerned with the stakeholders' intentions and their rationales. However, how to go from requirement models to the corresponding software products is still an open question.

Current model transformation technologies (such as ATL or QVT) propose the specification of model transformations driven by metamodels. Thus, the use of the *i** approach is a suitable alternative, because it has a well-defined syntax, and it is possible to find metamodel specifications, which can be used as a reference for the definition of modeling transformations.

Different works focused on adopting and adapting goal-oriented models to model-driven development processes were analyzed in the related work section. From this analysis, three key open issues were identified, as follows:

1) The identification of those analysis elements that will be considered for implementing the final software product.
2) The need for including additional information in the analysis models in order to align the specification to a particular development technology.

3) The verification of the goal-oriented models to ensure the proper generation of the system model; *i.e.*, to transform the requirements model to the corresponding design model.

Figure 2 presents an excerpt of the *i** model to better explain how these issues impact the use of goal-oriented models in model-driven development processes. This *i** model represents the interaction between a customer and a seller, which considers the emission of a purchase order from the customer side and the delivery of the product and invoice once the purchase is carried out from the seller side.

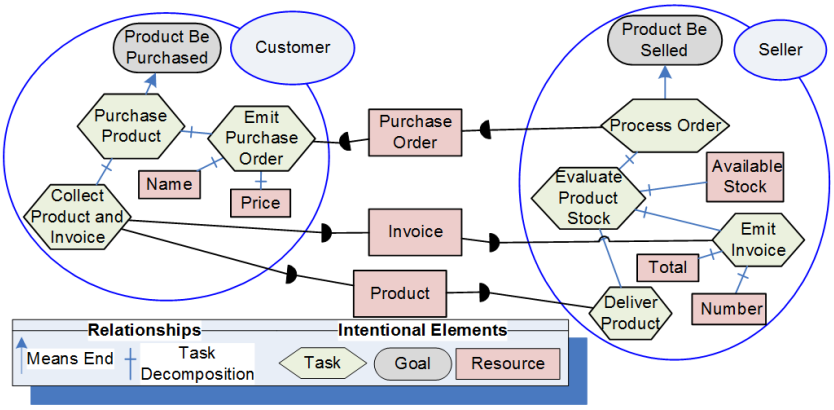

Figure 2. Example *i** model for customer and seller interaction

Analysis models can comprise elements that will be supported by information systems as well as elements that will be manually supported. Thus, it is important to properly differentiate these elements. In the example, depending on the automation decisions made, it would be possible to consider a system for invoice emission, a system for processing customer orders, a system for managing the stock of products, or a system that comprises all these elements.

However, in the *i** specification, it is not possible to indicate the elements that will be involved in a concrete software product development. This is the first problem to be solved: which solution can be considered as an extension of the *i** specification? Going on with the example, Figure 3 shows the *i** elements related to invoice emission and product delivery (*i.e.*, invoice, product, evaluate product stock, emit invoice, deliver product, available stock, total, and number). It is important to note that the selection of the *i** elements may vary depending of the analyst criteria and the functionalities that will be supported by the final system.

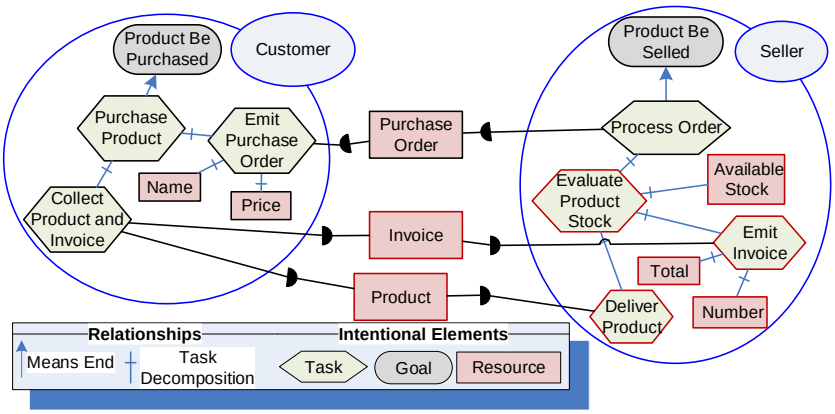

Figure 3. *i** model highlighted for invoice emission and product delivery

The second problem comes after the selection of the *i** model elements that will be automated: how to align the elements selected to a concrete software development technology?

In the particular context of model-driven development, this corresponds to inferring modeling elements of the target system model from the source *i** elements. In this paper, the interpretation is driven by concrete transformation guidelines to go from *i** models to Integranova class models. For the proper application of these guidelines, it is necessary to consider additional information to

perform the proper alignment of *i** to the target modeling approach. To exemplify the transformation problem, we will consider the following transformation guidelines (extracted from [23]).

G1: An *i** resource that represents a physical entity is transformed into a class of the target class model. The name of the generated class is obtained from the name of the resource.

G2: An *i** resource that represents an informational entity (informational resource) related to a resource (physical resource) of the *i** model is transformed into an attribute of the class generated from the physical resource involved. The name of the attribute is obtained from the name of the informational resource.

From these guidelines, it can be observed that a resource has a double interpretation: 1) a resource can be considered as a physical entity, which is an entity that has behavior and a specific data structure, and 2) a resource can be considered as an informational entity, which corresponds to information (data) that must be related to a physical entity. Figure 4 shows some class model generation alternatives that can be obtained from the elements related to invoice emission and product delivery of the example *i** model (Figure 3).

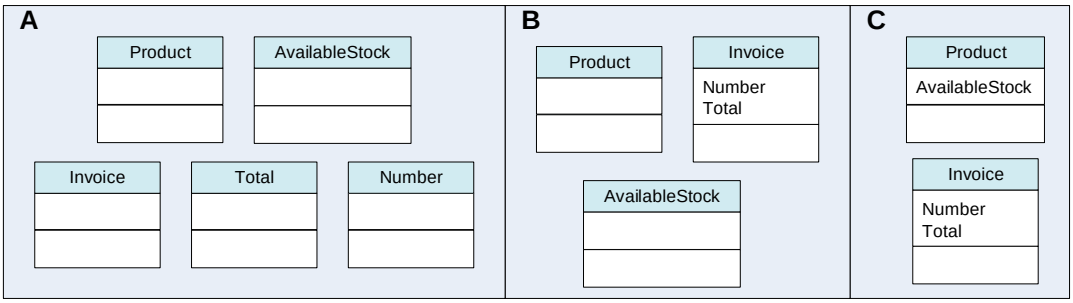

Figure 4. Class models generated from the elements related to the invoice emission and product delivery of the example *i** model

Figure 4-A shows that all the *i** resources were considered as physical resources and that they are transformed into classes according to the first transformation guideline (G1). Figure 4-B shows that the resources *Number* and *Total* were considered as informational resources related to *Invoice*, and thus transformed into attributes of the class *Invoice* (guideline G2). Finally, Figure 4-C shows that the resource *AvailableStock* was also considered as an informational resource related to the resource *Product*; therefore it generates the corresponding attribute of the class *Product* in the class model.

From Figure 4, it can be observed that depending on the characterization of the *i** resources, different class models are obtained. For this example, Figure 4-C will be considered as the correct alternative.

However, it is not possible to represent when a resource is informational or physical in an *i** model nor when an informational resource is related to a physical resource. Hence, the class model generation cannot be automated and needs to be manually guided for each resource involved. This is clearly a highly time-consuming and error-prone task, especially when the size of the *i** model involved becomes much larger.

For solving this second problem, it is necessary to use some extension mechanism for including into the *i** specification the additional information required to automate the application of the transformation guidelines. Thus, the extension mechanism will permit the automatic generation of class models from the input *i** models.

Finally, the third problem arises once the second problem is solved. The extensions defined over the *i** specification for alignment with MDD processes are potential failure points. These extensions need to be properly defined to perform a complete generation of the target class model. Otherwise, the resultant class model will be incomplete in relation to the input requirements. The verification of these extensions is an exhaustive task that can take long hours in large analysis models. Moreover, it demands that the analyst perfectly knows the conceptual formulation of each transformation guideline to detect modeling defects that may be present in the extended *i** model. These modeling defects cannot be automatically identified by existing *i** editors, since the modeling properties involved are not part of the original *i** specification.

To exemplify the third problem, Table 1 shows the information that extends the example *i** model for executing the transformation guidelines for *i** resources.

Table 1. Additional information for execution of example transformation guidelines

*i** Resource	Informational/Physical	Related to
Product	Physical	-
Invoice	Physical	-
Total	Informational	Invoice
Number	Informational	Total
AvailableStock	Informational	-

The information presented in Table 1 is not properly defined; it presents some problems, which are related to the resources *Number* and *AvailableStock*. Without knowing the formulation of the transformation guidelines involved, it is really difficult to know which and why some *i** elements are wrongly defined. When the transformation of the *i** model is executed, the resources *Number* and *AvailableStock* will not be considered in the generation of the target class model, and, hence, the resultant system model will be incomplete in relation to the original requirements specification. The correct values that must be defined in Table 1 for these elements are the following: a) the resource *Number* must be related to *Invoice*, and b) the resource *AvailableStock* must be related to *Product*.

To identify these issues, it is necessary to define some verification mechanisms to guarantee that the *i** model is perfectly aligned with the transformation process required to generate the target system model. For the definition of this verification mechanism, the reference transformation guidelines must be considered. This information can be used not only to identify the *i** elements that present some issues but also to provide additional information for solving the issues identified.

Coping with these three problems has leads to the VeMI approach presented in this paper, which is detailed in the next section.

4 Verification Approach for Aligning Goal-oriented Modeling and MDD Processes

VeMI is based on a model-driven interoperability approach specifically developed for MDD processes. We have presented this approach as a reference MDD interoperability model in [50]. This interoperability model has been applied to obtain an interoperability framework for goal-oriented modeling and MDD approaches. Details can be found in [51]. Figure 5 characterizes the components of the reference MDD interoperability model in three levels: the specification level, the implementation level, and the application level. In this figure, the elements defined in the original specification of the MDD interoperability model are at the left side of the figure. The new elements, which are included as part of the VeMI approach, are at the right side of the figure.

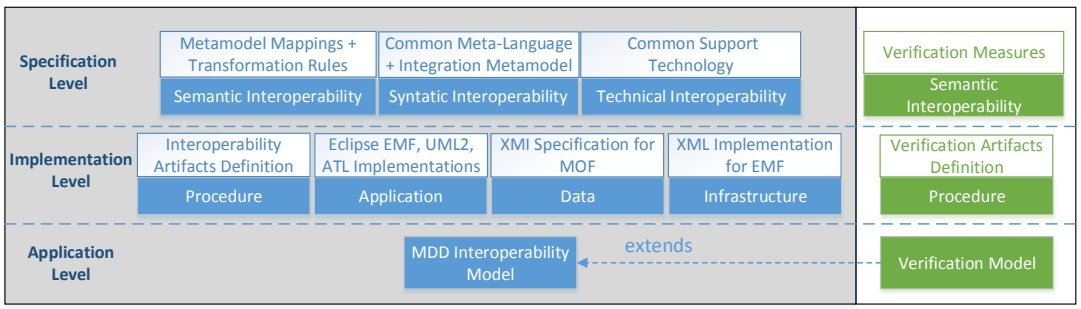

Figure 5. MDD interoperability model extended with the VeMI approach

The specification level states a model-driven interoperability solution in terms of three dimensions: semantic, syntactic, and technical interoperability. *Semantic interoperability* is achieved through mappings between the metamodels of the involved modeling languages and the rules for transforming the input goal-oriented model into a system model related to the target MDD solution. *Syntactic interoperability* (abstract syntax) is obtained by using a common metamodeling language for all the modeling approaches involved and by means of the definition of a specific bridge metamodel, which is called the integration metamodel (presented in [52]). The integration metamodel is used to solve the structural differences that may prevent the appropriate interchange of information and to

automatically identify the modeling extensions that are necessary to perform interoperability operations. *Technical interoperability* is achieved by using a common technology for supporting the implementation of modeling management tools (*i.e.*, model editors, model transformations, etc.).

At the specification level, the VeMI approach indicates the need for counting with concrete artifacts to assure the correct execution of interoperability operations. This involves the specification of measures to evaluate the input goal-oriented models to prevent the loss of information when the model transformation is performed. For this purpose, the concept of *verification measure* is defined, which has been introduced and explained in [44]. The need for counting with verification mechanisms for the execution of interoperability operations is founded on the systematic review presented in [53], where the lack of approaches for performing this kind of verification is clearly indicated.

The implementation level shows that it is possible to automate model-driven interoperability operations when the following four perspectives are supported: *procedure*, *application*, *data representation* and *infrastructure*. The *procedure* perspective specifies the artifacts that need to be defined to support the interoperability operations and the correct manner in which to perform this definition. The *application* perspective refers to the concrete technologies that support the definition and management of the artifacts involved in the interoperability operations. In particular, we have considered the specific facilities provided by the Eclipse Model Development Tool (MDT) [54]: EMF, UML2, and ATL. The *data* perspective corresponds to the reference standard for interchanging information among the applications involved. For the implementation of the VeMI approach, the XMI interchange specification defined according to the MOF standard is considered [45]. The *infrastructure* perspective specifies the implementation technology for supporting the data interchange. In particular, we have considered the XML format implemented for Eclipse EMF [55].

At the implementation level, the VeMI approach indicates the procedure for defining and implementing a concrete verification model. This verification model is focused on guaranteeing the completeness of the interoperability operations to be performed by means of measures that automatically identify modeling issues. Guidelines to solve the modeling issues identified are also obtained from the verification model definition.

At the application level, the elements defined at the implementation level are used for a specific interoperability scenario, thus generating an interoperability model that is extended with the verification measures implemented in a verification metamodel.

The resultant verification metamodel is finally used to generate modeling extensions that will be introduced in the goal-oriented specification. In the context of this paper, this corresponds to extending the *i** specification with the modeling information to automate the Integranova class model generation as well as to implement the verification mechanisms to guarantee the completeness of the model transformation process.

Therefore, the procedure for applying the VeMI approach to the *i** framework for the generation of an Integranova class model is comprised by the following steps (see Figure 6):

1. Definition of Transformation Rules. In this step, it is important to identify the *i** constructs and Integranova constructs involved with the extra information needed for performing the transformation process. The *i** model translation can be automated by means of model-to-model transformation rules by using technologies such as ATL [56] or QVT [57].
2. Definition of the Integration Metamodel. The integration metamodel is a pivot metamodel for the representations of mappings, new information, and the management of modeling heterogeneities. This pivot metamodel is defined according to the approach presented in [58].
3. Definition of Verification Measures. These measures are defined by taking as a reference the approach presented in [44], and they are implemented into the integration metamodel by means of OCL rules. As a result, the *verification metamodel* is obtained.
4. Definition of Fixing Guidelines. The fixing guidelines are alternatives to solve the issues identified from the evaluation of the verification measures. These guidelines are defined by considering the extra information that is integrated into the *i** framework to generate the Integranova class model, and the structure of the verification measures is specified.
5. Integration of the Verification Metamodel into the *i** Metamodel. This integration is performed by means of light-weight extensions, which are implemented through a UML profile specification by adapting to the approach presented in [59].

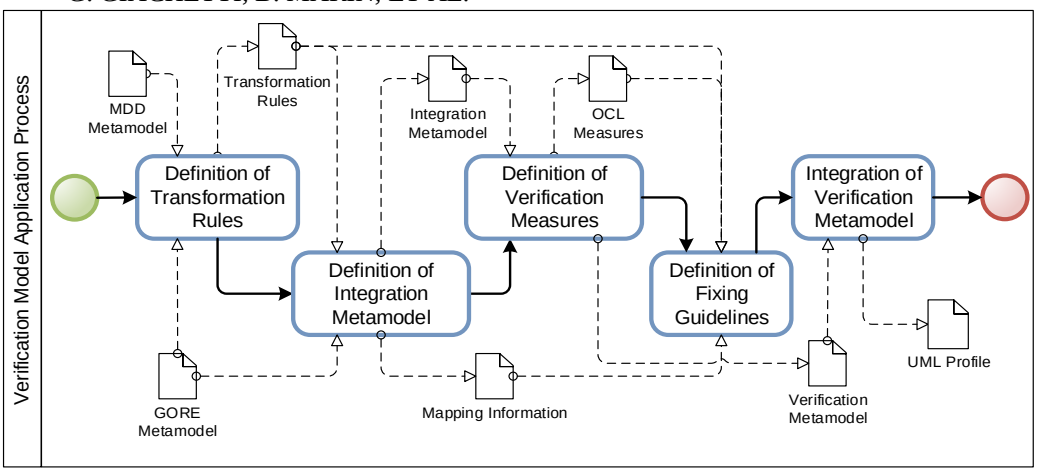

Figure 6. Systematic process for applying the verification approach

4.1. Step 1: Definition of Transformation Rules

The first step corresponds to the definition of the guidelines (or rules) related to the transformation of the input *i** models into the Integranova class model. From these rules, it is necessary to identify the *i** elements involved and the additional information that must be specified by the analyst to perform the transformation.

Table 2 summarizes a representative set of transformation guidelines for *i** and the Integranova technology, which have been selected due to their applicability to other MDD approaches based on the class model specification. These guidelines are used to evaluate the verification approach proposed. The rationale for these guidelines has been presented in [60]. Table 2 shows the *i** constructs that are involved in the transformation, the additional information that is required to perform the transformation, and the target constructs of the class model.

Table 2. Guidelines for the transformation of *i** models into Integranova class models

i* **Construct**	**Additional Information**	**Class Model Construct**
Actor	Marked for Class Model Generation	Class
Resource Dependency Link		Associations are automatically defined among the classes generated from the *dependum* resource and the classes that own the services generated from the involved tasks
Is-a Link		A generalization relationship is generated between the classes generated from the involved actors
Resource	Physical entity	Class
	Informational resource related to a physical resource or an actor	An attribute of the class generated from the actor or physical resource
	Informational resource inside of an actor boundary	An *agent relationship* between the classes generated from the actor and the attribute generated from the resource
Task	If generates an entity (physical resource or actor)	An instance creation service of the class generated from the corresponding entity
	If affects the state of a resource	A service of the class generated from the resource or from the owner physical resource
	If does not affect resources or generate entities	A service of the actor that contains the task
	If is decomposed in resources	Associations are automatically defined among the classes that contain the corresponding service and the classes generated from the decomposed resources
	Inside of an actor boundary	An *agent relationship* between the classes generated from the owner actor and the task

In the transformation guidelines presented (Table 2), it is possible to observe a specific Integranova construct, the *agent relationship*. This construct corresponds to a binary relationship that indicates the visibility and execution permissions that a class of the model has over other classes or over itself (recursive agent relationship). The classes that have an agent relationship to other classes are referred to as the *agents* of the system modeled. This semantics can be generalized to other MDD approaches that define system users and interaction aspects at the conceptual level.

4.2. *Step 2: Definition of Integration Metamodel*

To perform the definition of the integration metamodel we have put into practice the approach to integrate domain-specific information into the *i** metamodel presented in [61]. Figure 7 shows the reference *i** metamodel used in this work. This figure only shows the structural representation of the metamodel. For the elaboration of this *i** metamodel, the proposals presented in [62], [63], and [64] were considered.

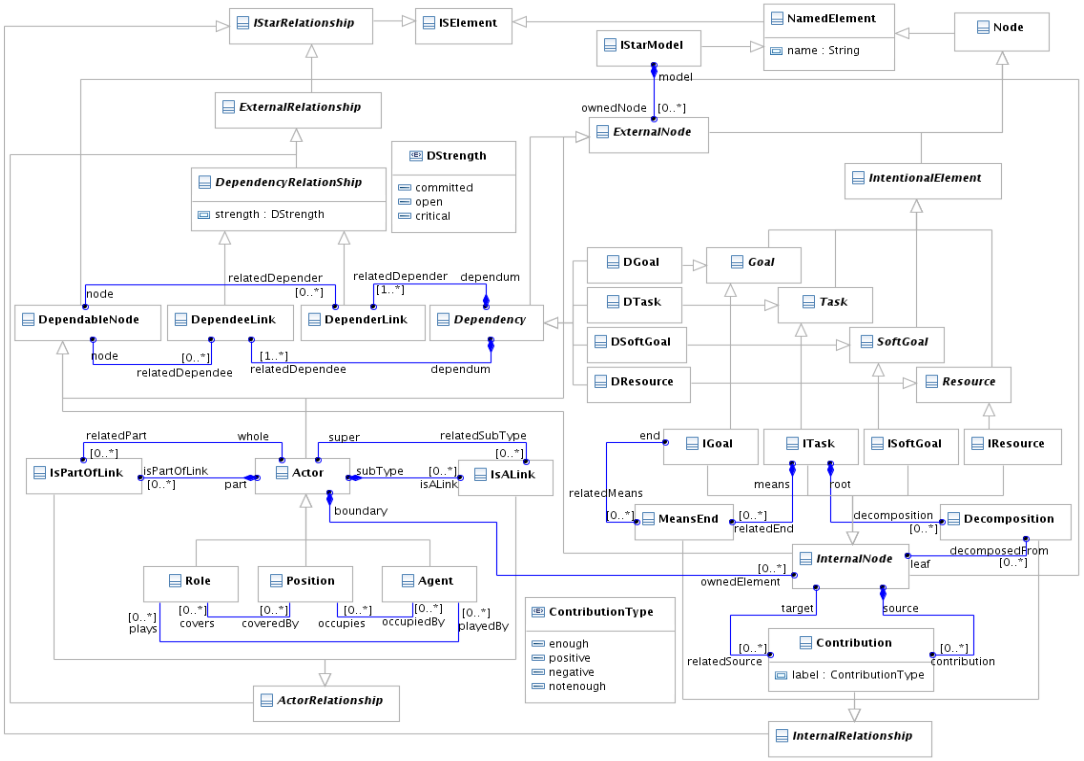

Figure 7. EMOF *i** metamodel

The integration metamodel is specified from the *i** metamodel definition and the transformation guidelines. The integration metamodel is an EMOF metamodel that includes the *i** constructs involved in the transformation process and the additional modeling information that is necessary to automate the generation of the Integranova class model.

Figure 8 shows the verification metamodel that corresponds to the integration metamodel plus the verification measures (in terms of OCL rules), which are defined according to the next step of the process (Step 3). The mapping between the constructs of the integration metamodel defined and the target *i** metamodel is also presented in this figure. All the metaclasses from the integration metamodel must be mapped to an *i** metaclass to guarantee the equivalency between both metamodels.

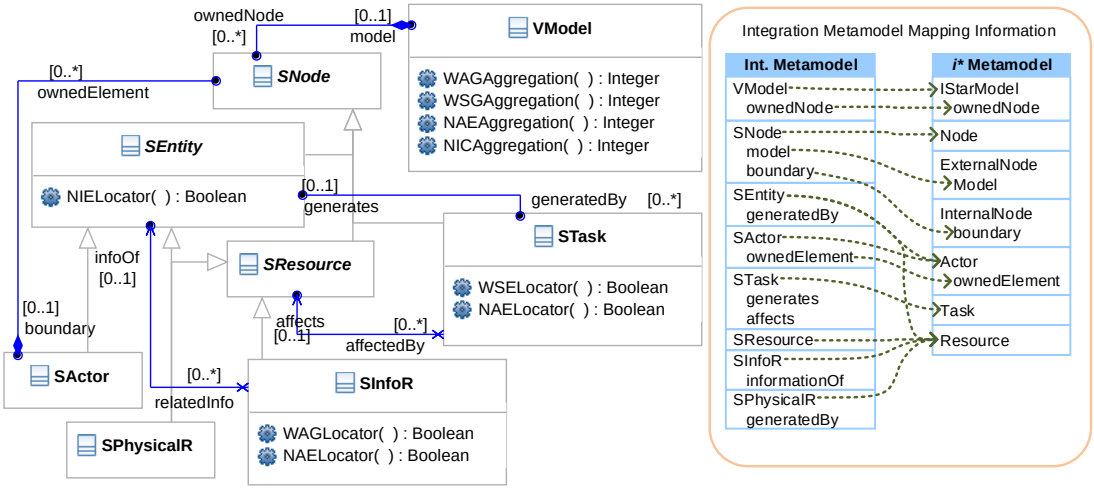

Figure 8. Verification metamodel and mapping information

From Figure 8, it is possible to observe that only the *i** elements involved in the transformation rules are mapped into the integration metamodel. Also, the additional information required by the transformations is represented in this new metamodel – for instance, the sub-classification of a Resource in Physical Entity and Informational Resource or the relation *generates* (between *STask* and *SEntity*) that indicates when an entity is produced by a specific task.

From the resultant integration metamodel, it is possible to implement complete M2M transformation rules to automatically transform input *i** constructs into the corresponding class model constructs. Moreover, from the integration metamodel it is also possible to implement automatic verification mechanisms to guarantee the adequacy of the *i** models used as input for the class model generation.

4.3. *Step 3: Definition of Verification Measures*

For the definition of the verification measures, it is important to consider the additional information, not present in the original *i** specification, that is required for the *i** model transformation. This modeling information is the critical point that must be verified to assure that the transformation can be performed correctly and automatically.

For the formulation of the verification measures, two severity levels are considered according to the information reported:

1. **Critical Verification Measures:** These measures report those goal-oriented elements that must be fixed, because they cannot be transformed, or they produce an incorrect MDD model.
2. **Warning Verification Measures:** These measures report input elements that can be transformed, but they can be improved or refined to obtain a better MDD model generation.

A brief description of the measures proposed is presented below. Additional information about these measures can be found in [44]. The customer and seller example presented in Section 2 will be used to facilitate the comprehension of these measures.

M1. Wrong Attribute Generation (WAG) – Critical Measure. This measure identifies those informational resources that are not related to any actor or physical resource. These resources cannot be transformed, since they generate attributes without a class that contains them. The formula to obtain the measure M1 – WAG is the following:

$$WAG_M = \sum_{\substack{r \in resources(M) \wedge \\ kind(r)=Informational}} conv\left(\neg relatedToActor(r) \wedge \neg relatedToPhysResource(r)\right) \quad Conv(x)\begin{cases}1, \text{if } x = true\\ 0, \text{if } x = false\end{cases} \tag{1}$$

An *i** resource can represent both a physical entity with a specific behavior and structure or an informational entity that corresponds to the data of a physical entity or an actor. According to the transformation guidelines, the physical resources and actors are transformed into classes, and the informational resources are transformed into class attributes. For this reason, it is necessary that the

informational resources involved in the class model generation be related to a physical resource or an actor of the *i** model.

Figure 9 shows the application of the WAG measure over the selected resources of the customer and seller example. In this *i** model, the resources *AvailableStock*, *Total*, and *Number* are indicated as informational entities without a physical entity or actor related. The WAG measure reports this situation. In this figure, the *Original Model* shows the class model generated from the *i** model without any improvements. The *Improved Model* shows the class model generated after fixing the *i** model according to the details provided by the WAG measure evaluation.

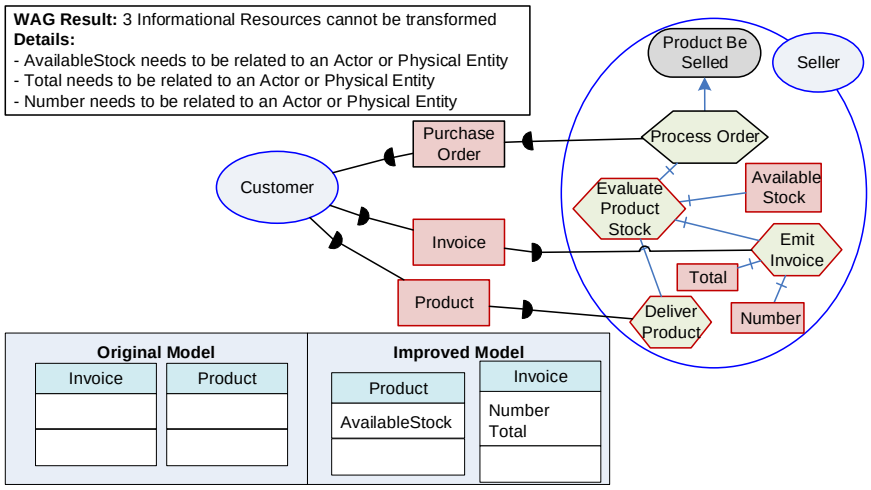

Figure 9. Application of the WAG measure over the customer and seller example

M2. Wrong Service Generation (WSG) – Critical Measure. This measure identifies those tasks that do not generate entities nor affect resources. These tasks cannot be transformed, since they generate services without a class that contains them. The formula to obtain the measure M2 – WSG is the following:

$$WSG_M = \sum_{t \in tasks(M)} conv(\neg generatesResource(t) \wedge \neg affectsResource(t) \wedge \neg hasSystemActor(t)) \quad (2)$$

Figure 10 exemplifies the effect of the measure WSG. In this example, the task *EvaluateProductStock*, *EmitInvoice*, and *DeliverProduct* were defined without indicating the related resource, and, for this reason, the original model obtained lacks class services. The WSG measure reports this situation. The *Improved Model* shows the class model generated after fixing the issues detected. The tasks *EvaluateProductStock* and *DeliverProduct* are defined as affecting the resource *Product*; and the task *EmitInvoice* is defined as the generator of the resource *Invoice*.

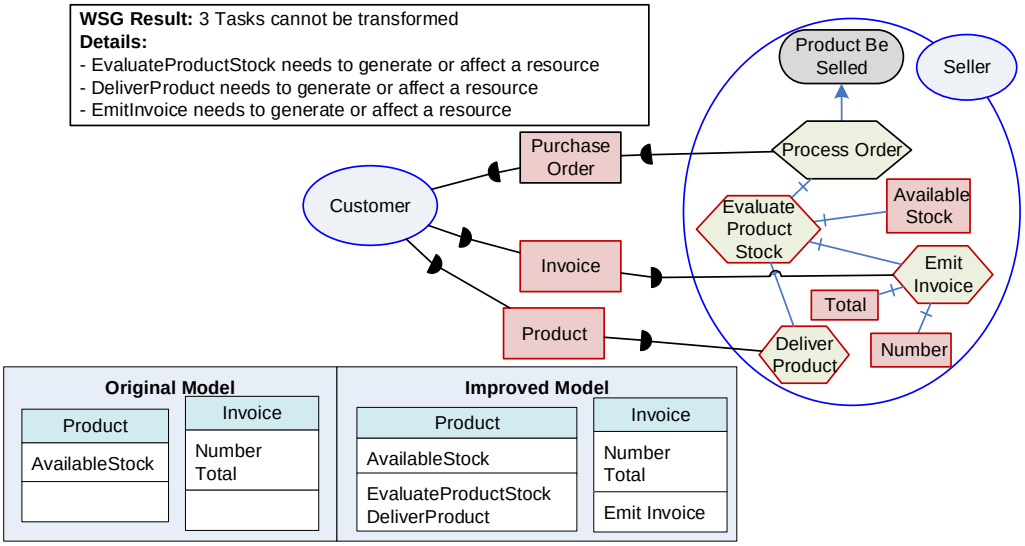

Figure 10. Application of the WSG measure over the customer and seller example

M3. Non-Accessible Element (NAE) – Warning Measure. This measure identifies those actors that are not selected for the MDD model generation. The tasks or informational resources that are inside of these actors' boundaries generate services or attributes that cannot be executed or visualized in the class model specification; *i.e.*, they do not have a user (agent in the Integranova domain) related. However, it is not mandatory to define an actor as part of the intended system. For instance, the analyst could consider that the involved actor must not be maintained in the final system. In this case, a new agent (special user) must be defined at design time during the refinement of the generated class model to execute and visualize the generated elements, such as an administrator user. The formula to obtain the measure M3 – NAE is the following:

$$NAE_M = \sum_{t \in tasks(M)} conv(\neg hasSystemActor(t)) + \sum_{\substack{r \in resources(M) \wedge \\ kind(r)=Informational}} conv(\neg hasSystemActor(r)) \quad (3)$$

Figure 11 exemplifies the effect of the measure NAE. This figure shows that the actor *Seller* is identified by the NAE measure, because some of its internal elements are marked (highlighted) for the generation of the class model (*i.e.* Evaluate Product Stock, Available Stock, Emit Invoice, Total, Number, and Deliver Product).

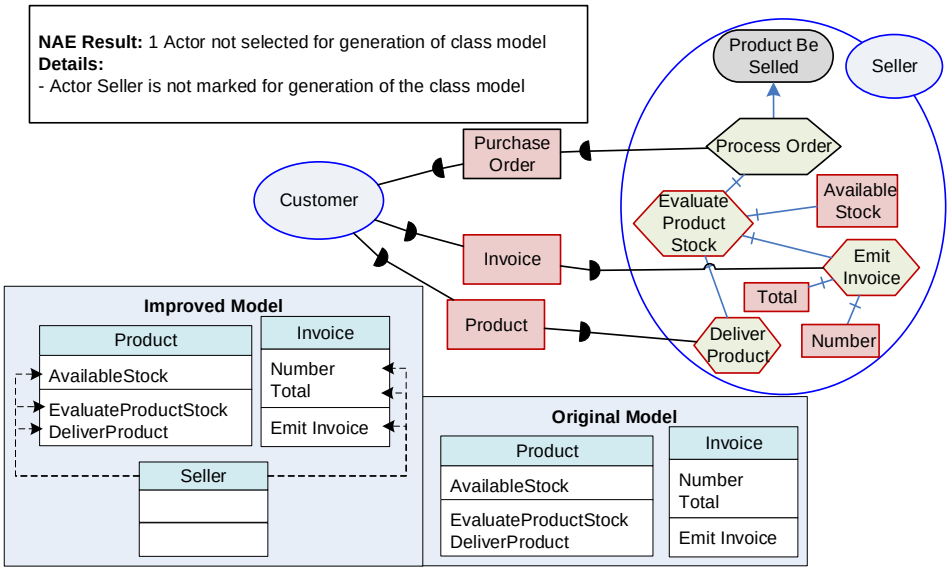

Figure 11. Application of the NAE measure over the customer and seller example

The NAE measure is a warning measure; therefore, the results obtained do not prevent the proper class model generation. However, the information provided by this measure can be useful for improving the *i** model to produce a more detailed class model generation. In this example, the recommendation provided by the measure is followed, and the actor *Seller* is marked for the class model generation. As result, the improved model is obtained. In this model, a new class, *Seller*, is generated, and this class has accessibility over the attributes and services of the classes *Product* and *Invoice*.

M4. Non-Instantiable Class (NIC) – Warning Measure. This measure identifies the system entities (physical resources or actors) without a production task related. These entities generate classes without an instance-creation service (see Table 1). The service that produces new instances of a class takes special relevance, since without this service, the class is not properly defined (all the defined classes must be capable of generating their instances). However, the definition of a production task for entities (actors or physical resources) is not mandatory, since specific instance-creation services can be defined at design time for the classes generated. The formula to calculate the measure M4 – NIC is the following:

$$NIC_M = \sum_{\substack{r \in resouces(M) \wedge \\ kind(r)=Physical}} conv\ (\neg hasProductionTask(r)) + \sum_{a \in actors(M)} conv(\neg hasProductionTask(a)) \quad (4)$$

Figure 12 exemplifies the effect of the warning measure NIC over the reference example. This measure shows that in the original model there are two entities without a production task related. For the example, only the issue of the resource *Product* has been solved. Thus, the improved model shows that a new task, *CreateProduct*, has been defined as solution for this issue. In the improved model, the NIC measure still indicates the issue related to the actor *Seller*. The creation service for the generated class *Seller* can be defined later, over the generated class model, at design time.

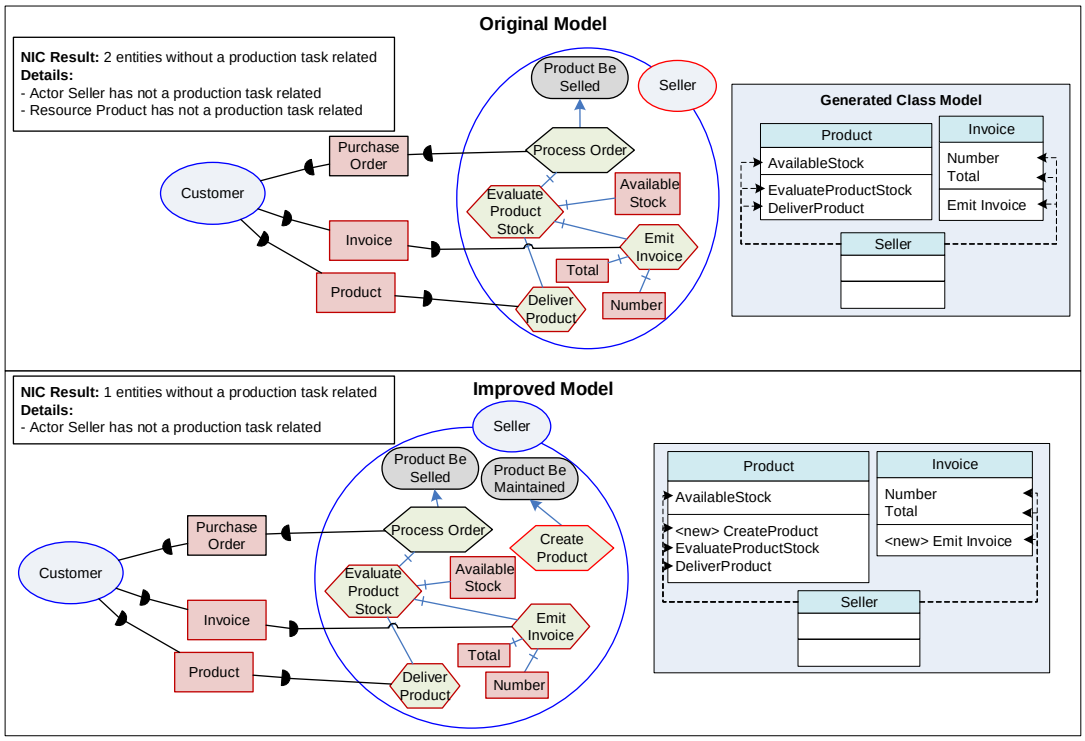

Figure 12. Application of the NIC measure over the customer and seller example

It is interesting to observe the effect that the application of the verification measures and fixing guidelines may have over the *i** model for the generation of the Integranova class model. In this running example, the final *i** and class models obtained (improved model in Figure 12) are clearly more detailed than the models initially presented in Figure 9.

Table 3. WAG measure specification in the OCL language

Measure	Subject of Measure	Alert Level
M2: Wrong Attributes Generation (WAG)	*i** Informational Resources	Critical

```
Context: VModel::WAGAggregation() : Integer

Body: result = self.ownedNode->select(irs|irs.oclIsKindOf(SInfoR)).oclAsType(SInfoR)
   ->select(irs|irs.WAGLocator())->size() + self.ownedNode
   ->select(act|act.oclIsKindOf(SActor)).oclAsType(SActor).ownedElement
   ->select(irs|rs.oclIsKindOf(SInfoR)).oclAsType(SInfoR)->select(irs|irs.WAGLocator())->size()

Context: SInfoR::WAGLocator() : Boolean

Body: result = self.infoOf->isEmpty()
```

Specific OCL rules are defined for implementing the verification measures proposed. For this OCL specification, the measure patterns presented in [49] are applied – specifically, the *aggregation* and *locator* patterns. The locator pattern is used to identify the elements involved in the measure evaluation, and the aggregation pattern is used to return the final value of the measure. A very useful aspect of the application of these patterns is that the *i** elements that must be fixed can be easily identified by means of the locator pattern. Thus, with the process and patterns proposed for the definition of the verification measures, the OCL implementation obtained have a very simple and standardized structure that is comprised by two elements: 1) an OCL rule (*locator*) for the identification of the elements to be measured, and 2) an OCL rule (*aggregation*) for counting the

number of occurrences of the element to be measured. For instance, the OCL definition of the measure WAG (see Table 3) is comprised by the OCL rule *WAGLocator* that identifies the corresponding resources by returning a Boolean value, and the OCL rule *WAGAggregation* that returns the final measure result by aggregating those resources where the OCL rule *WAGLocator* returns true.

4.4. *Step 4: Definition of Fixing Guidelines*

The information obtained from the measures' formulation and evaluation can be used to fix the modeling issues identified. Thus, specific fixing guidelines for each measure can be obtained.

In the case of the WAG measure, the issue that can be identified in the *i** model is the presence of an informational resource not related to a stereotyped actor or physical resource. In this case, there are three possible solutions, which we refer to as fixing guidelines:

1) Associate the informational resources to a system entity (stereotyped actor or physical resource).
2) Change the informational resource to a physical resource.
3) Remove the resource from the intended system (un-stereotyped resource).

In addition to these guidelines, removing the element identified from the *i** model also solves the issue, but we do not consider this to be a guideline for fixing the element identified. It is also important to mention that independently of the guidelines that can be derived from the different verification measures, this information is merely a reference for the analyst, who must decide how to solve the issues identified or how to improve the *i** model specification. Table 4 summarizes the fixing guidelines related to the four verification measures defined.

Table 4. Fixing guidelines related to the verification measures defined

Measure	**Wrong Attribute Generation (WAG)**
Guidelines	Associate the informational resources to a system entity (stereotyped actor or physical resource).
	Change the informational resource to a *physical resource*.
	Remove the resource from the intended system (un-stereotyped resource).
Measure	**Wrong Service Generation (WSG)**
Guidelines	Define the owner actor as part of the intended system.
	Indicate if the involved task participates in the generation or affects the state of a system entity (stereotyped actors or physical resources).
Measure	**Non-Accessible Element (NAE)**
Guidelines	Define the owner actor as part of the intended system.
	Change the informational resource to a *physical resource.*
Measure	**Non-Instantiable Class (NIC)**
Guidelines	Define a new task in the model as a production task of the involved entity (stereotyped resource or physical resource).
	Indicate a task that is already defined in the model as a production task of the entity (stereotyped resource or physical resource).
	Change the physical resource to an *informational resource.*

4.5. *Step 5: Integration of the Verification Metamodel into the Goal-oriented Metamodel*

Finally, in the fifth step of the process, the metamodel extensions that are necessary to integrate the verification metamodel into the *i** framework are generated. These extensions are implemented in a UML profile (see Figure 13), which is generated by means of the proposals presented in [59] and [52]. In [52], an approach to the adaptation of metamodels for the generation of UML profiles is presented, and [59] defines a set of transformation rules for automatic UML profile generation. These proposals use the mapping information presented in Figure 8.

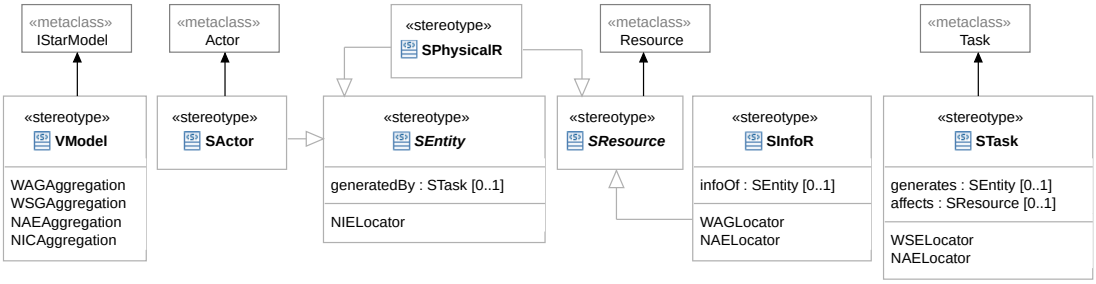

Figure 13. UML profile to integrate the verification metamodel into the *i** framework

The UML profile generation consists of the generation of one stereotype for each class of the verification metamodel and one tagged value for each property (attribute or association end) that has no correspondence in the target *i** metamodel (non-mapped properties). The abstract stereotype *SNode* is not represented, since it does not introduce new properties or operations into the *i** metamodel.

The UML profile is a lightweight extension mechanism that does not change the target metamodel; it has a standardized definition [32] and interchange format [46]. Therefore, it is a suitable alternative for the application of our verification proposal. Other proposals have also considered the use of lightweight extensions for goal-oriented modeling (*e.g.*, [65]). In the generated UML profile, the elements of the OCL specification must be changed according to the mapped elements of the *i** metamodel, the generated stereotypes, and tagged values.

5 Evaluating the Verification Approach

We have conducted an experiment to assess the effectiveness of the VeMI application. The evaluation process consists on the application of VeMI to a simplified version of an industrial case study defined using the Integranova technology. The complete case study description can be found in [66].

The experiment's goal is to analyze the the effectiveness of the VeMI approach referent to the completeness of the design models obtained from the transformation of goal-oriented models. In this respect, the ISO 9126 standard [67] distinguishes between two kinds of completeness: 1) the completeness of a system with respect to the requirement specification and 2) the completeness of the functionality that a system must support. The first kind of completeness corresponds to the completeness of the Integranova class models generated in relation to the system requirements that are defined in the *i** models. The second kind of completeness is related to the completeness of the class model in regard to the functionality of the software system – in other words, the completeness of the generated Integranova model to perform the automatic model compilation and, therefore, the generation of the software code.

The experiment has been designed according to the framework proposed by Wohlin *et al.* [68] for empirical software engineering. The research question addressed by the experiment is stated as follows:

RQ1: Is the completeness of the *i** model transformation for the generation of the Integranova class model improved by the application of the VeMI approach?

The rest of this section provides details about the design of the experiment as well as the results obtained from the experiment's execution.

5.1. *Subjects, Variables, and Hypothesis*

Four subjects were selected to participate in the study: two *i** analysts (identified as ANA1 and ANA2) and two measurement experts (identified as EXP1 and EXP2). These subjects are Computer Science PhD Professors who have similar backgrounds in the *i** framework and the Integranova MDD approach. Additionally, the experts have also worked in industrial MDD projects.

The independent variables in the experiment correspond to the photography agency *i** models, which have been defined by the *i** analysts.

We considered the following quantitative dependent variables:

1) Number of informational resources that cannot generate the corresponding class attributes in the Integranova class model. Obtained from evaluating the WAG measure.
2) Number of tasks that cannot generate the corresponding service definitions in the Integranova class model. Obtained from evaluating the WSG measure.
3) Number of tasks and informational resources that generate non-accessible elements in the Integranova class model. Obtained from evaluating the NAE measure.
4) Number of actors and physical resources that generate non-instantiable classes in the Integranova class model. Obtained from evaluating the NIC measure.

To answer our research question, we consider the following hypotheses related to critical and warning measures:

H_{RCOM}: The critical measures allow the verification of all the system requirements that are defined in the extended *i** model to generate the corresponding class model constructs.

H_{CCOM}: The warning measures allow the verification of those *i** elements that can be improved to generate a more complete specification of the class model, which represents the functionality of the final software product.

To test H_{RCOM}, each *i** element related to the intended system must have a direct relation with the constructs generated in the class model.

To test H_{CCOM}, the improvements performed on the *i** model with the information obtained from the warning verification measures must generate a more detailed specification of the class model.

5.2. *Instruments and Experimental Tasks*

To perform the experiment, two groups, each comprising one analyst and one measurement expert, execute the experimental tasks starting at the same time but in different rooms.

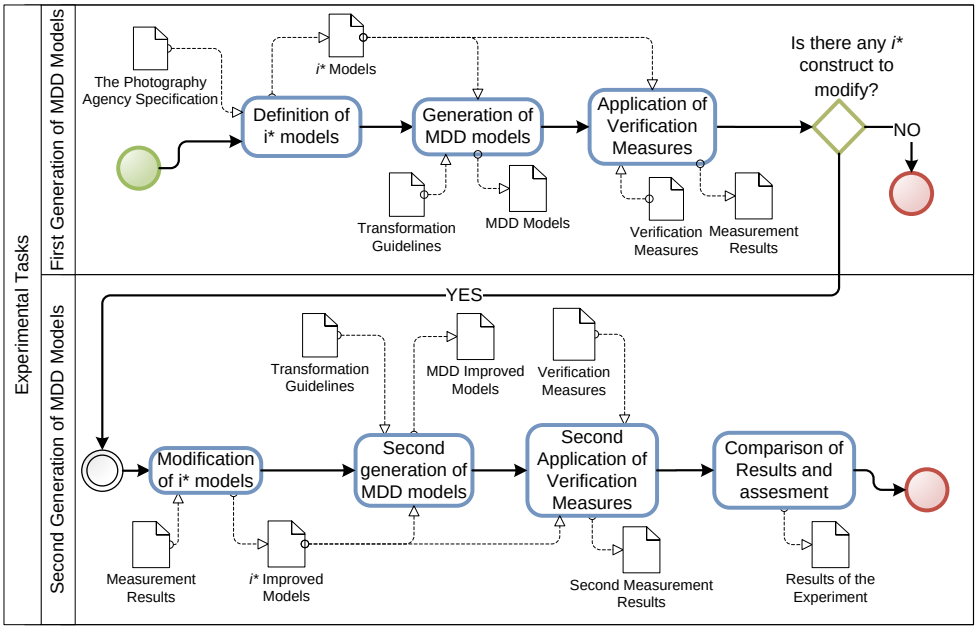

Figure 14. Experimental tasks

The experiment consisted on the execution of the following seven tasks (see Figure 14):

Task 1. ***Definition of i* models.*** Each analyst defines the corresponding *i** model according to a specific case study.

Task 2. ***Generation of MDD models.*** Each measurement expert performs the Integranova class model generation from the defined *i** models by applying the transformation rules.

Task 3. ***Application of verification measures.*** Each measurement expert evaluates the verification measures in the *i** models.

Task 4. ***Modification of i* models.*** The analysts use the results obtained from Task 3 to fix or improve the *i** models.

Task 5. ***Second generation of MDD models.*** The measurement experts generate new class models from the improved versions of the *i** models.

Task 6. ***Second Application of verification measures*****.** The measurement experts evaluate the verification measures over the *i** models improved.

Task 7. ***Comparison of results and assessment*****.** The results obtained from Tasks 2, 3, 5, and 6 are compared and analyzed by the two measurement experts to check the hypotheses proposed.

It is worth mentioning that softgoals defined by the *i** experts are omitted from the diagrams. They are not considered in the generation of the class model, because this *i** construct has no projection over the target design model in the transformation process; *i.e.*, the resultant class model is not affected by the representation of soft-goals in the *i** model. For this reason, the softgoals defined by the *i** experts are omitted in the diagrams presented to simplify the model representation.

In addition to the models themselves, the instruments used in the experiment were the Eclipse Model Development Tools [54], the EMF editor for the *i** metamodel extended with the UML profile generated, the ATL scripts that transform the *i** models into class models, and tables filled according to a predefined template to keep the results of the experiment. It is also important to mention that to improve the understanding of the *i** models presented, the pictures of these models correspond to manual transcriptions of the defined EMF models using the *i** notation.

5.3. *Execution of the Experiment*

The experiment is based on a simplified version of an industrial case study developed using the Integranova technology. This case study has been defined with the independency of the *i** framework modeling facilities, and it preceded the development of the VeMI approach. The case study considers the operation of a photography agency, in particular, the management of work requests for hiring new photographers. A brief description of the organizational scenario involved is presented below:

> *The photography agency is dedicated to the management of photo reports and their distribution to publishing houses. This agency operates with freelance photographers, who must present a request to the production department of the photography agency. This request contains the photographer's personal information, a description of the equipment owned, and a brief curriculum vitae. An accepted photographer is classified by the production department in one of three possible levels for which minimum photography equipment is required. The possible levels are defined by the commercial department that establishes the price that will be paid to the photographer and the price that will be charged to the publishing house for each photo.*

For the organizational scenario proposed, the first *i** expert has defined the SR model (called ISTAR1) presented in Figure 15. This ISTAR1 model shows that the *production department* depends on the reception of *work requests* (*i.e.*, job applications) that are produced by *photographers* that want a *work opportunity*. The *work requests* include the photographers' *personal data*. The *production department* is responsible for *refusing* or *accepting* the received work requests by indicating the final work request status. For the accepted requests, a *photographer level* is assigned according to the information provided by the *commercial department.* The information introduced in the ISTAR1 model with the application of the profile is presented in Table 5.

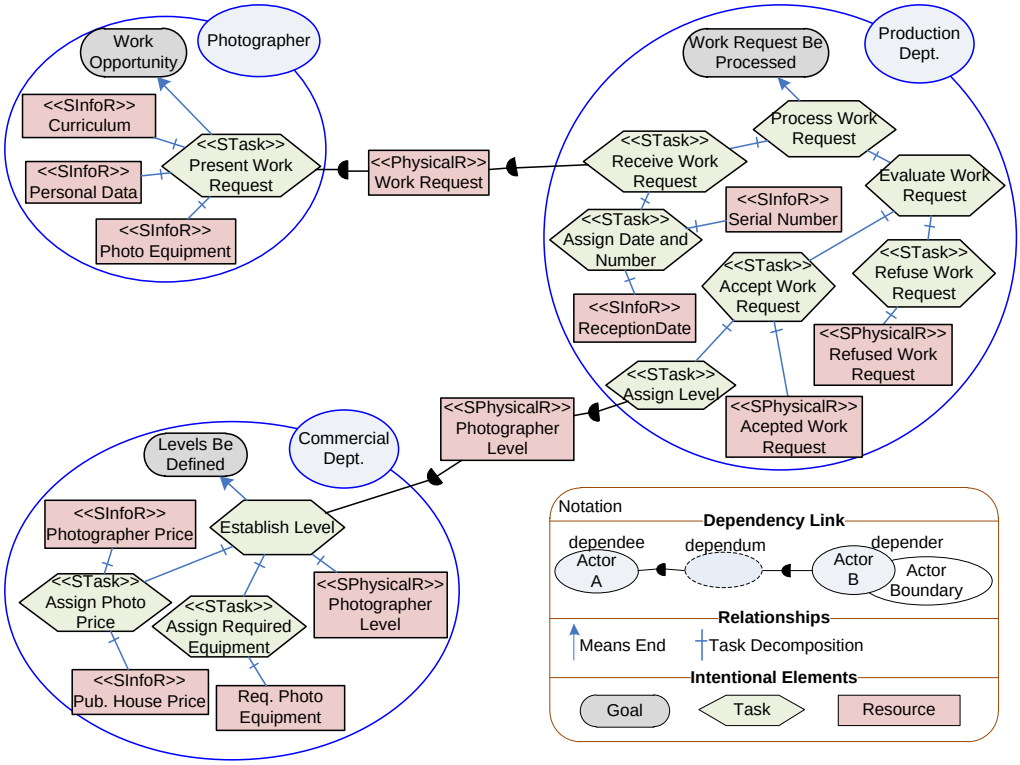

Figure 15. First *i** SR model (extended) for the photography agency case study (ISTAR1)

Table 5. Tagged values related to ISTAR1

TaggedValue	Value	TaggedValue	Value
Curriculum		**Photographer Price**	
.infoOf	--	.infoOf	Photographer Level
Photo Equipment		**Pub. House Price**	
.infoOf	--	.infoOf	Photographer Level
PersonalData		**Assign Required Equipment**	
.infoOf	--	.affects	--
Reception Date		.generates	--
.infoOf	Work Request	**Assign Date and Number**	
Serial Number		.affects	Work Request
.infoOf	Work Request	.generates	--
Assign Photo Price		**Assign Level**	
.affects	--	.affects	--
.generates	--	.generates	--
Present Work Request		**Refuse Work Request**	
. affects	--	.affects	--
.generates	Work Request	.generates	Refused Work Request
Receive Work Request		**Accept Work Request**	
.affects	--	.affects	--
.generates	Work Request	.generates	Accepted Work Request

Figure 16 shows the class model that is generated by the application of the transformation rules to the *i** model ISTAR1. Only those *i** elements related to the intended system are considered in the transformation process, which correspond to the stereotyped elements in the model ISTAR1.

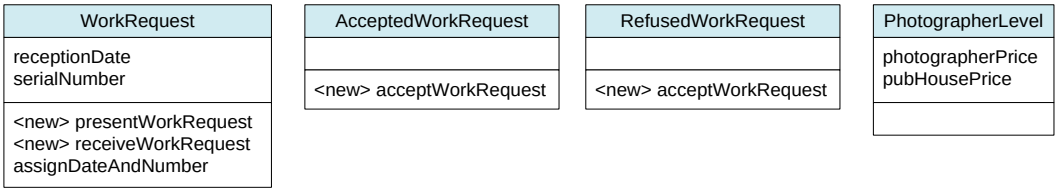

Figure 16. Class model generated from ISTAR1

After the generation of the class model, the verification measures are applied over the input *i** model ISTAR1. The application of the verification measures is made after the class model generation to prevent any manipulation of the original *i** model performed to solve potential issues identified by the measures. Table 6 shows the results obtained from the measures evaluation by indicating 1) the result of the measure (the values obtained from the aggregation OCLs) and 2) the *i** elements that return true from the evaluation of the locator OCLs.

Table 6. Results obtained from measures evaluation for ISTAR1

Measure	Alert	Result (Aggregation)	Locator
WAG	Critical	3 Resources	Curriculum, Photo Equipment, Personal Data
WSG	Critical	3 Tasks	Assign Photo Price, Assign Required Equipment, Assign Level
NAE	Warning	15 Elements	All stereotyped informational resources and tasks defined in actors' boundaries (none stereotyped actors in the model)
NIC	Warning	1 Entity	Photographer Level

It can be observed that those elements identified by the critical measures are not present in the class model generated, such as the resource *Curriculum* or the task *Assign Photo Price*. This demonstrates that the information reported by critical verification measures is really critical, since it prevents the correct transformation of the *i** elements that need to be considered in the system specification; *i.e.*, the generated system model is incomplete in relation to the input requirements. Therefore, it is necessary to fix the elements identified by the critical measures to assure the transformation of all the *i** elements selected (stereotyped).

For the improvement of the model, the analyst can consider the fixing guidelines obtained from the application of the VeMI approach (presented in Table 4). For instance, according these guidelines, the informational resources *Curriculum*, *Photo Equipment*, and *Personal Data* need to be related (or change their type) to a physical resource, or they need to be excluded from the class model generation by removing the corresponding stereotype. Thus, in the improved model *ISTAR1* (see Figure 17), the informational resources located by the WAG measure are now defined as information of the actor *Photographer*. The warning related to the NAE measure has been solved by defining the task *Establish Level* as a generation task for the resource *Photographer Level*. Table 7 shows the tagged values that have been changed in the improved *i** model.

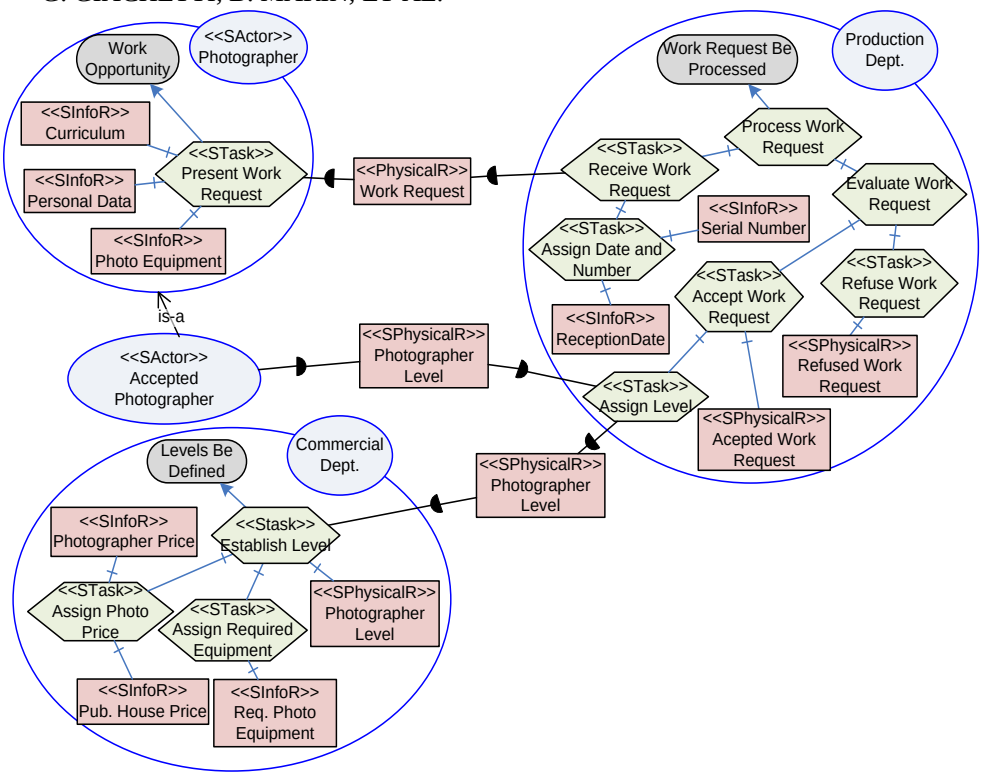

Figure 17. Improved *i** model ISTAR1

Furthermore, in the improved model *ISTAR1*, the task *Assign Level* now affects the actor *Accepted Photographer* (added to the model). The tasks *Assign Photo Price* and *Assign Photo Equipment* are now related to the resource *Photographer Level*. Another interesting change is the specification of the actor *Req. Photo Equipment* as an informational resource. Even though this resource has not been located by the verification measures, the analyst has decided that it must be included in the system as part of the *Photographer Level* after reviewing the measures results.

The informational resources located by the WAG measure are now defined as information of the actor *Photographer*. The warning related to the NAE measure has been solved by defining the task *Establish Level* as a generation task for the resource *Photographer Level*.

It is important to note that by solving the issues identified, an improved and more detailed requirement representation is obtained.

Table 7. Tagged values changed in the improved model ISTAR1

TaggedValue	Value	TaggedValue	Value
Curriculum		**Assign Required Equipment**	
.infoOf	Photographer	.affects	Photographer Level
Photo Equipment		.generates	
.infoOf	Photographer	**Assign Level**	
PersonalData		.affects	Accepted Photographer
.infoOf	Photographer	.generates	--
Req. Photo Equipment		**Establish Level**	
.infoOf	Photographer Level	.affects	--
Assign Photo Price		.generates	Photographer Level
.affects	Photographer Level		
.generates	--		

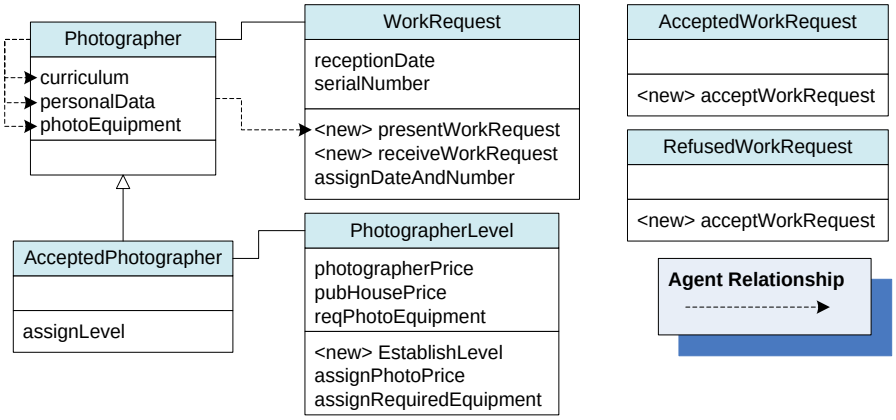

Figure 18. Class model generated from the improved model ISTAR1

Figure 18 shows the class model generated from the improved *i** model. This new class model provides a detailed system specification. It includes the classes *Photographer* and *AcceptedPhotographe*r. Also, associations among classes have been generated. In summary, all the stereotyped elements of the *i** model have been transformed to conceptual constructs of the target class model. Thus, the class model considers all the system requirements specified.

It is important to note that the generated class model is an initial class model; it must be refined at design time to obtain a fully compilable model. Some possible refinements are the specification of the specializations that exist between the class *PhotoWorkRequest* and the classes *AcceptedWorkRequest* and *RefusedWorkRequest*. Also, the cardinality of the associations and the appropriate specification of the services must be defined.

According to the process defined for the experiment, the generation of the class model from the improved model *ISTAR1* correspond to the fourth step. Now, the same fourth steps of the experiment is performed with the second *i** expert, which defines the model *ISTAR2*. Steps five and six of the experiment are presented in the next section.

Figure 19 shows the model *ISTAR2*, Table 8 presents the information related to its tagged values, and Figure 20 presents the initial class model (*MODEL2*) generated from *ISTAR2* without the information of the verification measures.

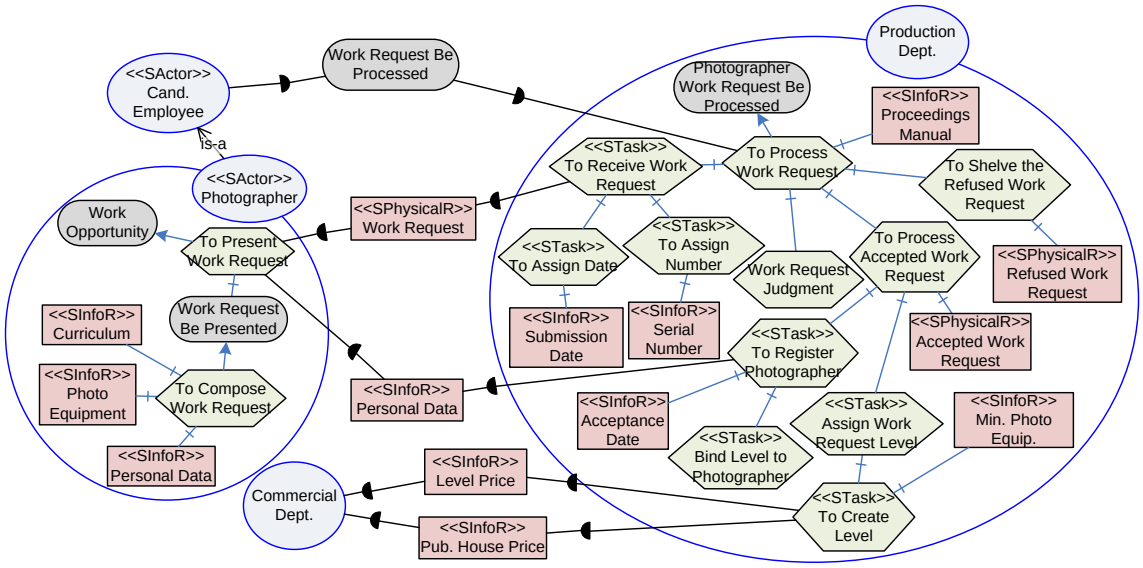

Figure 19. ISTAR2 model for photography agency description

Table 8. Tagged values related to ISTAR2

TaggedValue	Value	TaggedValue	Value
Curriculum		**To Create Level**	
.infoOf	Photographer	. affects	--
Photo Equipment		.generates	--
.infoOf	Photographer	**To Receive Work Request**	
PersonalData		.affects	--
.infoOf	Cand. Employee	.generates	Work Request
Level Price		**To Assign Date**	
.infoOf	--	.affects	Work Request
Proceedings Manual		.generates	--
.infoOf	--	**To Register Photographer**	
Min. Photo Equip.		.affects	Photographer
.infoOf	--	.generates	Photographer
Acceptance Date		**Bind Level to Photographer**	
.infoOf	--	.affects	Photographer
Submission Date		.generates	--
.infoOf	Work Request	**To Assign Number**	
Serial Number		.affects	Work Request
.infoOf	Work Request	.generates	--
Pub. House Price		**Assign Work Request Level**	
.infoOf	--	.affects	Accepted Work Request
		.generates	--

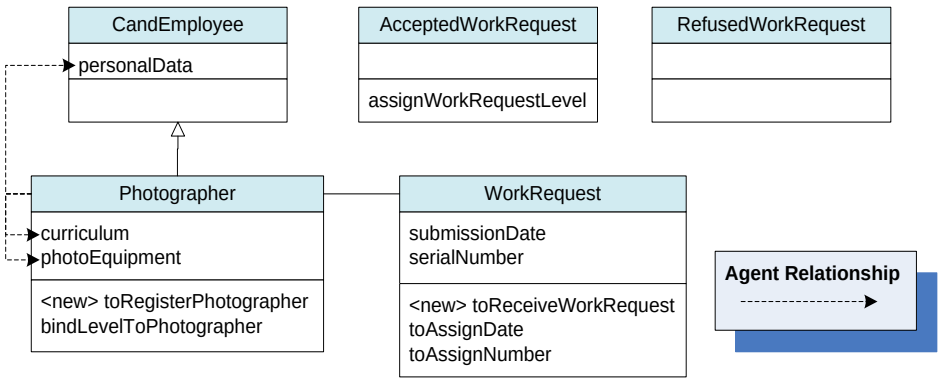

Figure 20. Class model generated from ISTAR2 model without improvements

After the generation of the class model from ISTAR2, the verification measures are applied. Table 9 shows the results obtained from the measures evaluation by indicating 1) the result of the measure and 2) the *i** elements that return true from the evaluation of the locator OCLs.

Table 9. Application of the verification measures to ISTAR2

Model	Measure	Alert Level	Measurement Result	Locator
ISTAR2	WAG	Critical	5 Resources	Level Price, Proceedings Manual, Min. Photo Equip., Acceptance Date, Pub. House Price
	WSG	Critical	1 Task	To Create Level
	NAE	Warning	12 Nodes	All the stereotyped informational resources and tasks defined in the Production Dept. Boundary

	NIC	Warning	3 Entities	Cand. Employee, Accepted Work Request, Refused Work Request

An interesting benefit that emerged while fixing the elements identified by the critical measures was that the analyst ANA2 detected a mistake in the understanding of the organizational description. The analyst initially defined the actor *Production Department* as responsible for the levels definition. However, the actual actor responsible is *Commercial Department*. As a consequence, the analyst defined a new physical resource *Level*, where the task *To Create Level* is the production task for this physical resource. Thus, the resources *Price Min.*, *Photo Equip.*, and *Pub. House Price* are defined as informational resources of the *Level* resource. Furthermore, in contrast to the reasoning performed by the first analyst (ANA1), the second analyst (ANA2) considered that all the actors involved in the *i** model must be part of the system-to-be. Thus, the improved *i** model did not generate non-accessible elements in the class model (measure NAE = 0). Additionally, the resource *Proceeding Manual* is changed from an informational entity to a physical entity. Figure 21 shows the improved ISTAR2 model, Table 10 present the tagged values changed, and Figure 22 shows the class model generated from the improved *i** model.

Table 10. Tagged values changed in the improved i* Model

TaggedValue	Value	TaggedValue	Value
Level Price		**Acceptance Date**	
.infoOf	Level	.infoOf	Photographer
Min. Photo Equip.		**To Create Level**	
.infoOf	Level	.affects	--
Pub. House Price		.generates	Level
.infoOf	Level		

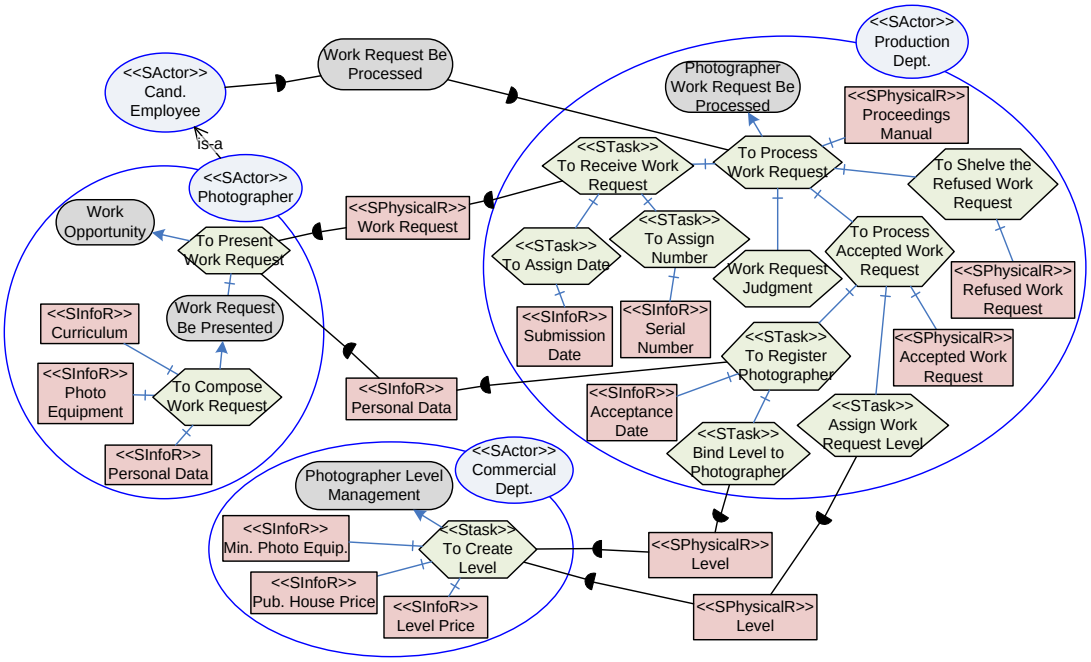

Figure 21. Model ISTAR2 improved with the verification measure results

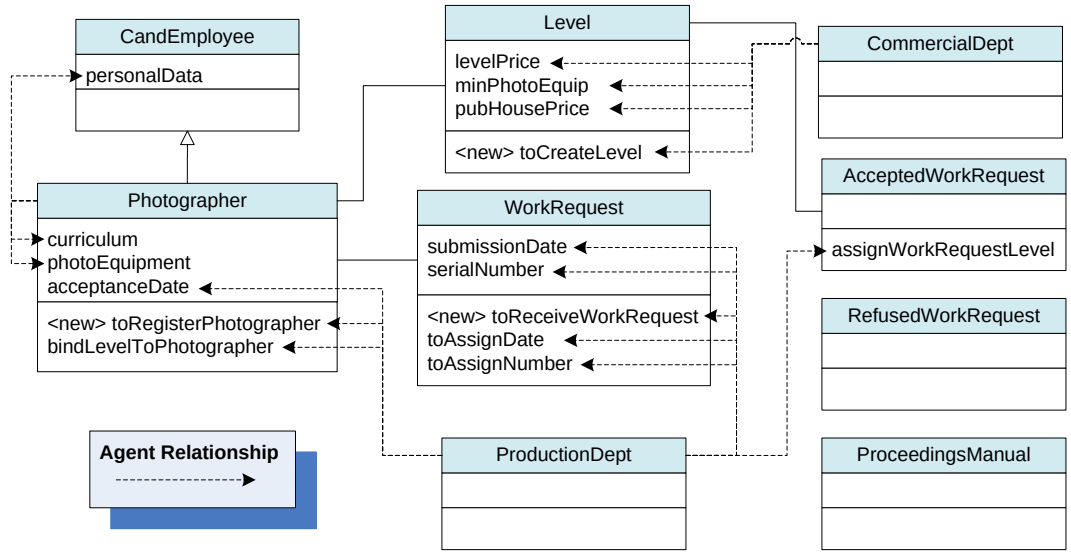

Figure 22. Class model obtained from the improved version of the second *i** model

5.4. *Results: Analysis and Interpretation Issues*

In the first generation of the class models, the resultant models suffered from several defects related to their lack of completeness. It is clear that not all the stereotyped elements were transformed into constructs of the class models; *i.e.*, the class models are not complete regarding to the requirements.

In addition to the identification of modeling issues and fixing guidelines, the information reported by the verification measures can be used to perform different analyses over the *i** models defined. For instance, we have defined the measure *PTE* to obtain early information about the completeness of the class model to be generated.

$$PTE = \left(\frac{TSE - (WAG + WSG)}{TSE}\right) \times 100$$

The measure PTE (*Percentage of Transformable Elements*) obtains the percentage of *i** elements related to the intended system that will be transformed into elements of the target class model. *PTE* is calculated using the TSE, WAG, and WSG measures.

TSE (*Total Stereotyped Elements*) counts the elements identified to be part of the intended system (the stereotyped elements). For *ISTAR1*, TSE = 19, and for *ISTAR2*, TSE = 23 (see Table 11). WAG and WSG correspond to the critical verification measures defined. Thus, for *ISTAR1*, WAG = 3 and WSG = 3. For *ISTAR 2*, WAG = 5 and WSG = 1. Note that the addition of the critical measures (WAG and WSG) is equal to the difference among stereotyped *i** elements and transformed *i** elements (see Table 11).

Thus, for *ISTAR1*, we obtain PTE = 68.4%. This means that only the 68.4% of the *i** stereotyped elements will be transformed during the design model generation; *i.e.*, 31.6% of the elements related to the system requirements will not be considered in the resultant class model. For *ISTAR2*, PTE = 73.9%.

Table 11. Experiment results

Measures →	**WAG**	**WSG**	**NAE**	**NIE**	**PTE**	**WIP**
First Generation (Initial *i** Models)						
ISTAR1	3	3	18	1	68.4%	--
ISTAR2	5	1	13	3	73.9%	--
Second Generation (Improved *i** Models)						
ISTAR1	0	0	16	0	100%	84.6%
ISTAR2	0	0	0	6	100%	77.3%

The warning measures support the identification of those *i** elements that can be improved to obtain a more complete specification of the class model generated. With this information, we have defined the WIP (*Warning Improvement Percentage*) measure, as presented below:

$$WIP = \left(\frac{(IMDD - ((OMDD + WAG + WSG))}{OMDD} \right) \times 100$$

The WIP measure returns the percentage of improvement (in terms of new elements) obtained in the MDD model generation after the evaluation of the verification measures. WIP is calculated using IMDD, OMDD, WAG, and WSG. IMDD (improved MDD model) corresponds to the number of Integranova class model constructs generated from the improved *i** model. OMDD (original MDD) corresponds to the number of Integranova class model constructs generated from the original *i** model. WAG and WSG correspond to the results obtained from the critical measures evaluation. The WIP measure is evaluated from the class models generated from the improved *i** models. Table 12 summarizes the results of the second MDD model's generation.

Table 12. Generation of MDD model from the improved *i** models

Improved *i** Model	Stereotyped Elements	Transformed Elements	Improved Class Model	Integranova Class Model Elements
ISTAR1	23	23	MODEL1	6 classes, 8 attributes, 9 services, 2 associations, 4 agent relations, 1 generalization (Total=30)
ISTAR2	25	25	MODEL2	9 classes, 9 attributes, 7 services, 3 association, 16 agent relations, 1 generalization (Total=45)

According to the information presented in Table 10, the measures WAG and WSG are equal to 0, which means that all the stereotyped *i** elements from the improved model ISTAR1 are transformed. Only the NAE measure is greater than zero (NAE=16), which means that 16 elements of the generated class model (*MODEL1*) will not have agent relationships defined. For *ISTAR2*, the measures WAG, WSG, and NAE are equal to zero, which means that the improved model ISTAR2 generates attributes, services, and accessible elements correctly. Only NIC was greater than zero (NIC=6). In fact, NIC's value is even greater than the result obtained from the initial *ISTAR2* model (NIC=3). This situation is produced by the two new actors defined as part of the system and the change in the stereotype of the resource *Proceeding Manual*, which is defined now as a physical resource. However, this is a warning verification measure that does not affect the completeness of the *i** model transformation. With the results obtained in the experiment, we can test the hypotheses H_{RCOM} and H_{CCOM}, and consequently answer our research question.

The experiment shows that by fixing the issues identified from the application of the critical measures (WAG and WSG) in the improved *i** models ISTAR1 and ISTAR2, the completeness of the resultant class models (improved MODEL1 and MODEL2) is achieved according to the system requirements. In both *i** models, 100% of the stereotyped elements are transformed into the corresponding class model constructs (see PTE measure results). Therefore, we can state that the hypothesis H_{RCOM} has been demonstrated.

Also, the experiment results show that by fixing the issues identified by the warning measures (NAE and NIC), the completeness of the class models in relation to system functionality is higher. This is observed in the number of MDD constructs generated from the improved *i** models in relation to the original *i** models (see WIP measure results). We find that 84.6% of additional class model elements are obtained from the improved model ISTAR1 and 77.3% from the model ISTAR2. Thus, since the class model elements are directly representing the functionality of the final software system, the hypothesis H_{CCOM} is also demonstrated.

Finally, we can conclude that the completeness of the generated Integranova class model from an i* model is supported by the application of the VeMI approach.

6 Overall Analysis

This section presents an overall analysis of the VeMI approach as well as some threats to the validity of the evaluation of VeMI.

A first element to analyze is the relevance of using the VeMI definition process as the starting point of our verification approach instead of a direct and intuitive definition of OCL verification rules. This decision comes from the maturity that the measurement specification has in the software engineering context, where we can find sound frameworks for the definition and implementation of measures. These frameworks have been considered for the systematic definition of the schema proposed for the appropriate identification of properties that must be measured and, in the context of this paper, also verified. It also assures the theoretical validity of the defined measures according to metrology concepts [56], which are designed independently of implementation platforms. Additionally, as we can observe in the definition and evaluation sections (sections 4 and 5, respectively), the VeMI approach can be used to infer fixing guidelines to the defined *i** models as well as to perform different analyses at early stages of the development process. These are clear advantages of the proposed verification approach in regard to other mechanisms for defect detection [43]. Moreover, we have observed that the fixing guidelines also facilitate the comprehension of the extensions generated for the *i** framework for integration with the MDD approach.

Another relevant aspect of the VeMI approach is that the entire measure specification is performed by following the model-driven philosophy, where the measures and the required modeling information are specified in a verification metamodel. The extensions over the *i** framework are defined by means of lightweight extensions (defined as a UML profile) that do not alter the original *i** metamodel specification, which permits compatibility with existent technologies that use the same metamodel as a reference. Also, we have considered mechanisms to automate the generation of the extensions. With these mechanisms, the main effort in the application of the VeMI approach is put into the appropriate definition of the transformation rules, the identification of the properties involved, the definition of the verification measures, and the definition of the mapping between the verification metamodel and the target *i** metamodel. Thus, once the VeMI Approach is defined for a particular model interplay scenario (such as *i** and Integranova), it can be used with few modifications over and over in different projects. In this respect, the analyst's work is centered only on the definition and improvement of the extended *i** model; the remaining tasks for the application of the VeMI approach are automatically performed.

It is important to consider that the transformation rules and verification measures formulated in this paper are specific for Integranova. Thus, other MDD approaches with different transformation guidelines will require different (or additional) verification measures. However, since we have intended to select a representative set of transformation rules, the resultant measures can provide relevant information to other object-oriented MDD approaches. Moreover, despite the fact that the VeMI approach has been applied to the *i** modeling framework, the concept involved can be easily applied to other works that propose the interoperability among goal-oriented modeling approaches, such as [69] [10] [70]. Also, the transformation rules proposed for the Integranova [20] can be applied to any other class model-based approach with minor changes.

6.1 *Threats to Validity*

Even though this study has been supported by a predefined study protocol, it has some limitations. In this section, we discuss all the aspects during the experiment design and execution that might lead to a threat to validity as well as the actions we have taken to mitigate them.

6.1.1 Internal Validity

To minimize the impact of the non-random selection of the subjects and technology used (i* and Integranova), we selected subjects with similar backgrounds in the i* framework and the Integranova MDD approach. We also asked the i* analysts to draw the i* models by hand, avoiding the interference of the EMF editor with the business analysis required for generating the models. This is due to the fact that the EMF editor does not provide *i** notation, which could affect the appropriate analysis of the business. In tasks 2 and 5, the measurement experts translated the hand-made *i** models with the corresponding EMF tree-like representation in order to apply the verification measures and to generate the corresponding design models automatically.

The model itself can influence in the validity of the experiment results. To minimize the impact, we considered data triangulation, which refers to using more than one data source or collecting the same data on different occasions. In this case, we have two sources (the two *i** models) produced by the two i* analysts representing the same data.

As we were not evaluating the performance, we defined the experimental tasks without any limitation on the execution time, thereby avoiding that the time factor would have any influence on the quality of the resulting artifacts.

Finally, to avoid the situation where the work done by one expert could affect that of the other, we located the *i** analysts and experts in separate rooms, and we ensured that the measurement experts did not share or comment on the content of their work during the experiment.

6.1.2 External Validity

The subjects participating in the experiment belong from academy as opposed to being actual practitioners. We are aware that this limits the generalization of the results, instead of some fresh results that state that there are just minor differences when we use subjects from academy and practitioners in software engineering experiments [71]. To minimize the impact, we select subjects with similar background and some experience working in industrial MDD projects.

Regarding the generalization of the VeMI approach, we are aware that we are using specific models – i* and the Integranova technology – which could impede the generalization of the results. However, to mitigate this threat, we have defined measures for transformation guidelines that could be easily used with other MDD approaches.

6.1.3 Construct Validity

Regarding the process used to verify the VeMI approach, it has been systematically designed and evaluated following several well-known guidelines for the definition and evaluation of measures, such as [72], [73], [74]; for instance, we defined the research question, then we identified the independent variables (which correspond to the i* models) and dependent variables and indicated how we can measure these variables, we identified the hypothesis, and later we systematically defined and executed the tasks of the experiment. It is important to note that using the measures defined in the VeMI approach alleviates the threat of the expertise of the subjects could provoque that there exists i* constructs that are not present in the MDD model. Thus, we consider that we have mitigated the possible threats regarding construct validity.

7 Conclusions and Further Work

From the results presented in this paper, we can conclude that the VeMI approach supports the verification of the goal-oriented models used in software model-driven development processes, specifically, in terms of assuring the completeness of the model-to-model transformations. Moreover, it facilitates that the definition of the goal-oriented models be properly aligned with the target MDD approach without demanding additional knowledge about the specific modeling constructs of the MDD approach. Thus, the VeMI approach is aimed to be applied to real development scenarios, where goal-oriented models are manually defined by system analysts, and they need to be properly verified to assure that the final software product is correctly aligned with organizational needs. With this verification approach, we intend to contribute with a new stone for paving the road of model-driven engineering (MDE) [75], which drives the development process from the requirements to the code generation by means of well-defined model transformations.

The quantitative information obtained from the evaluation of verification measures related to the VeMI approach allows the determination of the degree of completeness of the resultant design models in relation to the original requirements models. This information can also be used to compare different requirement models in a concrete MDD approach to determine their effectiveness in relation to the model compilation process – *i.e.*, which goal-oriented specification is capable of producing a major number of design artifacts that will be considered in the final software generation process. It is important to remark that a requirement model that provides a larger amount of information to the model design process also provides the clearest vision of the decisions involved in the definition of system models. This facilitates the alignment of the refinement and improvement tasks of design models with respect to business objectives and requirements.

The process applied to the evaluation of the VeMI approach and the artifacts obtained has been presented to facilitate the replication of the results and to guide practitioners in the application of the verification approach to different integrations of goal-oriented modeling and MDD approaches. In this context, there is an important aspect to be considered by MDD practitioners who are interested in putting into practice the VeMI proposal: the complexity of determining which elements must be maintained at the design level and which must be up scaled to the analysis level. For instance, it is

possible to introduce an extension to identify the generalization between resources in the *i** model, but we have considered that this task is part of the design effort. Therefore, further studies can be oriented toward identifying new extensions for requirement models without affecting the clarity of the business analysis. Consequently, the inclusion of new extensions also implies the definition of new verification measures.

We are aware that the VeMI approach can be improved and extended for obtaining a sounder verification. However, it already provides interesting features, such as the systematic process for guiding the application of verification measures and the definition of fixing guidelines as well as the use of standard modeling approaches that are supported by open-source tools to automate the verification process, which implies a time and effort reduction with respect to manual verifications.

From a tools perspective, it is important to mention that we did not find tools that provided transparent support for all the modeling features considered, and, hence, additional programming effort was necessary, for instance, to support the profile extension mechanisms in a non-UML metamodel. However, existing tools have continuously improved the support to the standards considered (such as the Eclipse MDT project [54]). This also motivates the emergence of new approaches for the verification of the integration of modeling approaches that improve MDD capabilities and the quality of the software products at the end.

Regarding the application of the VeMI approach to industrial contexts, we have performed an exploratory study with engineers from a software company that shows that the use of the VeMI approach provides interesting benefits for novel engineers who are adopting the Integranova MDD approach. We observe that learning *i** demands less time than learning the Integranova technology. From practical experience, we found that the average training time required for the *i** framework application models is one week, while the Integranova approach models involve one month of training (involving in both cases a full-time training process of six hours per day). However, it seems that once the engineers have gained experience with the Integranova technology, they prefer to skip the definition of the analysis model and work directly on the design model, especially for business scenarios of low complexity. Therefore, we consider as future work the development of studies to determine the impact of the VeMI approach for adopting goal-oriented modeling in industrial MDD developments. In addition, we plan to apply the VeMI approach to other model-driven development approaches.

ACKNOWLEDGEMENTS

This work has been developed with the support of FONDECYT under the projects AMoDDI 11130583 and TESTMODE 11121395.

REFERENCES

[1] B. Selic, "The Pragmatics of Model-Driven Development," *IEEE Software*, vol. 20, pp. 19–25, 2003.

[2] D. Schmidt, "Model Driven Engineering," *IEEE Computer*, vol. 39, pp. 25-31, 2006.

[3] S. W. Liddle, "Model-Driven Software Development," in *Handbook of Conceptual Modeling*, D. W. Embley and B. Thalheim, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 17-54.

[4] J. Cabot and E. Yu, "Improving Requirements Specifications in Model-Driven Development Processes," 1st Int. Workshop on Challenges in Model-Driven Software Engineering (MoDELS'08), 2008.

[5] R. Monteiro, J. Araújo, V. Amaral, and P. Patrício, "MDGore: Towards model-driven and goal-oriented requirements engineering," 18th IEEE International Requirements Engineering Conference (RE), 2010.

[6] O. Pastor and S. España, "Full Model-Driven Practice: From Requirements to Code Generation," 24th International Conference Advanced Information Systems Engineering (CAiSE), Gdansk, Poland, 2012.

[7] G. Loniewski, E. Insfran, and S. Abrahao, "A Systematic Review of the Use of Requirement Engineering Techniques in Model-Driven Development," 13th International Conference on Model Driven Engineering Languages and Systems (MoDELS), 2010.

[8] A. Lamsweerde, *Requirements Engineering - From System Goals to UML Models to Software Specifications*: Wiley, 2009.

[9] C. W. Lu, C. H. Chang, W. C. Chu, Y. W. Cheng, and H. C. Chang, "A Requirement Tool to Support Model-Based Requirement Engineering," 32nd Computer Software and Applications Conference (COMPSAC '08), 2008.

[10] R. Monteiro, J. Araújo, V. Amaral, M. Goulão, and P. Patrício, "Model-Driven Development for Requirements Engineering: The Case of Goal-Oriented Approaches," Eighth International Conference on the Quality of Information and Communications Technology (QUATIC), 2012.

[11] T. Ruiz-López, C. Rodríguez-Domínguez, M. Noguera, and M. J. Rodríguez, "A Model-Driven Approach to Requirements Engineering in Ubiquitous Systems," 3rd International Symposium on Ambient Intelligence (ISAmI 2012), 2012.

[12] OMG, "MDA Guide Version 1.0.1," 2003.

[13] S. J. Mellor, K. Scott, A. Uhl, and D. Weise, *MDA Distilled: Principles of Model-Driven Architecture*: Addison-Wesley Professional, 2004.

[14] I. Zikra, J. Stirna, and J. Zdravkovic, "Analyzing the Integration between Requirements and Models in Model Driven Development," 12th International Conference, BPMDS 2011, and 16th International Conference, EMMSAD 2011, London, UK, , 2011.

[15] D. Gross and E. Yu, "From Non-Functional Requirements to Design through Patterns," *Requirements Engineering Journal*, vol. 6, pp. 18–36, 2001.

[16] B. Hailpern and P. Tarr, "Model-driven develpment: The good, the bad, and the ugly," *IBM Systems Journal*, vol. 45, pp. 451–461, 2006.

[17] J. Horkoff, T. Li, F.-L. Li, M. Salnitri, E. Cardoso, P. Giorgini, *et al.*, "Using Goal Models Downstream: A Systematic Roadmap and Literature Review," *International Journal of Information System Modeling and Design (IJISMD)*, vol. 6, pp. 1-42, 2015.

[18] A. Lamsweerde, "Goal-oriented requirements engineering: A guided tour," 5th IEEE International Symposium on Requirements Engineering (RE01), 2001.

[19] A. Perini and A. Susi, "Automating model transformations in agent-oriented modelling," in *Agent-Oriented Software Engineering VI*, ed: Springer, 2005, pp. 167-178.

[20] O. Pastor and J. C. Molina, *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling*, 1st edition ed. New York: Springer, 2007.

[21] G. Giachetti, F. Alencar, B. Marín, O. Pastor, and J. Castro, "Beyond Requirements: An Approach to Integrate i* and Model-Driven Development," XIII Congreso Iberoamericano en Software Engineering - CIBSE 2010, Cuenca, Ecuador, 2010.

[22] A. M. L. de Vasconcelos, G. Giachetti, B. Marín, and O. Pastor, "Towards a CMMI-Compliant Goal-Oriented Software Process through Model-Driven Development," PoEM 2011, 2011.

[23] F. Alencar, B. Marín, G. Giachetti, O. Pastor, J. Castro, and J. H. Pimentel, "From i* Requirements Models to Conceptual Models of a Model Driven Development Process," 2nd Working Conference on The Practice of Enterprise Modeling (PoEM), 2009.

[24] IEEE, "IEEE 1012 Standard for Software Verification and Validation," 2004.

[25] ISO, "International vocabulary of basic and general terms in metrology (VIM)," Geneva, Switzerland2004.

[26] E. Yu, "Tech. Report DKBSTR-94-6: Modelling Strategic Relationships for Process Reengineering," Dept. of Computer Science. University of Toronto, Toronto, Canada1995.

[27] E. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering," 3rd IEEE Int. Symp. on Requirements Engineering (RE'97), 1997.

[28] E. Yu, P. Giorgini, N. Maiden, and J. Mylopoulos, *Social Modeling for Requirements Engineering*: The MIT Pres, 2011.

[29] C. Rolland, C. Souveyet, and C. B. Achour, "Guiding Goal Modelling Using Scenarios," *IEEE Transactions on Software Engineering (IEEE TSE), Special Issue on Scenario Management,* vol. 24, pp. 1055–1071, 1998.

[30] E. Yu, "Modelling Strategic Relationships for Process Reengineering," PhD Thesis PhD Thesis, University of Toronto, Toronto, Canada 1995.

[31] O. Pastor, J. Gómez, E. Insfrán, and V. Pelechano, "The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming," *Information Systems,* vol. 26, pp. 507–534, 2001.

[32] OMG, "UML Simplified. UML 2.5," 2013.

[33] R. Pohjonen and S. Kelly, "Domain-Specific Modeling," *Dr. Dobb's Journal,* 2002.

[34] M. A. Laguna and B. Gonzalez-Baixauli, "Requirements variability models: metamodel based transformations," Symposia on Metainformatics (MIS 05), 2005.

[35] A. Lapouchnian, Y. Yu, S. Liaskos, and J. Mylopoulos, "Requirements-driven design of autonomic application software," Conference of the Center for Advanced Studies on Collaborative Research (CASCON 2006), 2006.

[36] Z. Li, X. Zhou, A. Gu, and Q. Li, "A complete approach for CIM modelling and model formalising," *Information and Software Technology,* vol. 65, pp. 39-55, 2015.

[37] E. Letier and A. Lamsweerde, "Deriving Operational Software Specifications from System Goals," 10th ACM SIGSOFT Symp. on the Foundations of Software Engineering (FSE10), 2002.

[38] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. Yu, "Evaluating goal models within the goaloriented requirement language," *International Journal of Intelligent Systems (IJIS),* vol. 25, pp. 841–877, 2010.

[39] P. Giorgini, S. Rizzi, and M. Garzetti, "Goal-oriented Requirement Analysis for Data Warehouse Design," 8th Int. Workshop on Data Warehousing and OLAP, 2005.

[40] J. Pardillo, F. Molina, C. Cachero, and A. Toval, "A UML Profile for Modelling Measurable Requirements," 4th International Workshop on Foundations and Practices of UML (FP-UML) ER Workshop, 2008.

[41] M. Genero, M. Piattini, and C. Calero, "A Survey of Metrics for UML Class Diagrams," *Journal of Object Technology,* vol. 4, pp. 59-92, 2005.

[42] Y. Tong, W. Fangjun, and G. Chengzhi, "A comparison of metrics for UML class diagrams," *ACM SIGSOFT Software Engineering Notes,* vol. 29, 2004.

[43] B. Marín, G. Giachetti, and O. Pastor, "Applying a Functional Size Measurement Procedure for Defect Detection in MDD Environments " 16th European Conference EUROSPI, Alcalá (Madrid), Spain, 2009.

[44] G. Giachetti, B. Marín, and X. Franch, "Using Measures for Verifying and Improving Requirement Models in MDD Processes," 14th International Conference on Quality Software, Allen, TX, USA, 2014.

[45] OMG, "MOF 2.4.2 Core Specification," 2014.

[46] OMG, "XMI 2.4.2 Specification," 2014.

[47] F. Alencar, O. Pastor, B. Marín, G. Giachetti, and J. Castro, "Aligning Goal-Oriented Requirements Engineering and Model-Driven Development," 11th International Conference on Enterprise Information Systems (ICEIS), 2009.

[48] X. Franch, " A Method for the Definition of Metrics over i* Models," 21st International Conference on Advanced Information Systems (CAiSE 2009), 2009.

[49] X. Franch and G. Grau, "Towards a Catalogue of Patterns for Defining Metrics over i* Models," 20th International Conference on Advanced Information Systems (CAiSE 2008), 2008.

[50] O. Pastor, G. Giachetti, B. Marín, and F. Valverde, "Automating the Interoperability of Conceptual Models in Specific Development Domains," in *Domain Engineering: Product Lines, Languages, and Conceptual Models*, I. Reinhartz-Berger, A. Sturm, T. Clark, S. Cohen, and J. Bettin, Eds., ed: Springer, 2013, pp. 349-374.

[51] O. Pastor and G. Giachetti, "Linking Goal-Oriented Requirements and Model-Driven Development," in *Intentional Perpectives on Information Systems Engineering*, ed: Springer-Verlag, 2010, pp. 255–274.

[52] G. Giachetti, F. Valverde, and O. Pastor, "Improving Automatic UML2 Profile Generation for MDA Industrial Development," 4th International Workshop on Foundations and Practices of UML (FP-UML) – ER Workshop, 2008.

[53] G. Giachetti, F. Valverde, and B. Marín, "Interoperability for model-driven development: Current state and future challenges," RCIS 2012, Valencia - Spain, 2012.

[54] Eclipse. *Model Development Tools Project*. Available: http://www.eclipse.org/modeling/mdt/

[55] Eclipse. (Last accessed July 2015). *Modeling Framework Project*. Available: http://www.eclipse.org/modeling/emf/

[56] F. Jouault and I. Kurtev, "Transforming Models with ATL," Satellite Events at the MoDELS 2005 Conference, 2005.

[57] OMG, "QVT 1.1 Specification," 2011.

[58] G. Giachetti, M. Albert, B. Marín, and O. Pastor, "Linking UML and MDD through UML Profiles: a Practical Approach based on the UML Association," *The Journal of Universal Computer Science (JUCS)*, vol. 16, pp. 2353-2373 2010.

[59] G. Giachetti, B. Marín, and O. Pastor, "Using UML as a Domain-Specific Modeling Language: A Proposal for Automatic Generation of UML Profiles," 21st International Conference on Advanced Information Systems Engineering (CAiSE), Amsterdam, The Netherlands, 2009.

[60] F. Alencar, B. Marín, G. Giachetti, O. Pastor, J. Castro, and J. H. Pimentel, "From i* Requirements Models to Conceptual Models of a Model Driven Development Process," 2nd Working Conference on The Practice of Enterprise Modeling (PoEM 2009), 2009.

[61] G. Giachetti, B. Marín, and O. Pastor, "Integration of domain-specific modelling languages and UML through UML profile extension mechanism," *International Journal of Computer Science & Applications*, vol. 6, pp. 145-174, 2009.

[62] C. Ayala, C. Cares, J. P. Carvallo, G. Grau, M. Haya, G. Salazar, *et al.*, "A Comparative Analysis of i*-Based Goal-Oriented Modelling Languages," International Workshop on Agent-Oriented Software Development Methodologies (AOSDM'05), at the SEKE Conference, Taipei, Taiwán; China. , 2005.

[63] X. Franch, "Incorporating Modules into the i* Framework," 22nd International Conference on Advanced Information Systems (CAiSE 2010), Hammamet, Tunisia, 2010.

[64] M. Lucena, E. Santos, M. J. Silva, C. Silva, F. Alencar, and J. F. B. Castro, "Towards a Unified Metamodel for i*," 2nd IEEE Int. Conference on Research Challenges in Information Science (RCIS 2008), 2008.

[65] D. Amyot, J. Horkoff, D. Gross, and G. Mussbacher, "A Lightweight GRL Profile for i* Modeling," Third International Workshop on Requirements, Intentions and Goals in Conceptual Modeling (RIGIM) ER Workshops, 2009.

[66] B. Marín, G. Giachetti, and O. Pastor, "Technical Report DSIC-II/13/08. The Photography Agency: A case study of the OO-Method Approach," Universidad Politécnica de Valencia, Valencia, Spain2008.

[67] ISO/IEC, "ISO/IEC 9126-1, Software Eng. – Product Quality – Part 1: Quality model," 2001.

[68] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering - An Introduction*: Kluwer Academic, 2000.

[69] R. Monteiro, J. Araújo, V. Amaral, M. Goulão, and P. Patrício, "Adding Interoperability to Requirements Models," *SOFTWARE QUALITY PROFESSIONAL*, vol. 15, pp. 16-27, 2013.

[70] J. C. Nwokeji, T. Clark, and B. S. Barn, "A proposal for consolidated intentional modeling language," Proceedings of the Second Workshop on Graphical Modeling Language Development, 2013.

[71] I. Salman, A. T. Misirli, and N. Juristo, "Are students representatives of professionals in software engineering experiments?," Proceedings of the 37th International Conference on Software Engineering - ICSE 2015, Piscataway, NJ, USA, 2015.

[72] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, and B. Regnell, *Experimentation in Software Engineering*: Springer, 2012.

[73] A. Jedlitschka, M. Ciolkowski, and D. Pfahl, "Reporting Experiments in Software Engineering," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. Sjøberg, Eds., ed: Springer London, 2008, pp. 201-228.

[74] P. Runeson and M. Host, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering Journal*, vol. 14, pp. 131–164, 2009.

[75] S. Kent, "Model Driven Engineering," Integrated Formal Methods (IFM), 2002.