UNIVERSIDAD POLITECNICA DE VALENCIA

PROS
Centro de Investigación en Métodos de Producción de Software

Technical Report




Automatic Verification of Requirement Models for Their Interoperability in Model-Driven Development Processes

Giovanni Giachetti, Fernanda Alencar, Xavier Franch, Beatriz Marín, Oscar Pastor





Ref. #:	ProS-TR-2011-07				
Title:	Automatic Verification of Requirement Models for Their Interoperability in Model-Driven Development Processes				
Author (s):	Giovanni Giachetti, Fernanda Alencar, Xavier Franch, Beatriz Marín, Oscar Pastor				
Corresponding author (s):	Giovanni Giachetti, ggiachetti@pros.upv.es Fernanda Alencar, fernanda.ralencar@ufpe.br Xavier Franch, franch@essi.upc.edu Beatriz Marín, bmarin@pros.upv.es Oscar Pastor, opastor@pros.upv.es				
Document version number:	1.0	Final version:	Yes	Pages:	49
Release date:	April 2011				
Key words:	Domain-specific Modeling, Model-Driven Development, Measures, Verification, I* Framework				

Automatic Verification of Requirement Models for Their Interoperability in Model-Driven Development Processes[1]

Giovanni Giachetti[1], Fernanda Alencar[2], Xavier Franch[3], Beatriz Marín[1], Oscar Pastor[1]

[1]Centro de Investigación en Métodos de Producción de Software
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia, Spain
{ggiachetti, bmarin, opastor}@pros.upv.es
[2]Universidade Federal de Pernambuco
Av. Acad. Hélio Ramos s/n, 50.740-530, CDU, Recife, Brasil
fernanda.ralencar@ufpe.br
[3]Universitat Politècnica de Catalunya (UPC)
UPC-Campus Nord, Omega-122, 08034 Barcelona, Spain
franch@essi.upc.edu

1 Introduction

The present software development context is rapidly moving to the Model-Driven Development (MDD) paradigm [57], which has motivated the emergence of multiple MDD approaches oriented to automating the final software product generation by means of model compilation processes. Just as any software development process does, MDD processes also require an appropriate requirement elicitation activity to obtain software products that fit the customers' needs. Integrating the requirements elicitation activity into the MDD processes, it should be possible to obtain software products properly aligned with the stakeholders' needs [11]. In addition, it should be possible to

[1] This work has been submitted to SOSYM Journal on November 2010. We are waiting for reviewers feedback.

estimate the impact that the generated software systems have in the organizational objectives, and to identify different alternatives for the configuration of the intended software systems.

Among modeling approaches to requirement elicitation, the *i** framework [61] provides a suitable alternative for the analysis of complex scenarios. The *i** framework is a goal-oriented [12] approach used in several activities and contexts of software engineering, and in particular, in the early phases of requirements engineering [62]. The versatility and expressive power of *i** is extensively documented [15]. Thus, we consider that it is a good choice for the requirement elicitation in MDD processes.

However, to achieve this goal, there is still a gap to bridge between *i** and MDD processes, since *i** models are not oriented to automatic software generation but to intentional modeling. Therefore, to apply *i** models in a MDD process, it is necessary to transform them into an appropriate input for MDD processes. This input corresponds to initial MDD models that must be refined at design time to obtain appropriate artifacts for the automatic model compilation process. Certain approaches have defined guidelines to perform this transformation but, in general terms, these guidelines consider *i** models to already be perfectly defined for the generation of the corresponding MDD models. In real application contexts, this idealist scenario is not feasible and, hence, additional verification mechanisms are required (refer to related work section). Thus the goal of this paper is:

> To properly define and integrate automatic verification measures into the *i** framework to assess the adequacy of *i** models for automatic generation of initial models related to MDD process.

To achieve the proposed objective, this paper tackles the following elements:

Definition: Presents a specific proposal for the systematic definition of *i** verification measures, which assure the completeness of the MDD models that are generated from *i** models. Thus, the verification measures act as indicators of modeling issues, which identify the *i** elements that need to be fixed to assure the automatic generation of input models for MDD processes.

Integration: A process defined for domain-specific modeling and UML integration [22] is adapted to integrate into the *i** framework the defined verification measures and the modeling information that is necessary to automatically transform *i** models into MDD models.

Evaluation: The verification proposal is empirical validated through a laboratory experiment, which demonstrates that the measures obtained provide support to achieve the completeness of the generated MDD model. The execution of this experiment is also used to show how the verification measures are applied to improve *i** models in a specific MDD context. To do this, we select the OO-Method MDD approach, which has been successfully applied to industrial software development [54].

The rest of the paper is organized as follows. Section 2 shows the background and the related work. Section 3 presents a big picture of the transformation process defined for transforming *i** models into MDD models. Section 4 details the proposed *i** verification process. Section 5 explains how the verification measures can be used to fix and improve *i** models. Section 6 presents an empirical study performed to evaluate the efficacy of the verification proposal. Section 7 presents an overall analysis of the proposal. Finally, Section 8 presents our conclusions and further work.

2 Background and Related Work

In this section, the main metrology concepts used in this paper are clarified. Also, the *i** framework and the OO-Method MDD approach used to explain and evaluate our proposal are briefly introduced. Afterwards, a discussion about the related work is presented.

2.1 Clarifying Verification and Measure Concepts

In the literature, there is no consensus for the concepts used in the software measurement field. This has provoked that different concepts are used to refer to the same things, or even the same concept is used to refer to different things. Thus, we have carefully analyzed the measurement standards to properly use the terms involved in this paper, which are related to the concepts of verification and measure.

Verification is defined in the International Vocabulary of Basic and General Terms in Metrology [29] as "*confirmation through examination of a given item and provision of objective evidence that it fulfils specified requirements*". In contrast, validation is defined in the same standard vocabulary as "*confirmation through examination of a given item and provision of objective evidence that it fulfils the requirements for a stated intended use*". These definitions are also agreed with the widely used Barry Boehm's definitions to verification (doing the system right) and validation (doing the right system) [10]. Thus, we use the term verification instead of validation since we focus in the correct use of the transformation guidelines defined to go from *i** models to MDD-oriented models.

Even though the most of referenced works related to software measurement use the term metric instead of measure, we use the term measure because it is more appropriate to the objectives of our approach (we measure *i** elements)

and it prevents ambiguous interpretations. This term distinction is clearly presented in the paper [26]. Extending the concept of measure we have introduce in this paper the term *verification measure*.

It is important to clarify that a *verification measure* is not referring to a verification mechanism by itself; it is a special measure that supports the verification and improvement of an *i** model by means of a proper analysis of the reported information.

2.2 The *i** Goal-Oriented Requirements Framework

In general terms, the *Goal-Oriented Requirement Engineering* (GORE) approaches are oriented to obtain the 'what' of the intended systems through the analysis of organizational scenarios [34][56]. Among several existing GORE approaches, the *i** framework [60] 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 a sense that they do not simply carry out activities and produce entities, but also they 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. The SR model provides the internal decomposition of SD actors' intentions. We have considered the *i** SR model to perform the integration of *i** models into MDD processes since it offers a detailed representation of the analyzed problem scenario, which provides extra information that is relevant to generate appropriate inputs for MDD processes.

Figure 1 shows a example of an *i** SR model that is a partial representation of the OO-Method study presented in [42], which is related to the management of work requests in a Photography Agency. In order to simplify the *i** model representation, the soft goals are omitted in the example since this *i** construct does not participate in the generation of the target MDD models that are considered in this paper. Even though a similar situation occurs for *i** goals, this constructs are represented to be consistent with the *i** framework notation.

The organizational description related to the example *i** model definition 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 about 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, who establishes the price that will be paid to the photographer and the price that will be charged to the publishing house for each photo.

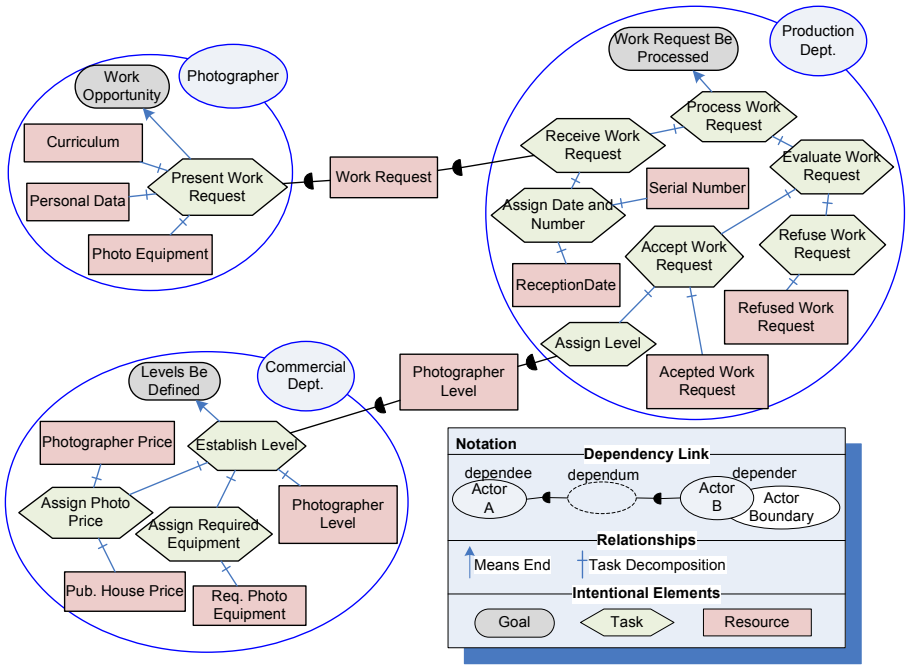

Figure 1. Example *i** SR Model.

In general terms, the presented *i** model shows how 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* are comprised by the photographer's *personal data*. The *production department* is the 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*.

2.3 The OO-Method MDD Approach Overview

OO-Method is an MDD approach that allows the automatic code generation from the conceptual representation of software systems (see Figure 2). This conceptual representation is defined according to the OO-Method Conceptual

Model, 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 the *Presentation Model*.

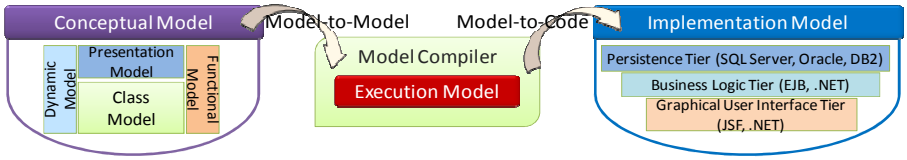

Figure 2. The OO-Method Software Production Process

From the four models that comprise the OO-Method Conceptual Model, the class model is the most important. The other models are defined (or derived) from this central model. For this reason, the OO-Method class model has been considered to evaluate the approach proposed in this paper. More details about the OO-Method approach and its industrial application can be found in [54].

2.4 Related Work

As we can observe in the systematic review about requirement engineering and MDD presented in [38], several approaches (such as [11][35][39]) have encouraged the use of high-level analysis models (i.e., requirement models) as part of a sound MDD process. A representative example is the MDA approach [46], which proposes the definition of a *Computation-Independent Model* as starting point of the development process [43]. However, most of the current requirement approaches are not automatically applied, or are not based on modeling standards [25]. Thus, an effective solution that includes requirement models as part of a complete, standardized, and automatic MDD process [27] is still an unsolved challenge [38].

Probably, one of the main issues to achieve this requirement modeling and MDD linkage is the proper definition of the requirement models for the

automatic generation of domain-specific models [55] related to MDD processes. Most of the proposals oriented to translate requirement models into MDD models (such as [33] and [37]) are considering the input requirement models to be properly defined to perform the translation. We know this idealist scenario is not applicable in practice, and verification mechanisms are necessary to assure the generation of the corresponding MDD models.

To assure the automatic requirement transformation, certain proposals suggest the manual translation of the defined requirement documents to a specific computable format [36][39]. These approaches restrict the flexibility of the original specification, which, together with the manual translation of the requirements, may cause lost of information.

Other approaches suggest to add quantitative information to existent requirement modeling approaches [5][24][52], which allows the automatic measure and analysis of the defined models without restricting their original specification. However, there is a lack of measures to support the verification of requirement models for generation of domain-specific models [55] related to MDD processes. Hence, to fill this gap, we have considered the approaches related to object-oriented models verification [20][58], and definition of measures to verify the correct compilation of domain-specific models [41].

Thus, in this in this paper, we present a systematic approach for the definition of automatic verification measures related to requirement models, which support the automatic generation of MDD-oriented design models. This proposal is based on current modeling standards and it has been developed by considering our experience related to linking *i** and MDD modeling [3][21], definition of *i** measures [17][18], and industrial application of MDD approaches [17].

3 Transforming *i** Models into MDD-Oriented Models

In this section, we briefly introduce our proposal for the transformation of *i** models into MDD models (presented in [53]), which provide the basis for the verification approach defined in this paper.

For the transformation of an *i** model, we assume that it is possible to partially infer an *initial MDD model* from both the information that is represented in the *i** model and from extra information that is added when it is necessary (see Figure 3). This MDD model generation is feasible if there is a mapping from *i** constructs (actors, tasks, etc.) to constructs of the MDD modeling language (e.g., for a class model: classes, attributes, etc.). The abstract syntax of the involved constructs is represented by means of the corresponding metamodels.

It is important to note that we are referring to an initial MDD model and not a complete MDD model because there are aspects related to specific system functionality that cannot be obtained from *i** models. These aspects must be specified later in the refinement of the initial MDD model that is obtained.

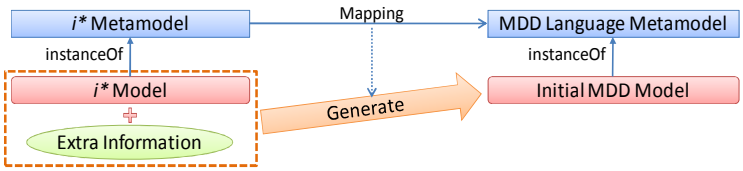

Figure 3. General *i** and MDD transformation schema

The *i** model transformation can be automated by means of well-defined model-to-model transformations by using technologies such as ATL [31] or QVT [48]. For the representation of the extra information that is required, we use light weight extensions, which are defined by means of a process that generates them automatically [22]. This transformation approach has been

applied to a particular MDD proposal, the OO-Method approach [54], but it can be applied to any other MDD approach with minor changes.

Table 1 summarizes a representative subset of transformation guidelines (adapted from [2]) for *i** and OO-Method, which have been selected due to their applicability to other MDD approaches based on class model specification. These guidelines are used to exemplify the proposed verification approach throughout this paper. The rationale of these guidelines has been presented in [2]. Table 1 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 1. Guidelines for the transformation of *i** models into OO-Method class models

i* **Construct**	**Additional Information**	**Class Model Construct**
Actor		Class
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 class 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 class that contain the corresponding service and the classes generated from the decomposed resources
	Inside of an actor boundary	An *agent relationship* between the class generated from the owner actor and the task
Resource Dependency Link		Associations are automatically defined among the class 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

The guidelines presented in Table 1 can be combined, for example, a physical resource that is a *dependum* in a dependency link generates a class, but also, associations between the classes that own the services generated from the involved tasks.

For the transformation guidelines related to tasks and dependency links, when the resource involved corresponds to an informational resource, the rule is applied to the physical resource related to the informational resource. For instance, a task that affects the state of an informational resource is transformed into a service of the class generated from the physical resource that owns the attribute generated from the informational resource.

In the transformation guidelines presented in Table 1 it is possible to observe a specific OO-Method 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 agent relationship to other classes are named *Agents* of the modeled systems. This construct is relevant for the specification of interaction models, such as the OO-Method presentation model (see Figure 2). Even though the agent construct is specific for OO-Method, its semantics can be generalized to other MDD approaches that define system users and interaction aspects at conceptual level.

Thus, for the automatic application of the transformation guidelines it is important to determine if the defined *i** models provide a proper specification, and, hence, the verification of the *i** models becomes necessary.

4 Integration of Verification Measures into the *i** Framework

This section explain the process for the definition and integration of verification measures into the *i** framework. For the elaboration of this

process, we have considered existing standards and modeling technologies to facilitate its application for different MDD approaches. The technologies and standards involved are: approaches for the specification of measures [8][9][26], the last version of the *i** framework [28], approaches for the definition of *i** measures [17][18], OMG Standards for metamodeling [47] and model extensions definition [19], and Eclipse Model Development Tools [13]. The steps of the process are described below (see Figure 4).

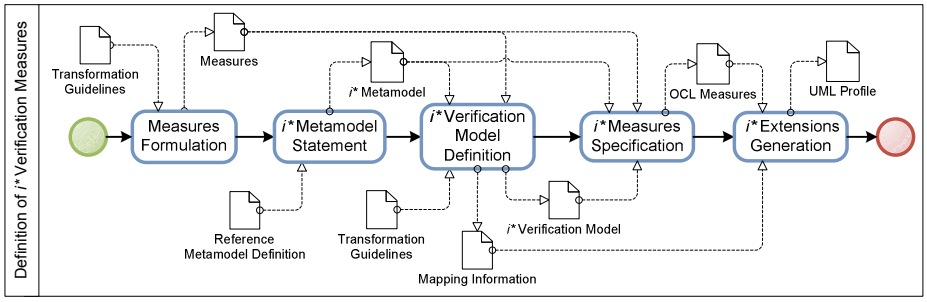

Figure 4. Process for definition of *i** verification measures.

4.1 Step 1: Measures Formulation.

The first step of the process considers the appropriate formulation of the *i** verification measures. This means identifying the *i** constructs that participate in the MDD model generation, and, from these, identifying the aspects that must be verified for a correct *i** model transformation.

To perform the identification of the involved *i** elements, it is necessary to know the transformation guidelines (or rules) related to the target MDD model generation. In particular, we focus on the additional information that must be specified by the analyst to perform the corresponding transformation, which is the critical point that must be verified to assure that the transformation can be performed correctly and automatically. The measures formulation is performed by applying the Goal-Question-Metric (GQM)

approach [8]. Figure 5 shows an excerpt of the application of the GQM approach to the transformation guidelines presented in Table 1.

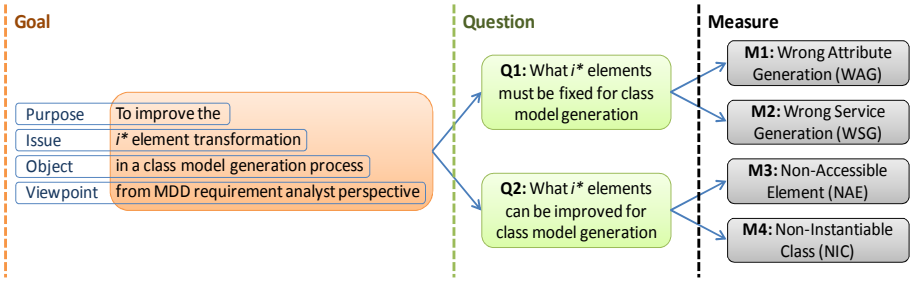

Figure 5. Application of the GQM approach.

For the formulation of the questions related to the GQM approach, we suggest to consider two verification levels. These levels are related to: 1) the *i** elements that must be necessarily fixed because they cannot be transformed or produce a wrong class model generation (i.e., Q1 in Figure 5); and 2) The *i** elements that can be correctly transformed, but they still can be improved to obtain a more complete class model generation (i.e., Q2 in Figure 5).

It is important to consider that the verification measures formulated in this step are specific for the transformation guidelines presented Table 1[2]. 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 guidelines that can be generalized for MDD-oriented class model generation, the resultant measures can provide relevant verification information to other object-oriented MDD approaches.

The measures that are related to answer each of the presented GQM questions are specified by considering the framework presented in [26]. This framework specifies that the empirical world, the numerical world, the

[2] The complete linking framework defined for *i** and OO-Method considers 17 transformation guidelines and 9 verification measures.

measurement method, and the measurement procedure must be defined in the design of measures. In the definition of the empirical world, the entity and the attributes to be measured must be identified. An *entity* corresponds to an input artifact used to perform a measurement, and the concepts related to that artifact correspond to measurable *attributes*. In the definition of the numerical world, the measurement *scale* must be defined. In the specification of the measurement method, the *measurement principle* must be identified. Finally, in the specification of the *measurement procedure*, details of the application of the measurement method must be defined. This measure specification framework is applied to the four measures previously formulated.

M1. Wrong Attribute Generation (WAG).

Rationale. The informational resources are involved in the generation of class attributes (see Table 1). Therefore, for the correct generation of informational resources, they must be related to a system entity (actor or a physical resource), which is transformed into a class in the class model. Otherwise, it is impossible to transform these resources into attributes because the class that contains them cannot be identified. Table 2 details the characteristics related to this measure according to the considered definition framework.

Table 2. Characteristics of measure Wrong Attribute Generation (WAG)

Characteristic	Definition
Measurement Entity	*i** model
Measurement Scale	Ratio scale
Attribute to be measured	Informational resources not related to a physical resource nor to an actor
Measurement principle	An informational resource that is not related to a physical resource nor to an actor is directly proportional to a wrong attribute generation in the MDD model
Measurement procedure	The attributes to be measured must be counted to obtain the number of informational resources that cannot be transformed into attributes

Following, the formula to obtain the measure M1 – WAG is presented:

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

M2. Wrong Service Generation (WSG). *Rationale.* According to the transformation guidelines, the tasks that do not generate entities (physical resources or actors) or that do not affect resources are transformed into services of the class generated from the owner actor (according to the corresponding actor boundary). Therefore, if the corresponding actor is not marked for the generation of the intended system, the involved task cannot be transformed since it is not possible to generate a service in the class model without a class that contains it. See WSG characteristics in Table 3.

Table 3. Characteristics of measure Wrong Service Generation (WSG)

Characteristic	Definition
Measurement Entity	*i** model
Measurement Scale	Ratio scale
Attribute to be measured	Tasks that not generate entities nor affect resources and the related actor is not marked for the generation of the intended system
Measurement principle	A task that not generates entities nor affects resources and it is related to an actor not marked for the generation of the intended system is directly proportional to wrong service generation in the generated MDD model
Measurement procedure	the attributes to be measured must be counted to obtain the number of wrong services specified in the generated MDD model

Following, the formula to obtain the measure M2 – WSG is presented:

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

M3. Non-Accessible Element (NAE). *Rationale.* According to the presented transformation guidelines (see Table 1), agent relationships are defined between the classes generated from actors and the elements generated from services or informational resources contained in the corresponding actor boundaries. However, if the involved actors are not selected for the MDD model generation, the actor is not transformed in a class, and, hence, the

involved agent relationships are not defined. This produces that the transformed tasks or informational resources (that are inside of the actor boundary) cannot be executed or visualized in the final application. 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. See NAE characteristics in Table 4.

Table 4. Characteristics of measure Non-Accessible Element (NAE)

Characteristic	Definition
Measurement Entity	*i** model
Measurement Scale	Ratio scale
Attribute to be measured	Internal tasks or resources related to the system that are defined in the boundary of an actor that is not related to the system
Measurement principle	An internal task or resource related to the system that is defined in the boundary of an actor that is not related to the system is directly proportional to the a non-accessible element in the generated MDD model
Measurement procedure	the attributes to be measured must be counted to obtain the number of non-accessible elements in the generated MDD model

Following, the formula to obtain the measure M3 – NAE is presented:

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

M4. Non-Instantiable Class (NIC). *Rationale.* The system entities (physical resources or actors) without a production task related are transformed into 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 production task for entities (actors or physical resources) is not mandatory since this issue

does not prevent the appropriate transformation of the *i** elements. Thus, specific instance-creation services can be defined at design time for the classes generated without this kind of services. See NIC characteristics in Table 5.

Table 5. Characteristics of measure Non-Instantiable Class (NIC)

Characteristic	Definition
Measurement Entity	*i** model
Measurement Scale	Ratio scale
Attribute to be measured	Actors and physical resources without a task related to their production
Measurement principle	An actor or a physical resource without a related production task is directly proportional to a non-instantiable class in the generated MDD model.
Measurement procedure	The attributes to be measured must be counted to obtain the number of non-instantiable classes

Following, the formula to obtain the measure M4 – NIC is presented:

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

4.2 Step 2: *i** Metamodel Statement

The second step corresponds to stating the target *i** metamodel, which must be defined according to the EMOF specification [47]. The use of EMOF is mandatory for the appropriate application of the considered approach for modeling language integration [22]. Therefore, we have defined the EMOF *i** Metamodel presented in Figure 6. This figure only shows the structural representation of the metamodel. Additional features such as derived values, constraints, and operations are not necessary for the application of the proposed verification approach, and, hence, they have been omitted to simplify the metamodel representation.

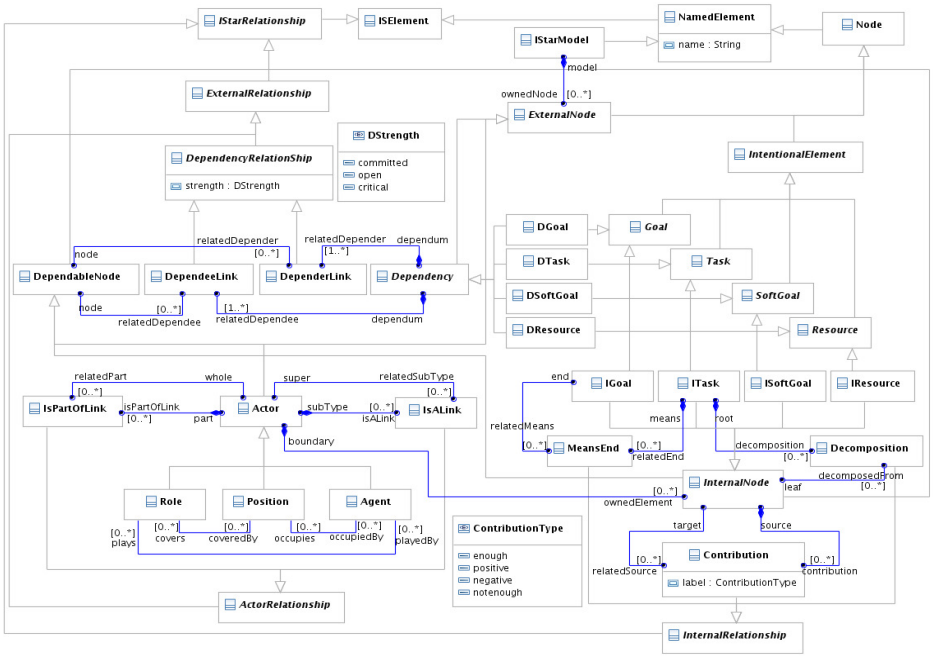

Figure 6. EMOF *i** Metamodel

For the elaboration of this *i** metamodel, the proposals presented in [7], [16], and [40] were considered. It also has been considered the specification presented in the *i** guide v3.0 [1], and the metamodel related to the User Requirements Notation (URN) [4], which is a variant of the *i** Framework.

4.3 Step 3: *i** Verification Model Definition

The third step of the process consists in the definition of a verification model. This is an EMOF model that includes the information required for the correct application of the measures (see Figure 7).

The verification model must include those elements that are not present in the reference *i** metamodel, which are also relevant for the correct generation of the corresponding MDD class models according to the transformation guidelines presented in Table 1.

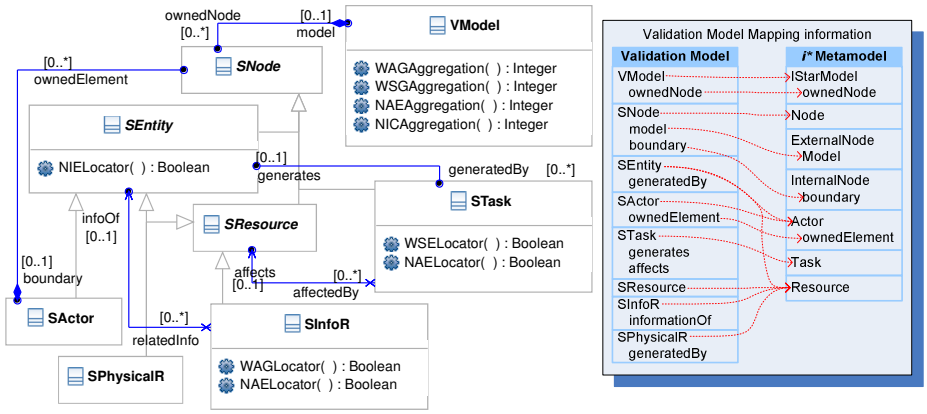

Figure 7. Verification Model and Mapping Information

Figure 7 also shows the mapping information that indicates the correspondences among the elements of the verification model and the *i** metamodel.

4.4 Step 4: *i** Measures Specification

The fourth step of the process corresponds to the OCL specification of the measures, which must be included in the verification model. This specification is performed by considering the modeling information that is contained in the verification model. Figure 7 shows the names and outputs of the different OCL rules defined. For the measure specification, we have applied the measure patterns presented in [18], 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. For instance, the OCL definition of the measure WAG (Wrong Attribute Generation) is comprised by two OCL rules (see Table 6), these are: the 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.

Table 6. 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()
```

In addition, since the proposed measures have been defined for verification purposes, we have introduced a new property in the measure specification, which corresponds to the alert levels that are related to the defined measures. These levels are: 1) Critical, which indicates that the situation identified by the measure prevents the transformation of the corresponding *i** elements; and 2) Warning, which indicates that there is a modeling issue that can be fixed to improve the class model generated. These alert levels are derived from the separation proposed for the definition of the questions related to the GQM application (see Step 1 of this section). Thus, WAG and WSG measures have a *critical* level, and NAE and NIE measures have a *warning* level.

4.5 Step 5: *i** Extensions Generation.

Finally, in the fifth step of the process, the verification model and the OCL specification of the measures are used to generate the metamodel extensions that are necessary to integrate the proposed measures into the *i** framework. These extensions are implemented in a UML profile (see Figure 8), which is

generated by means of the proposals presented in [22] and [23]. In [23] is presented an approach for the adaptation of metamodels for generation of UML profiles, and [22] defines a set of transformation rules for automatic UML profile generation. These proposals use the mapping information presented in Figure 7.

In general terms, the UML profile generation consists in the generation of one stereotype for each class of the verification model, and the definition of one tagged value for each property (attribute or association end) that has not correspondence in the target *i** metamodel (non-mapped properties). In particular, for those abstract classes that have the child classes mapped to different classes of the *i** metamodel, only the extensions related to the child classes are represented (stereotype *Actor*). Otherwise, if the abstract and the child classes are mapped to the same class in the *i** metamodel, only the extension of the concrete class is represented (stereotype *SResource*) and the extensions related to the child classes are omitted (stereotypes *SPhysicalR* and *SInfoR*). Additionally, the abstract stereotype *SNode* is not represented since it does not introduces new properties or operations into the *i** metamodel.

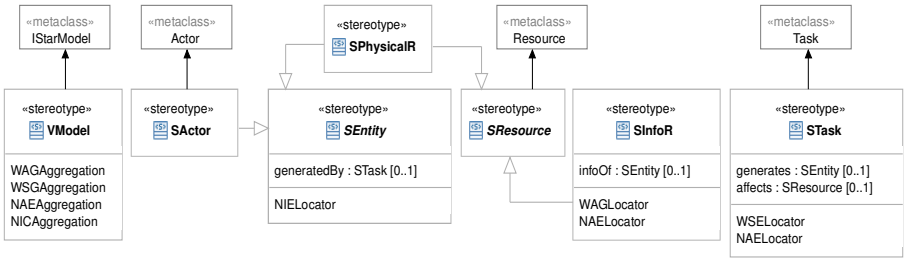

Figure 8. UML Profile to extend the *i** metamodel with the verification measures

The UML profile is a lightweight extension mechanism that does not change the target metamodel, and it has a standardized definition [49] and interchange format [51]. 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. [6]).

In the generated UML profile, the elements of the OCL specification must be changed according to the mapped elements of the *i** metamodel (see Figure 7), and the generated stereotypes and tagged values. For instance, the specification for measure WAG (Wrong Attributes Generation) is finally defined as follows:

```
Context: VModel::WAGAggregation() : Integer
Body: result = self.ownedNode->select(irs|irs.isStereotyped(SInfoR))
               .oclAsType(Resource)->select(irs|irs.WAGLocator())->size() +
               self.ownedNode->select(act|act.oclIsKindOf(Actor)).oclAsType(Actor)
               .ownedElement->select(irs|irs.isStereotyped(SInfoR))
               .oclAsType(Resource)->select(irs|irs.WAGLocator())->size()

Context: SInfoR::WAGLocator() : Boolean
Body: result = self.infoOf->isEmpty()
```

It is important to mention that the OCL operation *isStereotype* is not part of the OMG specification and it must be defined or implemented according to the OCL interpreter used. For instance, in ATL this operation can be implemented as follows:

```
helper context UML!Element def : isStereotyped(name:String):Boolean =
not self.getAppliedStereotypes()->select(s|s.name=name)->isEmpty();
```

Finally, for the definition *i** models extended with the generated UML profile, we have used the eclipse UML2 project. Thus, we take advantage of the already implemented support for UML profiles that this tool provides. However, since there is a dependency between the UML profile application and the UML metamodel, we need to introduce a little adaptation to the defined *i** metamodel in order to use the implemented UML profile extension capabilities. This adaptation consists in the definition of two generalization relationships from the classes *IsModel* and *IsElement* of the *i** metamodel to the classes *Model* and *Element* of the UML metamodel, respectively.

5 Applying the *i** Verification Measures

This section exemplifies how the proposed *i** measures are used to verify and improve the generation of the corresponding class model. The process to apply the verification measures is presented in Figure 9.

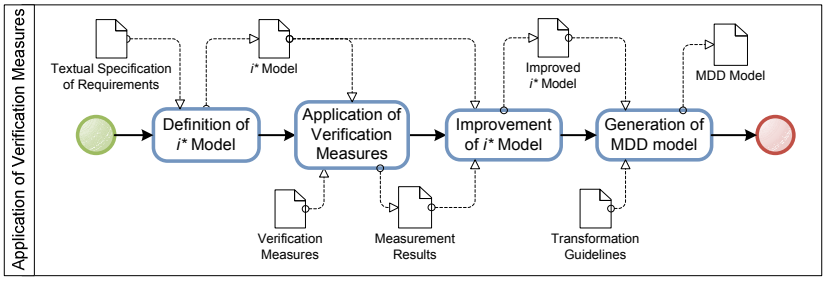

Figure 9. Process for the application of verification measures.

To specify the *i** models, the corresponding EMF editor has been generated by using the *i** metamodel that has been defined as reference (implemented with the Eclipse UML2 tool).

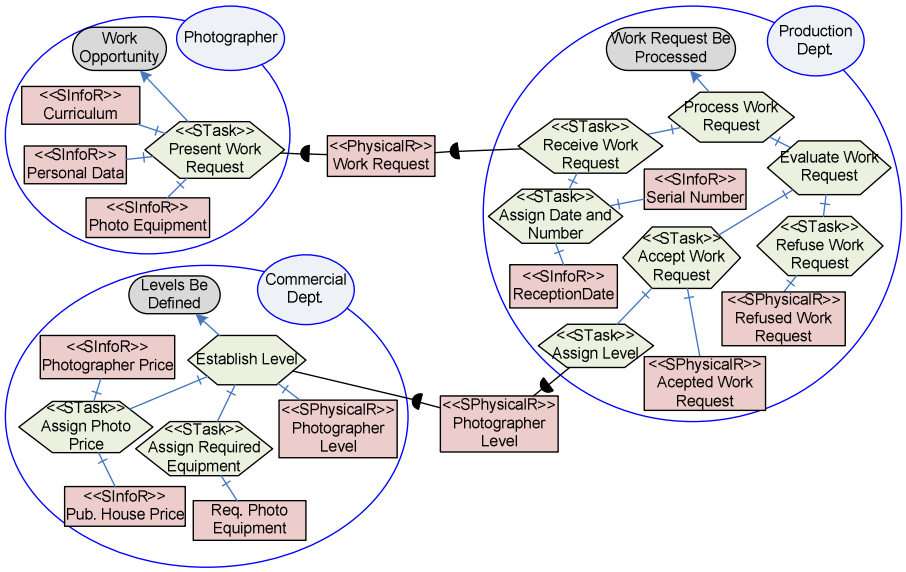

Figure 10. Example *i** Model extended with the generated UML Profile.

In order to improve the understanding of the *i** models presented in this paper, the pictures of these models corresponds to manual transcriptions of the defined EMF models using the *i** notation. Therefore, the example *i** model presented in Figure 1 has been extended with the information that is required for the automatic measures application. Figure 10 shows the *i** model extended with the generated UML profile.

Only those *i** elements related to the intended system are considered in the transformation process. These elements are the stereotyped elements. Table 7 shows the values related to the tagged values of each stereotyped element.

Table 7. Tagged values related to the example *i** Model

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

Table 8 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* for evaluation of locator OCLs.

Table 8. Results obtained from measures evaluation.

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

Figure 11 shows the class model that may be generated (applying the transformation guidelines presented in Table 1) from the example *i** without considering the information reported by the verification measures.

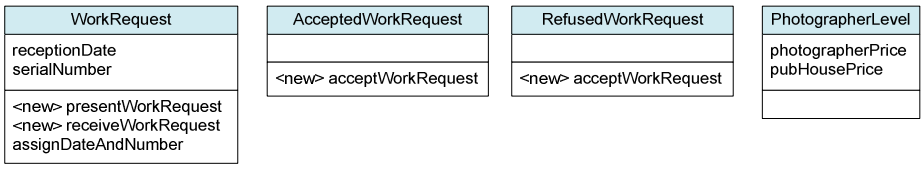

Figure 11. Class model generated from the example *i** model

In Figure 11 can be observed that those elements identified by the critical measures are not present, such as the resource *Curriculum* or the task *Assign Photo Price*. Therefore, it is necessary to improve the defined *i** model in order to assure the transformation of all the selected *i** elements, i.e.; to fix the issues related to elements identified by critical measures.

5.1. Improving the *i** Models for MDD Model Generation

The results obtained from the measures application provide useful information to fix the detected modeling issues. Thus, it is possible to identify specific

fixing guidelines for each measure formulated. For the four measures defined, the alternative guidelines presented in Table 9 have been inferred.

Table 9. Fixing guidelines related to the verification measures

Measure	**Wrong Attribute Generation (WAG)**
Guidelines	Associate the informational resources to a system entity (stereotyped actor or physical resource).
	Change the kind of the informational resource to *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 affect 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 *physical resource.*
Measure	**Non-Instantiable Class (NIC)**
Guidelines	Define a new task in the model as production task of the involved entity (stereotyped resource or physical resource).
	Indicate a task that is already defined in the model as production task of the entity (stereotyped resource or physical resource).
	Change the physical resource to *informational resource.*

In addition to the guidelines presented, it is also possible to remove the corresponding element from the intended system (i.e.; remove the stereotype), or even remove the element from the *i** model.

However, independently of the guidelines that can be derived from the different verification measures, this information is just a reference and the analyst is who must decide the guidelines to apply to improve the *i** model. Figure 12 shows the *i** model improved by the analyst after analyzing the results obtained from the application of the verification measures.

In the improved *i** model, the task *Assign Level* affects the state of the new defined actor *Accepted Photographer*. 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 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*.

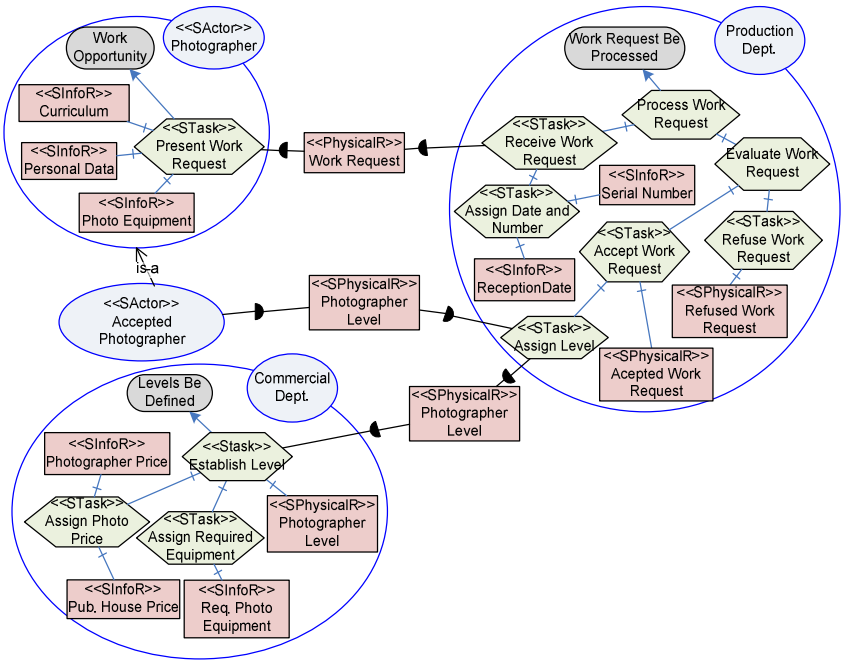

Figure 12. Improved *i** model

The informational resources located by the WAG measure are now defined as information of the actor *Photographer*. The warning related to the NIE measure has been solved by defining the task *Establish Level* as a generation task for the resource *Photographer Level*. Table 10 shows the tagged values that have been changed in the improved *i** model.

It is important to note that solving the issues identified by the verification measures, the stereotyped elements are properly performed, but also, an improved and detailer requirement representation is obtained. Figure 13 shows the class model generated from the improved *i** model.

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

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	--		

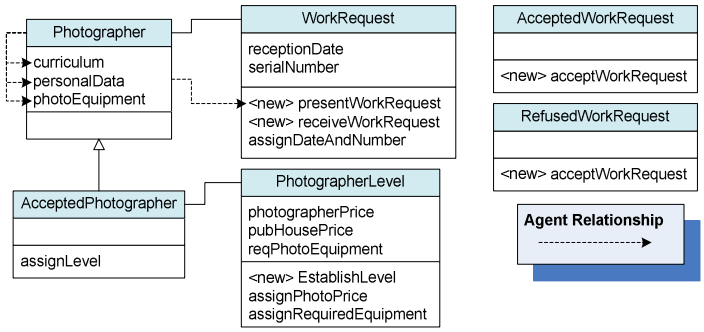

Figure 13. Class model generated from the improved *i** model

Figure 13 shows that the class model generated from the improved *i** model has a more detailed system specification. Essential elements generated from the improved *i** model are the classes *Photographer* and *AcceptedPhotographer*. 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 MDD model represents all the system requirements considered.

Since the generated class model is an initial MDD model, it must be refined at design time. 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.

In the improved *i** model, there still exist warning issues related to the NAE measure. These warning issues are derived from the stereotyped resources and tasks that are defined inside of the boundaries of the actors *Production Dept.* and *Commercial Dept.*, which are not considered as part of the system-to-be by the analyst. In this case, the analyst has considered that an administrator user must be specified at design time to visualize and execute the corresponding resources and tasks. However, since the NAE measure is just a warning measure, it does not prevent the correct transformation of the *i** model. It just indicates that the MDD constructs obtained from identified elements must be refined at design time to obtain a complete specification.

6. Evaluating the Verification Approach

To evaluate the verification approach, we have focused on the efficacy of the measures obtained with the proposed definition process to achieve the completeness of the generated MDD model.

The ISO 9126 standard [30] 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 supports. Thus, the first kind of completeness is related to the completeness of the MDD models 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 initial MDD model regarding to the functionality of the software system, i.e.; the completeness of the MDD model to perform the automatic model compilation.

The evaluation of our verification approach has been conducted as a laboratory experiment, which has been designed using the framework proposed by Wohlin et al. [59] for Empirical Software Engineering. The research question addressed by the experiment is stated as:

RQ1: Is the completeness of the generated MDD model supported by the measures obtained from the application of the proposed verification approach?

The rest of this section provides details of the design of the laboratory experiment, as well as the results obtained from the experiment execution.

6.1. 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 students from the Universidad Politécnica de Valencia, which have similar backgrounds in the *i** framework and the OO-Method MDD approach.

The independent variables in the experiment correspond to the Photography Agency *i** models, which have been defined by the *i** analysts. The first *i** model (called ISTAR1) is already detailed in Section 5 (see Figure 10). The details about the second *i** model (called ISTAR2) are presented in Appendix I, which contains the model diagram, tagged values specification, and class model generation obtained during the experiment execution.

The quantitative dependent variables considered in the experiment are the following:

a) Number of informational resources that cannot generate the corresponding class attributes in the initial MDD model. Obtained from WAG measure.

b) Number of tasks that cannot generate the corresponding service definitions in the initial MDD model. Obtained from WSG measure.
c) Number of tasks and informational resource that generate non-accessible elements in the initial MDD model. Obtained from NAE measure.
d) Number of actors and physical resources that generate non-instantiable classes in the initial MDD model. Obtained from NIC measure.

Thus, in order to answer our research question, we consider the following hypotheses related to the critical measures and the 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 MDD conceptual 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 initial MDD model, which represents the functionality of the final software product.

To test H_{RCOM}, each *i** element related to the intended system (the stereotyped elements in the extended *i** model) must have a direct relation with the constructs generated in the initial MDD model (the OO-Method class model in the experiment).

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

6.2. Instruments and Experimental Tasks

Figure 14 shows the tasks performed in the experiment, which are modeled with the BPMN notation [45]. These tasks are described below.

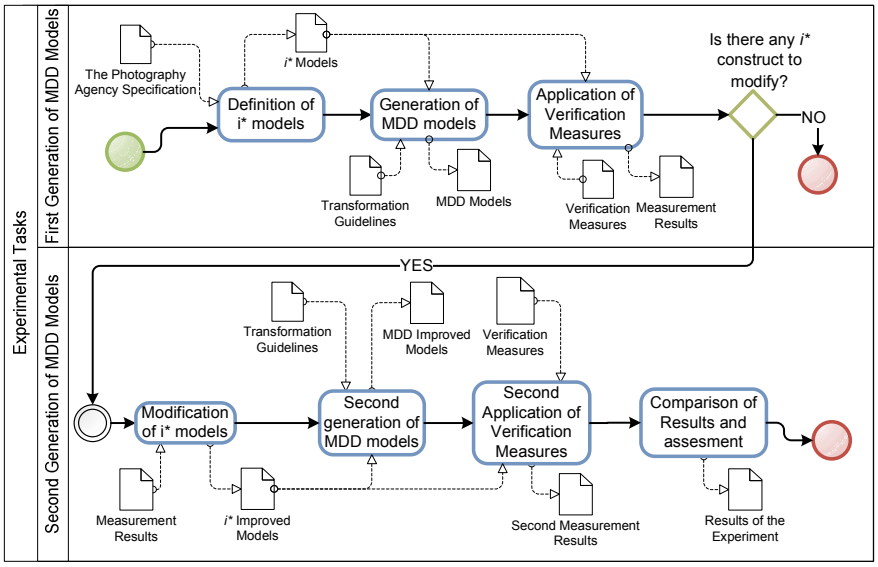

Figure 14. Experimental tasks

Task 1. ***Definition of i* models.*** Each one of the two involved *i** models (see Figure 10 and Figure 15) is defined by one *i** analyst, which considers the problem statement presented Section 2.2. The defined *i** models include the information that is required for the automatic application of the transformation guidelines, as described in Section 3.

Task 2. ***Generation of MDD models.*** Each measurement expert performs a class model generation from one of the defined *i** models by applying an ATL transformation script. The verification measures are not applied in this task.

Task 3. ***Application of verification measures.*** Each measurement expert applies the verification measures over its corresponding *i** model.

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

Task 5. ***Second generation of MDD models.*** The measurement experts generate a new class model from the improved version of the *i** model that they have transformed in Task 2.

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

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.

In addition to these sequential tasks, the measurement experts controlled all the experiment execution.

Regarding the instruments, in addition to the models themselves, the instruments used in the experiment were: the Eclipse Model Development Tools [13], the EMF editor for the *i** metamodel extended with the UML profile related to the verification measures and transformation extensions (see Figure 8), the ATL scripts to transform the *i** models to MDD models according to the transformation guidelines presented in Table 1, and tables filled according to a predefined template to keep the results of the experiment.

6.3. Execution and Data Collection Procedures

To perform the experiment, the *i** analysts were located in separated rooms to avoid any kind of influence on each other's results. The *i** models were defined manually to prevent that the use of the EMF editor (that does not provide *i** notation) affects to the appropriate analysis of the business. In tasks 2 and 5, the measurement experts translate the hand-made *i** models with the corresponding EMF representation in order to apply the verification measures and to generate the corresponding MDD models automatically. No time limit was set for any experimental tasks, such as the EMF models specification, the generation of the corresponding MDD models, the application of the verification measures, the improvement of the *i** models, etc.

In this study, data triangulation was considered (which refers to using more than one data source or collecting the same data on different occasions) since

we used two sources (the two *i** models). Thus, the following common steps were defined to collect data from the two *i** models in the study:

1. Each *i** model was transformed into an OO-Method class model by measurement experts applying an automatic transformation process.
2. Work diaries were completed by each measurement expert for each model transformed. In those diaries, the information obtained from the verification measures was registered.

6.4. Results: Analysis and Interpretation Issues

In the first generation of the MDD models (see Task 2 in Section 6.2), the resultant models suffered from several defects related to their completeness. Table 11 shows the amount of constructs of the *i** models that must transformed, which correspond to the stereotyped elements. Also, Table 11 shows the amount of effectively transformed elements. Thus, it is clear that not all the stereotyped elements were transformed into constructs of the class models, i.e.; MDD models are not complete regarding to the requirements.

Table 11. First generation of the MDD models.

*i** Model	Stereotyped Elements	Transformed Elements	MDD Model	MDD Elements
ISTAR1	19	13	MODEL1	4 classes, 4 attributes, 5 services. (Total = 13)
ISTAR2	23	17	MODEL2	5 classes, 6 attributes, 6 services, 1 association, 3 agent relationships, 1 generalization. (Total = 22)

Then, EXP1 and EXP2 apply the verification measures to the defined *i** models. The results obtained are presented in Table 12, which shows for each *i** model the measures applied, the alert level of the measures, the result of the measure obtained from the evaluation of *aggregation* OCLs, and the *i** elements that return *true* from the evaluation of *locator* OCLs.

Table 12. Application of the verification measures to ISTAR1 and ISTAR2

Model	Measure	Alert Level	Measurement Result	Locator
ISTAR1	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
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

It is important to point that the verification measures can be used to define additional measures to obtain relevant information of the defined i^* models. For instance, we have defined the following measure to obtain information about of the completeness of the MDD 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 are transformed in elements of the target MDD model. PTE is calculated using 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. Thus, for ISTAR1, WAG = 3 and WSG = 3. For ISTAR 2, WAG = 5 and WSG = 1 (see Table 12). Note that the sum of the critical measures (WAG and WSG) is coincident with the difference of transformed *i** elements (see Table 11).

Thus, for ISTAR1 we obtain PTE = 68,4 %. It means that only the 68,4% of the *i** stereotyped elements can be transformed in the MDD model generation. i.e. 31,6% of elements related to the system requirements will not be represented in the software model. For ISTAR2, we obtain PTE = 73,9 %.

The warning measures support the identification of those *i** elements that can be improved to obtain a more complete specification of the initial MDD model generated. Thus, we have defined the following measure to identify this situation in the improved *i** models:

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

The measure *Warning Improvement Percentage* (WIP) returns the percentage of new MDD constructs of the improved MDD models obtained with the information of the warning measures. WIP is calculated using IMDD, OMDD, WAG, and WSG.

IMDD (Improved MDD) corresponds to the number of MDD constructs generated from the Improved *i** model. OMDD (Original MDD) corresponds to the number of MDD constructs generated from the original *i** model. WAG and WSG correspond to the critical measures previously defined. In the experiment, the WIP measure is evaluated after the second generation of MDD models.

The next task in the experiment (Task 4) corresponds to improve the *i** models using the information obtained from the verification measures WAG,

WSG, NAE, and NIE. Following, we enumerate the improvements performed by ANA1 to the model ISTAR1 (see Section 5.1):

(1) 1 actor, 1 is-a relationship, and 1 task were added to the *i** model.

(2) 2 actors were added to the system requirements.

(3) 4 resources and 4 tasks were modified in the *i** model.

The same cognitive process explained in Section 5 to improve the ISTAR1 model is applied by ANA2 to obtain the improved ISTAR2 model (see Appendix I). The improvement actions performed were the following:

(1) 2 actors were added to the system requirements

(2) 1 goal, and 1 resource were added to the *i** model.

(3) 2 tasks and 6 resources were modified in the *i** model.

(4) 1 stereotype application was changed to physical resource in the *i** model.

Table 13 shows the results obtained from the transformation of the improved versions of the models ISTAR1 and ISTAR2.

Table 13. Second generation of the MDD models.

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

0 shows the results obtained from the application of the verification measures to the improved *i** models. For ISTAR1, the measures WAG, WSG, and NIC are equal to 0, which means that the improved model ISTAR1 generates correctly attributes, services, and instantiable entities. Only NAE was greater to zero (NAE=16), which means that 16 elements of the generated MDD model (MODEL1) do not have agent relationships defined.

Table 14. 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%

For ISTAR2, the measures WAG, WSG and NAE are equal to 0, which means that that the improved model ISTAR2 generates attributes, services, and accessible elements correctly. Only NIC was greater to 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 application of the resource *Proceeding Manual*, which is defined now as a physical resource. However, this warning measure does not prevent the proper generation of the MDD model (MODEL2).

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 MDD 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 MDD constructs (see PTE measure in 0). Therefore, we can state that the Hypothesis H_{RCOM} is demonstrated.

Also, the experiment results shows that fixing the issues identified by the warning measures (NAE and NIC), the completeness of the MDD models in relation to the system functionality is higher. This is observed in the amount

of MDD constructs generated from the improved *i** models in relation to the original *i** models (see WIP measure in 0). For ISTAR1, we obtain that the improvements performed from warning measures generate 84,6% of additional MDD constructs. For ISTAR2, we obtain that the improvements related to warning measures increase the number of generated MDD constructs in 77,3%. Thus, since MDD constructs are directly representing functionality of the final software system (such as system users), the hypothesis H_{CCOM} is also demonstrated with the results obtained.

7. Overall Analysis

A first element to analyze is the relevance of using a measure definition process as 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 found sound frameworks for the definition and implementation of measures. This has been considered for the definition of the systematic schema proposed for the appropriate identification of properties that must be measured and, in the context of this paper, verified. It also assures the theoretical validity of the defined measures according to metrology concepts [29], which are designed independently of implementation platforms. Additionally, as we can observe in the application and evaluation sections (sections 5.1 and 6.4 respectively), the verification measures can be used to infer fixing guidelines to the defined *i** models, but also, to perform different analyses at early stages of the development process. These are clear advantages of the proposed verification approach regarding to other mechanisms for defect detection [41].

Another relevant point is related to the benefits that the approach proposed for the integration of the verification measures in the *i** framework provides. One of the main advantages of this integration 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 model by using current metamodeling standards. 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 the compatibility with existent technologies that use the same metamodel as reference. Also, we have considered mechanisms to automate the generation of the extensions. With this, the main effort in the application of the verification proposal is in the appropriate definition of the required measures, the correct specification of the involved properties and OCL rules in the verification model, and the definition of the mapping between the verification model and the target *i** metamodel. Thus, once the verification framework is defined for a particular MDD approach (such as OO-Method); it can be used with no or little modification over and over in different projects.

There is an important aspect to be considered by MDD practitioner who be interested in to put in practice and improve the proposal presented. It is the complexity to determine which elements must be maintained at design level, and which must be up scaled to the analysis level. For instance, it is possible to introduce an extension to identify 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 to identify new extension for requirement models without affecting the clarity of the business analysis. Evidently, the inclusion of new extensions also implies the definition of new verification measures.

Finally, it is important to mention that for the application of our verification proposal, we did not find tools that provided transparent support for all the modeling features considered, and, hence, additional programming effort was necessary. However, the current development of tools that provide support to the standards considered (such as the Eclipse UML2 project [14]) and the increasing number of research works that take advantage of these technologies (such as the *Moskitt Project* [44] developed in the context of our research group) are indicators that improved tools will appear in a not-too-distant future. This also motivates the emergence of new approaches for integration and verification to improve the MDD capabilities and the quality of the generated software products.

8. Conclusions and Further Work

The integration of *i** models for requirements elicitation in MDD process is a step beyond going from Model-Driven Development (MDD) to Model-Driven Engineering (MDE) [32], where different modeling approaches can be integrated to obtain improved software products [53]. This paper has presented new results in this direction by introducing a process for the definition and integration of verification measures into the *i** framework. The main advantages of the proposed verification approach are:

1) The verification measures assure the completeness of the generated MDD models in relation to the requirements and improve the completeness in relation to the system funtionality.

2) The definition of the verification measures is driven by a systematic process that supports the correct identification of the elements to be verified.

3) The evaluation of the verification measures is automatic, which implies a time and effort reduction with respect to manual verifications.

4) Specific alert levels are defined to distinguish the *i** constructs that must be fixed from the *i** constructs that can be improved.
5) The definition and implementation of the verification measures is performed by using current open-source technologies and standards.

Thus, using our proposal, the defined analysis models are not just documentation artifacts; they also play an active role in the development process by providing an entry point for the definition of the necessary MDD models. Additionally, according to the results obtained from the *i** and OO-Method integration, the use of *i** models facilitates the refinement of design models, providing a clear vision of the purpose of the modeled systems.

We are aware that additional evaluation of our proposal to real development scenarios is necessary. Therefore, we consider as future work the development of more empirical studies to validate the effectiveness of our approach and to obtain results about the benefits of using *i** models in real MDD processes. Additionally, we have planed to publish the complete suite of transformation guidelines and verification measures defined for *i** and OO-Method, which can be used to as reference by different MDD approaches. Also, the *i** softgoals and their role in generating non-functional requirements to be considered in the MDD process are subject of ongoing work.

Acknowledgments. This work has been developed with the support of MEC and GVA under the projects ADICT TIN2007-64753 and ORCA PROMETEO/2009/015 respectively. Also, it is partially supported by CAPES.

References

1. Abdulhadi, S.: *i** Guide version 3.0 (2007)
2. Alencar, F., Marín, B., Giachetti, G., Pastor, O., Castro, J., Pimentel, J.H.: From i* Requirements Models to Conceptual Models of a Model Driven Development Process. In:

2nd Working Conference on The Practice of Enterprise Modeling (PoEM 2009), vol. LNIBP 39, pp. 99–114. Springer (2009)
3. Alencar, F.M.R., Pastor, O., Marín, B., Giachetti, G., Castro, J.: Aligning Goal-Oriented Requirements Engineering and Model-Driven Development. In: 11th International Conference on Enterprise Information Systems (ICEIS), pp. 347–350 (2009)
4. Amyot, D.: New draft Recommendation Z.151: User Requirements Notation (URN) (2008)
5. Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., Yu, E.: Evaluating goal models within the goal-oriented requirement language. To appear In: International Journal of Intelligent Systems (IJIS) (2010)
6. Amyot, D., Horkoff, J., Gross, D., Mussbacher, G.: A Lightweight GRL Profile for i* Modeling. In: Third International Workshop on Requirements, Intentions and Goals in Conceptual Modeling (RIGIM) - ER Workshops, vol. LNCS 5833, pp. 254–264. Springer-Verlag (2009)
7. Ayala, C., Cares, C., Carvallo, J.P., Grau, G., Haya, M., Salazar, G., Franch, X., Mayol, E., Quer, C.: A Comparative Analysis of i*-Based Goal-Oriented Modelling Languages. In: International Workshop on Agent-Oriented Software Development Methodologies (AOSDM'05), at the SEKE Conference, pp. 657–663 (2005)
8. Basili, V., Caldeira, G., Rombach, H.D.: The Goal Question Metric Approach. Encyclopedia of Software Engineering, Wiley (1994)
9. Basili, V., Rombach, H.: The TAME Project: Towards Improvement Oriented Software Environments. IEEE Transactions on Software Engineering, vol. 14 nº 6, 758–773 (1988)
10. Boehm, B.W.: Software Engineering Economics, isbn: (10) 0138221227; (13) 978-0138221225. Prentice-Hall Inc., Englewood Cliff, New Jersey (1981)
11. Cabot, J., Yu, E.: Improving Requirements Specifications in Model-Driven Development Processes. In: 1st Int. Workshop on Challenges in Model-Driven Software Engineering (MoDELS'08) (2008)
12. Dardenne, A., Van Lamsweerde, A., Fickas, S.: Goal-Directed Requirements Acquisition. Science of Computer Programming, vol. 20 nº 1-2, 3–50 (1993)
13. Eclipse: Model Development Tools Project, http://www.eclipse.org/modeling/mdt/
14. Eclipse: UML2 Project, http://www.eclipse.org/uml2/
15. Eric Yu, P.G., Neil Maiden and John Mylopoulos: Social Modeling for Requirements Engineering, isbn: (10) 0-262-24055-6; (13) 978-0-262-24055-0 (2011)
16. Franch, X.: Incorporating Modules into the i* Framework. In: 22nd International Conference on Advanced Information Systems (CAiSE 2010), vol. LNCS 6051, pp. 439–454. Springer-Verlag Berlín Heidelberg (2010)
17. Franch, X.: A Method for the Definition of Metrics over i* Models. In: 21st International Conference on Advanced Information Systems (CAiSE 2009), pp. 201-215. Springer-Verlag LNCS (2009)
18. Franch, X., Grau, G.: Towards a Catalogue of Patterns for Defining Metrics over i* Models. In: 20th International Conference on Advanced Information Systems (CAiSE 2008). LNCS, pp. 197–212. Springer (2008)
19. Fuentes-Fernández, L., Vallecillo, A.: An Introduction to UML Profiles. In: The European Journal for the Informatics Professional (UPGRADE), vol. 5 nº 2, 5–13 (2004)
20. Genero, M., Piattini, M., Calero, C.: A Survey of Metrics for UML Class Diagrams. Journal of Object Technology, vol. 4 nº 9 (2005)
21. Giachetti, G., Alencar, F., Marín, B., Pastor, O., Castro, J.: Beyond Requirements: An Approach to Integrate i* and Model-Driven Development. In: XIII Conferencia Iberoamericana en Software Engineering (CIbSE 2010) (2010)
22. Giachetti, G., Marín, B., Pastor, O.: Using UML as a Domain-Specific Modeling Language: A Proposal for Automatic Generation of UML Profiles. In: 21st International Conference on Advanced Information Systems (CAiSE 2009), vol. LNCS 5565, pp. 110–124. Springer (2009)

23. Giachetti, G., Valverde, F., Pastor, O.: Improving Automatic UML2 Profile Generation for MDA Industrial Development. In: 4th International Workshop on Foundations and Practices of UML (FP-UML) – ER Workshop, vol. LNCS 5232, pp. 113–122. Springer (2008)
24. Giorgini, P., Rizzi, S., Garzetti, M.: Goal-oriented Requirement Analysis for Data Warehouse Design. In: 8th Int. Workshop on Data Warehousing and OLAP, pp. 47–56. ACM Press (2005)
25. Gross, D., Yu, E.: From Non-Functional Requirements to Design through Patterns. Requirements Engineering Journal, vol. 6, 18–36 (2001)
26. Habra, N., Abran, A., Lopez, M., Sellami, A.: A framework for the design and verification of software measurement methods. Journal of Systems and Software, vol. 81 nº 5, 633–648 (2008)
27. Hailpern, B., Tarr, P.: Model-driven development: The good, the bad, and the ugly. IBM Systems Journal, vol. 45 nº 3 (2006)
28. i*: Wiki Web Page, http://istar.rwth-aachen.de/, last Accessed October 2009
29. ISO: International vocabulary of basic and general terms in metrology (VIM), International Organization for Standarization, Geneva, Switzerland (2004)
30. ISO/IEC: ISO/IEC 9126-1, Software Eng. – Product Quality – Part 1: Quality model (2001)
31. Jouault, F., Kurtev, I.: Transforming Models with ATL. In: Satellite Events at the MoDELS 2005 Conference, vol. LNCS 3844, pp. 128–138 Springer (2006)
32. Kent, S.: Model Driven Engineering. In: Integrated Formal Methods (IFM), pp. 286–298. Springer (2002)
33. Laguna, M.A., Gonzalez-Baixauli, B.: Requirements variability models: metamodel based transformations. In: Symposia on Metainformatics (MIS ’05). ACM (2005)
34. Lamsweerde, A.v.: Goal-oriented requirements engineering: A guided tour. In: 5th IEEE International Symposium on Requirements Engineering (RE’01) (2001)
35. Lamsweerde, A.v.: Systematic Requirements Engineering - From System Goals to UML Models to Software Specifications. Wiley (2008)
36. Lamsweerde, E.L.a.A.v.: Deriving Operational Software Specifications from System Goals. In: 10th ACM SIGSOFT Symp. on the Foundations of Software Engineering (FSE’10). ACM (2002)
37. Lapouchnian, A., Yu, Y., Liaskos, S., Mylopoulos, J.: Requirements-driven design of autonomic application software. In: Conference of the Center for Advanced Studies on Collaborative Research (CASCON 2006). ACM (2006)
38. Loniewski, G., Insfran, E., Abrahao, S.: A Systematic Review of the Use of Requirement Engineering Techniques in Model-Driven Development. In: 13th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2010), vol. LNCS 6395, pp. 213–227. Springer-Verlag (2010)
39. Lu, C.-W., Chang, C.-H., Chu, W.C., Cheng, Y.-W., Chang, H.-C.: A Requirement Tool to Support Model-Based Requirement Engineering. In: 32nd Computer Software and Applications Conference (COMPSAC '08), pp. 712–717 IEEE (2008)
40. Lucena, M., Santos, E., Silva, M.J., Silva, C., Alencar, F., Castro, J.F.B.: Towards a Unified Metamodel for i*. In: 2nd IEEE Int. Conference on Research Challenges in Information Science (RCIS 2008), pp. 237–246 IEEE (2008)
41. Marín, B., Giachetti, G., Pastor, O.: Applying a Functional Size Measurement Procedure for Defect Detection in MDD Environments In: 16th European Conference on Systems & Software Process Improvement and Innovation (EuroSPI 2009). CCIS. Springer (2009)
42. Marín, B., Giachetti, G., Pastor, O.: The Photography Agency: A case study of the OO-Method Approach. Technical Report DSIC-II/13/08, Universidad Politécnica de Valencia, Valencia, España (2008)
43. Mellor, S.J., Scott, K., Uhl, A., Weise, D.: MDA Distilled: Principles of Model-Driven Architecture, isbn: (10) 0-201-78891-8; (13) 978-0-201-78891-4. Addison-Wesley Professional (2004)

44.Moskitt: Modeling Software KIT, http://www.moskitt.org/eng/moskitt0/
45.OMG: Business Process Modeling Notation version 1.1 (2008)
46.OMG: MDA Guide Version 1.0.1 (2003)
47.OMG: MOF 2.0 Core Specification (2006)
48.OMG: QVT 1.0 Specification (2008)
49.OMG: UML 2.1.2 Infrastructure Specification
50.OMG: UML 2.3 Superstructure Specification (2010)
51.OMG: XMI 2.1.1 Specification
52.Pardillo, J., Molina, F., Cachero, C., Toval, A.: A UML Profile for Modelling Measurable Requirements. 4th International Workshop on Foundations and Practices of UML (FP-UML) – ER Workshop, Vol. LNCS 5232. Springer-Verlag (2008) 123–132
53.Pastor, O., Giachetti, G.: Linking Goal-Oriented Requirements and Model-Driven Development. Intentional Perpectives on Information Systems Engineering. Springer-Verlag, 255–274 (2010)
54.Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. 1st edition, isbn: 978-3-540-71867-3. Springer, New York (2007)
55.Pohjonen, R., Kelly, S.: Domain-Specific Modeling. Dr. Dobb's Journal (2002)
56.Rolland, C., Souveyet, C., Achour, C.B.: Guiding Goal Modelling Using Scenarios. IEEE Transactions on Software Engineering (IEEE TSE), Special Issue on Scenario Management, vol. 24 nº 2, 1055–1071 (1998)
57.Selic, B.: The Pragmatics of Model-Driven Development. In: IEEE Software, vol. 20 nº 5, 19–25 (2003)
58.Tong, Y., Fangjun, W., Chengzhi, G.: A comparison of metrics for UML class diagrams. ACM SIGSOFT Software Engineering Notes vol. 29 nº 5 (2004)
59.Wohlin, C., Runeson, P., Host, M., Ohlsson, M., Regnell, B., Wesslén, A.: Experimentation in Software Engineering - An Introduction. Kluwer Academic (2000)
60.Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD Thesis. University of Toronto, Toronto, Canada (1995)
61.Yu, E.: Modelling Strategic Relationships for Process Reengineering, Ph.D. thesis, also Tech. Report DKBS-TR-94-6. Dept. of Computer Science. University of Toronto, Toronto, Canada (1995)
62.Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In: 3rd IEEE Int. Symp. on Requirements Engineering (RE'97), pp. 226–235 (1997)

APENDIX I: The ISTAR2 Model

Figure 15 shows the departing ISTAR2 model used in the experiment, Table 15 presents the information related to its tagged values, and Figure 16 presents the initial MDD model (MODEL2) generated from ISTAR2 without the information of the verification measures.

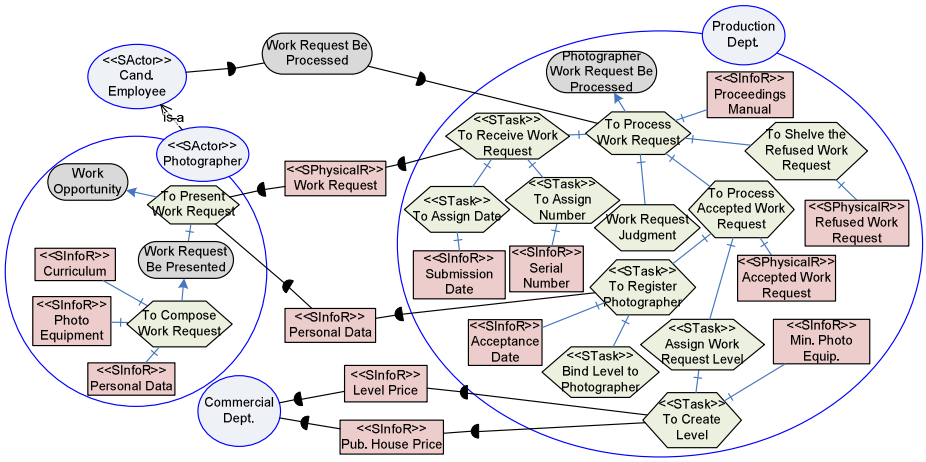

Figure 15. Second *i** Model for Photography Agency Description

Table 15. Tagged values related to the Second *i** Model for Photography Agency

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	--

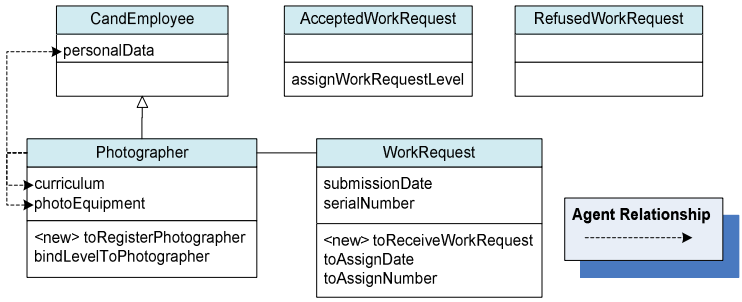

Figure 16. Initial class model generated from second *i* model* without improvements

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 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, 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 MDD model (measure NAE = 0). Additionally, the resource *Proceeding manual* is changed from informational entity to physical entity. Figure 17 shows the improved ISTAR2 model, Table 16 the tagged values changed, and Figure 18 shows the MDD model generated.

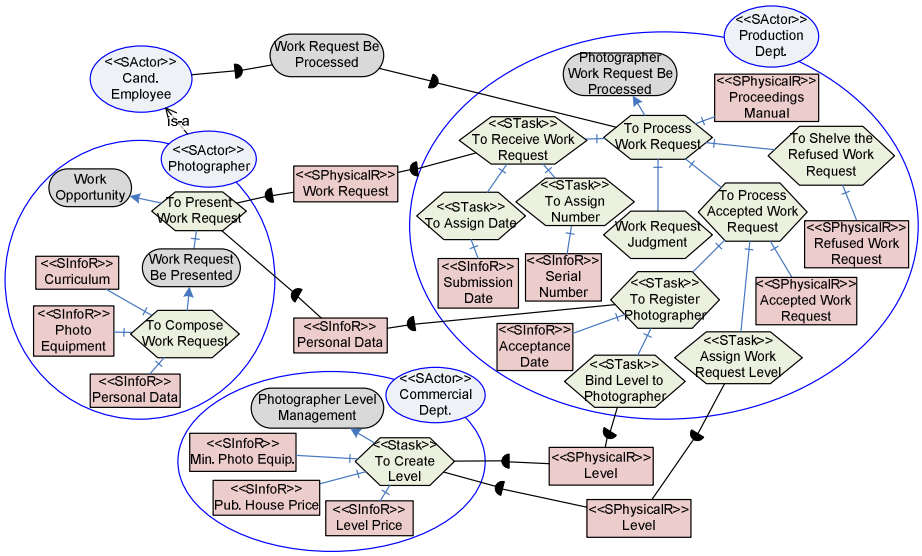

Figure 17. Second *i** model improved with the verification measure results

Table 16. 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		

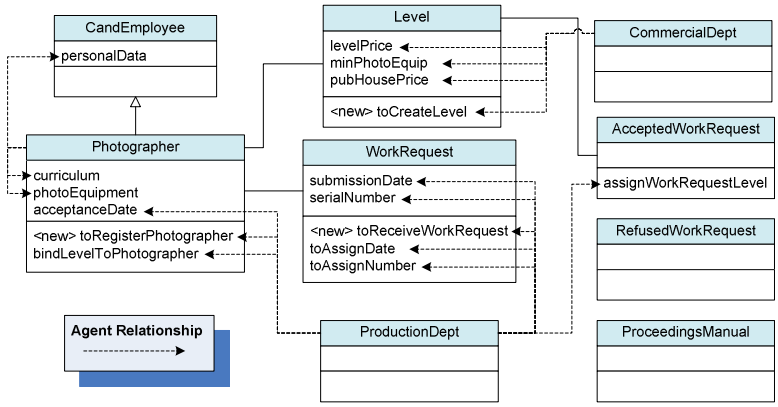

Figure 18. Class model obtained from the improved version of the second *i** model