Using Measures to Improve *i** Models for Automatic Interoperability in Model-Driven Development Processes*

Giovanni Giachetti[1], Xavier Franch[2], Beatriz Marín[1], Oscar Pastor[1], Carlos Cares[2], Lidia López[2]

[1]Centro de Investigación en Métodos de Producción de Software
Universitat Politècnica de València
Camino de Vera s/n, 46022 Valencia, Spain
{ggiachetti, bmarin, opastor}@pros.upv.es
[2]Grupo de investigación en Ingeniería del Software para los Sistemas de Información
Universitat Politècnica de Catalunya (UPC)
UPC-Campus Nord, Omega-122, 08034 Barcelona, Spain
franch@essi.upc.edu, {ccares, llopez}@lsi.upc.edu

Abstract. The use of requirement modeling approaches in model-driven development (MDD) is a relevant topic that has received special attention during the last years. However, there still are several issues that must be taken into account to obtain a sound interoperability of requirement models in MDD processes. Among these, the generation of appropriate input artifacts for model-compilation processes from the requirement artifacts defined is a key aspect to be faced. In this paper, we tackle this issue with an approach for the definition of specific measures that are used to verify *i** models for the automatic interoperability with MDD processes. From the definition and execution of these verification measures, relevant information for identifying and fixing *i** interoperability issues is obtained. From the improved *i** models, initial MDD-models can be automatically generated, which guarantees the completeness of the obtained software models in relation to the requirement specification.

Keywords: DSM, MDD, Verification, *i** Framework, i-Star, interoperability.

1 Introduction

In general terms, the Model-Driven Development (MDD) [1] approaches propose the automatic generation of software products through the automatic transformation of the defined software models into the final program code. Thus, MDD is oriented to reduce (or even eliminate) the hand-made programming, which is an error-prone and time-consuming task.

In this context, interoperability can be considered as a natural trend for the future of model-driven technologies, where different modeling approaches

* **Acknowledgments.** This work has been supported by MICINN under projects TIN2010-19130-C02-01, TIN2010-19130-C02-02, and PTA2008-1443-P. Co-financed with ERDF

can be integrated and coordinated to reduce the implementation and learning time of MDD approaches as well as to improve the quality of the final software products. In particular, through the integration of the requirement elicitation approaches into MDD processes, it should be possible to obtain software products properly aligned with the stakeholders' needs.

We consider the *i** framework [2] as a suitable alternative for requirement modeling since it is a versatile, expressive, and well documented analysis approach [3]. The interoperability of *i** and MDD processes is centered on transforming the defined *i** models into initial MDD models, which are the starting point of the MDD processes involved. In this context, it is necessary to count with adequate mechanisms to identify and fix those *i** model issues that prevent an automatic MDD model generation. We faced this situation by means of an approach for the definition of specific verification measures that are oriented to automatically identify interoperability conflicts. This paper introduces this approach by putting special emphasis on how the involved verification measures provide information to improve the defined *i** models for the generation of the corresponding MDD models. The proposed verification approach is explained through an *i** and MDD interoperability example, which is part of the empirical experiment presented in [4].

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. Finally, Section 6 presents our conclusions and further work.

2 Background and Related Work

The *i** framework [2] is a *Goal-Oriented Requirement Engineering* (GORE) approach that is oriented to obtain the 'whys' of the intended systems through the analysis of organizational scenarios [5]. It emphasizes the analysis of strategic relationships among organizational *actors* capturing the intentions behind software requirements.

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 interoperability of *i** and MDD processes since it provides extra information to generate appropriate inputs for MDD processes.

Figure 1 shows an *i** SR, which is related to the management of work requests in a Photography Agency. Softgoals 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.

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*.

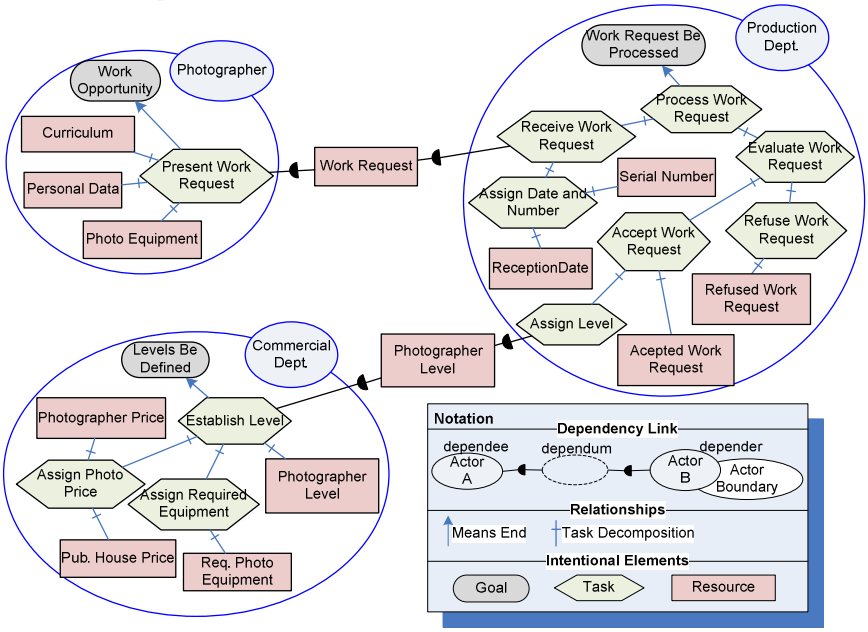

Figure 1. Example *i** SR Model

2. Related Work

As we can observe in the systematic review about requirement engineering and MDD presented in [6], several approaches have encouraged the use of requirement models as part of a sound MDD process. However, a sound solution that includes requirement models as part of a complete, standardized, and automatic MDD process is still a pending challenge [6].

Most of the proposals oriented to translate requirement models into MDD models (such as [7]) 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 automate requirement model transformation, certain proposals manually transform the requirement documents to specific computable formats [8]. This restricts the flexibility of the original specification, which, together with the manual translation of the requirements, may cause loss of information.

Other approaches add quantitative information to existent requirement modeling approaches [9, 10], 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 models related to MDD processes.

3 Interoperability of *i** Models and MDD Processes

For the interoperability of *i** and MDD processes, we propose to partially infer an *initial MDD model* from both the information that is represented in the *i** models and from extra information that is added when it is necessary. This MDD model generation is feasible through a mapping (or model weaving [11]) from the *i** metamodel to the target MDD metamodel. Additional details about our interoperability approach can be found in [12]

Table 1. Mapping for the transformation of *i** models into OO-Method class models

i* Construct	Additional Information	Target 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*[1] 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

[1] This construct corresponds to a binary relationship that indicates the visibility and execution permissions that a class has over others or over itself (recursive agent relationship).

The *i** model transformation can be automated by using technologies such as ATL [13] or QVT [14]. For the representation of the extra information that is required, we use a UML profile, which is automatically generated with the proposal presented in [15].

Table 1 summarizes a representative subset of transformation guidelines (adapted from [16]) for *i** and an industrially applied MDD approach, which is called OO-method [17]. This table indicates the *additional information* that is necessary to perform the transformation of the *i** construct involved. Thus, the *i** guidelines and the involved OO-method constructs are used to exemplify our verification approach throughout this paper.

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.

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

This section briefly explains the process for the definition and integration of verification measures into the *i** framework. For the elaboration of this process, existing standards and modeling technologies have been considered, such as the last version of the *i** framework [18], approaches for the definition of *i** measures [19, 20], OMG Standards for metamodeling [21] and model extensions definition [22], and Eclipse Model Development Tools [23]. The steps of the process are described below (see Figure 2).

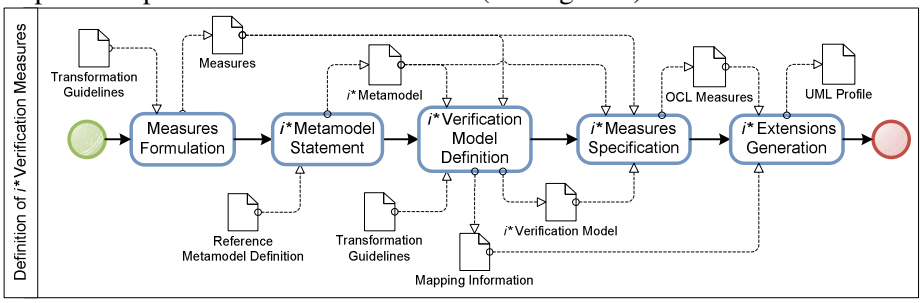

Figure 2. Process for definition of *i** verification measures

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 properties that must be verified for a correct *i** model transformation. These properties can be obtained from the defined transformation guidelines (or rules), in particular, from the additional information that is required to properly perform the involved transformation. We have applied the Goal-Question-Metric (GQM) approach [24] to the transformation guidelines presented in Table 1 to obtain the required verification measures (see Figure 3).

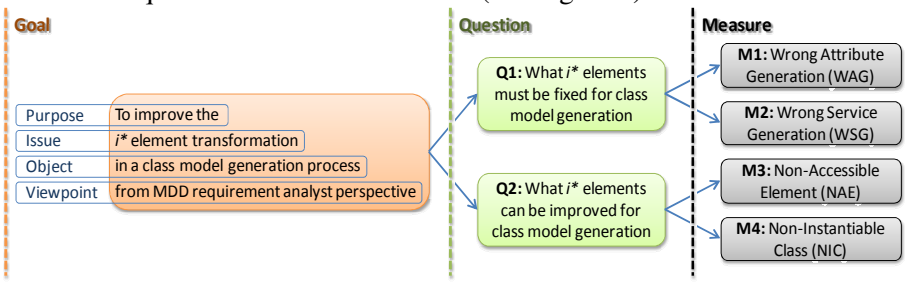

Figure 3. Application of the GQM approach

Two questions have been considered for the GQM approach application: 1) the *i** elements that must be necessarily fixed because they cannot be transformed or produce a wrong class model generation (Q1 in Figure 3); and 2) the *i** elements that can be correctly transformed, but they can be improved to obtain a more complete class model generation (Q2 in Figure 3).

The measures that are related to answer each of the presented GQM questions are specified by considering the framework presented in [25]. Due to space constraints, we only present a brief description of each obtained measure. The detailed definition is presented in [4].

M1. Wrong Attribute Generation (WAG). The informational resources are involved in the generation of attributes (see Table 1). Therefore, for the correct transformation of informational resources, they must be related to a system entity (actor or a physical resource), which is transformed into a class. Otherwise, the informational resources cannot be transformed into attributes because the lack of a class that contains them.

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

M2. Wrong Service Generation (WSG). The tasks that do not generate system 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 MDD model generation, 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.

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

M3. Non-Accessible Element (NAE). 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, they are not transformed into classes, and the necessary agent relationships are not defined. This demands the definition of specific agent classes (such as a system administrator) at design to allow the execution and visualization of the generated services and attributes.

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

M4. Non-Instantiable Class (NIC). The system entities (physical resources or actors) without a production task related are transformed into classes without an instance-creation service. In OO-Method, all the classes must be capable of generating their instances. Thus, specific instance-creation services must be defined at design time for those non-insatiable classes generated.

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

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 [21]. The use of EMOF is mandatory for the appropriate application of the considered interoperability approach [15]. For the elaboration of the *i** metamodel, the proposals presented in [26-29], can be considered. Details about the *i** metamodel used for the application of the defined measures can be found in [4].

4.3 Step 3: *i** Verification Model Definition

The third step of the process consists in the definition of a verification model (see Figure 4). This is an EMOF model that includes the information required for the correct application of the measures. In particular, those elements that represent the additional information (not present in the *i** metamodel) that is necessary for the execution of the transformation guidelines (see Table 1).

Figure 4 also shows the mapping that indicates the correspondences among the elements of the verification model and the *i** metamodel.

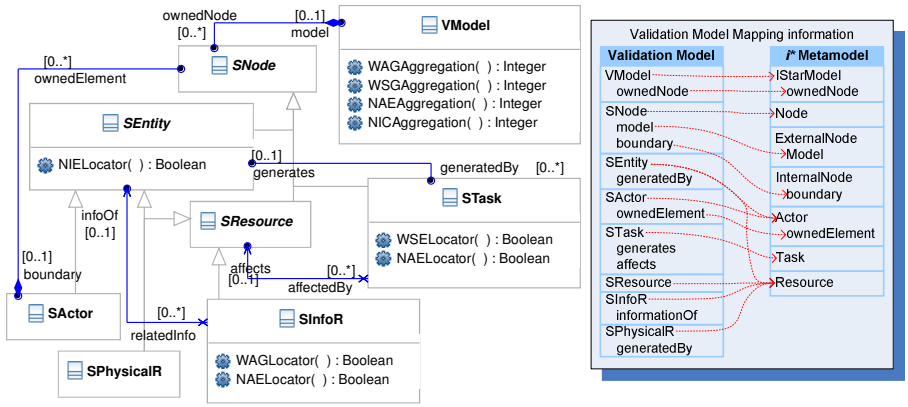

Figure 4. Verification Model and Mapping Information

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 4 shows the names and outputs of the different OCL rules defined. For the measure specification, we have applied the *aggregation* and *locator* patterns presented in [20]. The *locator* pattern identifies the elements involved in the measure evaluation, and the *aggregation* pattern returns the final value of the measure. Thus, the elements that must be fixed are identified by means of the *locator* pattern. For instance, for the measure WAG (Wrong Attribute Generation), the OCL rule *WAGAggregation* returns the WAG measure result by aggregating those resources where the *WAGLocator* returns true.

For the measures execution two alert levels have been considered: 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 generation. These alert levels are derived from the questions proposed for the GQM application. 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 5), which is generated by means of the proposals presented in [15] and [30]. This kind of extensions do not alter the target metamodel, which guarantees the compatibility with the original *i** specification and already implemented tools [31]. However, UML profile depends of the UML metamodel. Thus, a specialization of the classes *Model* and *Element* from the UML metamodel has been performed for the definition of the *i** metamodel.

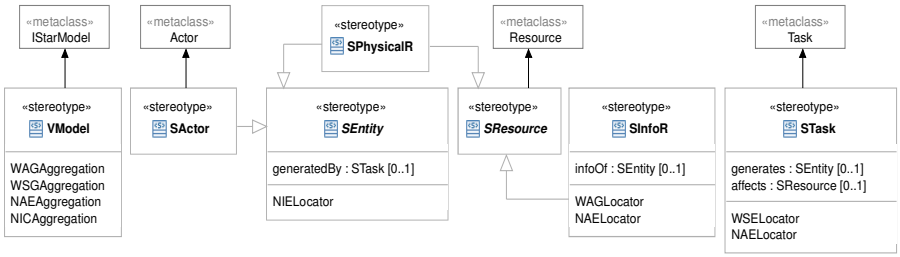

Figure 5. UML Profile to extend the *i** metamodel with the verification measures

5 Applying the *i** Verification Measures

This section shows how the proposed *i** measures are used to verify and improve the generation of the corresponding class model. Only those *i** elements related to the intended system are considered in the transformation process. These elements are the stereotyped elements.

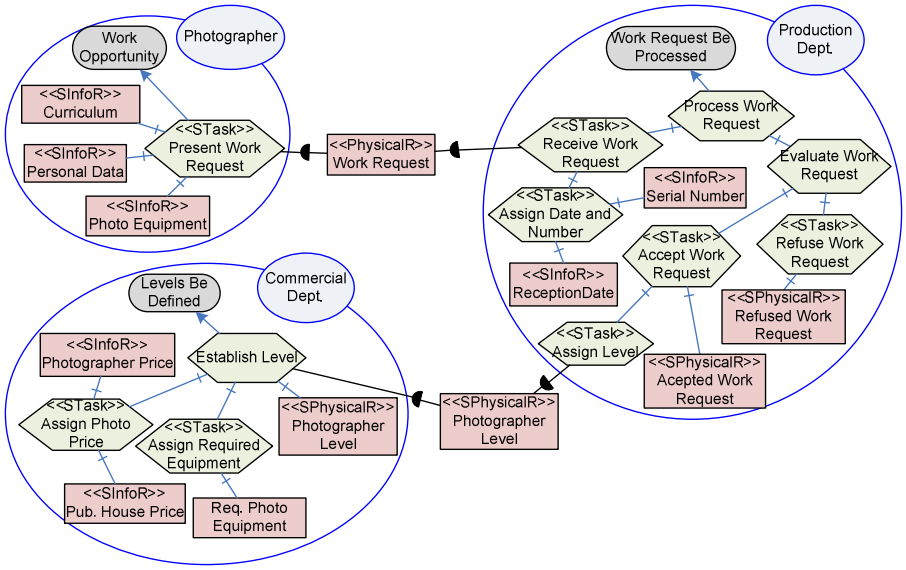

Figure 6. Example *i** Model extended with the generated UML Profile

Table 2 shows the values related to the tagged values of each stereotyped element. Table 3 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 2. 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 3. 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 7 shows the class model that is generated (applying the transformation guidelines presented in Table 1) from the example *i** without considering the information reported by the verification measures.

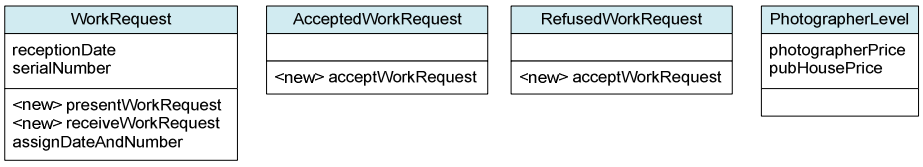

Figure 7. Class model generated from the example *i** model

Figure 7 shows that those elements identified by the critical measures are not present, such as the resource *Curriculum*. Therefore, it is necessary to fix the interoperability issues identified by critical measures.

5.1. Improving the *i** Models for MDD Interoperability

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 4 have been inferred.

Table 4. 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.

Figure 8 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*.

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 5 shows the tagged values that have been changed in the improved *i** model.

Figure 9 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.

Table 5. 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	--		

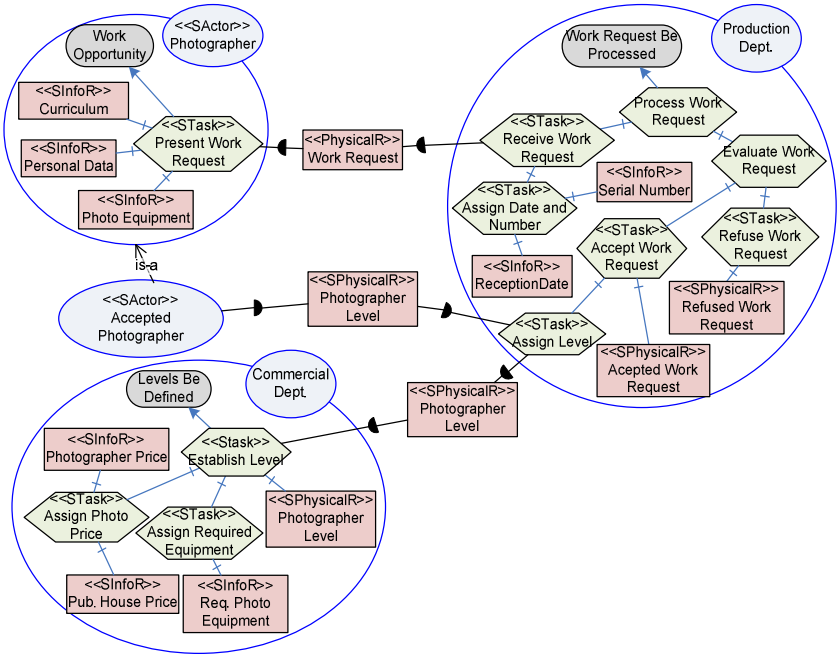

Figure 8. Improved *i** model

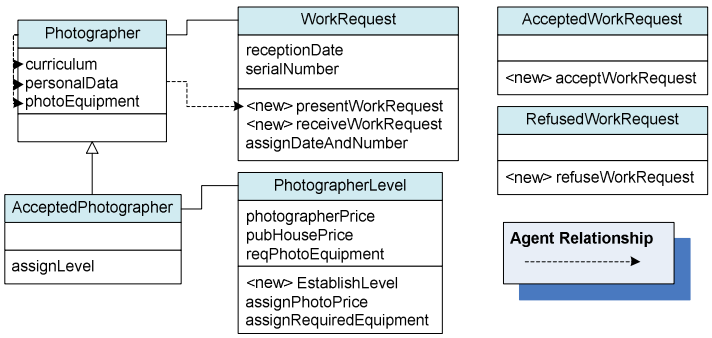

Figure 9. Class model generated from the improved *i** model

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.

8. Conclusions and Further Work

This has presented an approach for the definition verification measures, which improve the interoperability of the *i** framework in MDD processes. Thus, using our proposal, the defined analysis models are not just documentation artifacts; they also play an active role in the development process.

From the *i** and OO-Method interoperability example, we observe that the fixing guidelines obtained from the verification reduce the refinement effort of the generated class models. Furthermore, the critical measures clearly indicate those *i** elements that cannot be transformed. Thus, by fixing the identified critical issues, all the transformation rules can be executed properly. This implies that all the requirement elements considered for the specification of the intended system have correspondence in the generated MDD model. i.e. the generated system model is complete in relation to the requirements. This completeness assurance is demonstrated in the experiment presented in [4].

We consider as future work the development of empirical studies to obtain results of using *i** models in real MDD processes. Additionally, we plan to publish the complete interoperability framework defined for *i** and OO-Method, which can be used as a reference by different MDD approaches.

References

1. Selic, B.: The Pragmatics of Model-Driven Development. IEEE Software, vol. 20, pp. 19–25 (2003)
2. Yu, E.: Modelling Strategic Relationships for Process Reengineering, Tech. Report DKBS-TR-94-6. Dept. of Computer Science. University of Toronto, Canada (1995)
3. Eric Yu, P.G., Neil Maiden and John Mylopoulos: Social Modeling for Requirements Engineering (2011)
4. Giachetti, G., Alencar, F., Franch, X., Marín, B., Pastor, O.: Technical Report ProS-TR-2011-07: Automatic Verification of Requirement Models for Their Interoperability in Model-Driven Development Processes. Universidad Politécnica de Valencia (2011)
5. Lamsweerde, A.v.: Goal-oriented requirements engineering: A guided tour. 5th IEEE International Symposium on Requirements Engineering (RE'01), (2001)
6. Loniewski, G., Insfran, E., Abrahao, S.: A Systematic Review of the Use of Requirement Engineering Techniques in Model-Driven Development. MoDELS 2010, vol. LNCS 6395, pp. 213–227. Springer-Verlag (2010)

7. Lapouchnian, A., Yu, Y., Liaskos, S., Mylopoulos, J.: Requirements-driven design of autonomic application software. CASCON 2006. ACM
8. Lu, C.-W., Chang, C.-H., Chu, W.C., Cheng, Y.-W., Chang, H.-C.: A Requirement Tool to Support Model-Based Requirement Engineering. COMPSAC'08, pp. 712–717 IEEE (2008)
9. Pardillo, J., Molina, F., Cachero, C., Toval, A.: A UML Profile for Modelling Measurable Requirements. In: FP-UML, vol. LNCS 5232, pp. 123–132. Springer-Verlag (2008)
10. Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., Yu, E.: Evaluating goal models within the goal-oriented requirement language. In: International Journal of Intelligent Systems (IJIS) (2010)
11. Fabro, M.D.D., Valduriez, P.: Towards the efficient development of model transformations using model weaving and matching transformations. Software and Systems Modeling 8, 305–324 (2009)
12. Pastor, O., Giachetti, G.: Linking Goal-Oriented Requirements and Model-Driven Development. In: Nurcan, S., Salinesi, C., Souveyet, C., Ralyté, J. (eds.) Intentional Perpectives on Information Systems Engineering, pp. 257–276. Springer-Verlag (2010)
13. Jouault, F., Kurtev, I.: Transforming Models with ATL. Satellite Events at the MoDELS 2005 Conference, vol. LNCS 3844, pp. 128–138 Springer (2006)
14. OMG: QVT 1.0 Specification. (2008)
15. Giachetti, G., Marín, B., Pastor, O.: Using UML as a Domain-Specific Modeling Language: A Proposal for Automatic Generation of UML Profiles. CAiSE 2009, vol. LNCS 5565, pp. 110–124. Springer (2009)
16. 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. PoEM 2009, vol. LNIBP 39, pp. 99–114. Springer (2009)
17. Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer, New York (2007)
18. http://istar.rwth-aachen.de/, last Accessed October 2009
19. Franch, X.: A Method for the Definition of Metrics over i* Models. CAiSE 2009, pp. 201-215. Springer-Verlag LNCS (2009)
20. Franch, X., Grau, G.: Towards a Catalogue of Patterns for Defining Metrics over i* Models. CAiSE 2008, pp. 197–212. Springer (2008)
21. OMG: MOF 2.0 Core Specification. (2006)
22. Fuentes-Fernández, L., Vallecillo, A.: An Introduction to UML Profiles. The European Journal for the Informatics Professional (UPGRADE), vol. 5, pp. 5–13 (2004)
23. http://www.eclipse.org/modeling/mdt/
24. Basili, V., Caldeira, G., Rombach, H.D.: The Goal Question Metric Approach. Encyclopedia of Software Engineering, Wiley (1994)
25. Habra, N., Abran, A., Lopez, M., Sellami, A.: A framework for the design and verification of software measurement methods. Journal of Systems and Software 81, 633–648 (2008)
26. 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. AOSDM'05, at the SEKE Conference, pp. 657–663, Taipei, Taiwán; China. (2005)
27. Franch, X.: Incorporating Modules into the i* Framework. CAiSE 2010, vol. LNCS 6051, pp. 439–454. Springer-Verlag Berlín Heidelberg, Hammamet, Tunisia (2010)
28. Lucena, M., Santos, E., Silva, M.J., Silva, C., Alencar, F., Castro, J.F.B.: Towards a Unified Metamodel for i*. In: RCIS 2008, pp. 237–246 IEEE (2008)
29. Amyot, D.: New draft Recommendation Z.151: User Requirements Notation (URN). (2008)
30. Giachetti, G., Valverde, F., Pastor, O.: Improving Automatic UML2 Profile Generation for MDA Industrial Development. In: FP-UML – ER Workshop, vol. LNCS 5232, pp. 113–122. Springer (2008)
31. Amyot, D., Horkoff, J., Gross, D., Mussbacher, G.: A Lightweight GRL Profile for i* Modeling. RIGIM - ER Workshops, vol. LNCS 5833, pp. 254–264. Springer-Verlag (2009)