Information Systems and e-Business Management
Reusing enterprise models to build computer models
--Manuscript Draft--

Manuscript Number:	ISEB-D-15-00083R2
Full Title:	Reusing enterprise models to build computer models
Article Type:	S.I. : Model-based engineering for next-generation EIS
Keywords:	Enterprise Models; Model Transformation; MDA
Corresponding Author:	ricardo chalmeta SPAIN
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	
Corresponding Author's Secondary Institution:	
First Author:	Veronica Pazos
First Author Secondary Information:	
Order of Authors:	Veronica Pazos
	ricardo chalmeta
Order of Authors Secondary Information:	
Funding Information:	
Abstract:	Enterprises use enterprise models to represent and analyse their processes, products, decisions, organisation, information flows, and so forth. Nevertheless, the enterprise knowledge that exists in enterprise models is not used beyond these purposes. The main goal of this paper is to present a framework that allows enterprises to reuse enterprise models to build software. The framework includes these dimensions: 1) a methodology that guides the use of each dimension in the reutilisation of enterprise models in software generation, 2) a set of metamodels to represent enterprises at CIM level, 3) a modelling guide to make enterprise models using the metamodels proposed in this paper, 4) an extraction algorithm to discriminate the part of the CIM model to reuse; and 5) a set of transformation rules to reuse enterprise models to build computer models. In addition, a case example is shown to validate the work that was carried out and to detect limitations.
Response to Reviewers:	We have made all the corrections but three. see attach file

We have made all the **corrections but the following.**

Referee 1 comments	28. Description of Use -- why the "I" prefix on "IInput, IParameter"? These names are from MDK Proposal. To the best of our knowledge the "I" prefix is to distinguish them from the BRInput. Para starting "The selection of the elements" -- talks about selecting elements from CIM (i.e., CIM1, presumably) that must be part of the CIM2 level, but right after the description of the propagation, the text says "that we want to be represented in the PIM-level conceptual model". Is this stated as intended? Yes, because the elements in the CIM2 level are those which are going to be transformed, so they are going to be represented in PIM-level conceptual model. 55. Wondering about the lifecycle of the "exclusive" object. Seems like it could have states like pending, obtained, delivered, billed, paid, etc. Are those considered? These states are not considered explicitly. However some of them can be deduced from the properties of the exclusive. It is pending if photographer property is not set to any value, it is obtained when delivery date property has value for example. 57. Case study analysis and discussion -- the section is fairly self-serving, since the evaluation was done by the authors. Would be good to have independent evaluation/assessment of the methodology. It is a good idea. We leave it for a future work.
Referee 4 comments	As a personal note on the research proper: The justification to use O-O methods is basically that it is popular, well founded and 'a lot of work' has been done with it in this domain. That is as strong as one can say in the scope of the paper. And the general approach outlined here is the only one a skilled engineer could use, given the tools of today. But I believe based on underlying formal issues, that O-O is not capable of doing what is intended here. There is room for a look at first principles, and after the authors have some experience using this method, I encourage them to revisit their basic assumptions. Thank you for your suggestion. We are going to explore other alternatives to OO methods for a future work.

Honestly, we wish to thank the reviewers for the time they have spent on this paper, as their comments and suggestions have enabled us to improve the paper considerably.

Reusing enterprise models to build Platform Independent Computer Models

Verónica Pazos Corella, Ricardo Chalmeta Rosaleñ

[1]Grupo de Investigación en Integración y Re-Ingeniería de Sistemas (IRIS), Departamento de Lenguajes y Sistemas Universitat Jaume I. 12006 Castellón. Spain. Tel: +34964728329. Fax: +34964728435. vpazos@lsi.uji.es, rchalmet@lsi.uji.es

Corresponding autor: Ricardo Chalmeta Rosaleñ. rchalmet@uji.es

Reusing enterprise models to build Platform Independent Models

Abstract.

Enterprises use enterprise models to represent and analyse their processes, products, decisions, organisation, information flows, etc. Nevertheless, the enterprise knowledge that exists in enterprise models is not used beyond these purposes. The main goal of this paper is to present a framework that allows enterprises to reuse enterprise models to build software. The framework includes these dimensions: 1) a methodology that guides the use of the other dimensions in the reutilisation of enterprise models in software generation; 2) a set of metamodels to represent enterprises at the Computation Independent Model (CIM) level; 3) a modelling guide to make enterprise models using the metamodels proposed in this paper; 4) an extraction algorithm to discriminate the part of the CIM model to reuse; and 5) a set of transformation rules to reuse enterprise models to build Platform Independent Models. In addition, a case example is shown to validate the work that was carried out and to identify limitations.

Keywords: Enterprise Models, Model Transformation, Model Driven Architecture

1 Introduction

Enterprise Modelling can be defined as the art of representing the structure and behaviour of an enterprise (Vernadat, 1996) in a way that allows it to be used to support decision-making, to gain a better understanding of the different dimensions of the enterprise (such as its organisation, business processes, information, decisions, products, resources, etc.) and to share this knowledge with others. Many languages, standards, methodologies, architectures and tools for Enterprise Modelling have emerged since the 1970s (Devedzic, 1999).

Although existing proposals have proved to be adequate for the aims that they were initially designed for, today there is a new challenge to be met; i.e., the automatic reuse of the information contained within the enterprise models to create Platform Independent Models (PIM) that lend support to enterprises (Chen & Sairamesh, 2006). It is generally agreed that the only realistic way to address this problem, and to continue to provide software benefits to the public at large, is to develop software using appropriate methods of abstraction based on a model-driven approach (Kramer, 2007).

This new challenge allows advantage to be taken of the effort that went into constructing these enterprise models by reusing the knowledge depicted in them. At the same time it allows some of the current objectives of software engineering to be achieved, such as: (1) improving communication and reaching agreements among the final users and developers of the computer system, because models can be used as a communication mechanism (Huet & Martin, 1980); (2) aligning the functioning of the enterprise with the software solution that supports it, because if the requirements are obtained from the enterprise models, the software solution is more likely to better fit the needs of the enterprise; (3) reducing the probability of errors and the time and costs of software development because the development is partially automated (Zhu et al, 2013).

Nevertheless, in their present state there are a number of issues that prevent any of the existing approaches from creating and transforming enterprise models into Platform Independent Models that are wholly satisfactory and capable of being used to support the development of a computer system (Garner & Raban, 1999). The main problems can be summed up as follows:

1. There are no enterprise models that take into consideration all the aspects of a business that need to be included in the future computer system of the enterprise.

2. The purpose of the model, the nomenclature and the point of view followed in creating the models differ according to the view of the enterprise that is being modelled (Dijkman et al, 2008). For instance, when the organisational structure is being modelled, the models usually focus on the organisational elements, like departments and human resources of the enterprise, while if the business processes are being modelled, they focus on the tasks, activities and processes needed to carry out the business process. However, even if elements that are present in the organisational structure model also appear in the

business models, there is no explicit mechanism that allows the modellers to establish an explicit relationship between the elements that appear in different models, despite the fact that they represent the same concept. This means that the semantic wealth of the models does not possess the information necessary for them to be used to create the conceptual model of the future computer system automatically.

3. Lack of strong links between enterprise models and software generation. This problem becomes more pronounced due to the existence of various modelling languages for each of the different views of the enterprise (processes, products, resources, etc.), which makes it more difficult to define suitable mechanisms for transforming enterprise models into Platform Independent Models.

4. Enterprises do not have a clear understanding of the potential advantages of enterprise modelling. As a result, some enterprises, especially SMEs, implement few enterprise models and, if they do, it is very hard for them to maintain those (Vergidis et al, 2008).

5. There are few examples of success stories. This may be due to the many open and unresolved research issues in this field, like how to deal with very large models or the designing of model management tool chains. Success stories may reveal new directions for development of the theory as well as lessons of transferable value for future practice (Ruscio et al, 2014).

6. The benefits of using abstract models to develop software are frequently cited and are often considered to be obvious, but there are no empirical data to test or support the claims of these benefits (Hutchinson et al, 2014).

7. Enterprises create models for several purposes, since they are very useful to understand how enterprises work, among other benefits. As stated in the Literature Review section, to the best of our knowledge, most approaches for enterprise modelling have a lack regarding their level of formalism. They are not formal enough to be able to use them to create Platform Independent Models automatically.

Hence, there is a need for methodological proposals that allow the main views of the enterprise to be modelled, which use the enterprise models as yet another resource in software development and which include a definition of the rules of transformation so that the information from the enterprise models can be reused in the automatic creation of Platform Independent Models.

With the aim of solving this problem, this paper proposes the MDKe-IRIS (Model Driven Knowledge extended – IRIS) Framework for developing software from enterprise models with a significant practical impact. It allows the main enterprise views to be modelled; it solves the semantic problems in the interoperability among the different models of each of the enterprise views; it defines the transformation rules to help to build software from enterprise models; and it develops models at the Computation Independent Model (CIM) level that are formal enough to be able to reuse them to build software.

The main goal of the MDKe-IRIS Framework is to allow practitioners to derive software solutions for enterprises starting from their enterprise models. This framework includes these dimensions: (1) a methodology that guides the use of each dimension in the reutilisation of enterprise models in software generation; (2) a set of metamodels for the enterprise model; (3) the definition of a guide to carry out enterprise modelling; (4) an algorithm for extracting the elements of the enterprise models that are going to be included in the Platform Independent Models; and (5) the definition of a set of transformation rules to be applied to the instances of the metamodels in order to turn them into Platform Independent Models. Moreover, in order to illustrate how the MDKe-IRIS Framework works, this paper includes a case study. The case study was developed with the aim of validating the research conducted during the development of the MDKe-IRIS Framework, specifically to answer this question: *is it possible to reuse the knowledge depicted in enterprise models for code generation using the MDKe-IRIS Framework*?

This paper is structured as follows. Section 2 briefly reviews the concepts, proposals and approaches related to the use of business models in software development. Section 3 explains some concepts related to the development of the MDKe-IRIS Framework. A detailed description of the MDKe-IRIS Framework is given in section 4 and a practical case is described in section 5 as an example of how it can be used. Certain aspects of the framework are discussed in section 6 and, finally, section 7 contains the conclusions from the study.

2 Literature Review

The work presented in this paper is based on the Model Driven Development (MDD) paradigm, which is in turn based on the idea that models direct the process of developing a software application. In this paradigm, the models represent the functioning of the enterprise by using a language that is expressive enough to allow part of the software system that supports it to be generated automatically (Pham et al, 2007). Within MDD, the Model Driven Architecture (MDA) paradigm (OMG, 2003) is probably the most commonly used. It was proposed by the Object Management Group (OMG) in 2001 as an architecture for developing applications by using a hierarchy of models at three levels of abstraction:

• Computation Independent Model (CIM). This is used to represent the domain to be modelled (called CIM1) and the computer system requirements of the domain (called CIM2). CIM is based on business models and views of the enterprise from a holistic perspective that is independent of the computation.

• Platform Independent Model (PIM). This is used to model system functionality but without defining how it will be implemented or on what platform. It focuses on information and sets out from a computational point of view.

• Platform Specific Model (PSM). The PIM model is transformed into a platform-dependent model according to the platform selected for use and is focused on a technological point of view.

A lot of work is being carried out within the OMG framework to develop PIM and PSM models and to achieve direct transformations from PIM to PSM, as well as from PSM to executable code (OMG, 2003), (Soler et al, 2009), (Xiao-mei et al, 2009).

The work presented in this paper is focused on the transformation of CIM models into PIM models following a set of transformation rules. To the best of our knowledge, the creation of CIM models of enterprises and their transformations into PIM models is still in progress (De Castro et al, 2011). The rest of this section explains the works in the literature regarding the concept of reusing knowledge depicted in models with a high level of abstraction (CIM models) to generate PIM models. These CIM2PIM works can be classified in three categories, depending on the enterprise view that is being modelled at the CIM level: one focused on software requirements modelling, another focused on decisional and organisation modelling and the last one focused on business process modelling.

2.1 CIM2PIM works focused on software requirements modelling

Regarding the CIM2PIM works focused on software requirements modelling at the CIM level, all the proposals contribute to establish the foundations of how to reuse the information depicted at the CIM level. However, they have been developed mainly from the point of view of software development. Thus, the CIM models only model the software requirements of the enterprise without considering other enterprise views; i.e., they only model what they are going to transform. There are two sub-categories: one related to the reutilisation of models generated with i* language (Nicolás & Toval, 2009), and the other based on the processing of natural language specifications to obtain PIM models.

The I* language (Dalpiaz et al, 2008) is a goal-oriented modelling language in which the models represent the objectives to be accomplished by a software solution and the agents that are involved in the accomplishment of these objectives. Regarding i*, in (Guizzardi & Guizzardi, 2010) the authors present the ARKnowD Methodology that combines Tropos, a methodology for making models using i* at the CIM level (Bresciani et al, 2004), and the Agent-Object-Relationship (AOR) modelling approach (Wagner, 2003) for making models at the PIM level. The ARKnowD Methodology allows practitioners to obtain models for system engineering starting from i* models. The idea is to map Tropos and AOR modelling language to an ontology in order to establish the semantic correspondence between the elements of the metamodels of both languages. The elements in the resulting AOR model, derived automatically from Tropos, are agents, objects, interactions and relationships, but the properties of the objects cannot be extracted from the i* model automatically.

Other works based on i* are (Alfonso et al, 2010), (Luna et al, 2010) and (Alencar et al, 2009). The first two explain explain CIM to PIM transformations in the context of a web application. Requirements are modelled using the i* framework (Yu, 1997) and the resulting models are transformed to the Adaptive Object Oriented Hypermedia method (A-OOH) (Garrigós, 2008). In these three works, the authors present a goal-oriented approach to specify web requirements and a set of transformation rules that allow users to obtain A-OOH Domain and Navigational models. The difference between these two works is that (Luna et al, 2010) includes the use of mock-ups to facilitate the end-users' comprehension of the functioning of the final application. Moreover, in (Luna et al, 2010) an automatic evaluation of i* models is included so as to be able to know the level of satisfaction of the goals described in them. On the other hand, (Alencar et al, 2009) explains the transformation guidelines to automatically obtain OO-Method approach models (Pastor & Molina, 2007) from i* models. It establishes a total of eight guidelines that allow the modellers to reuse the knowledge depicted in i* diagrams. To reuse the knowledge at the CIM level, an analysis of the i* elements is carried out. The processes to be automated at the PIM level are obtained from tasks at the CIM level and actors in the i* diagrams are transformed into classes at the PIM level if it is necessary to maintain information about them. Regarding this guideline, from the information we found in (Alencar et al, 2009), it is not clear if there exists an automatic way to distinguish actors to be transformed and actors that do not need to be transformed. Moreover, attributes of classes are obtained from the resources associated to an actor at the CIM level if these resources express information about the actor but, again, it is not clear if it is possible to know whether a resource has this characteristic automatically.

The other sub-category deals with issues of CIM to PIM transformation based on the processing of natural language specifications to obtain PIM models. In (Chen & Zeng, 2010)) an approach is proposed to transform product requirements expressed in natural language into UML (Unified Modelling Language) diagrams using the Recursive Object Model (ROM). The text is analysed using ROMA (ROM Analysis) and a ROM diagram is generated. Once the ROM diagram has been created, a set of transformation rules is applied over it to automatically obtain use case and object UML (Object Management Group, 2000) diagrams. From our point of view, this is an interesting approach because text specifications are easy for the end users of the software solution to understand, but its main problem is that it is very difficult to get the most from the text; for instance, it is not clear how the approach deals with business rules expressed in the text, or what happens if the writer of the text uses synonyms to designate the same concept. In (Lopata & Ambraziunas, 2010) the authors explain the generation of the PIM level, represented by the UML class, use case and sequence diagrams, from the requirements represented with structured questionnaires, modified workflows and free-form diagrams (CIM level). The idea is to enrich the CIM level with the knowledge base subsystem, which represents an enterprise metamodel, in order to establish the semantics of the elements at the CIM level. CIM level elements are mapped onto the knowledge base subsystem and the transformation is performed from the knowledge base subsystem to the PIM level. Finally, (Martinez-Fernandez et al, 2009) introduces the use of business rules at all levels of an MDA architecture. In this way, the business restrictions are present in the whole software development process. Business rules are defined over concepts represented in an ontology that can be matched against business objects, represented by UML classes.

2.2 CIM2PIM works focused on decisional and organisation modelling

Regarding the CIM2PIM works focused on decisional and organisation modelling at the CIM level, the only work we have found in the literature is (Grangel et al, 2010), where it is explained how to transform models made with the GRAI method develop at the LAP/GRAI Laboratory, University Bordeaux, France (Doumeingts et al, 1993) into UML use case and activity diagrams. This work shows how it is possible i) to define UML profiles for filling the semantic gap between GRAI Grids and UML Use Case Diagrams or Activity Diagrams, and ii) to implement the profile-based mapping using a transformation language.

2.3 CIM2PIM works focused on business process modelling

Regarding the CIM2PIM works focused on business process modelling at the CIM level, there are different approaches. In (Cox et al, 2005) the authors propose a guide to derive software from process models and represent them in terms of Jackson's problem frames (Jackson, 2001). The study proposes a set of guidelines to map role activity diagrams (Ould, 1995) to problem frames. In (Rodríguez, 2010) a set of transformation rules are explained for transforming Secure Business Process (SBP) models into UML analysis classes and UML use cases with the aim of obtaining useful artefacts for software development. SBP models were defined as an extension of UML 2.0 – Activity Diagram and BPMN – BPD (Business

Process Modelling Notation – Business Process Diagram) (OMG, 2009). The work presented in (Rodríguez, 2010) is about how to take advantage of CIM models in software development processes. However, they do not explain what the specific usages of the generated PIM models are and, currently, since the authors do not define transformation rules for transforming PIM to PSM and code, it is not possible to use those PIM models for automatic software generation. (De Castro et al, 2011) includes a complete set of metamodels for the CIM and PIM levels as well as the mapping rules to perform the transformations from CIM to PIM models. The authors use the e3 value model (Gordijn et al, 2006) and BPMN models for the CIM level and a set of Domain Specific Languages defined specifically for the service-oriented development of information systems. Transformations are defined using the language ATL (Atlas Transformation Language) (Jouault et al, 2008). In (Mazanek & Hanus, 2011) the authors show how to transform CIM models, represented with BPMN, into PIM models, represented with BPEL (Business Process Execution Language) (OASIS, 2011). The main difference of this work with regard to the previous ones is the transformation engine they use. The transformation engine is based on functional logic programming: it uses rewriting rules specified with Curry language (Antoy & Hanus, 2002).

2.4 Synthesis

As a result of the review of the state of the art in the literature regarding the CIM to PIM transformation, it may be concluded that there is no approach that models the enterprise from a holistic point of view at the CIM level; i.e., considering all the enterprise views. This is an important challenge inside the software engineering community, as was pointed out in (Kardos & Drozdová, 2010).

To solve this problem, the MDKe-IRIS Framework has been developed. This Framework takes into account the developer's point of view, but also business people's point of view. The types of models that the Framework works with are enterprise models that allow most of the views of the enterprise to be modelled automatically, as well as automatically extracting the elements that will make up the software application. Moreover, the framework includes a modelling guide to help practitioners create the enterprise models and a set of sixteen rules to automatically obtain a software prototype from the CIM model. The prototype created is a prototype that follows the Object Oriented Programming paradigm. It includes the set of classes and the signature of the methods that represent the behaviour of the objects. The main contributions of the MDKe-IRIS Framework compared with earlier proposals are:

- MDKe-IRIS allows practitioners to model the main views of the enterprise. Most of the proposals reviewed here are focused on one view. MDKe-IRIS takes into account the structure of the enterprise, its business rules, business processes and the organisational structure, among others.
- The framework includes the mechanisms needed to establish relations among the models of the different enterprise views, which facilitates interoperability of the CIM-level models by preventing semantic inconsistencies.
- One of the biggest problems of the previous models is the automatic extraction of the requirements. The business models may include aspects that do not necessarily have to appear in the platform-independent models. Hence, with the aim of finding a solution to the extraction of the requirements from the business models, the framework includes a mechanism for discriminating the elements in the business model that must appear in the Platform Independent Models by means of a tagging process, and their automatic extraction using an algorithm.
- All the reviewed proposals are based on the fact that all the information at the CIM level must be transformed to create the PIM level. They do not take into account the possibility that CIM models represent the enterprise itself and not only its information system. On the CIM level of the MDKe-IRIS Framework it is possible to represent the enterprise model independently of the elements that will make up the PIM level.
- It includes a top-down modelling guide that helps modellers obtain the information from business people and to represent it using the MDKe-IRIS Framework.
- It is possible to generate a prototype automatically from the CIM level.

3 Background

In this section the different approaches that form the basis of the MDKe-IRIS Framework are reviewed. First, the model-driven knowledge (MDK) proposal (Grangel, 2007) is explained, which is the foundation for the metamodels that are used in the MDKe-IRIS Framework to generate the enterprise models. Next, the OO-Method (Pastor & Molina, 2007) is outlined, which is the modelling language chosen to represent the Platform Independent Models derived from the enterprise models. Finally, the justification underlying the choice of these two approaches is explained.

3.1 Model Driven Knowledge Proposal

The modelling language used to generate the Enterprise models in the MDKe-IRIS Framework is based on the MDK proposal. MDK is a reference framework for generating models that represent enterprise knowledge with the aim of later producing a knowledge management system (Chalmeta & Grangel, 2008).

The MDK proposal defines an enterprise ontology and a set of metamodels and diagrams that make it possible to generate models which represent different views of the enterprise. The metamodels that were defined and later implemented by developing a series of UML profiles are:

- The Knowledge Metamodel, which generates the 'Knowledge Model'. This model offers a holistic view of the knowledge possessed by an enterprise. It makes it possible to identify and represent the conceptual blocks of knowledge (which are understood as being any elements belonging to the organisation or to its surroundings that contain a particular type of knowledge), the ontological categories and subcategories, the target knowledge of a particular enterprise, and the variables, sources and procedures that are employed to extract and calculate them.
- The Organisation Metamodel, which generates the 'Organisation Model'. This represents the enterprise from the point of view of its organisation, business rules and goals.
- The Structure Metamodel, which makes it possible to obtain the 'Structure Model'. This models the products, resources, employees and customers/suppliers of an enterprise.
- The Behaviour Metamodel, which allows the 'Behaviour Model' to be generated. This models the business processes and the services offered by the enterprise.

3.2 OO-Method

The enterprise models developed with the MDKe-IRIS Framework are transformed into Platform Independent Models that follow the OO-Method approach.

The Object Model is a graphic model that includes a definition of the system classes, their attributes and the relations among them; the services available for each class, which specify the changes of state that can take place in the instances of the classes to which they are associated; and the agents, which represent the users of the system.

The Dynamic Model specifies the valid lifecycle of an object, which is the order in which the services associated to a class must take place. The valid life sequence of an object is defined by means of a State Transition Diagram, which contains two kinds of elements: states and transitions. States represent the different specific situations in which an object may find itself. Transitions, in contrast, associate two states of the object and indicate that change of state is allowed. To do so, each transition is associated to a service so that if service X is executed on an object that is in a state N, then the state of the object will become the state associated with N by means of X. Furthermore, this model allows the user to define global transactions that include services of different classes but which are executed as a single unit. It is also possible to determine which services can be executed if a certain condition is fulfilled.

The Functional Model captures the operational semantics associated with the changes of state that are produced in these objects. The effects that different events produce on the objects are defined and the conditions for evaluating their attributes are specified, among other things.

Finally, the Presentation Model represents the way in which the information in the system is captured by means of agent interaction models.

3.3 Justification

The MDK proposal was adopted because (1) the metamodels defined in this proposal allow CIM models of the main views of the enterprise to be generated; (2) the diagrams for producing the CIM models are visual and similar to the ones that are usually employed in enterprises, for example IDEF0 (Cantamessa & Paolucci, 1998) for modelling the process view, GRAI (Roboam et al, 1989) for modelling decisions, or EXPRESS for products (ISO 10303 STEP: Standard for the Exchange of Product model data); and (3) it uses a single modelling language to generate the models of the different views. The MDK proposal has these characteristics because it is based on the stereotype mechanism offered in version 2 of UML, developed by the OMG, which allows UML to be customised in a better way (Fuentes et al, 2002). Moreover, MDK was developed from the principles defined in the ATHENA (Advanced Technologies for interoperability of Heterogeneous Enterprise Networks and their Applications) project (IST-2001-507849, 2010) and is based on the UEML (Unified Enterprise Modelling Language) (Opdahl & Berio, 2007) and POP* (Process, Organisation, Product) (Grangel et al, 2006) metamodels for establishing common formats, which facilitate the exchange of models and provide an environment in which they can be reused. UML has been selected because it has been successfully used to model and develop IT systems in different domains, and it can also be useful in the context of Enterprise Modelling (Marshall, 2000).

The OO-Method was chosen as the target of the transformations because it is very well documented and is implemented within a commercial tool. Thus, it is possible to validate the PIM models generated from the MDKe-IRIS Framework in practice.

Finally, although functional programming is gaining popularity nowadays most companies choose web and mobile applications based on object-oriented programming implemented with Java, asp.net, ios or android as programming languages. We therefore chose object-oriented modelling as the representation of the PIM level because, in our opinion, it can have a major practical impact.

4 MDKe-IRIS Framework for Enterprise Modelling and Knowledge reutilisation

In this section the MDKe-IRIS Framework is explained. It is based on five dimensions (Fig. 1). The first dimension is a methodology that guides the use of each dimension in the reutilisation of enterprise models in software generation. The second dimension is the modelling language of the CIM level based on the MDK Proposal, which is called MDKe-IRIS (Model Driven Knowledge extended). The third dimension is the Modelling Guide, which is a set of steps about how to create enterprise models using the MDKe-IRIS language. The fourth dimension is an extraction algorithm to obtain the elements of the CIM2 level from the CIM1 level. The fifth dimension consists of a set of rules for mapping the metamodel of the MDKe-IRIS language onto the OO-Method modelling language. These mapping rules are used as transformation rules to derive the PIM model that conforms to the OO-Method metamodel, from the CIM2 level that conforms to the MDKe-IRIS metamodel. The rest of this section explains each dimension.

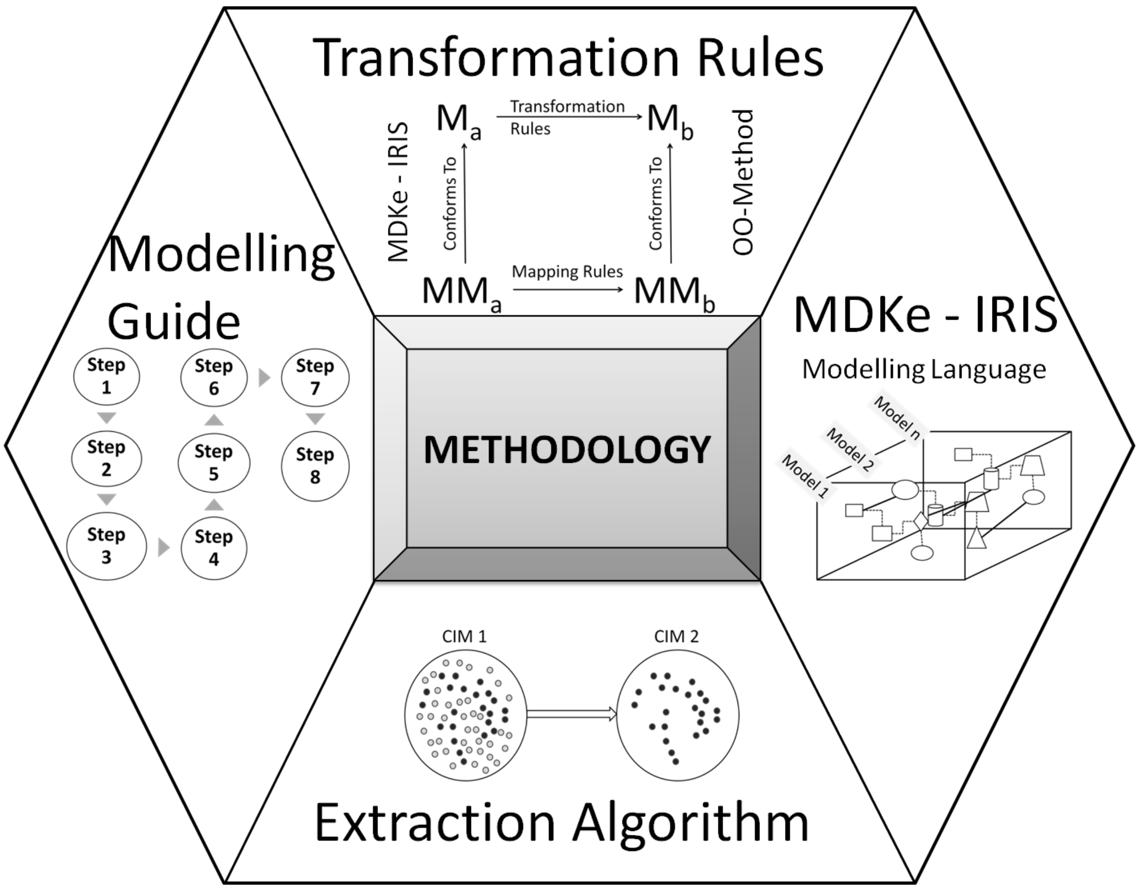


Fig. 1 MDKe-IRIS Framework Overview

4.1 Dimension 1: Methodology

Fig. 2 shows the methodology that allows the other dimensions of the framework to be employed to reuse enterprise models to generate PIM models. The methodology is split into three phases.

Phase 1: Apply the modelling guide (dimension 3) to obtain the CIM1 model using the MDKe-IRIS modelling language (dimension 2).

Phase 2: Apply the extraction algorithm (dimension 4) to the CIM1 model to obtain the CIM2 model. The CIM2 model is the part of the enterprise models that is going to be transformed in the next phase.

Phase 3: Apply the transformation rules (dimension 5) to the CIM2 Model to obtain the OO-Method model, which belongs to the PIM level.

The result of the application of the methodology is a PIM model that conforms to the OO-Method metamodel. Finally, this PIM model, together with additional information about the technological platform that must be specified, will be used to generate a PSM model.

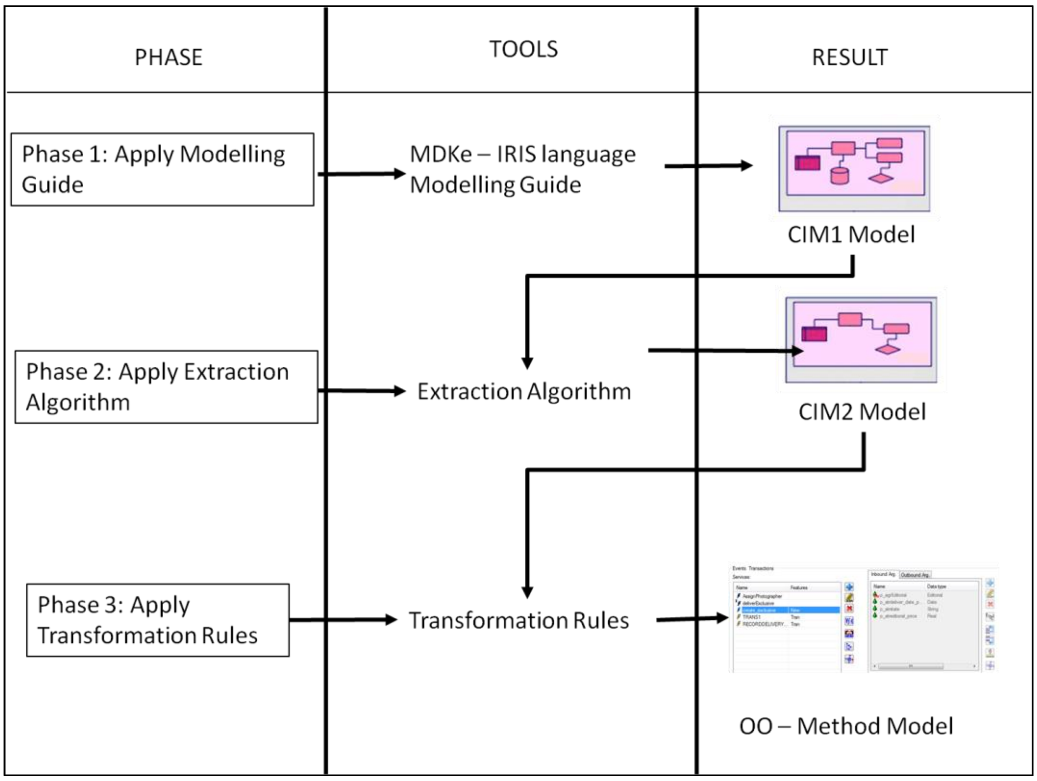

PHASE
TOOLS
RESULT
Phase 1: Apply Modelling Guide
MDKe – IRIS language Modelling Guide
CIM1 Model
Phase 2: Apply Extraction Algorithm
Extraction Algorithm
CIM2 Model
Phase 3: Apply Transformation Rules
Transformation Rules
OO – Method Model

Fig. 2 Methodology of the MDKe-IRIS Framework

4.2 Dimension 2: MDKe-IRIS Modelling Language

The MDKe-IRIS Modelling Language is an extension of the MDK Proposal. In this section, the modifications performed on the MDK Proposal are explained. The MDK Proposal is oriented towards modelling the knowledge handled within an enterprise to create an information system for knowledge management. Although it does propose metamodels for creating CIM1 models of the different views of the enterprise using a single modelling language (UML adapted with different profiles and stereotypes), these metamodels had to be improved in order to fit the purposes of the MDKe-IRIS Framework. This is essentially due to the fact that (1) in some cases the models that were proposed were too simple (for example the part of the Organisational Metamodel for the business rules model); (2) the terminology used in the metamodels was oriented towards modelling the knowledge to be implemented in a knowledge management system, instead of being a terminology that was closer to that used in the enterprise in each of the views to be modelled; and (3) the metamodels are independent, meaning that the elements that are identical in each of them are not explicitly related.

To solve these problems the MDK metamodels were improved by incorporating a set of artefacts and a system for relating the common elements in the different metamodels. More specifically, two kinds of artefacts were incorporated: metaclasses in the 'Knowledge Metamodel' and a set of inter-metamodel relations that allow the user to connect the common elements in the different metamodels and to relate the different elements in order to specify the distinct types of relations between them.

Description of the metaclasses that were incorporated

The main change in the MDK Proposal was made in the 'Knowledge Metamodel', which was renamed as the 'Domain Metamodel' in the MDKe-IRIS modelling language. The 'Domain Metamodel' does not include the elements related to the knowledge modelling that were present in the MDK 'Knowledge Metamodel', such as *KnowledgeBlock*, *OntologicalCategory*, *TargetKnowledge* and *InstanceKnowledge*. Rather, the following elements were incorporated (surrounded by a dotted square in Fig. 3):

- *DomainBlock*, which represents a block from the enterprise domain. An enterprise domain block is considered to be the conceptual grouping of elements belonging to the same view of the enterprise, as defined in the enumeration *DomainBlockType*.
- *Domain*, which represents subdivisions of the *DomainBlocks* used to group the elements of the enterprise domain conceptually.
- *DomainElement*, which represents an abstraction of the different elements used by the enterprise in its day-to-day activities and which need to be managed, such as products, documents, information, resources, and so on. It is made up of: a name, which must be unique in the model; a value to indicate the type of persistence of an element in the model, which can have two values, *True* or *False*, depending on whether an element of the domain can be discarded or not in the future within the enterprise system; and a set of associated properties represented by means of the metaclass *Property*. Even if the type of persistence could appear to be information that is hard to know at the CIM level, in our opinion, this is not so because business people know if they discard the information about the elements related with the enterprise or not.
- *Property*: this element allows *DomainElements* to be characterised and each of them must have at least one associated *Property*. The following aspects must be indicated for each property:
    - *Defining*. This indicates whether a property is defining or not. In this context the set of defining properties is considered to be the set of properties that are necessary and sufficient to characterise a *DomainElement*.
    - *Constant*. If this value is true, it shows that once the value of the property has been established it cannot be modified. Otherwise, it will be possible to modify the value of the property.
    - *Type*. This may have the following values: numerical, text, defined domain, date, currency, irrational or element population. The possible types of property were defined taking into account the fact that it must be possible to represent the basic types of data that appear in the day-to-day operation of an enterprise (either in its paperwork or in its information system). These may include numbers or dates, and more complex types of data represented by other *DomainElements*, as is the case of the element population type. This type indicates that a property takes values from the instances (the population) of another *DomainElement* present in the model. Moreover, the defined domain type indicates that the value of the property is restricted to a particular clearly defined set of values that does not coincide with the population of another *DomainElement*.
    - *Multiply*. This indicates the cardinality of a property.

The relation *isBasedOn* was also added to the model. This is a directed relation between two *DomainElements* so that if A' *isBasedOn* A, it means that the incorporation of an instance of A' within the enterprise domain depends on the existence of A. Furthermore, the attribute *type* was added for the *input* type of information flows in the 'Behaviour Metamodel' with the aim of determining the types of elements that participate in the processes and facilitate the PIM-level transformation. It can have the same values as the attribute *Type* of the properties.

Fig. 3 shows the 'Domain Metamodel', which includes the metaclasses for modelling the *DomainElements* that exist within the enterprise, their characteristics, the relations between them, and the processes in which they are involved and the sources of information associated to them. The other elements not explained before belong to the original metamodel. The elements *TacitSource* and *ExplicitSource* represent sources from where information can be obtained about the *DomainElements;* the difference between them is that an *ExplicitSource* is another *DomainElement*, so the information is represented in a computer-readable way, while *TacitSource* represents sources of information that are not computer-readable. The element *Source* is an abstraction of *TacitSource* and *ExplicitSource*. On the other hand, the *InputVariable* element represents the information inputs to characterise a *DomainElement*. The *ExtractionProcedure* element represents the procedures to be performed in order to extract the *InputVariables* from the *Sources*. The *CalculationProcedure* represents the transformations that should be applied over the *InputVariables* in order to be compatible with the target *DomainElement*. Finally, the *Procedure* element was included in the original metamodel to represent the procedural knowledge. However, in the extension of the MDK Proposal its meaning has changed. In the MDKe-IRIS Modelling Language it represents the procedures that modify the values of the *DomainElement* attributes. The representation is made by enumeration; i.e., in this metamodel he implementation of these procedures is not included because its implementation is part of the Behaviour Metamodel, which is very similar to an IDEF0 metamodel.

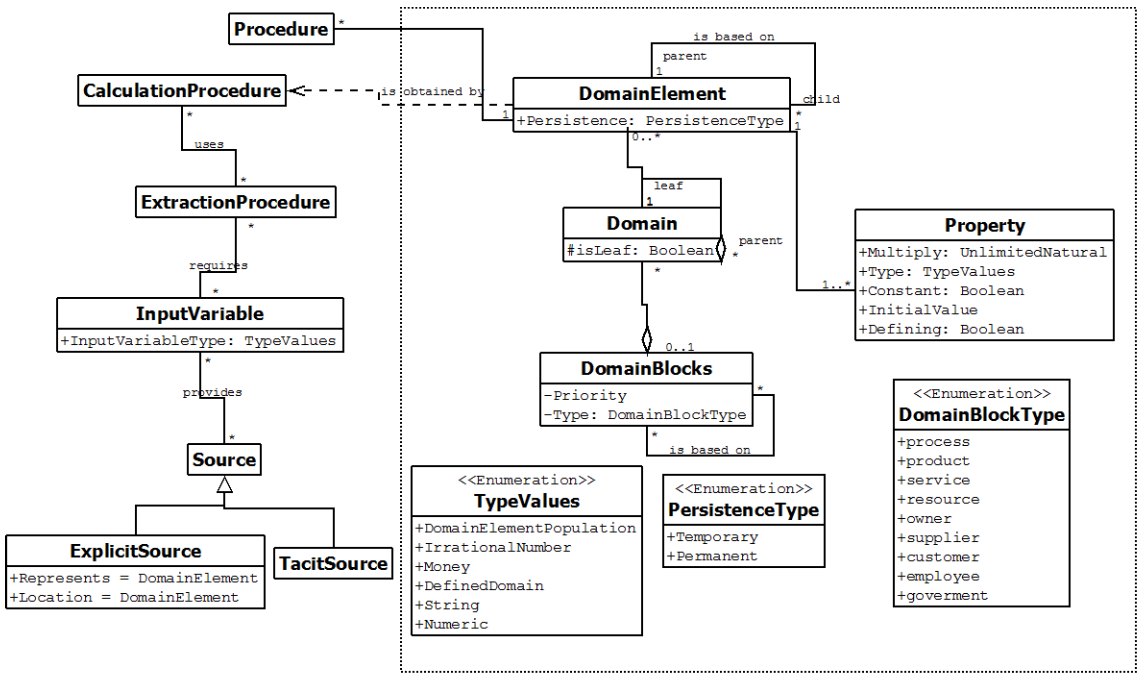


Fig. 3 Domain Metamodel

Inter-metamodel relations

The MDK proposal consists of a set of metamodels with which the main views of an enterprise can be modelled. Yet the metamodels are not explicitly related to each other, which is a drawback when it comes to automatically processing the models generated from the MDK Proposal and reusing the information that they contain to develop a computer-based conceptual model. A set of inter-metamodel relations were therefore incorporated to enable associations to be established between different elements from different metamodels. These interrelations ensure that the set of models that represent an enterprise are related to each other by means of relations with a precise semantics that makes it possible to complete the information about the enterprise offered by the model.

One of the purposes of inter-metamodel relations is to specify the relations among procedures and the business rules that these procedures must fulfil, while also establishing links between sources of information and the workers employed by the enterprise. These relations also make it possible to determine the function of a particular element in the different views of the enterprise. The *DomainElements* and the sources of information can therefore be associated to their behaviour and integrity constraints (which are formulated as business rules) at the CIM level.

These kinds of associations correspond to the structure of an application designed according to an object-oriented approach. Thus, using a typical structure from the object-oriented approach to model at the CIM level makes it easier to transform the enterprise models into object-oriented Platform Independent Models, while at the same time satisfying the objective of reusing the information that exists in the enterprise models to generate Platform Independent Models.

Including the relations among the metamodels, the resulting structure can be seen as a multilevel graph in which each level is a model and they are related to each other by means of inter-metamodel relations.

Fig. 4 shows the inter-metamodel relations that were defined among the metamodels of the MDKe-IRIS metamodels. Next it is explained the relationship among the elements in the figure and the metamodels.

BRInput, Business Rule and Task belong to the Organisation Metamodel. BRInput represents the inputs of a Business Rule, and Task represents a set of procedures to be carried out to perform a business process. The Human element belongs to the Structure Metamodel and represents the employees of the enterprise. The elements Process, Service, Process Role and Flow belong to the Behaviour Metamodel. Process and Service represent the implementation of a business process and are composed of a set of activities. Process Role represents the role played by the different resources related to a Process or a Service element and the Flow element represents the information fluxes among the processes. Finally, the elements such as Element, Uses, Possible Instances and Possible Tacit source show the elements that do not really exist in the metamodels but represent elements that have been grouped together for the sake of simplifying the graphical representation. The rest of the elements belong to the Domain Metamodel and were explained in the previous subsection.

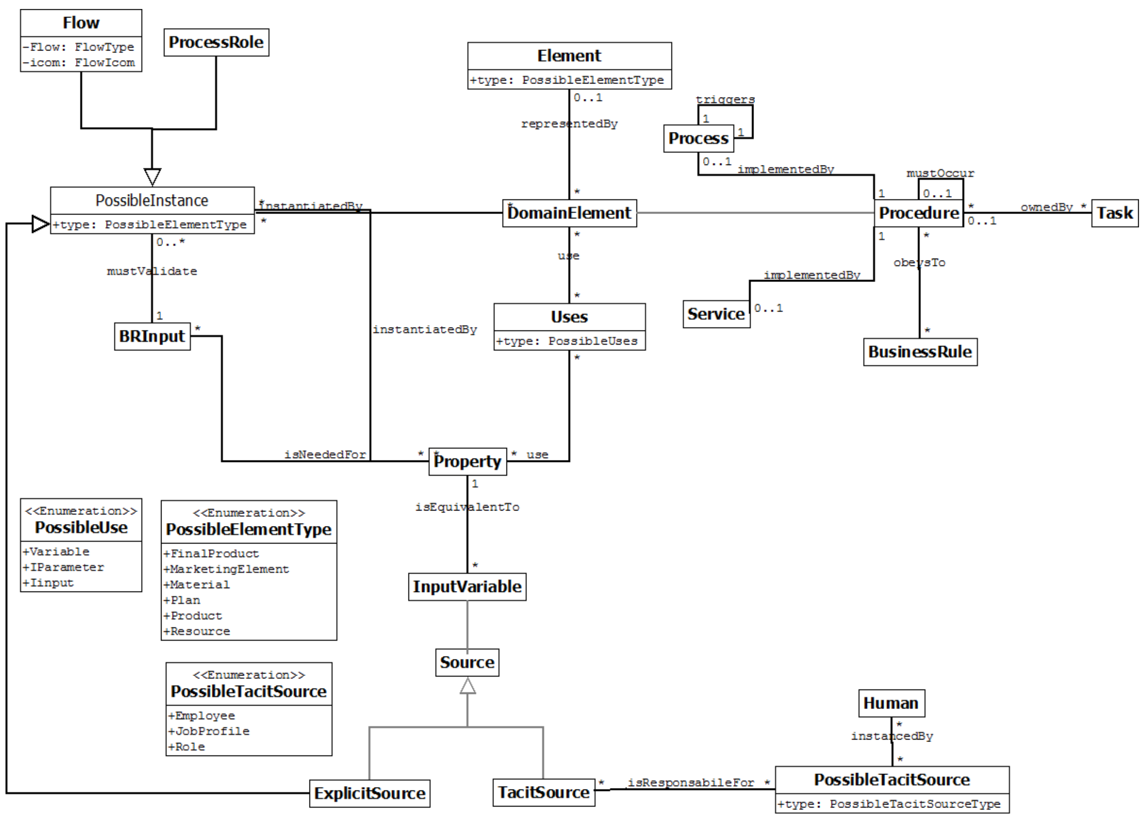


Fig. 4 Inter-metamodel relations

In the following, the meaning of each of these inter-metamodel relations is explained. Most of these relations do not represent cardinality constraints and thus, unless indicated otherwise, the cardinality of the relation must be considered to be N:N.

isResponsibleFor relates elements of the *Profile*, *Role* or *Employee* type from the ‘Organisation Metamodel’ with elements of the *TacitSource* type from the ‘Domain Metamodel’. It makes it possible to define which people are responsible for being the tacit source of information associated with an element of the domain.

instancedBy relates elements of the *Human* type from the ‘Structure Metamodel’ with elements of the *Profile*, *Role* or *Employee* type from the ‘Organisation Metamodel’. It makes it possible to define which particular people belong to a specific profile or play a particular role in the enterprise.

Use relates elements of the *Property* or *DomainElement* type from the ‘Domain Metamodel’ with elements of the *IInput*, *IParameter* or *Variable* type from the ‘Organisation Metamodel’. It makes it possible to define what properties are used as inputs, parameters or variables in the strategic business plans and centres of analysis.

mustValidate relates elements of the *inputVariable* type from the 'Domain Metamodel' with elements of the *BRInput* type from the 'Organisation Metamodel'. It makes it possible to indicate that the value of an input variable of a procedure for obtaining a *DomainElement* must fulfil the business rule.

isNeededFor relates elements of the *Property* type from the 'Domain Metamodel' with elements of the *BRInput* type from the 'Organisation Metamodel'. It is similar to the previous one, but in this case the properties do not have to be the element to be validated but instead are part of the functioning of the business rule.

implementedBy relates elements of the *Procedure* type from the 'Domain Metamodel' with elements of the *Process* or *Service* type from the 'Behaviour Metamodel'. It indicates what procedure is implemented by means of what process or service. The minimum cardinality of this relation was set to 0 so as not to impose any kind of constraint on the modelling and thereby avoid forcing the final user to model all the enterprise procedures as processes or services. The maximum cardinality is 1, since a procedure cannot be implemented by more than one process or service at the same time. This also means that a procedure should not be allowed to relate to a process and a service at the same time. The representation of the 'Relations Metamodel', however, does not allow this fact to be expressed graphically and to do so it would have to be specified by means of a constraint expressed in a language like Object Constraint Language (OCL), which is not currently envisaged in this research (Alencar et al, 2009).

ownedBy relates elements of the *Procedure* type from the 'Domain Metamodel' with elements of the *Task* type from the 'Organisation Metamodel'. It allows tasks to be related to the procedures that they are made up of and to create a bridge to the work profiles that each task must perform (which are directly related in the 'Organisation Metamodel' of the MDK Proposal).

obeysTo relates elements of the *Procedure* type from the 'Domain Metamodel' with elements of the *BusinessRule* type from the 'Organisation Metamodel'. The aim is to associate the constraints that it must fulfil at the procedure level.

mustOccur relates *Procedure* type elements from the 'Domain Metamodel' to each other. It makes it possible to model an ordered relation in which the business procedures are to be executed. The maximum cardinality was limited to 1 in order to avoid indeterminism in the execution of these procedures.

triggers relates *Process* type elements from the 'Behaviour Metamodel' to each other. The aim is to be able to model an ordered relation in which the processes belonging to the different procedures are to be executed and which must always be executed sequentially and completely. As in the case of the relation *mustOccur*, the maximum cardinality was limited to 1 in order to avoid indeterminism in the execution of these processes.

representedBy is an identity relation between an element of the *DomainElement* type from the 'Domain Metamodel' and another element from another metamodel that represents it. It indicates that the instance of a *DomainElement* is present in another model in the form of the elements defined in the *PossibleElementType* enumeration. To represent this relation graphically, a new class was created, *Element*, whose attribute *type* and its possible values, which are defined in the *PossibleInstanceType* enumeration, indicate the elements with which this relation can be established. Cardinality was defined as 0:1 because for each type of element this relation can be established with one element that instantiates it at the most.

isEquivalentTo relates elements of the *Property* type with elements of the *inputVariable* type, which both belong to the 'Domain Metamodel'. It links the values of the characteristics of the domain elements represented as input variables to the properties that characterise it in the model. In this case the maximum cardinality was limited to 1 because a property can only be associated to an element of the *inputVariable* type; in the opposite direction, however, the cardinality does not present any constraints because an *inputVariable* can be associated to more than one property.

instantiatedBy relates elements of the *Property* and *DomainElement* type from the 'Domain Metamodel' with elements of the *Flow* or *ProcessRole* type from the 'Behaviour Metamodel'. It associates the structural elements expressed in the 'Domain Model' with the function that they perform in the dynamics of the enterprise.

4.3 Dimension 3: Modelling Guide

An enterprise is a complex system and therefore creating models that reflect it is also a complex task. With the aim of making the modelling task easier, the MDKe-IRIS Framework includes a guide on how to carry it out, taking into account the professional profiles of the personnel who play a part in its development. In general, there will be two kinds of profiles: modellers and users. Modellers are the specialists in enterprise modelling who are familiar with the metamodels that are going to be used, the semantics of each element and the general functioning of the enterprise. Users are the members of staff at the enterprise who will use the models to make decisions in their areas of work. Users do not need to have any knowledge at all of the modelling language or of the concept of modelling itself.

The modelling guide proposed here consists in a series of steps to be followed. The result of the application of the guide is the CIM1 model. Thus, the process of modelling will run from aspects that are better known by the users, such as the organisational chart of the enterprise, to more abstract elements for the users, such as the modelling of the different *DomainBlocks* that exist within the enterprise. This facilitates participation by the users and allows the modellers to gradually get to know the enterprise they are working on as they construct the model. We propose that the steps should be applied sequentially but this is not mandatory. Moreover, it could be possible to change and to improve the models that have already been done if necessary. However, following the guide ensures that modellers make all the necessary models and include all the necessary information to make the resulting models suitable for transformation. The steps of the modelling guide are listed and described in the following.

Step 1: Develop the Organisation Model. It is a representation of the organisational structure. Starting with the creation of the organisational chart of the enterprise to be modelled enables the modellers to gain a fairly clear idea of the departments that make up the enterprise and the job roles and profiles that exist, while also providing them with a complete list of the tasks carried out by each department. This allows users to begin within an environment that they feel comfortable in and which they have a command over so that they can start the modelling task from something that they feel is solid ground.

Step 2: Break down the tasks detected in Step 1 into their constituent procedures. Work begins on creating the Domain Model from the point of view of the transactional functioning of the enterprise, which is the most tangible for users. At the same time, the Business Rules Model is also created according to the constraints that the procedures must fulfil (relation *mustValidate*) and the valid sequence of execution of those procedures is established, if necessary (relation *mustOccur*).

Step 3: Incorporate the elements of the domain over which the procedures defined in Step 2 are executed and complete the Business Rules Model with the characteristics that participate in the validation of the business rules (relation *isNeededFor*).

Step 4: Implement the procedures defined in Step 2 using the Behaviour Model by representing the processes and instantiate (relation *instantiatedBy*) the process flows and roles with the domain elements and properties that participate; establish the relations *implementedBy* between procedures and processes and the relation *triggers* if necessary.

Step 5: Create the part of the Structure model related to the Products and/or Services. Complete the Domain Model if new elements appear that have not been taken into account in the previous steps.

Step 6: Complete, if necessary, the Domain Model with the relations *instantiatedBy* and *representedBy* between domain elements and process, product and service roles in the models created in steps 4 and 5.

Step 7: Group the domain elements in categories and create the *DomainBlocks*.

Step 8: Create the strategic plan of the enterprise with the goals of the enterprise and the centres of analysis of the Organisation Model. This task has been left until last so that it can be performed once a holistic view of the enterprise has been obtained, since it will allow the plan to be created with full knowledge of all the details of the organisation.

4.4 Dimension 4: Extraction algorithm

The CIM model can represent a whole enterprise whereas the PIM model is a computer-based conceptual model which does not necessarily have the same scope. For this reason it is necessary to have a mechanism that allows us to distinguish and to extract the elements on the CIM level that are also part of the PIM level. This problem is solved by using the inter-metamodel relations. First, the different models that make up the CIM1 model and which are related to one another by means of the inter-metamodel relations are represented as a graph comprising a series of levels, each level being a model. On each level there are nodes and edges. The nodes represent the elements of the model. The edges represent relations between the nodes and can be of two types: those that relate nodes belonging to the same level (intra-metamodel) and those that relate nodes that are on different levels (inter-metamodel). Thus, the transformation from the model to the graph is very simple: each element in the model corresponds to a node that represents it on the graph and each of the relations between the elements of the model is an edge.

After the graph has been obtained, it is necessary to distinguish between the CIM-level elements that are going to make up the PIM level and those that are not. It must be borne in mind that the elements that comprise the CIM model represent the conceptual model of a computer system that must fulfil the requirements imposed by users and are therefore subjective. The users must select, considering the goal of the software to be generated, the domain elements, tasks and procedures that they want to be included in the computer system.

The selection of the elements that comprise the CIM1 level is carried out by a tagging mechanism represented by a check. That is, each element of the CIM1 model has a check that, if it is marked, means that this element must be part of the CIM2 level. Once chosen, the selections are automatically propagated across the graph with the elements that are needed to create a conceptual model which will satisfy the user's functional and informational needs. The propagation is performed as follows:

```
For each DomainElement selected by the user:
  Select the defining properties.
  Select the inputVariables that are equivalent to the defining
  properties (isEquivalentTo).
  Select the extraction and calculation procedures and the sources.
  For each defining property of the 'DomainElement Population' type:
  check that the corresponding DomainElement is selected.
For each task:
  Select all the associated procedures.
For each procedure:
  Select the task it belongs to.
  Select each process that implements it.
  Select the properties which appear as an input flow or with the
  stereotype other that are not defined and which belong to a
  selected DomainElement.
  Select the profile or role responsible for carrying out that task.
```

Hence, all we have to do is to select the domain elements that we want to be represented in the PIM-level conceptual model and the tasks that it must include if all the procedures associated to those tasks or the individual procedures are included. The rest of the elements are selected automatically.

To perform the discrimination task it is necessary to include a new Boolean-type attribute for each element of each metamodel so that if the attribute is true in one instance of the element, then this indicates that it must become part of the conceptual model.

Once the final user has decided what requirements the computer system must fulfil and has selected the elements of the CIM1 model that represent it at this level, the selected elements can be extracted using an extraction algorithm. This extraction algorithm is based on model slicing. Model slicing is a method that allows fragments of a model to be obtained by means of a fragmentation criterion (Hatcliff et al, 2000), (Kagdi et al, 2005). Model slicing was chosen because, in addition to fit perfectly to the goal of obtaining a subset of the CIM1 level to create the CIM2 level, it was successfully applied in several studies, such as (Bae and Chae, 2008). Following the guidelines of this method, an algorithm must be implemented to traverse the graph and if an element has been selected, it is added to an intermediate graph. The relations between other elements are transferred to this intermediate graph if these elements have also been selected (which corresponds to the fragmentation criterion). The intermediate graph is the one that contains the elements that must be transformed and the one taken as the basis on which to generate the PIM-level conceptual model. More specifically, an algorithm like the one below can be followed:

```
Let $\mathbf{G}$ be the graph of the CIM1 model
Let $\mathbf{N_c}$ be the set of nodes of $\mathbf{G}$
Let $\mathbf{A_c}$ be the set of edges of $\mathbf{G}$, where each element $\mathbf{a} \in \mathbf{A_c} = (\mathbf{n_1}, \mathbf{n_2}, \mathbf{r})$, where $\mathbf{n_1}$ and $\mathbf{n_2} \in \mathbf{N_c}$ and $r$ represents the relation that connects them
Let $\mathbf{G_p}$ be an initially empty intermediate graph
Let $\mathbf{N_p}$ be the set of nodes of $\mathbf{G_p}$
Let $\mathbf{A_p}$ be the set of edges of $\mathbf{G_p}$, where each element $\mathbf{a} \in \mathbf{A_p} = (\mathbf{n_{p1}}, \mathbf{n_{p2}}, \mathbf{r})$, where $\mathbf{n_{p1}}$ and $\mathbf{n_{p2}} \in \mathbf{N_p}$ and $\mathbf{r}$ represent the relation that connects them
For every $\mathbf{n_i} \in \mathbf{Nc}$:
    If $\mathbf{n_i}$ is selected:
        Add $\mathbf{n_i}$ to $\mathbf{N_p}$
    For every $\mathbf{a} \in A_c$ and ($\mathbf{a}.\mathbf{n_1} = \mathbf{n_i}$ or $\mathbf{a}.\mathbf{n_2} = \mathbf{n_i}$):
        If $\mathbf{a}.\mathbf{n_1} = \mathbf{n_i}$ and $\mathbf{a}.\mathbf{n_2}$ is selected:
            Add $\mathbf{a}$ to $\mathbf{A_p}$
        If $\mathbf{a}.\mathbf{n_2} = \mathbf{n_i}$ and $\mathbf{a}.\mathbf{n_1}$ is selected:
            Add $\mathbf{a}$ to $\mathbf{A_p}$
    End for
End for
```

By following this algorithm, the nodes (elements of the model) and the edges (relations) to be transformed in order to generate the computer-based conceptual model are left in N_p and A_p. The next step is to convert the graph into the CIM2 model that only represents the user's requirements regarding the computer system to be constructed, and the transformation rules outlined in section 0 are executed on this CIM2 model. This is achieved by performing the inverse operation to the one carried out when creating the graph. Each node is an element of the CIM2 model and each edge represents a relation between the elements.

4.5 Dimension 5: CIM2 to PIM transformation rules

In this section we describe the transformation rules that make it possible to generate the different models that comprise a computer-based conceptual model according to the OO-Method approach. First, we explain the relations established between the metamodels in the MDKe-IRIS Framework and the Object Model, and then the generation of the Dynamic Model is described. To be able to establish the rules for transforming metamodels at a CIM level to a PIM level, a well-structured CIM level language is needed, because otherwise it would be almost impossible to perform the transformation (Zhang et al., 2005). In this regard, the modelling language used in the MDKe-IRIS Framework simplifies the transformation process at the PIM level because the elements that denote both behaviour ('Behaviour Diagram' and the *Procedure* type elements from the 'Domain Diagram') and structure (the rest) are well defined. This is because the role of each element depends on whether the type of UML element from which it inherits its attributes is structural or behavioural. Furthermore, the inter-metamodel relations facilitate the extraction of the types of relations among the different elements because they establish explicit relations between the elements that denote structure and the associated behaviour.

4.5.1 Generation of the Object Model

In this section we describe the rules established for transformation between the components of the metamodels in the MDKe-IRIS Framework and the elements that make up the Object Model of the OO-Method. First, details are given of the generation of the structure of the classes, attributes and relations, then the generation of services and, finally, the generation of the system of agents are both outlined.

Structure of classes and relations

Rule 1: DomainElement → Class: Every DomainElement is transformed to a Class. This rule allows the creation of the set of classes for the future computer application. The name of the *DomainElement* becomes the name of the new class.

Rule 2: Property → Attribute: Each Property is transformed into a class attribute. This rule allows the creation of the set of attributes of a class for characterising the real-world object that it represents. The name of the property now becomes the name of the new attribute. Moreover, the characteristics of the property are transformed into characteristics of the attributes. Table 1 provides details of the values that each characteristic has, depending on the values defined at the CIM level in the properties. The first column shows the rule number that has been assigned. Two characteristics of the properties, Multiplicity and the Element Population type, have been omitted in this table because their transformation follows an algorithm and is not a one-to-one transformation as in the other cases. The way the values of each of these characteristics are treated is explained in the following.

Table 1. Transformation rules for characterising an OO-Method attribute from the properties of the domain elements

	MDK Proposal		OO-Method	
R	Property	Value	Attribute	Value
2.1	isEquivalentTo relation	True	Request upon creation	True
	Not isEquivalentTo relation	False	Request upon creation	False
2.2	Constant	True	Type	Constant
	Constant	False	Type	Variable
			Add to edit event	
2.3	Defining	True	Nulls allowed	False
	Defining	False	Nulls allowed	True
2.4	Type	Numeric	Data Type	Integer
	Type	Text	Data Type	String
	Type	Money	Data Type	Real
	Type	Irrational	Data Type	Real
	Type	Date	Data Type	Date
		Defined Domain	Not supported	

Rule 3: Treatment of Multiplicity: The following algorithm describes how to treat the multiplicity of a property in the transformation process. However, in an informal way it could be summarised as follows: if a DomainElement can have more than one value for a property, then a new class is added to the Conceptual Model at the PIM level that respects the cardinality established at the CIM level.

```
procedure treatMultiplicity (Property p)
    if p.Multiplicity = 1 then: do nothing
    else
        create new Class c
        c.name = p.name
        // p.parent is the DomainElement that the attribute belongs to
        createRelation(relationName, c, p.parent)
        if p.Defining then: relationName.cardMin = 1
        else: relationName.cardMin = 0
        fi
        relationName.cardMax = p.Multiplicity
    fi
end treatMultiplicity
```

Rule 4: Treatment of Element Population: If a DomainElement has a property whose values belong to a DomainElement, it is transformed into a relation between the two classes that represent the DomainElements involved. Moreover, if the property which is being treated is defined, then the minimal cardinality of the resulting relation must be one. Next, a pseudo-algorithm that implements this rule is shown.

```
procedure treatElementPopulation(Property p)
    if p.type.name = "elementPopulation" then
        if (createRelation(relationName, p.type.DomainElement, p.parent)) then
```

```
            if p.defining then: p.parent.relationName.cardMin = 1
            else: p.parent.relationName.cardMin = 0
            fi
            p.parent.relationName.cardMax = p.Multiplicity
            // Cardinality restrictions are defined in both directions
            p.type.DomainElement.relationName.cardMin = 0
            // N cardinality means that there is not a specific limit
            p.type.DomainElement.relationName.cardMax = N
            if not p.Constant then: relationName.dynamic = True
            fi
        fi
    fi
end treatElementPopulation
```

Rule 5: relation isBasedOn. The aggregation and composition of objects is calculated from the instances of the relation *isBasedOn* between two *DomainElement* type elements (see 'Domain Metamodel'). Depending on whether participation is mandatory or not in the relation that is instantiated in the model, it will be an aggregation or a composition, respectively.

Services

Rule 6: Add a creation service to all the object model classes. Since all the classes of the Conceptual Model have a creation service, for each class a creation service is added automatically. The creation service is completed with the information in the CIM model as follows: the input parameters of the creation service coincide with the input variables that were defined as equivalents of the properties (*isEquivalentTo*) and were defined in the conceptual model by 'request Upon Creation'=True.

Rule 7: Add an edit service to all the object model classes if the domain element has properties that are not constant. That is, if there are properties that could be modified during the life cycle of the object, then it is necessary to add an edit service to change these values. In this case the input parameters of the edit service coincide with the properties that are not constant at the CIM level.

Rule 8: Add a delete service to all the object model classes if the persistence of the domain element from those which have been created is established as being false. Keep in mind that if the persistence is false, then it is possible that the DomainElement may be discarded in the enterprise in the future, so the class that represents it at the PIM level must have a delete service. From the authors' experience, even if the information of a DomainElement could be discarded in the future, it is never physically deleted from an information system; it is usually marked as *not active*. However, from our point of view, to specify this information at the CIM level implies breaking the abstraction level, since it is a detail of the implementation of the software and not a detail of the functioning of the enterprise.

Rule 9: Process → Service. A new service is created in the conceptual model for each procedure selected at the CIM level. The name of the CIM level procedure becomes the name of the service created at the PIM level. The class to which this service is associated will be the one generated from the domain element that the procedure belongs to. It must be noted that different types of services are defined in the OO-Method: own and shared. Own services affect the state of a single object, whereas the shared ones affect the state of several objects. The reason for making such a differentiation is based on the types of flows related to the procedure that is being dealt with, and is as follows:

9.1: If all the flows stereotyped as *other* have as their target properties that belong to the domain element that owns the procedure, then it is an own-type service. That is, if all the flows point to the same object, then the procedure only modifies the state of a single object, so its type must be *own*.

9.2: If there is any flow stereotyped as *other* that has as its destination properties of other domain elements, a shared service is added between the classes that correspond with the target domain elements of this type of flow. So, if a procedure modifies different objects, its type must be *shared*.

9.3: If the process includes a stereotyped flow such as *other* whose target is a domain element (and not one of its properties), a transaction is created. The transaction includes an own service that carries out the changes of state on the corresponding properties of the domain elements that the procedure belongs to.

Additionally, it also invokes the service creating the domain element that is the destination of the flow *other*. If several domain elements are registered, this is not a problem because each of the corresponding creation services is invoked. The order in which these services are invoked will be the same as the one established by the activities that set up the processes at the CIM level.

9.4: If the process gives rise to a relation *triggers*, create a new transaction that includes all the processes linked together by means of the relation *triggers*. Note that the input arguments of the transaction coincide with the set of input arguments of all the services involved and that they are not the destination of an *other* type flow from a previously involved process (they change their state) in the chain of triggers. The name of the transaction is that of the service that originates the relation *triggers* and the services that include the transaction are flagged as internal so that they are not visible to the user.

Rule 10: Flow that is stereotyped as *input* → input parameter of the service created from the process it belongs to. That is, the input flows are transformed into input parameters.

Rule 11: Flow that is stereotyped as *other* → changes of state carried out by the service created from the process it belongs to. If the destination of the flow is a property of a domain element, then it is an evaluation of that property. If the destination is a domain element itself, then the creation process of that element is invoked.

Rule 12: Preconditions. A precondition is added to a service if there is an *obeysTo* relation between the procedure which that service was created from and a business rule. The added precondition is then used to create the signature and the input parameters are indicated from the relation *isNeededFor* and *mustValidate* of the properties and input variables respectively.

Agents

Rule 13: From the *isResponsibleFor* relation between a JobProfile/Role/Employee and a tacit source: an agent is created with visibility over the creation, edit and delete services of the domain element related with this tacit source. The agent thus created has visibility over the class attributes and the relations obtained from rules 3 and 4 (treatment of multiplicity and of the population element type). That is, for each relation isResponsibleFor an agent is created with permissions to execute the creation, edit and delete services.

Rule 14: From the *ownedBy* relation between a procedure and a task, an agent is created to represent the job profile responsible for carrying out this task. The agent thus created has permission to carry out procedures associated with the task and visibility over its input and output parameters. That is, even if an agent is not responsible for a domain element, like in the previous case, it is possible that it is responsible for performing other tasks. This rule covers this case.

Rule 15: Incorporate a global system agent. Generally, all information systems have an administrator with permissions over all the classes and services. This rule does not transform anything. However, it covers this case.

4.5.2 Generation of the Dynamic Model

The Dynamic Model of the OO-Method consists of a set of states and transitions that determine the sequences of actions that are valid for an object. The OO-Method proposes a default status diagram for each class in which there are no constraints on the sequence in which the services are executed. The default status diagram is valid provided that there are no *mustOccur*-type relations between the procedures from which the services have been created. Otherwise, the order in which the services are executed must respect the order imposed at the CIM level. Furthermore, it must be remembered that it is possible that not all the procedures are flagged to become part of the PIM level and will therefore not be shown on the status diagram. However, this is not a problem because the *mustOccur* relation is an order relation and is thus transitive. Hence, if the procedure is not represented at the PIM level and is located between two *mustOccur*-type relations (one input and the other output), the input relation goes to the procedure targeted by the *mustOccur* output relationship and the intermediate procedure, and its output relationships are not represented. The algorithm that allows the Dynamic Model to be generated is given in the following.

Rule 16: For each class of the Object Model whose CIM-level procedures do not have mustOccur interrelations, create a default Dynamic Model. Otherwise, create the Dynamic Model using the following algorithm:

```
STEP 1: Eliminate procedures not selected to be part of the PIM level.
STEP 2: Group procedures that are part of the same transaction in a single node.
STEP 3: Create an initial and an intermediate state and join them by means of an edge that indicates the transition of the creation of the object.
STEP 4: Create the final state.
// Procedures are scanned following the order established by MustOccur relation
STEP 5: For each procedure:
        STEP 5.1: Create a new node in the Dynamic Model
        STEP 5.2: For each node that is created:
                STEP 5.2.1: Add an edge connected with the previous node. The transition tag matches the name of the service or transaction.
STEP 6: If there is a delete service:
        STEP 6.1: Add a transition between the final node of the sequence of procedures and the final node.
STEP 7: If the edit service exists:
        STEP 7.1: Create a cyclical transition at each node for the edit service, except for the initial and final node.
```

5 Case Study

In this section an example application of the MDKe-IRIS Framework is explained. In keeping with the MDKe-IRIS methodology, the CIM1-level model was carried out following the modelling guide. Then the part of the CIM model on which the computer system is to be created (CIM2 model) was extracted and finally the transformation rules were executed to create the corresponding PIM model. It must be noted that the practical example outlined here focuses on only one business process of the enterprise, and more specifically on the management of exclusive reports carried out in a photographic agency.

The case study has been developed following the guidelines defined in (Runeson & Höst, 2009). In this work, the authors establish that a case study must be developed following these four phases:

- Case study design: In this phase, the objectives of performing a case study are defined.
- Preparation for data collection: In this phase, procedures and protocols for data collection are defined.
- Collecting evidence: This is the application of the proposal, the MDKe-IRIS Framework in this case.
- Analysis of collected data: This is the validation of the research hypothesis.

Next, the application of these guidelines to a case study to validate the MDKe-IRIS Framework is explained.

5.1 Case Study Design and Planning

During this phase, the objective of the case study, its context, research questions, methods and selection strategies were defined. In our case, the objective of the development of this case study is to demonstrate that *it is possible to reuse the knowledge depicted in enterprise models for code generation following the MDKe-IRIS Framework*. To validate this hypothesis we establish three research questions to be analysed after the application of the MDKe-IRIS Framework to the case study:

Q1. – Does the modelling guide help? The objective of this question is to consider whether the application of the modelling guide was useful or not.

Q2. – Is the modelling language selected (extension of the MDK Proposal) expressive enough to allow practitioners to build software starting from enterprise models? This question was established to study the scope of the modelling language designed regarding the enterprise.

Q3. – Do the transformation rules and the extraction algorithm make the most of enterprise models? The aim of this question is to establish which elements of the PIM level were not obtained from the CIM level and to study whether it is possible to obtain them by changing the transformation rules.

The case study selected is a holistic case study that embraces all the processes carried out in a photographic agency (small-sized enterprise). Nevertheless, in this paper we only show a unit of analysis from it. Specifically, the process of managing exclusive reports (hereinafter: exclusives) in the photographic agency. The process is as follows:

The clients, in this case publishing houses or publishers, ask for exclusives about topics that are not covered by the agency in the general reports it has available. The publishers state the price they are willing to pay for the exclusive. Each request is dealt with by the technical department, where a paper index card is filled in with details of the exclusive and a code that is assigned to each one. A copy of this 'Exclusive index card' is posted on a notice board in the agency.

The photographers are in permanent contact with the agency to see which exclusives have been requested by the publishers. If a photographer is interested in a particular exclusive, he/she takes the index card from the notice board and goes to the technical department to have it assigned. Each exclusive is assigned to a single photographer, the only condition being that they do not have any previously assigned exclusive still pending submission. When the exclusive is assigned, the photographer is given a copy of the Exclusive index card so that he/she has all the details of the request.

When a photographer finishes the exclusive, it is delivered to the agency's Production Department together with a copy of the Exclusive index card. There, the date of delivery is recorded and a cheque is made out in the payee's name for 40% of the price of the exclusive (which is the photographer's part). The exclusive is then sent with a delivery note to the publisher by courier. Once the exclusive has been delivered, the courier returns the signed delivery note to the Commercial Department at the agency. The process of expedition of the bill for the total amount of the price of the exclusive is not covered in this case study.

The case study design and planning as well as a summary of the MDKe-IRIS Framework were embodied in the Case Study Protocol Document that is the result of the first phase. Once the protocol had been accepted by the agency, procedures for collecting data were carried out.

5.2 Preparation for data collection

The MDKe-IRIS Framework does not define the techniques for data collection from the users. In this case, study data were collected by open-ended interviews (Goguen & Linde, 1993) and a review of the paper documentation that the enterprise uses in its daily work. The following strategy was used to select the people to be interviewed. First, interview the CEO of the company to obtain data with which to represent the organisational structure of the enterprise. Then, the rest of the data were obtained from the person responsible for each department of the enterprise and one employee that performs operational tasks per department. That was possible because it was a small enterprise and so the total number of people was six and by interviewing only six people we had a representation of each role in the enterprise.

5.3 Collecting evidence

In this case study, collecting evidence is the application and evaluation of the MDKe-IRIS framework. Following the MDKe-IRIS methodology, the application is organised in three phases.

5.3.1 Phase I: Applying the modelling guide

The first phase is the application of the modelling guide. The result is the CIM1 model of the case study. Next, each of the eight steps of the modelling guide are explained.

STEP 1: Organisational Structure Diagram. The first step is to create the Organisational Structure Diagram (in other words, the organisational chart) for the Photographic Agency, including the tasks carried out in each department. This diagram is created by using the metamodels of the MDK modelling language. In this case, there are three departments: Technical, Commercial and Production, which are responsible for the different business processes carried out by the Agency. Moreover, the Management

Department is made up of the managers from the other three departments. It should be noted that the tasks related with the management of exclusives are performed in the Technical and Production departments. Fig. 5 shows the result of this step for the case study.

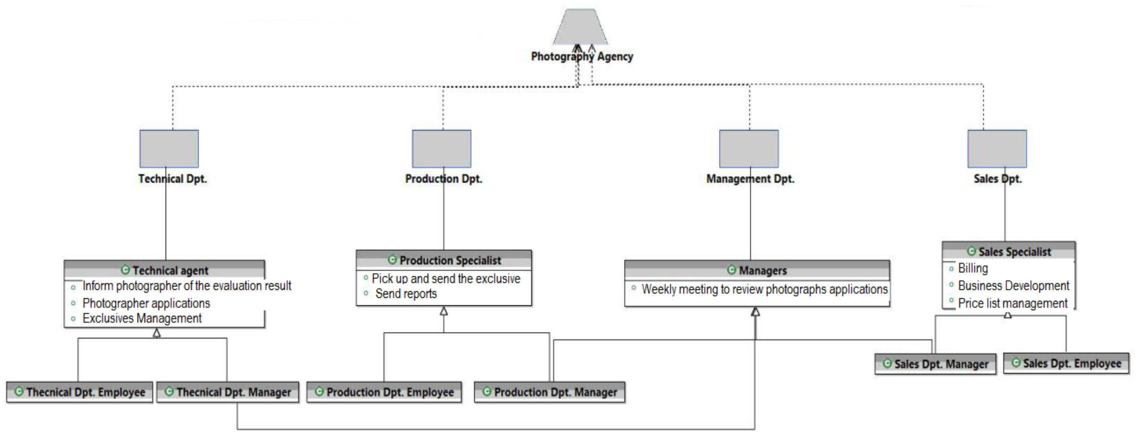


Fig. 5 Organisational Structure Diagram for the Photographic Agency

STEP 2: Breakdown of tasks into procedures. Interviews were conducted to establish the procedures that the enterprise carries out in each department. Moreover, paper documentation was reviewed for each procedure and the interviewer had a copy of them to allow him to check them if necessary. Subsequently, the tasks and procedures carried out for each department (relation OwnedBy) are explained:

For the Technical Department:

• Photographer applications: If a photographer wants to work in the Agency, they should fill in an application form that the manager will evaluate. This is composed of only one procedure: 'Manage photographer application'.

• Inform photographer of the result of the evaluation: The photographers' applications are evaluated in the managers' weekly meeting. It is composed of the procedure 'Inform photographer'.

• Task 1: Management of exclusives (Technical Department): This task is composed of the procedures: 'Record the details of the exclusive' procedure, 'Post the notice on the board' procedure, and 'Assign a photographer' procedure.

For the Production Department:

• Task 2: Pick up and Send the exclusive (Production Department): This task is composed of the procedures: 'Record delivery of the exclusive' and 'Send the exclusive' procedures.

• Send reports: This refers to sending a report to a publisher. It is composed of the procedure 'Send report to publisher'.

For the Managers:

• Weekly meeting to review photographers' applications: On Monday mornings, the managers hold a meeting to talk about the applications of the photographers, among other questions. This task is composed of one procedure 'Weekly meeting'.

For the Sales Department:

• Billing: This task is composed of the procedures: 'Send Quotation of reports to publishers', 'Manage delivery notes' and 'Billing'.

• Publisher Gaining: This task is composed of 'Design of advertising campaign' and 'Mailing'.

• Price list management: This task is composed of 'Price list review', 'Offers review' and 'Special Customer's price list'.

In this step, it is also necessary to establish the order in which the procedures are to be executed by means of the mustOccur relationship, if it is important. For the practical case, the order of execution is as follows: 'Post the notice on the board', 'Assign a photographer', 'Record delivery of the exclusive' and 'Send the exclusive'.

Finally, the business rules associated to the procedures (relation obeysTo) are represented. In this case the rule to be fulfilled by the procedure 'Assign a photographer' is the fact that photographers cannot ask for a new exclusive if they still have one outstanding.

All this data is archived in two ways: MDKe models and cards. Models were defined following the remaining steps of the modelling guide. Next, these steps are explained.

STEP 3: Domain Elements

The elements involved are: the actual exclusive itself, the publisher that requests the exclusive, the photographer that does it and the courier firm that delivers it to the publisher, which are the domain elements represented in Fig. 7. In section 5.3.2. Since the graphic model does not allow the properties of the domain elements to be viewed, some of these characteristics are detailed in Table 2 for the DomainElement Photographer. The processes of extraction and calculation of the domain elements, as well as the input variables, are also incorporated in this step.

Table 2. Examples of the characterisation of the properties of the DomainElements

Exclusive DomainElement

Property	Defining	Const.	Multi.	Type
Photographer	NO	YES	1	Population element Photographer
Title	YES	YES	1	Text
Price of the exclusive	YES	NO	1	Currency
Publisher	YES	YES	1	Population element Publisher
Request Date	YES	YES	1	Date
Assignment Date	NO	YES	1	Date
Photo number	YES	YES	1	Number

The case of the input variables is managed as follows: there is an input variable for each defining property (relation *isEquivalentTo*). Furthermore, other input variables can also be defined, although no defining variables can be part of the record of an object in the enterprise system. Finally, details must be given about the sources of information that provide the values of variables and associate them to elements of the Organisational Structure Diagram, if necessary (relation *isResponsibleFor*).

STEP 4: Process Diagrams

The next step is to represent the business procedures by means of process and service diagrams, and to establish the *implementedBy* relations. In the following, the functioning of the processes related with the exclusives is explained.

Post notice on the board. This process consists of two activities: making a copy of the exclusive and posting it on the board. Posting the exclusive on the board has a flow from type element population of Exclusive as an input, and the mechanism associated to it is a member of the technical department and an output flow (o*ther* type), which indicates the change of state on the notice board and instantiates an Exclusive-type element.

Assign photographer. This process consists of only the activity of making a record of the photographer responsible for the exclusive and the date it was requested. The inputs of the process are a date-type data item and a Photographer element population-type data item. The changes of state, stereotyped with *other*, are instantiated with the properties Date of Assignation and Photographer from the element Exclusive.

Record delivery of the exclusive. This process consists of two activities: filling in the Exclusive index card with the date on which it was delivered and issuing the cheque to pay the photographer his or her fee of 40% of the total cost of the exclusive.

Send exclusive. The process of sending the exclusive to the publisher, which is made up of the following activities: generating the delivery note and contacting the courier service so that it can be sent.

***implementedBy* relation**. Fig. 6 shows an example of the relationships between tasks, procedures and processes. In particular, it offers a graphical representation of the *ownedBy* relationships between the procedures and the tasks that they belong to, as well as the *implementedBy* relationship between procedures and processes. In addition, it indicates that the procedure ‘assign Photographer’ has a business rule associated to it, i.e. ‘ConstraintsPhotographer’. This business rule must validate the fact that the photographer, who is the input of the process, has no outstanding exclusives. Note that the procedure of recording the details of the exclusive are not reflected as such in Fig. 6. This is due to the fact that the incorporation of instances of domain elements is characterised by means of the extraction and calculation procedures, and the modellers must know that this process is not to be represented as a procedure.

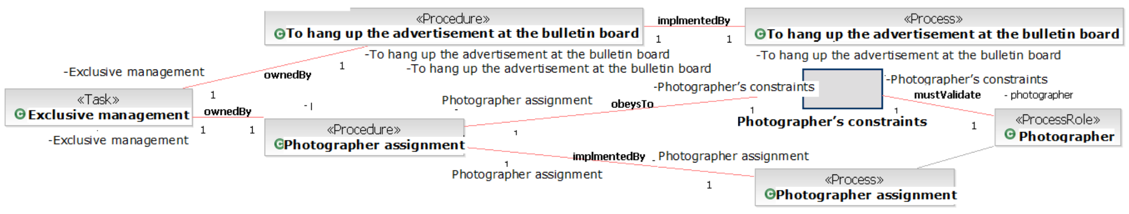

«Procedure»
To hang up the advertisement at the bulletin board
implmentedBy
«Process»
To hang up the advertisement at the bulletin board
-To hang up the advertisement at the bulletin board
-To hang up the advertisement at the bulletin board
-To hang up the advertisement at the bulletin board
-Exclusive management
ownedBy
«Task»
Exclusive management
ownedBy
-Exclusive management
Photographer assignment
obeysTo
-Photographer’s constraints
-Photographer’s constraints
mustValidate
- photographer
Photographer’s constraints
«ProcessRole»
Photographer
«Procedure»
Photographer assignment
implmentedBy
Photographer assignment
Photographer assignment
«Process»
Photographer assignment
1
1
1
1
1
1
1
1
1
1
1
1

Fig. 6 ownedBy and implementedBy relationships

STEP 5: Product Diagram

In this case the product diagram only includes the exclusive together with the materials that it is made up of (i.e., the photographs taken by the photographer), who is the person responsible for that material, and the production processes (the taking of the photographs). Note that if the production processes were carried out by the enterprise, the procedure and the corresponding process that is implemented would be part of the domain and the process diagrams, respectively.

STEP 6: Completing the domain diagram

In this case, a new element appears in the process of recording the delivery of an exclusive, i.e. the cheque that is handed over to the photographer as payment for the exclusive. On the other hand, in the process of sending the exclusive two elements appear: the delivery note and the individual photos that make up the exclusive. Fig. 7 shows the Domain Diagram of the practical case. It must be noted that the extraction and calculation procedures have not been represented in order to make the model easier to read. In this case, and in the same way as in step 2, the properties of the domain elements incorporated into this step are also characterised.

STEP 7: Abstraction in ontological categories and conceptual blocks

The criterion used to group the domain elements in conceptual blocks was based on whether they belonged to the same view within the enterprise or not. In this case, there are four conceptual blocks: Product, Client, Resources and Administration, which encapsulate the DomainElements represented in Fig. 7.

STEP 8: Creation of Analysis and Objectives diagrams

This practical case does not include these diagrams because the original specification of requirements does not include information in this respect.

5.3.2 Phase II: Applying the extraction algorithm

Following the MDKe-IRIS methodology, the second phase is to represent the CIM 1 model obtained in the previous phase as a graph. After that, the domain elements, tasks and procedures that set up the CIM2 level are identified and selected. Then the selection is propagated over the graph. Finally, the extraction algorithm is applied to obtain the CIM2 model, which reflects the characteristics of the computer system in terms of its information requirements and its functional requirements.

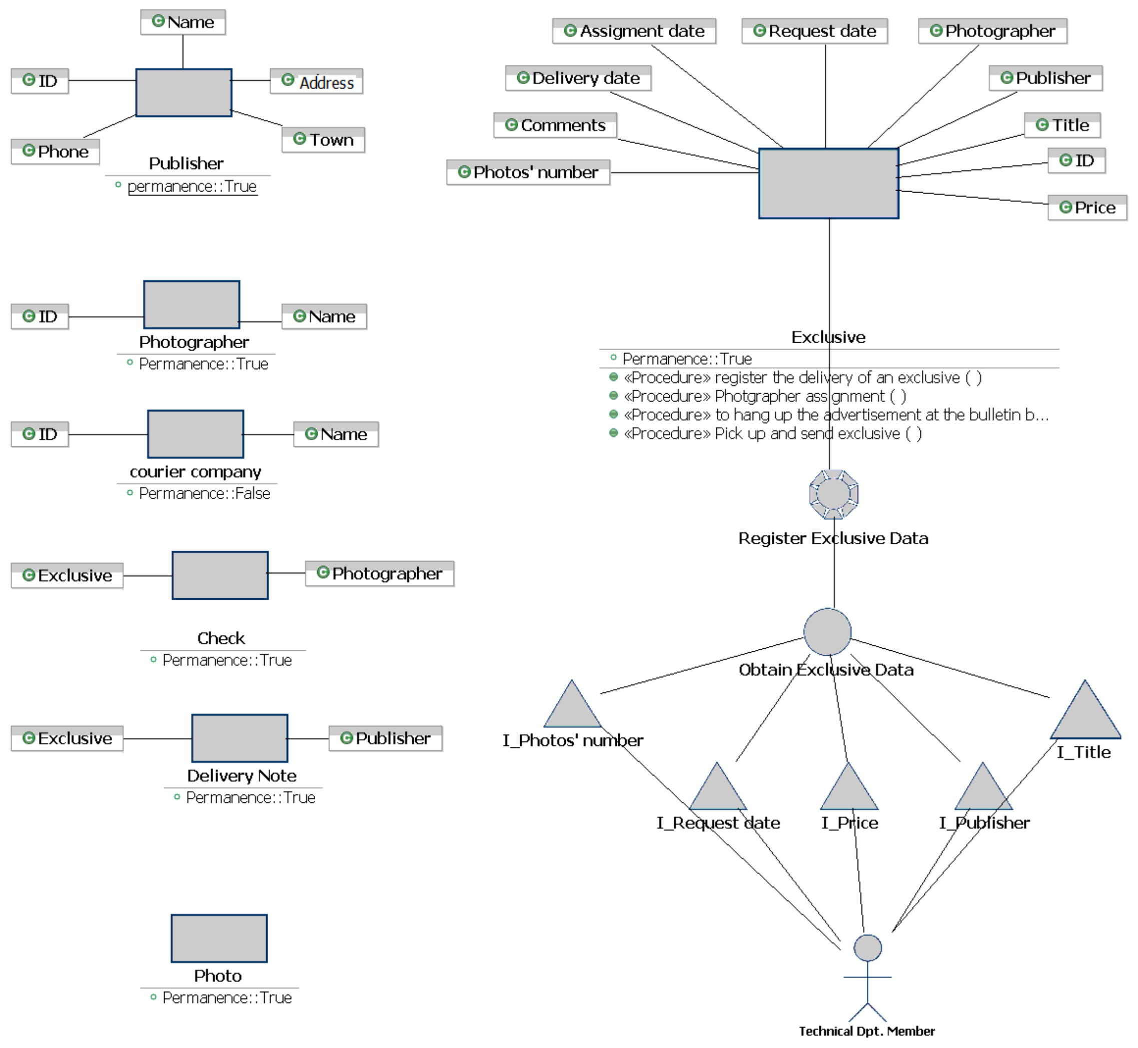


Fig. 7 Domain Diagram

Fig. 8 shows the graph obtained from the CIM1 model of the practical case. With the aim of simplifying the representation and making it easier to read, the properties have been grouped as a single node and most of the elements of the organisational structure, as well as those of the product diagram, have been omitted. In the figure, the elements that are selected manually are highlighted in grey. The elements surrounded by a diamond shape and those filled in dark grey are those that have been selected automatically from the initial selection. Finally, the elements of the organisational structure diagram that will be associated with the agents of the system at a later stage are surrounded by a rectangle. These elements are selected automatically because they are responsible for the basic services of the domain elements that are selected manually. The elements that are selected both automatically and manually are the ones that will become part of the PIM model and are therefore the only ones the transformation rules will be applied to. The elements can be extracted by means of the Dimension 4: extraction algorithm.

5.3.3 Phase III: Applying the transformation rules

Following the MDKe-IRIS methodology, the third and last phase is to apply the transformation rules over the CIM2 model. Next, the results of applying each of the rules to the case study are indicated. Furthermore, screenshots of the resulting PIM model created with the computer application Olivanova (Care Technologies, 2010) are included.

Rule 1: DomainElement → Class. Applying rule 1 to the domain elements of the CIM2 model to be transformed produces the following classes: Exclusive, Publisher, Photographer and Delivery Note.

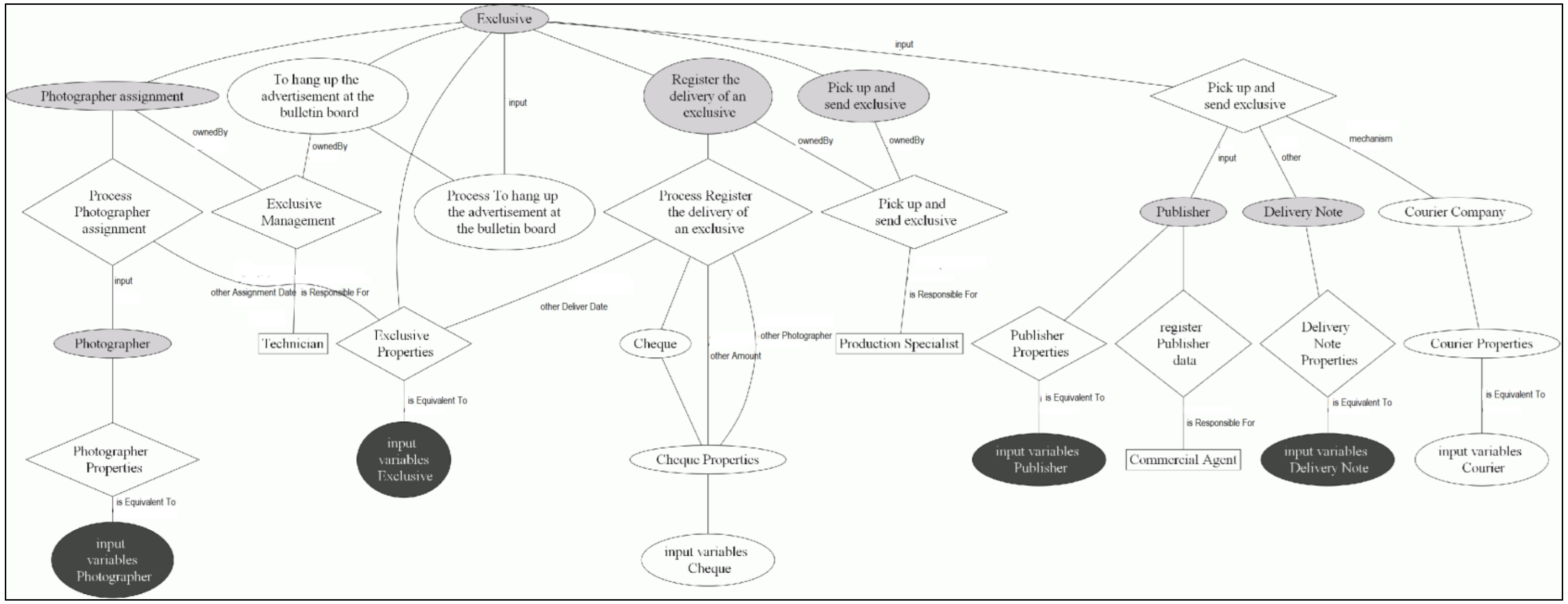


Fig. 8 Representation of the CIM1 model as a graph

Rule 2: Property → Attribute, Rule 3 and Rule 4. Applying rule 2 to the properties of the domain elements produces a set of class attributes generated from them. The attributes are characterised by applying rules 2.1 to 2.4, rule 3 and rule 4. As an example, it is shown how the rules are applied for the property Photographer of the domain element Exclusive; the rest of the attributes are characterised in the same way.

Property Photographer:

Rule 2.1: Relationship of equivalence with an input variable: NO -->'request Upon Creation' = False.

Rule 2.2: MDK::Constant = YES --> OOM::Type = Constant.

Rule 2.3: MDK::Defining = False --> OOM::Nulls allowed = True.

// The attribute OOM::type is modified.

OOM:: 'request Upon Creation' = False --> OOM::Type = Variable.

Rule 4: MDK::Type = Population Element Photographer --> OOM::Relationship between Photographer and Exclusive.

Minimum cardinality = 0 (defining = no).

Maximum cardinality = 1 (multiplicity = 1).

Fig. 9 shows the Object Model resulting from the application of rule 1 and rule 2 to the properties of the Exclusive.

Rule 5: Since the domain diagram does not include any examples of the relation *isBasedOn*, this rule was not applied.

Rules 6, 7 and 8: These rules refer to the incorporation of creation, edit and delete services respectively. In the case study, no class has a delete service because the data concerning the domain elements are stored permanently. As regards the edit service, since the exclusive and the publisher are the only domain elements that have non-constant properties, the classes that are obtained from them will also include an edit service.

Fig. 10 shows the creation and edit services incorporated into the class Exclusive. The creation and edit service arguments are added automatically from the characteristics of the attributes 'Request Upon Creation' and 'add to Edit event'.

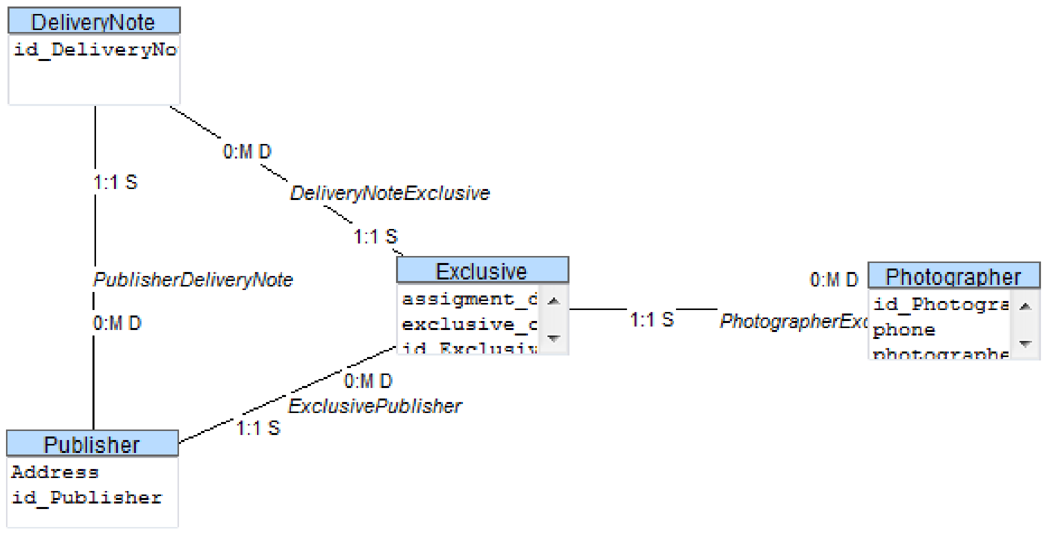


Fig. 9 Rules 1, 2, 3, 4: Structure of classes and attributes

Rule 9: Creation of services: In this case we have only a simple service, 'Assign Photographer', since the other services are part of a transaction. Thus, the following rules are applied:

Rule 9.1: Process Assign Photographer → Own service assignPhotographer.

Rule 9.3: From the process ‘Send Exclusive’ a transaction is created to invoke the delivery note creation service (called ‘Trans1’). In this case, as there is no change of state for the exclusive, no change of state service is created.

Rule 9.4: From the relationship *triggers* between Record Delivery of Exclusive and Send Exclusive: a transaction that includes the service created for ‘Record Delivery of Exclusive’ and ‘Trans1’. The name of the transaction is ‘recordDeliveryExclusive’. In addition, ‘Trans1’ and the ‘Record Delivery of Exclusive’ service are flagged as internal.

Rules 10 and 11: Input and changes of state arguments

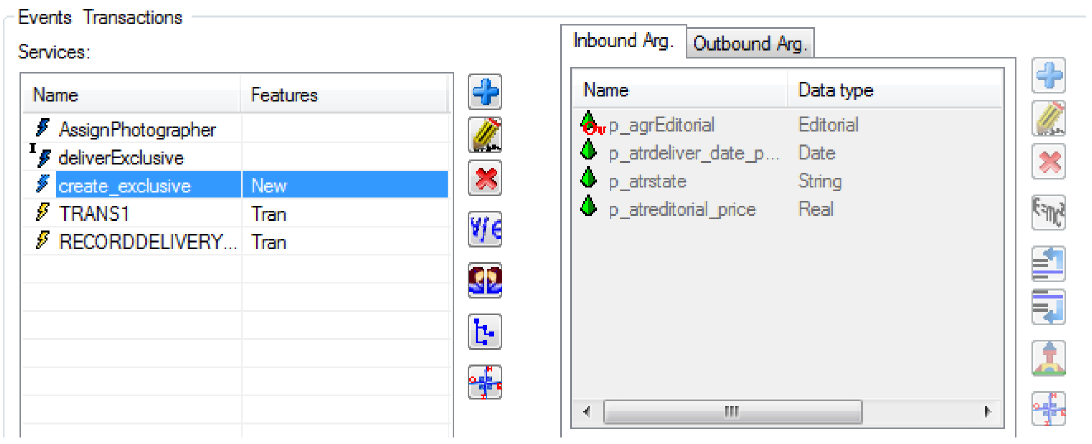
The input arguments of the services and the changes of state they undergo are obtained from the process diagrams, *input* and *other*-type flows, and *instantiatedBy* relations. In the following, details are given of the input arguments and the changes of state that take place in the service assignPhotographer.

AssignPhotographer service:

Input Arguments: Elements from the Domain Photographer and a date

Change of state: Exclusive.photographer and Exclusive.assignationDate

Rule 12: Preconditions. The preconditions that the conceptual model must satisfy are obtained from the business rules diagrams and are associated to the services if the source procedure of the service and the business rule are related, at the CIM level, by means of ObeysTo. In this case there is a relationship between the procedure ‘Assign Photographer’ and the rule ‘Photographer Constraints’, and therefore a precondition is added to that service.



Name	Features
AssignPhotographer	
deliverExclusive	
create_exclusive	New
TRANS1	Tran
RECORDDELIVERY...	Tran

Name	Data type
p_agrEditorial	Editorial
p_atrdeliver_date_p...	Date
p_atrstate	String
p_atreditorial_price	Real

Fig. 10 Rules 6 and 7: Creation and edit services

Rule 13: Agents. The inter-metamodel relations allow agents to be associated to the procedures that they are responsible for. Fig. 8 shows, using rectangles, the elements that operate as system agents. From the tacit source (represented as actors) shown in Fig. 7, a system agent is created whose name is determined by the name of the job profile, role or employee with whom it is related by means of isResponsibleFor. Then, the Technical agent with visibility over the processes of creating and editing the exclusives and photos is created, as is the Commercial agent with visibility over the creation and editing processes of the publishers.

Rule 14: Agents from the interrelationship *ownedBy*. The agents from the other services are deduced from the *ownedBy* relationship between the tasks and procedures that give rise to them. Thus, for the service assignPhotographer, which has its origin in the procedure ‘Assign Photographer’ that belongs to the task Management of Exclusives carried out by the Technical Department, it will be the Technical agent created earlier that will have visibility over this procedure. Furthermore, a new Production agent is created with visibility to perform the DeliverExclusive transaction, whose services derive from the task ‘Send Exclusive’, which is the responsibility of the Production Department.

Rule 15: Global system agent. Once all the elements in the object model have been defined, a global agent is created that will have visibility and permission to execute over all of them.

Rule 16: Dynamic Model. The Dynamic Model is created from the *mustOccur* relations between procedures following the algorithm explained previously.

Fig. 11 shows the state diagram that is generated for the exclusive. The basic services, the service Assign photographer itself and the transaction DeliverExclusive all participate in the state diagram.

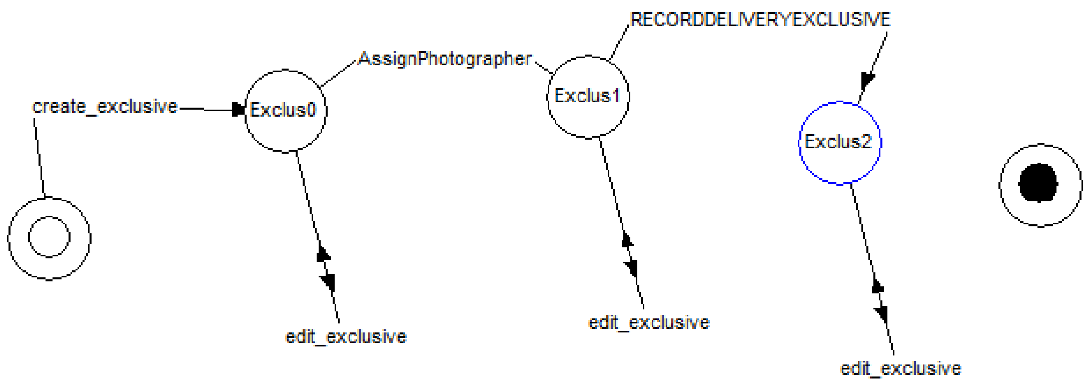


Fig. 11 State Diagram

5.4 Case study analysis and discussion

Once the application of the MDKe-IRIS Framework was finished, we validated the research hypothesis and questions using the results obtained, as well as the case study. In this section, we discuss the research questions we set out at the beginning of this section.

Q1. - Does the modelling guide help?

After the experience of the case study, our opinion is that having a step-by-step guide to performing the modelling was useful both for researchers and employees. The fact that the application of the framework began with simple questions such as: "what departments does the enterprise have?" or "what tasks does department X carry out?" help the interviewee to explain to the researchers the knowledge about the enterprise they needed to be able to depict MDKe models. However, the modelling guide does not consider the problem of requirement elicitation to obtain the knowledge needed to depict the models. Then, there are other situations in which more complex questions and analyses are needed before the elicitation. The modelling guide establishes the models and relations that must be depicted and suggests a sequence of steps to obtain them based on the authors' experience. Hence, the modelling guide establishes the inputs and outputs of each step so that it also serves as a checklist to be taken into account while the modelling guide is being executed.

Q2. - Is the modelling language selected (extension of the MDK Proposal) expressive enough to allow practitioners to build software starting from enterprise models?

The MDK language with the adjustments performed by the authors is very complete to capture the knowledge of the enterprise, as it allows all the enterprise views to be modelled. Nevertheless, there are some issues of the code generation that are outside the scope of the enterprise models, such as the interfaces and the algorithmic specification of procedures and processes. Thus, they cannot be considered. This point is extended in the *Discussion* section.

Q3. - Do the transformation rules and the extraction algorithm make the most of enterprise models in order to build Platform Independent Models?

To validate this question we automatically generate code using the tool OLIVANOVA (Care Technologies, 2010) and the models obtained from the application of the transformation rules over the partial CIM model. We thereby obtain a prototype without an implemented functionality. But both the persistence structure and the menu entries with which to perform the procedures and the users' permissions structure were obtained. So, we consider that the transformations rules do indeed make the most of the business models.

6 Discussion

The MDKe-IRIS Framework presented in this paper has two main parts: the enterprise modelling and the definition of the set of transformation rules that allow the information contained in the enterprise models to be reused to generate a preliminary software application representing the conceptual model. Nevertheless, there are some aspects of the code generation that have not been taken into account, such as the human interfaces and the algorithmic specification of procedures and processes.

Specification of the interaction between the user and a computer application depends on the characteristics that the user wishes to introduce into the computer and must take into account aspects such as usability or accessibility. In consequence, it was thought that specifying them at the CIM level would make the resulting model lose part of the abstraction with respect to the computation that characterises this level. Therefore, this part of the modelling must be performed at the PIM level.

As regards the algorithmic specification of the procedures and processes, the 'Behaviour model' makes it possible to represent the inputs, outputs and mechanisms used by the enterprise services and processes, but it does not give any specific indication of how the data is processed. Because specifying the particular processing of each data item in each enterprise process is a tedious, time-consuming task with respect to the richness offered by the CIM model (interpreted as an enterprise model), the idea of specifying the processing of each item of data was ruled out. Another point that was taken into account was the fact that the enterprise models are used by members of staff in the firm who are not specialists in information technology, and the algorithmic specification of a process can be confusing if the user is not familiar with these kinds of representations.

On the other hand, in order to carry out the specification, it would be necessary to create an algebra whose types should be capable of representing the elements handled by the enterprise and whose operations are expressive enough to create algorithms that indicate how to process those elements in a way that is computation-independent and makes it possible to represent what each process must do. This would entail creating an algebra that can be adapted to each enterprise depending on the elements they use and the processes they perform, which would greatly increase the cost in terms of time and the complexity involved in creating a model. Another important factor is the complexity involved in creating the rules for transforming from the algebra created for the CIM level towards the algebra of the PIM metamodel to which the user wants to transform from the enterprise model, and not from the metamodel, because if the algebra is adapted to each CIM model, these adaptations to the algebra must also be transformed. All these reasons explain why the decision was made to leave the modelling of the process logic at the PIM level, since compilers and models at this level have already defined algebras that are computable and can be transformed into code. Thus, the complete Object Model is obtained from the CIM-level model. This includes its classes, attributes and relations; the signature of the services including the input parameters and the changes of state that must be carried out; the signature of the preconditions to be fulfilled by each service (relation *ObeysTo*); and the Dynamic Model that establishes the order in which the services must be executed, if relevant (relation *mustOccur*). Hence, all that is needed to finish the PIM model is the definition of the Functional Model, indicating the changes of state carried out by the services, the integrity constraints, preconditions and the Presentation Model.

Finally, in this paper a solution to the CIM to PIM transformation problem is provided. However, its application is limited by the fact that it is necessary to develop enterprise models from scratch using the modelling language provided in the paper.

The reason for using a specific modelling language is because a high level of detail and a high level of formalism are needed, for example, to establish the relations between the different elements of the models in order to make information depicted in the CIM models useful to build a software prototype. If it is not provided in the CIM model, it should be provided at a lower level like PIM or PSM.

Moreover, as future work, we are considering the possibility of defining model-to-model transformation rules in order to provide a solution to the challenge of reusing existing enterprise models made with other modelling languages. The metamodels in the MDK proposal for modelling the process, decision and product views are based on IDEF0, GRAI and EXPRESS, which means that in these cases defining the transformation rules that map the elements modelled with those languages onto the MDK elements can be expected to be an easy task.

7 Conclusion

This paper explains the MDKe-IRIS Framework, which was designed to: (1) guide the process of enterprise modelling, using the models proposed in the MDK Proposal, and (2) reuse the information contained in the models to obtain most of the Platform Independent Models used in the OO-Method approach. Furthermore, a description of a practical case has been included to illustrates how the framework is applied. More specifically, the knowledge that is reused from the CIM level is useful for generating the structures of a computer-based conceptual model (classes, attributes and relations), the services to be executed over these structures and the constraints that must be fulfilled. Thus, we have created a guide for enterprise modelling that completes the process of three-level modelling defined in the MDA standard in the area where the greatest research efforts are most needed; i.e., CIM-level modelling and its transformation into PIM.

The idea underlying the development of the MDKe-IRIS Framework is the fact that enterprise models must drive the requirements elicitation phase of a software development process. From our experience, the result of the requirement elicitation is a more or less structured “Functional Document” which records the functional and information requirements that a software solution must satisfy to be suitable for the needs of an enterprise. However, the use of enterprise models makes it possible to:

1. Extract functional and information requirements automatically by selecting from the CIM level the elements that made up the PIM level, taking into account the context of the software application to be generated from the PIM level.
2. Speed up the creation of software prototypes that allow enterprises to validate the software application in the first phase of the software development process.
3. If there are strategic changes in the enterprise, and the CIM model is changed to represent them, it is possible to find out which DomainElements and tasks are affected through the *use* inter-metamodel relation. Moreover, it is possible to regenerate the prototype of the software application with the changes made to the model.

The MDKe-IRIS Framework is useful for both practitioners and researchers. Practitioners may be either members of staff that are empowered to make decisions in different areas of the firm, who use the models to understand the current and future situation of their area of work and thus put themselves in a position to make better decisions, or they could be software developers, who take these models as the starting point to implement an information system. The advantage for the decision-making staff members is that they will continue to work with models as they have been done until now, but those models will be more than just a graphical representation of the enterprise. They will be models that are interoperable with the models of other departments and enterprises, because they use a set of metamodels that are related to one another, and an asset to be reused in the process of computerising the enterprise. The advantage for software developers is that these models are based on UML metamodels, which will favour their implementation in computer systems. Yet, as has been mentioned earlier in the discussion section, the PIM model thus generated does not contain the functional implementation of services and transactions. But it does contain all the information needed to generate a first non-functional prototype that can be shown to future users. If the need for more capabilities is detected, they are selected in the CIM model and the prototype is created again automatically. Once the prototype is considered complete, the capability is incorporated into each service and the software development process would be finished. This would increase feedback between users and developers, while also speeding up the process of developing computer applications.

Moreover, for researchers, this is an area where there is still a lot of work to be done. Hence, some possible future lines of work that could be followed may include: improving the organisational metamodel in order to enrich the representation of business rules and the reuse at the CIM level of the additional information that is added at the PIM level, so that if the model is generated again at the CIM level, this information is automatically reused. On the other hand, we could consider the possibility of including elements from outside the enterprise (such as collaborators, thus modelling CIM-level interoperability) and defining the type of transformations required to generate an interoperable PIM model. Furthermore, research could be conducted to examine how to transform the domain model into an ontology so as to be able to reuse even more information expressed in the model. Finally, studies could be carried out to refine the UML profiles and incorporate constraints in OCL and research ways to transform these types of constraints.

References

Alencar, F., Marin, B., Giachetti, G., Pastor, O., Castro, J. & Pimentel, J. H. (2009). From i* Requirements Models to Conceptual Models of a Model Driven Development Process, in *Proceeding of Practice of Enterprise Modeling*, vol. 39, 99-114.

Alfonso Aguilar, J., Garrigos, I., Mazon, J.-N. & Trujillo, J. (2010). An MDA Approach for Goal-oriented Requirement Analysis in Web Engineering, *Journal of Universal Computer Science*, 16 (17), 2475-2494

Antoy S. & Hanus, M. (2002). Functional Logic Design Patterns, in *Proceedings of the 6th International Symposium on Functional and Logic Programming*, London, UK, 67–87.

Bae, J. H., Chae, H.S. (2008). UMLSlicer: A tool for modularizing the UML metamodel using slicing, *Computer and Information Technology*, (CIT'08), 8th IEEE International Conference, pp.772-777

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. & Mylopoulos, J. (2004). Tropos: An Agent-Oriented Software Development Methodology, *Autonomous Agents and Multi-Agent Systems*, 8 (3), 203–236

Cantamessa M. and Paolucci, E. (1998). Using organizational analysis and enterprise modelling in SMEs IDEF0, *International Journal of Computer Integrated Manufacturing*, 11 (5), 416-429

Care Technologies, Olivanova, (2005). [Online]. Available: http://www.care-t.com/index.asp. [Accessed: 25-Mar-2010]

Chalmeta R. & Grangel, R. (2008). Methodology for the implementation of knowledge management systems, *Journal of the American Society for Information Science and Technology*, 59 (5), 742-755

Chen L. & Zeng, Y. (2010). Automatic Generation of UML diagrams from product requirements described by natural language, in *Asme International Design Engineering Technical Conference and Computers and Information in Engineering Conference, Proceeding, VOL 2, PTS A AND B*, vol. 2, 779-786.

Chen M. & Sairamesh, J. (2006). A Knowledge Model-driven Recommender System for Business Transformation, in *2006 IEEE International Conference on Services Computing (SCC'06)*, Chicago, IL, USA, 77-84.

Cox, K., Phalp, K. T., Bleistein, S. J. & Verner, J. M. (2005). Deriving requirements from process models via the problem frames approach, *Information and Software Technology*, 47 (5), 319-337, Mar.

Dalpiaz, F., Ali, R., Asnar, Y., Bryl, V. & Giorgini, P. (2008). Applying Tropos to Socio-Technical System Design and Runtime Configuration., in *Evolution of Agent Development: Methodologies, Tools, Platforms and Languages (WOA08)*, Palermo, Italy,

De Castro, V., Marcos, & Vara, J. M. (2011). Applying CIM-to-PIM model transformations for the service-oriented development of information systems, *Information and Software Technology*, 53 (1), 87-105

Devedzic, V. (1999). A survey of modern knowledge modeling techniques, *Expert Systems with Applications*, 17 (4), 275-294

Dijkman, R. M., Quartel, D. A. C. & van Sinderen, M. J. (2008). Consistency in multi-viewpoint design of enterprise information systems, *Information and Software Technology*, 50 (7-8), 737-752.

Doumeingts, G., Chen, D., Vallespir, B. & Fenie, P. (1993). GIM (GRAI Integrated Methodology) and its evolutions - A Methodology to Design and Specify Advanced-Manufacturing-Systems, in *Information Infrastructure Systems for Manufacturing*, 14, 101-117.

Fuentes, L., Troya, J. M. & Vallecillo, A. (2002). Using UML Profiles for Documenting Web-Based Application Frameworks, *Annals of Software Engineering*, 13 (1-4), 249-264

Garner B. & Raban, R. (1999). Context management in modeling information systems (IS), *Information and Software Technology*, 41 (14), 957-961

Garrigós, I. (2008). A-OOH Extending Web Application Design with Dynamic Personalization, University of Alicante, Alicante, Spain

Goguen J. A. & Linde, C. (1993). Techniques for requirements elicitation, in *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, 152 -164.

Gordijn, J., Yu, E. & van der Raadt, B. (2006). E-service design using i* and e/sup 3/ value modeling, *IEEE Software*, 23 (3), 26-33

Grangel, R. (2007). Business Knowledge Modelling Proposal, Universitat Jaume I

Grangel, R., Bigand, M. & Bourey, J.-P. (2010). Transformation of decisional models into UML: application to GRAI grids, *International Journal of Computer Integrated Manufacturing*, 23 (7), 655-672

Grangel, R., Chalmeta, R., Schuster, S. & Pena, M. (2006). Exchange of Business Process Models using the POP* meta-model, in *Business Process Management Workshops*, 3812.

Guizzardi R. S. S. & Guizzardi, G. (2010). Applying the UFO ontology to design an agent-oriented engineering language, in *Proceedings of the 14th east European conference on Advances in databases and information systems*, Berlin, Heidelberg, 190–203.

Hatcliff, J., Dwyer, M., & Zheng, H. (2000). Slicing Software for Model Construction, *Higher Order Symbol. Comput.*, 13 (4)

Huet B. & Martin, J. (1980). Modeling and simulation of Information Systems on Computer - Methodological Advantages, *Medical Informatics*, 5 (3), 193-203

Hutchinson, J., Whittle, J., Rouncefield, M. (2014). Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. *Science of Computer Programming*. 89:144-161.

IST-2001-507849, (2010). ATHENA - Advanced technologies for interoperability of heterogeneous enterprise Networks and their applications, 2007-2004. [Online]. Available: http://www.ist-world.info. [Accessed: 27-Abr-2010].

Jackson, M. (2001). *Problem frames : analysing and structuring software development problems*. Harlow [u.a.]: Addison-Wesley [u.a.]

Jouault, F., Allilaire, F., Bézivin, J. & Kurtev, I. (2008). ATL: A model transformation tool, *Science of Computer Programming*, 72 (1-2), 31-39

Kagdi, H., Maletic, J. I. & Sutton, A. (2005). Context-Free Slicing of UML Class Models, in *21st IEEE International Conference on Software Maintenance (ICSM'05)*, Budapest, Hungary, pp. 635-638.

Kardos M. & Drozdová, M. (2010). Analytical method of CIM to PIM transformation in Model Driven Architecture (MDA), *Journal of Information and Organizational Sciences*, 34 (1), 89-99

Kramer, J. (2007). Is abstraction the key to computing? *Communications of the ACM*, 50, 37–42.

Lopata A. & Ambraziunas, M. (2010). MDA Compatible Knowledge Based IS Development Process, in *Business Information Systems Workshops*, 57, 33-38.

Luna, E. R, Garrigós, I., Mazón, J.-N., Trujillo, J. & Rossi, G. (2010). An i*-based Approach for Modeling and Testing Web Requirements, *Journal of Web Engineering*, 9 (4), 302-326

Marshall, C. (2000). Enterprise modeling with UML. Designing successful software through business analysis. Addison Wesley.

Martinez-Fernandez, J. L., Martinez, P. & Gonzalez-Cristobal, J. C. (2009). Towards an Improvement of Software Development Processes through Standard Business Rules, in *Proceeding of Rule Interchange and Applications*, 5858, 159-166.

Mazanek S. & Hanus, M. (2011). Constructing a bidirectional transformation between BPMN and BPEL with a functional logic programming language, *Journal of Visual Languages & Computing*, 22 (1), 66-89

Nicolás J. & Toval, A. (2009). On the generation of requirements specifications from software engineering models: A systematic literature review, *Information and Software Technology*, 51 (9), 1291-1307

OASIS: Organization for the Advancement of Structured Information Standards, (2007). Web Services Business Process Execution Language (WSBPEL) TC | OASIS, 2007. [Online]. Available: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel. [Accessed: 06-Dic-2011].

Object Management Group, (2000). Unified Modeling Language, 2000. [Online]. Available: www.uml.org.

OMG - Object Management Group, (2009). http://www.omg.org/spec/BPMN/1.2/, *BPMN*, 2009. [Online]. Available: http://www.omg.org/spec/BPMN/1.2/. [Accessed: 28-Nov-2011].

OMG, Model Driven Architecture, (2003). [Online]. Available: http://www.omg.org/mda/. [Accessed: 27-Abr-2010].

Opdahl A. & Berio, G. (2007) A roadmap for UEML, in *Enterprise Interoperability: New Challenges and Approaches*

Ould, M. (1995). *Business processes : modelling and analysis for re-engineering and improvement.* Chichester: John Wiley & Sons.

Pastor O. & Molina, J. C. (2007). Model *Driven Architecture in Practice*. Springer

Pham, H. N., Mahmoud, Q. H., Ferworn, A. & Sadeghian, A. (2007). Applying Model-Driven Development to Pervasive System Engineering, in *Proceedings of the 1st International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments*, Washington, DC, USA.

Roboam, M., Zanettin, M. & Pun, L. (1989). GRAI-IDEF0-Merise (GIM): Integrated methodology to analyse and design manufacturing systems, *Computer Integrated Manufacturing Systems*, 2 (2), 82-98

Rodríguez, A., de Guzmán, I. G.-R., Fernández-Medina, E. & Piattini, M. (2010). Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach, *Information and Software Technology*, 52 (9), 945-971

Runeson P. & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering, *Empirical Softw. Engg.*, 14, 131–164

Ruscio, D., Paige, R. & Pierantonio, A. (2014). Guest editorial to the special issue on Success Stories in Model Driven Engineering, *Science of Computer Programming*, 89(B), pp 69-222

Soler, E., Trujillo, J., Blanco, C. & Fernandez-Medina, E. (2009). Designing Secure Data Warehouses by Using MDA and QVT, *Journal of Universal Computer Science*, 15 (8), 1607-1641

Vergidis, K., Turner, C. J. & Tiwari, A. (2008). Business process perspectives: Theoretical developments vs. real-world practice, *International Journal Of Production Economics*, 114 (1), 91-104

Vernadat, F. (1996). *Enterprise modeling and integration : principles and applications*. London;New York: Chapman & Hall

Wagner, G. (2003)The agent-object-relationship metamodel: towards a unified view of state and behavior,. *Inf. Syst.*, 28 (5), 475–504

Xiao-mei, Y., Heng, D. & Ping, G. (2009). Mapping approach for model transformation of MDA based on xUML, in *Computer Science Education, 2009. ICCSE '09. 4th International Conference of Computer Science and Engineering*, pp. 862 -865.

Yu, E. S. K. (1997). Towards modelling and reasoning support for early-phase requirements engineering, in *RE `97 - Proceedings of the third IEEE International Symposium on Requirements Engineering*, pp. 226-235.

Zhang, W., Mei, H., Zhao, H. & Yang, J. (2005). Transformation from CIM to PIM: A feature-oriented component-based approach, in *Model Driven Engineering Languages and Systems, Proceedings*, vol. 3713.

Zhu, Y., Fei, L. & Yang, N. (2013). Trustworthy Software Development Based on Model Driven Architecture, in I*nformation computing and applications*, ICICA 2013, Proceedings, Vol. 391 pp 193-202.