CIM to PIM Transformation: A criteria Based Evaluation

Abdelouahed KRIOUILE *, Taoufiq GADI, Youssef BALOUKI

Univ Hassan 1, LAVETE Laboratory, 26000 Settat, Maroc
* E-mail of the corresponding author: kriouile1970@gmail.com

Abstract

The Model Driven Architecture (MDA) of the Object Management Group (OMG) represents an approach of software development based on the use of models. The transformation of models is at the heart of the MDA (Model-Driven Architecture) approach. CIM to PIM transformation can be of a great support for domain experts and business analysts, but is not mentioned enough by OMG. Thus, we have decided to focus on this higher level of MDA and these transformations at the PIM level, first by this work of studying a number of research on this subject and then by a future work to improve contribution in this stage. In this paper, we provide an assessment based on criterions deducted from the framework MDA™ of the OMG and similar evaluations of the existing works for this stage of construction and transformation of CIM. The results of this study will make it easier, for future researchers, to provide a relevant method of CIM construction and transformation.

Keywords: MDA, CIM, PIM, CIM to PIM transformation

1. Introduction

The Model Driven Architecture (MDA) of the Object Management Group (OMG) represents an example of the Model Driven Engineering (MDE) that is a software development approach family based on the use of models in the software construction. This allows the exploitation of models in order to produce code through a set of model transformations.

MDA is an initiative adopted on 2001 by the OMG. It is defined as the realization of MDE principles around a set of standards like [1] : Meta Object Facility (MOF) [2], XML Metadata Interchange (XMI) [3], Common Warehouse Meta-model (CWM) [4], Unified Modeling Language (UML) [5] [6], Query/View/Transformation (QVT) [7], Object Constraint Language (OCL) [8], etc.

The Model-Driven Architecture specifies three viewpoints on a system, a computation independent viewpoint, a platform independent viewpoint and a platform specific viewpoint [9]. Thus, the priority in MDA was the separation between platform independent aspects and platform dependant aspects.

MDA uses models in various steps of software development cycle. It recommends the elaboration of [9]:
- Computation Independent Model (CIM) is a view of a system from the computation independent viewpoint.
- PIM is a view of a system from the platform independent viewpoint. It is independent on technology detail.
- PSM is a view of a system from the platform specific viewpoint. It combines the specifications in the PIM with the details that specify how that system uses a particular type of platform.
- Implementation Specific Model that describe the last detail of programming.

And to build a software system, a series of transformations is performed: transformation from CIM to PIM, transformation from PIM to PSM, and transformation from PSM to code. However, the transformation from CIM to PIM is not part of the MDA lifecycle, which starts from an analysis model and ends with deployed code [9]. Thus, on the transformation of PIM to PSM and PSM to Code, several studies have been made. But little works have covered the construction of CIM and its transformation to PIM.

MDA is an approach, not a method [1]. So, to exploit it, we need to use methods. Thus several researchers have proposed methods in this context of MDA using models as productive elements, but differ from each other by the degree of automation and the set of model and operations they propose.

In MDA, the CIM is the main source of communication between domain experts and software developers. Thus it has a special role in software development that is neglected in practical approaches to MDA.

On the other hand, several models are used to represent, both the CIM and the Platform Independent Model (PIM). For example, in [10] and [11] the use cases diagram is a component of the proposed CIM, while in [12] and [13] the use case diagram is the one
component of the proposed PIM. In order to determine which types of artifacts are expected on the one hand at
the CIM and secondly at the PIM, this study analyzes the
models used by some CIM to PIM transformations, found in a survey.

Our main research question in this paper is: “what different methods are available for CIM to PIM
transformation”?

The main research question is then divided into four
detailed questions:
- What are the components of the CIM?
- What methods can be used to build PIM?
- How these methods transform CIM to PIM?
- How can we assess these methods?

Thus, we will identify relevant methods for transforming CIM to PIM: identify relevant quality
criteria for evaluation of these methods and finally
evaluate the selected methods for transforming CIM to PIM.

There are some proposals to transform CIM to PIM, for example [10], [11], [12], [13], [14], [15] and [16].

The contribution of this paper is a critical review of the
proposed methods for modeling and transforming
CIM to PIM in the context of the MDA. It reports on
a systematic review that focuses on methods for
transforming CIM to PIM. The aim is to determine
whether these methods have the capability to transform
CIM into PIM. This survey selected, investigated, and
compared several primary studies (9 methods) for
building CIM and transforming it to PIM.

In order to facilitate the synthesis and comparison of
these methods, we considered the OMG framework
MDA™ and the papers [17], [18], [19] and [20] that
provides a review of some CIM to PIM transformation
methods.

This paper is organized as follows: in section 2, we
provide a global overview of the Model Driven
Architecture; it, firstly, introduces the necessity of
models in software development, then explains an
important term behind the world of MDA which is the
transformation process and at last gives an overview of
the life cycle of MDA approach. Section 3, gives an
overview on the CIM level, PIM level and its
transformation found in a survey. The analysis of
building CIM and PIM and its transformations are
presented in section 4. Finally, the conclusions are left
in section 5.

2. Overview of the OMG approach for
software development: MDA

In 2001 the Object Management Group (OMG)
adopted a framework, the Model Driven
Architecture™ or MDA™[9], as an approach to using
models in software development. Its three primary
goals are portability, interoperability and reusability
through architectural separation of concerns.

MDA intends to provide an approach for:
- specifying a system independently of the platform
  that supports it,
- specifying platforms,
- choosing a particular platform for the system, and,
- transforming the system specification into one for a
  particular platform.

2.1. Using Models in software development
with MDA

MDA is centered on models and how those models
may be used together to create the resulting software
system. As companies requirements increase in the
same way and interoperability with other enterprises
becomes a necessity, the OMG managed to resolve this
situation using models to derive code for platform-
specific architectures. The MDA recommends the
widespread use of models and provides initial answers
to how, when, what and why model. It aims at
highlighting the three main qualities of the models,
such as sustainability, productivity and consideration
of execution platforms [21].

Four kinds of models with a particular role are
distinguished in the MDA:

- Computation Independent Model (CIM)
  According to the MDA Guide [9] a computation
  independent model (CIM) is a view of a system from
  the computation independent viewpoint. A CIM does
  not show details of the structure of the systems. It
  focuses on the requirements for the system and the
  environment of the system. It is sometimes called a
  domain model and serves as a vocabulary that is
  familiar to the practitioners of the system’s domain. A
  CIM is more than a domain model; it also expresses
  the system requirements using the domain concepts as
  a vocabulary. Its role is “bridging the gap between
  those that are experts in the domain (business analysts
  and domain expert) and its requirements on the one
  hand, and those that are experts in the design and
  construction (software analysts) of the artifacts that
together satisfy the domain requirements, on the
other”.

  CIM is the initial point in MDA approach since it
  includes on the one hand the business processes used to
  execute the business of the enterprise and a domain
  model that represents the intra- or inter-organizational
  understanding of the domain the application operates in
  [22]. And on the other hand the requirements of the
  system.

  According to the above, the CIM usually includes
  several distinct models that describe system
  requirements, business processes and business objects.
  It represents all aspects of the system that are important
  from a domain expert’s point of view. Then, the CIM
  should be modeled in a language that is understandable
for domain experts. This language should be adapted towards specific needs of thereof. This can be done in UML or in a language that is specialized for modeling business processes.

- **Platform Independent Model (PIM)**

  A platform independent model is a view of a system from the platform independent viewpoint. It describes the system, but does not show details of its use of its platform. Thus, a Platform Independent Model exhibits a specified degree of platform independence, such that it is suitable for a number of different platforms of similar type. This implies that the targeted platforms must be known beforehand to a certain degree. It also implies that a PIM is still specific to the chosen group of platforms.

  The role of PIM is to be sustainable and to make the link between the CIM and the application code. These models must also be productive because they are the foundation of the whole process of code generation defined by MDA.

  PIM should be modeled in a language understandable for software developers.

  In MDA the standard modeling language is UML. This fits for technical description of the system as in the PIM, but can lead to problems in modeling the CIM, because many domain experts are not able to model their intentions in UML. A way to overcome this is the introduction of Business Process Modeling Notation (BPMN) [23], an easy notation that is understandable by managers, business analysts and software developers.

- **Platform Specific Model (PSM)**

  A platform specific model is a view of a system from the platform specific viewpoint. The platform specific viewpoint combines the platform independent viewpoint with an additional focus on the detail of the use of a specific platform by a system [9]. In other words, the PSM is a more detailed version of a PIM. Platform specific elements are added. When defining a PSM a target Platform Model has to be available.

- **Platform Model (PM)**

  A Platform Model provides the technical concepts that represent the parts that make up a platform and the services provided by that platform. It also provides concepts for specifying the use of a platform by a software system, which can be used in a PSM.

2.2. Model Transformation

We have to review the types of the most important models of MDA are that CIM, PIM and PSM. The key challenge of model-driven development is in transforming higher-level models into platform-specific models that can be used to generate implementation level models.

A transformation consists, as visualized in figure 1, of creating a target model from a source model by rules that describe how one or more constructs from the source model should be replaced by one or more constructs in the target model. A desirable characteristic of a transformation is that it is traceable. This means that it is possible to trace back parts in the target model to their corresponding parts in the source model. This is particularly useful when the target model has to be maintained. Different methods can be used for defining the transformation rules.

![Figure 1. MDA Transformation Process](image)

2.3. MDA Software life Cycle (General architecture of the MDA)

The construction of a new application begins with the development of one or several CIM models. It continues with the development of PIM models. These should theoretically be partially generated from CIM so that traceability links are established. PIM models are perennial models that contain no information on platforms running. Then we need to build PSM models, which should in principle be obtained by a transformation of PIM by adding technical information.
platforms. PSM is not intended to be permanent. Their main function is to facilitate the generation of code. Code generation from models is the last step consisting of a translation of the PSM in a textual formalism. In Figure 2 the dependencies between the different types of models are shown.

Figure 2. MDA software life cycle

3. CIM and PIM and their transformation: Literature review

To analyze the methods of CIM to PIM transformation, we first conducted a study of ways to build CIM and PIM, based on articles that tackle such methods. Our research started with a literature review in order to identify relevant methods for transforming CIM to PIM. We started by reviewing well known articles about these methods, in order to distinguish the most prominent techniques that could be used for a CIM to PIM transformation.

We found seven important candidates that transform CIM to PIM and two who propose only how to build CIM.

These different methods will be discussed from the viewpoint what constitutes a “goods” modeling techniques for developing a CIM and PIM based on the evaluation criteria as defined in section 4.

In [10] proposed a disciplined method for transformation of CIM into PIM. Business processes and system requirements are modeled in a CIM using two activity diagrams. The business process model shows all the business activities to be accomplished independently of their automation, while the requirement model specifies the system which best supports the business activities, by representing its use cases and considering it as an actor. In this paper, PIM which represented by class diagram, is obtained from the requirement model: The requirement model is transformed into a model of the system components. The latter, provides a first sketch of the system structure, a set of business archetypes help to transform the system components to the PIM layer in details.

[15] propose a transformation from CIM to PIM using feature-oriented and component-based method. In this paper, requirement in CIM is represented by feature model which includes a set of features and relationship between them. The PIM is represented by software architecture that includes a set of components and interaction between them. This method uses an intermediate model that is neither CIM nor PIM.

In research articles of Alfonso Rodríguez and its partners ([13], [24], [25]), the CIM is composed of a business process model, using the secure business process in BPMN. The CIM is transformed, with the help of QVT (Query/View/Transformation [7]) rules, checklists, and refinement rules into two models that are part of the PIM: a use case diagram and a class diagram. The use cases must be detailed to obtain actions, and the class diagram is considered as an initial analysis model.

In [12] presented an analytical solution for CIM modeling and then transform it to PIM. In this paper, for modeling business processes, the author used DFD for CIM level representation. While PIM level represented by four UML diagrams: use case diagram, activity diagrams, sequence diagrams, and domain models.

In paper [16], authors present a semi-automated method for the generation of web-based applications from high-level requirements expressed as use cases in accordance with model-driven architecture (MDA). His first step is to transform CIM to PIM. He considers that CIM is represented by a description of the Use case and the default domain objects. And PIM includes the state machine, the user interface model and the refined domain model.

[14] presents a systematic methodology for MDA transformation that includes creating a platform independent model (PIM), transforming PIM into platform specific model (PSM), and transforming PSM into a Code model. But consider first that CIM is composed by use case, activity diagram and robustness diagram. While PIM is modeled by two parts: the behavior part which presented using the sequence diagram and the structure part which depicted using the class diagram.

[26] presents a method for generating CIM using artifacts and concepts of RUP methodology. This method presents a CIM that covers both aspects of CIM that include business model and requirement model. The CIM is composed by three models:
Business use case model, Business analysis model and Use Case Model. But in this article, it is not clear how to transform this CIM built in PIM, and what are the components of the latter model.

Janis Osis and al. in their papers ( [27], [28], [29]) proposed a method called Topological Functioning Modeling for Model Driven Architecture (TFMMDA) uses formal mathematical foundations of Topological Functioning Model. It introduces more formal analysis of a business system (“as is”), enables defining of what the client needs, textual functional requirements validation, and missing requirements checking in conformance with the problem domain “as is” model. At CIM level, it define firstly a use case model of the planned application, then a conceptual class diagram presenting the domain concepts and their relations to be established. PIM modeling and CIM to PIM transformation method are not undertaken in this work.

[11] provides a method to build the CIM that can be transformed (semi-) automatically later to lower levels of abstraction in PIMs. Thus, this paper presents an method for CIM to PIM transformation where the CIM level is represented by two models: the Business process model (BPM) that will represents both the behavior and static aspect that will represents the different activities necessary to model businesses and resource used by those activities enabling thus the capability of transforming this CIM to a low level of abstraction in the PIM level; the second model is use case model that represents the functional aspect of the system that will represent the business actors and business functionalities that are intended to be realized. Whereas the PIM level is represented using the Domain Diagram Class (DCD) that shows the static aspect of the system and the sequence diagram of system’s external behavior (SDSEB) that is a UML sequence diagram that shows only interactions between actors and the whole system as unique entity which is represented by one lifeline without focusing on system objects interactions, it represents a high level of interaction.

4. Evaluation, analysis and discussion

In this section we will start by present evaluation criterions, and thereafter we evaluate the proposed methods on the basis of these criteria.

4.1. Evaluation criteria

The evaluation criteria will be derived from two sources: the OMG framework, the Model Driven Architecture™ (MDA™) [9] and the paper [17] that provide a review of four CIM to PIM transformation methods, and present a criteria-based evaluation. In the first we derive the evaluation criteria for the CIM, then for the PIM, and finally for CIM to PIM transformation.

These criteria can be adapted to evaluate future research works on the same topic.

4.1.1. Criterions for CIM. To assess the CIM, we start first by listing the recommendations for CIM in the framework MDA™ of the OMG. Then enrich the evaluation criteria introduced in the similar works.

- A CIM does not show details of the structure of systems
- A CIM includes the domain model. Thus, CIM is a model of a system that shows the system in the environment in which it will operate.
- CIM consider the vocabulary that is familiar to the practitioners of the domain in question and used in its specification, as a source of a shared vocabulary for use in other models.
- The primary user of the CIM should be the domain practitioner who does not know the models or artifacts used to realize the functionality for which the requirements are articulated in the CIM.
- CIM should bridge the gap between experts about the domain and its requirements, and experts of the design and construction of the artifacts that together satisfy the domain requirements, on the other.
- CIM should modeled the requirements for the system, it describe the situation in which the system will be used
- CIM requirements should be traceable to the PIM and PSM constructs that implement them, and vice versa.

We can, therefore, say that best CIM, must positively meet three criteria:

- CIM Criterion 1: CIM Coverage of domain objects that indicates if CIM proposed covers the business objects representing the static aspect of CIM.
- CIM Criterion 2: CIM Coverage of Business Process that indicates if CIM proposed covers the Business Process representing the Behavior aspect of CIM.
- CIM Criterion 3: CIM Coverage of Requirements that indicates if CIM proposed covers the Business Process representing the Functional aspect of CIM.

4.1.2. Criterions for PIM. Similarly, to assess the CIM, we start first by listing the recommendations for PIM in the framework MDA™ of the OMG [9]. Then enrich the evaluation criteria introduced in the similar works.

- PIM describes the system without showing details of its use of its platform. It focuses on the operation of a system while hiding the details necessary for a particular platform.
- PIM provides a specification that does not change from one platform to another.
- A PIM is prepared using a platform independent modeling language.
- PIM will be suited for a particular architectural style, or several.

PIM Criterion 1: Static aspect of PIM
PIM Criterion 2: Behavior aspect of PIM

4.1.3 Criterions for transformation. From framework MDA™ of the OMG [9], we can deduce the following suggestions for the CIM to PIM transformation:
- Model transformation is the process of converting one model to another model of the same system.
- A model type mapping specifies a mapping from any model built using a source model language to models expressed using a target model language.
- Transformation can be manually, with computer assistance, or automatically.

Based on the foregoing, we evaluate the CIM to PIM transformation of the methods proposed with respect to their automation, traceability and completeness of transformation rules, according to three criteria:
- Transformation Criterion 1: Automation transformation

The automation criterion evaluates whether a transformation is automated, semi-automated or manual. A transformation is automated if it has been fully implemented. When user interventions are required, the transformation is semi-automated. Finally, a transformation is manual, when it is entirely manual.

- Transformation Criterion 2: Completeness of Transformation Rules

Transformation rules between CIM and PIM are expected to be complete and well-structured. Thus, if the transformation rules proposed in a method can transform most or all CIM constructs into PIM elements, then we assume that this set of transformation rules is complete; otherwise, incomplete or absent.

- Transformation Criterion 3: Traceability from CIM to PIM

For the criterion of traceability, we are trying to say if the traceability links between CIM and PIM are expected to be established when a transformation is performed or no.

Table 1 summarizes all the evaluation criteria for the CIM, PIM, and CIM to PIM transformation.

<table>
<thead>
<tr>
<th>Table 1: General evaluation criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Criterion Name</strong></td>
</tr>
<tr>
<td>--------------------------------------</td>
</tr>
<tr>
<td>CIM Coverage</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>PIM Completeness</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Automation transformation</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Completeness of Transformation Rules</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Traceability from CIM to PIM</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Legend: Y: Yes; N: No; P: Partial

4.2. Analysis and discussion

Analysis consists of two sub section. Analysis related to graphical models used respectively for each CIM and PIM level and another based of the criterions presented at the top of this section.

4.2.1 Analysis of graphical modeling. On reviewing different methods discussed earlier in this paper, seven out of nine methods use UML for representing the elements of CIM and five out of nine methods use other graphical techniques for representing the components of CIM, which two uses BPMN.

Regarding the graphical representation of the elements of the PIM, most methods use UML. The analysis of graphical modeling in table 2 depicts for each level CIM or PIM, the graphical models used for representing their elements.
Table 2: Graphical representations of CIM and PIM

<table>
<thead>
<tr>
<th>Papers studied</th>
<th>CIM Graphic representation</th>
<th>PIM Graphic representation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UML</td>
<td>Others</td>
</tr>
<tr>
<td>Kherraf and al. [10]</td>
<td>BPM (AD)</td>
<td>Use Cases</td>
</tr>
<tr>
<td>Kardoš and al. [12]</td>
<td>DFD</td>
<td>Use Cases, Activity Diagram</td>
</tr>
<tr>
<td>Wu and al. [14]</td>
<td>Activity Diagram</td>
<td>Robustness Diagram</td>
</tr>
<tr>
<td>Zhang and al. [15]</td>
<td>Feature Model</td>
<td></td>
</tr>
<tr>
<td>Fatolahi and al. [16]</td>
<td>Use Cases Domain Objects</td>
<td></td>
</tr>
<tr>
<td>Sharifi and al. [26]</td>
<td>Business U.C. Model</td>
<td></td>
</tr>
<tr>
<td>Erika and al. [28] [29] [27]</td>
<td>Use Cases Conceptual Model</td>
<td></td>
</tr>
</tbody>
</table>

4.2.2. Analysis from results of evaluation criteria.

The result from the analysis of evaluation criteria is presented in table 3. The rows in the table are nine works studied and the columns in the table are each one of the nine criterions presented in table 3.

Such as we can see from table 3, among the nine methods tree can cover Business Objects and two partially. And regarding Business Process, five methods that take care of them, and one timidly. And finally, seven methods can represent the requirement model. But neither method fully covers the three aspects of the CIM.

As shown in table 3, all the methods generate structural model elements while five out of seven methods only can generate behavioral model of the system.

We can see from the table 3, that:

- All the methods are based on semi-automatic transformation and therefore user interventions are required.
- All of the methods do not propose any method for traceability support.
- However, four of the methods define partially the transformation rules while five of seven do not even describe transformation rules.

From the above, we see that no method perfectly meets the three criteria for CIM to PIM transformation.

From the point of view of the CIM, and according to the results of the comparison in Table 3, we found four ways to represent the CIM, which we present in ascending order of importance:

- In [12], and [15], a single model representing the business process is used as CIM. Therefore, this model do not describes business objects and the requirements.
- In [16], [26] and [28] CIM does not cover business process. In the three proposals, the CIM represents the business objects and requirements.
- Similarly at the previous methods, in [10] and [13] the CIM only cover two domains, but this time neglecting the business objects. Thus, they represent only business process and requirements.
- Finally, [11] and [14] focus on the three areas of coverage of the CIM. Although, it remains to further develop the representation of component business objects.

Lastly, from the results of the evaluation criteria for CIM to PIM transformation, even partially, only two studied methods, that meet the criteria, which are [11] and [13]. While for others, satisfies one or two criteria.

In the end, combining the different results obtained, we can deduce that one method [11] which satisfies the different evaluation criteria. In this paper CIM can be transformed (semi-) automatically later to lower levels of abstraction in PIMs. It cover essentially the different domains of the system with the BPMN, uses cases model and business rules. Meanwhile, the PIM level is represented by the domain diagram class and sequence diagram of systems external behavior covering both static and dynamic aspects.
Table 3: Evaluation summary of the methods proposed in the studies

<table>
<thead>
<tr>
<th>Papers Studied</th>
<th>CIM coverage</th>
<th>PIM completeness</th>
<th>CIM to PIM transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Business Object (Static View)</td>
<td>Business Process (Behavioral View)</td>
<td>Requirement (Functional View)</td>
</tr>
<tr>
<td>Kherraf and al. [10]</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Bousseta and al. [11]</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Kardoš and al. [12]</td>
<td>N</td>
<td>P</td>
<td>N</td>
</tr>
<tr>
<td>Rodríguez and al. [13]</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Wu and al. [14]</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Zhang and al. [15]</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Fatalahi and al. [16]</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Sharifi and al. [26]</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Erika and al. [28] [29] [27]</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Legend: Y: Yes; N: No; P: Partial

4.3. Summary of evaluation results

An ideal method for modeling CIM and transforming CIM into PIM would have the following characteristics:
1. Modeled CIM should cover the different views static, dynamic and functional of the business domain.
2. Generated PIM should be complete: contain structural and behavioral aspects of a system.
3. The CIM to PIM transformation should be automated, supports traceability and respects the completeness of transformation Rules.

Yet, none of the reviewed methods conforms to the ideal proposition, as described above. But [11] seems to us closest to this ideal description.

5. Conclusion and Future Work

The MDA approach of software development guided by the model begins with the development of CIM and its automatic transformation into a PIM, which, subsequently, can be transformed into PSM to generate the code. Nevertheless, this very important step is often overlooked in most research work focusing on development cycle based on MDA approach. Few attempts have tried to automate part of this stage of software development. Similarly, the coverage of CIM differs widely in the work of research examining its representation.

Our work tries to help researchers to a proper understanding of the state of the art and to indicate directions for future research to fill the gap felt in this context, with an assessment based on the criteria of existing works for this stage of construction and transformation of CIM.

We have surveyed several prominent methods for CIM to PIM transformation and have evaluated them using a predefined set of evaluation criteria. According to the evaluations results we can conclude that:
- The methods studied herein are not matured enough, specially are covering some stages but not all of the transformation.
- CIM does not cover, in most cases, all the views static, dynamic and functional of the business domain.
- Most of the methods do not even provide guidelines to assure traceability between CIM and PIM.
- Definitions of the methods are not complete. The transformation rules are not always specified.

Also we found that the use of a mix of graphical modeling techniques in order to produce a CIM seems ideal: BPMN to represent business process and UML.
for representing the other aspects of CIM that are the business objects and the requirements.

In our future work, we will try to fill the gaps in methods of CIM to PIM transformation, and, thus, propose a method ideally meets the evaluation criteria proposed in our study. This method must be able to build a CIM, covering all aspects static, functional, and behavioral to translate automatically into a useful PIM will be transforming even PSM and then to the code.

6. References


Independent Modeling within the MDA," in Software-
Science, Technology & Engineering, 2007. SwSTE
2007. IEEE International Conference on (pp. 22-34).
IEEE, October 2007.

independent representation of the problem domain," in