Transformation From CIM to PIM Using Patterns and Archetypes

Samir Kherraf
GELOG Laboratory
École de technologie supérieure
Montreal – CANADA
samir.kherraf@ieee.org

Éric Lefebvre
GELOG Laboratory
École de technologie supérieure
Montreal – CANADA
eric.lefebvre@etsmtl.ca

Witold Suryn
GELOG Laboratory
École de technologie supérieure
Montreal – CANADA
witold.suryn@etsmtl.ca

Abstract

Model transformations form a key part of MDA (Model-Driven Architecture). Most of the studies deal with the transformations from PIM (Platform-Independent Model) to PSM (Platform-Specific Model) and PSM to Code, but very few deal with the transformation from CIM (Computation-Independent Model) to PIM. This last transformation usually depends on business analysts’ and software architects’ experience and creativity.

This paper proposes a disciplined approach to transform a CIM into a PIM. It first uses UML2 activity diagrams to model the business processes up to the users’ tasks. The activity diagrams are then detailed to specify the system requirements. The system components are directly deduced from the requirement model elements. Finally, a set of business archetypes helps detail the system components to yield the PIM.

The same approach applies equally to CIM and PIM built to model inter-enterprise processes and systems.

A case study illustrates our approach. It demonstrates how it reinforces the components traceability and reusability and how it globally improves the modeler’s efficiency. Furthermore, the use of the activity diagrams, as a single technique to build business process and requirement models, is an important facilitator which prepares our further work to automate this approach.

1. Introduction

Under the constant pressure to more rapidly develop IT systems that support frequent business changes, as well as e-commerce, the Object Management Group (OMG) has specified a new approach, the Model Driven Architecture (MDA) [1]. This approach clearly categorizes four models:

1. the Computation-Independent Model (CIM), which represents the business processes, i.e. the activities to be accomplished to meet the business objectives;
2. the Platform-Independent Model (PIM), which represents the system to support the business processes, independently of the IT platform;
3. the Platform-Specific Model (PSM), which adds to the PIM the technological aspects of the target platforms;
4. the code to execute the system.

As code generation is MDA main objective, model transformation from one level to the next one is of primary importance. But most of the research work until now has focused on the transformation from PIM to PSM, whereas few have addressed the CIM to PIM transformation. [2] presents an approach of which features and components are adopted as the key elements of CIM and PIM building, and responsibilities as the connector between features and components to facilitate CIM to PIM transformation. This approach provides a disciplined way to CIM to PIM transformation but not a fully automatic way. In [3] a promising approach based on multiple views using subject-oriented design and composition patterns in UML is presented to consistently preserve separation of concerns between CIM, PIM and PSM models but did not show how to transform a CIM into a PIM. Requirements modeled in a CIM often lack a sound structure, and it is thought to be impossible to automate the transformation from a CIM to a PIM [4].

The main problem which remains in order to more effectively generate an IT system from business requirements is how to build a CIM so that it can be automatically transformed into a PIM.

In order to automate the current process for this CIM to PIM transformation, we have investigated how to effectively use current methods and techniques. Our approach applies 1) patterns to structure a CIM so that it simplifies its transformation to a PIM, and 2) a set of four archetypes which constitutes the building blocks to generate a PIM.
This paper is organized as follows: section 2 presents a description of how to build a CIM, so that its two parts, the business process model and the requirement model can be thoroughly related; section 3 proposes a method to transform a CIM, as previously built, into a PIM, by applying the business archetypes; section 4 illustrates a case study based on the proposed method; finally, we conclude by presenting our current and future work.

2. CIM Building

A CIM is the view of a system from a computation independent viewpoint [5]. It does not show details of the system structure but plays an important role in bridging the gap between those that are experts about the domain and its requirements, and those that are experts of the system design and development.

Business processes and system requirements are modeled in a CIM, which describes the situation in which the system will be used and the environment in which it will operate. Thus, it helps in presenting exactly what the system is expected to do. It is useful, not only as an aid to understand a problem, but also as a source to share vocabulary with other models. In a MDA approach, requirement specifications expressed in the CIM, should be traceable to the PIM and PSM constructs that implement them, and vice versa.

According to the above definition, we propose to build a CIM on two models:
1) the business process model shows all the business activities to be accomplished independently of their automation;
2) the requirement model specifies the system which best supports the business activities.

2.1 Business Process Modeling (BPM)

Business process modeling is widely discussed in the literature. The Workflow Management Coalition [6] defines business processes as: ‘a set of one or more linked procedures or activities which collectively realize a business objective or policy goal, normally within the context of an organizational structure defining functional roles and relationships’.

Many different BPM languages exist [7]. Such languages must be:
- obviously convenient to model business process;
- widely used;
- understandable to both business analysts and system architects;
- conform with MDA.

Most used BPM languages are UML, mainly with its activity diagram, and BPM Notation (BPMN) [8].

MDA recommends UML as a modeling standard. It is already proven to be convenient to build the PIM as well as the PSM. Activity diagrams have been proven to be useful for business process modeling in [9, 10].

UML [11] is the most referred modeling language. Its last version, UML2, is widely adopted for modeling not only the application structure, but also its behavior, and architecture.

We therefore recommend to use UML2 to model business processes with activity diagrams, as it clearly answers the criteria listed above.

A major question in BPM is the granularity of the model elements. In general, we build the BPMs by decomposing the processes into Elementary Business Processes (EBPs), that is, all the way down to a level that we are not interested to decompose any further. An EBP is defined as “A task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leaves the data in a consistent state” [12]. It corresponds to a well-defined and well-delimited user’s task, based on the chronology of events and activities. It also identifies the business entities required by the task. Business entities encompass both resources used and objects created by the activity.

An EBP can be described in a generic fashion [13], [14] as a human activity which:
- Has an objective
- Is triggered by a previous event/activity
- Is taken in charge by someone
- Needs resources to be accomplished. These can be of the following type:
  - Human,
  - Material (consumable),
  - Facilities (fixed assets),
  - Financial,
- Has a result dependent on the objective,
- Triggers the next event/activity.

Thus, the process is decomposed into a collection of EBPs which are related according to a workflow specifying the handling over of the tasks from one actor to another. Possible relations are shown on figure 1, according to UML activity diagram notation.

A BPM describes both the activities (the dynamic aspect) and the objects required for their realization (the static aspect). The objects can be any of the categories, expressed on figure 2:
- the actors, responsible for the activities;
- the resources required (human, material, informational);
- the products or services resulting from the activity.
A BPM should therefore identify each EBP, the business entities involved in each EBP execution and the workflow relating the EBPs.

### 2.2 Requirement Modeling

From the BPMs, we define the system requirements in order to optimally support the processes. To this end, we introduce in a BPM the system as an actor. For inter-enterprise processes, we introduce two systems as two actors. Exchanges between these two actors will specify the choreography between the two processes.

The system as an actor deals with the activities of the process for which automation is desired. The requirement model expresses the automation choices.

Most of the current software development processes recommend use case models to specify requirements. Many authors have expressed their views on how to write use cases and have proposed their own templates [15, 16]. The current technique to write use cases is in a textual format, with a word processor, but the main weakness of this technique is that use case models are uneasy to read and to transform into system models, and not integrated in the modeling tools.

Jacobson [17] defines a UML use case as: ‘...a sequence of actions, including variants, that the system can perform, and that yield an observable result of value to a particular actor’. According to this definition, a use case consists of activities (or actions), which are ordered in some way, are sequential and are aimed at delivering a particular result. According to Larman [16], this definition relates to that of an EBP: ‘a use case should specify the actor-system interactions of an EBP’.

Larman [16] and Jacobson [18] have recommended, as a sound alternative, to write use cases with activity diagrams. This technique has also been successfully experimented for several years by different practitioners [19].

We propose to use UML2 activity diagrams to write use cases. Activity diagrams can vehicle more information than texts, such as decision branches [19]. Other important benefits are to:

- facilitate the quality control, by helping to find the missing flows;
- identify user interfaces to build, messages to write and flows to test.

BPMs and use case models can therefore be built with the same notation.

To obtain a fair level of understanding of the system by both business and technical experts, it is a significant advantage to model their respective view of the supported business processes by the same language and to map these models to the technical implementation.

The following example of a standard order by Internet shows how to transform a textual use case into a visual use case (an UML2 activity diagram). Table 1 illustrates the textual use case according to RUP [20] use case template. It assumes that the customer has already selected the products and intends to buy.
Table 1. Textual use case

<table>
<thead>
<tr>
<th>Order by Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Brief description</strong></td>
</tr>
<tr>
<td><strong>Actor(s)</strong></td>
</tr>
<tr>
<td><strong>Flow of events</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Alternative</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Special requirements</strong></td>
</tr>
<tr>
<td><strong>Pre-conditions</strong></td>
</tr>
<tr>
<td><strong>Post-conditions</strong></td>
</tr>
<tr>
<td><strong>Extension points</strong></td>
</tr>
</tbody>
</table>

Table 2. Transforming a textual use case into a visual use case (an activity diagram)

<table>
<thead>
<tr>
<th>Textual format</th>
<th>Visual format</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Brief description</strong></td>
<td>Legend of the diagram (a note)</td>
</tr>
<tr>
<td><strong>Actor(s)</strong></td>
<td>Swim lane(s) or Stereotype(s)</td>
</tr>
<tr>
<td><strong>Flow of events</strong></td>
<td>Activity</td>
</tr>
<tr>
<td></td>
<td><strong>Basic</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Altérnative</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Special requirements</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Pre-conditions</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Post-conditions</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Extension points</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Include</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Extend</strong></td>
</tr>
</tbody>
</table>

We think that visual use cases are fairly easier to read than their corresponding textual use cases. All decision points are clearly visualized, as well as the loops in the processing sequence. They constrain use case writers to limit the size of each diagram and to be concise in specifying each step within the rectangle of an activity. Whereas textual use cases are written in a word processor with hyperlinks to the modeling tool, visual use case writing is possible with most of the modeling tools. The model transformation from requirements to system is therefore much easier to follow.

Use case diagrams can also be replaced by activity diagrams, which can contain more information, such as dependencies between use cases. Consequently, business process models and use case models can now evolve with the same notation.

Table 2 presents how to transform these same items into the elements of an activity diagram.

The same example as in Table 1, ‘Order by Internet’, illustrates the corresponding visual use case, which is presented in the Figure 3.
3. PIM Building

In this section, we evaluate how applying business archetypes improves the CIM to PIM transformation [13, 21].

A PIM describes the system, but does not show details of its use of the execution platform. A PIM exhibits a specified degree of platform independence so as to be suitable for execution on different platforms. It could be transformed into PSMs executing on different platforms of different enterprises as required by e-business applications.

The requirement model, which represents the “what” of the new system, is transformed into a model of the system components. This system component model provides a first sketch of the system structure.

As seen above, a use case specifies the automated activities of an EBP. It generally includes a main activity, the required resources for its completion, and the product or service resulting from the activity.

From the use case model, we identify two distinct categories of components [22]:
1. the business process components, which correspond to the process;
2. the business entity components, which correspond to the resources and products or services.

Thus, each use case is generally transformed into a process component which is then linked to various entity components (See figure 4). One process component is generally specific to one use case. An entity component can in turn participate in various use cases, but generally according to a role specific to each use case concerned. These process and entity components are a direct mapping of the elements encompassed in the corresponding BPM (Compare figure 4 to figure 2).

### Figure 3. Visual use case with UML2 activity diagram

### Figure 4. Generic system component model of an EBP

Because the activities are linked with one another according to the workflow, the process components are linked together. The entity components are linked to each process component that requires them, according to their role in the process.

Once system components of each category are identified, they can be modeled by using the four archetypes defined by Coad et al.[21]:
- **Moment-Interval;**
  A moment-interval object represents the main event or activity of an EBP, i.e. it is the heart of the system analysis model. It often comes with its details (for instance an order and its line items).
- **Role;**
  A role is a way for a business entity categorized as Party, Place or Thing (PPT) of participating in an activity.
- **Party, Place, Thing (PPT);**
  A PPT object serves to track a PPT individually.
• Description;
  A description object represents a catalogue-
  entry-like description of a PPT individual
  object.

These four archetypes were identified from the
authors’ experience and tested in many instances. Most
of information system object models can be built from
these four archetypes [13] [21].

The archetypes include generic attributes and
methods, which help detail the model. They can be
supported by a modeling tool. To facilitate model
building and reading, a color has been assigned to each
archetype, along with a UML stereotype, which adds a
new dimension to an object model.

The Moment-Interval (MI) archetype (with its
details, in most of the cases) is the first building block
of the model and is at its center. It represents the
process component. Besides representing the activity
and the event, the MI objects have a state, which
evolves during the accomplishment of the activity or
when the event is answered. Each process component
(i.e. each MI) is generally triggered by a prior one and
in turn triggers a next one.

Finally, because MIs represent the dynamics of an
EBP, they usually define the objects which control the
interface with the users.

MIs and their related events/activities are therefore
central in the system development artifacts, such as in:
  • the business process model,
  • the use case,
  • the object model,
  • the business component model,
  • the user interface

The three other archetypes, namely the PPT, the
description and the role archetypes, are our elementary
building blocks to model the business entities (the
entity components) which are required by the business
processes (the process components) for their
accomplishment. One archetype or a combination of
the three archetypes defines each business entity (the
entity component), whether the entity needs to be
identified individually and/or generally and whether it
plays a specific role in the context of the EBP under
study (See Figure 5).

Figure 5. Transformation from component to
class

We complete the PIM by selecting the attributes and
methods from the generic elements of each archetype,
In summary, the PIM components are directly
derived from the use case model through the EBPs and
their related business entities. Components are then
detailed into classes using the Coad’s set of four
archetypes.

4. Case Study

In this section, we use a case study about a car rental
company to illustrate our approach. This example is
taken from the Business Rules Group site [23]. EU-
RENT is the name of the fictitious company.

4.1. EU-RENT business

EU-RENT is a typical car rental company. It has
many branches. At each branch, cars (classified by
group and model) are available for rental and most of
the rentals are by advance reservation. A customer can
have several reservations but only one car can be
rented at a time.

4.2. Business Process Modeling

According to EU-RENT specifications, we have
defined five EBPs (figure 6) which describe the
business processes involving the clients: Reserve,
Assign, Rent, Pay and Return. These EBPs are
illustrated by five successive actions in the BPM.
Modeling the ‘rent a car’ process with an activity diagram identifies the same use cases as the use case diagram (Figure 7), but better expresses the use case relationships. The activities of the BPM can then be decomposed in the use case specifications within the same tool and documented in a common repository, which would be fairly complex to do with the use case diagram and the use case textual format.

4.3. From Business Process Model to Requirement Model

In this case study and as seen in 4.2, each activity in the BPM represents one use case (Figure 6). But in general, detailing a use case from one activity in the BPM depends on the need to automate it.

In this section, we will restrict our illustration to describe the ‘reserve a car’ use case. It specifies that a client intends to reserve a car, at a specific branch, for a specific period of time. If (s)he is eligible and a car is available, a reservation is recorded (See figure 8).

This use case clearly describes the main scenario:
- Client requests to reserve a car of a specific group or model, for a specific period of time and from a specific branch.
- System checks the availability.
- System proposes a reservation.
• System asks for personal information.
• Client provides personal information
• System validates client’s personal information
• System provides a reservation number
• System asks for confirmation.
• Client confirms the reservation.

It also clearly describes the alternative scenarios, according to the following decisions:
• Is a car available?
• Is personal information valid?
• Does the client confirm the reservation?

The four other use cases can be described by the same technique.

4.4. From CIM to PIM

In the “reserve a car” use case, reservation is the component process. Reservation is transformed into a Moment-Interval archetype.

There are three states of Reservation: 1) requested 2) assigned and 3) completed. These states are represented by an attribute “Status” of the Moment-Interval Reservation class.

Rental is the next activity enabled by the reservation. There are three states of Rental: 1) current 2) ended and 3) closed. These states are, as for the Reservation, represented by an attribute “Status” of the Moment-Interval Rental class.

Entity components are Client, Car and Branch (See figure 9). They are modeled by the Place, Thing, Role and Description archetypes:

Car:
The ‘CarModel’ and the ‘CarGroup’ are defined by ‘Description’ archetypes.

Client:
Is a ‘Role’ archetype; ‘CreditCard’ is related to the Client and is a ‘Thing’ archetype.

Branch:
Is a ‘Place’ archetype

The analysis of the process components and of their related entity components yields the system analysis model (See figure 10).

This PIM has been completed by class attributes and methods, identified in the use cases and with the help of the generic elements defined with each archetype.

4.4. From CIM to PIM

In the “reserve a car” use case, reservation is the component process. Reservation is transformed into a Moment-Interval archetype.

There are three states of Reservation: 1) requested 2) assigned and 3) completed. These states are represented by an attribute “Status” of the Moment-Interval Reservation class.

Rental is the next activity enabled by the reservation. There are three states of Rental: 1) current 2) ended and 3) closed. These states are, as for the Reservation, represented by an attribute “Status” of the Moment-Interval Rental class.

Entity components are Client, Car and Branch (See figure 9). They are modeled by the Place, Thing, Role and Description archetypes:

Car:
The ‘CarModel’ and the ‘CarGroup’ are defined by ‘Description’ archetypes.

Client:
Is a ‘Role’ archetype; ‘CreditCard’ is related to the Client and is a ‘Thing’ archetype.

Branch:
Is a ‘Place’ archetype

The analysis of the process components and of their related entity components yields the system analysis model (See figure 10).

This PIM has been completed by class attributes and methods, identified in the use cases and with the help of the generic elements defined with each archetype.

5. Conclusion and future work

In this paper we have proposed an original approach based on existing techniques:
• business process modeling by activity diagram;
• EBPs as the primitive entities of BPMs;
• use case modeling by activity diagram;
• use case model transformation into a system analysis model, based on the direct mapping of EBP elements and system components;
• a set of archetypes to build the system analysis model from the system components.

We think that this approach provides a disciplined way towards automation of the CIM-to-PIM transformation.

Topics of our current and future work are: 1) formalization of the transformation rules from CIM elements to PIM elements, 2) automatic transformation from CIM to PIM, 3) building of Business Process Models for specific industry domains, and 4) creation of catalogues of CIM and PIM models ready to be assembled [24].
The transformation techniques should give the possibility to develop applications within the enterprise or across enterprises, using service-oriented standards.

6. References