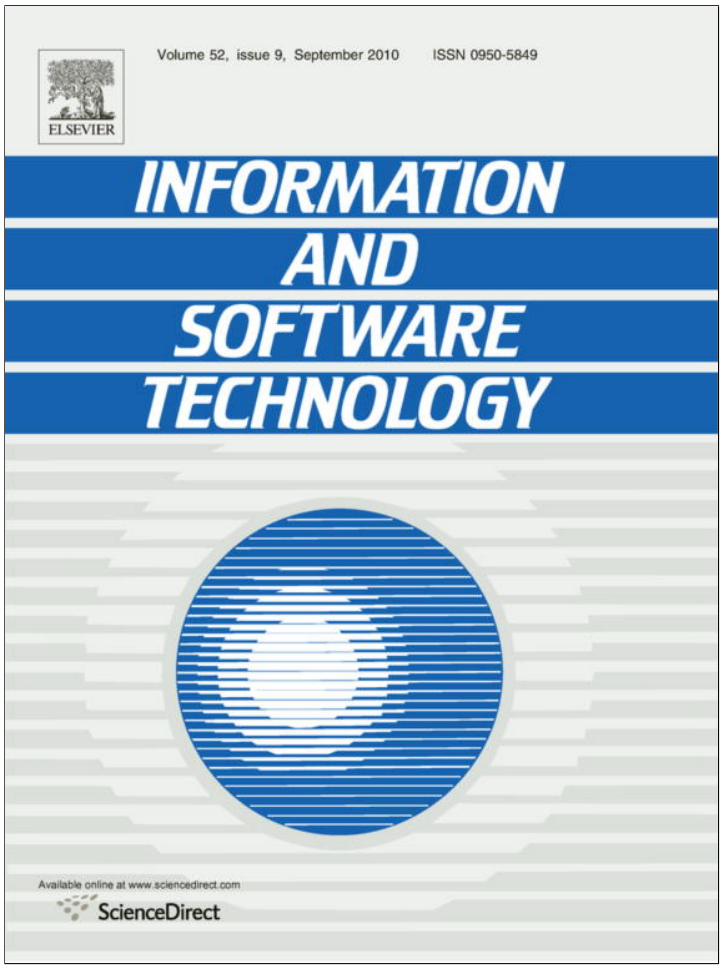


Information and Software Technology 52 (2010) 945–971

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach

Alfonso Rodríguez [a], Ignacio García-Rodríguez de Guzmán [b], Eduardo Fernández-Medina [b,*], Mario Piattini [b]

[a] Department of Computer Science and Information Technology, University of Bio-Bio, Casilla 447, Chillán, Chile
[b] ALARCOS Research Group, Information Systems and Technologies Department, University of Castilla-La Mancha, Paseo de la Universidad 4, 13071 Ciudad Real, Spain

ARTICLE INFO

Article history:
Received 19 August 2009
Received in revised form 31 March 2010
Accepted 31 March 2010
Available online 9 April 2010

Keywords:
MDA
Secure business processes
BPMN
UML

ABSTRACT

Context: Model-Driven Development (MDD) is an alternative approach for information systems development. The basic underlying concept of this approach is the definition of abstract models that can be transformed to obtain models near implementation. One fairly widespread proposal in this sphere is that of Model Driven Architecture (MDA). Business process models are abstract models which additionally contain key information about the tasks that are being carried out to achieve the company's goals, and two notations currently exist for modelling business processes: the Unified Modelling Language (UML), through activity diagrams, and the Business Process Modelling Notation (BPMN).
Objective: Our research is particularly focused on security requirements, in such a way that security is modelled along with the other aspects that are included in a business process. To this end, in earlier works we have defined a metamodel called secure business process (SBP), which may assist in the process of developing software as a source of highly valuable requirements (including very abstract security requirements), which are transformed into models with a lower abstraction level, such as analysis class diagrams and use case diagrams through the approach presented in this paper.
Method: We have defined all the transformation rules necessary to obtain analysis class diagrams and use case diagrams from SBP, and refined them through the characteristic iterative process of the action-research method.
Results: We have obtained a set of rules and a checklist that make it possible to automatically obtain a set of UML analysis classes and use cases, starting from SBP models. Our approach has additionally been applied in a real environment in the area of the payment of electrical energy consumption.
Conclusions: The application of our proposal shows that our semi-automatic process can be used to obtain a set of useful artifacts for software development processes.

1. Introduction

Model-Driven Development (MDD) offers a significant change in the way in which software is developed. The 'object' is the head element, which has been considered for the development of software in recent decades. However, the complexity of the information systems that are now being developed is so great that a change in paradigm is necessary for the industrialization of software construction. The scientific community has spent several years studying this paradigm change, through which software development is guided by the idea of 'everything is a model', rather than 'everything is an object' [10]. This discipline (also known as Model Driven Engineering – MDE) therefore considers models as being the most important element for software development, maintenance and evolution through model transformation [41]. Model Driven Architecture (MDA) is an instance of MDE developed by OMG [44], and its objectives include separation from business-neutral descriptions and platform-dependent implementations, modelling specific aspects of a system under development with specialized domain-specific languages, the establishment of precise relations between these different languages in a global framework and, in particular, the ability to express operational transformations between them [10]. As a consequence, MDA provides portability, productivity, interoperability and reusability by means of the architectural separation of concerns and throughout the complete development lifecycle, covering analysis and design, programming, testing and component assembly, along with coding and maintenance [32,40]. In short, MDA allows for the fast and effective development of systems that make use of new technological platforms, but which are based on the organization's existing business processes [35].

* Corresponding author. Tel.: +34 926295300; fax: +34 926295354.
E-mail addresses: alfonso@ubiobio.cl (A. Rodríguez), Ignacio.GRodriguez@uclm.es (I.G.-R de Guzmán), Eduardo.FdezMedina@uclm.es (E. Fernández-Medina), Mario.Piattini@uclm.es (M. Piattini).

0950-5849/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2010.03.015

The MDA approach defines models at three different levels of abstraction [26]: The *computation independent viewpoint* (represented by the Computation Independent Model or CIM) which focuses on the environment of the system and is usually built by business analysts; the *platform independent viewpoint* (represented by the Platform Independent Model or PIM) which is built by system analysts and software architects and focuses on the operation of a system while hiding the details necessary for a particular platform; and the *platform specific viewpoint* (represented by the Platform Specific Model or PSM) which combines the platform independent viewpoint with an additional focus on the details of using a specific platform by a system [44]. In the context of MDA, Query/View/Transformation (QVT) [45] can be considered as one of the most appropriate model transformation language since it has been proposed by the OMG, and can currently be seen as a standard built [34] from the collaboration of many parties interested in model transformations. QVT language plays an important role in the OMG metamodel family composed of, among others, Unified Modelling Language (UML), which is intended to assist in the software life cycle development [46], Common Warehouse Metamodel (CWM), which is intended to model data sources such as relational databases, data warehouses, among others [43], Knowledge Discovery Metamodel (KDM), which offers support to represent information obtained from legacy systems in re-engineering processes) [42] and Meta Object Facility (MOF), which is the basis for all the OMG metamodels and provides a common meta–meta language which makes it possible to work jointly with the aforementioned metamodels [45]. Since all the OMG metamodels are defined from MOF, QVT transformations can be carried out among them.

On the other hand, business processes (BP) are strategic for enterprises, and they play an interesting role in the information system development process. Business processes include details of the activities that enterprises need to perform to obtain a service or a product which meets their clients' needs. Business processes therefore offer a detailed view of the activities the companies perform, and how they are executed. Business processes thus help the business analyst to have a better understanding of the business, and this consequently helps to improve it and to make it much more flexible to the changes that arise in the market, that is to say, to be more competitive. Moreover, possessing business process models helps them to understand the sphere of the problem, and the fact that they contain very precise details about the needs that the information systems that computerise these processes must meet signifies that they can be used as a source of exceptional requirements and used in the first stages of the software development. It is consequently important to have a description of BP in a language that provides the models available in order to understand, adapt and improve them. During the last decade, this need has led to the appearance of the Business Process Modelling Notation, Business Process Diagram (BPMN–BPD) [12], and the Unified Modelling Language 2.0 Activity Diagrams (UML 2.0-AD) [46] have been improved to make it possible to represent business processes. Both of these – the notation and the language – constitute valuable tools for describing BPs because they fulfil a dual purpose: firstly, they facilitate the work of business analysts (graphical language) and secondly, they serve as a starting point for a software development process for system analysts [37]. With an MDA focus, business process models are located at the CIM level and from there, by applying the appropriate transformations, it is possible to construct system views nearer to implementation.

In other respects, the proliferation of the connectivity of information systems and the increasing complexity of applications and services have increased possibilities of security breaches

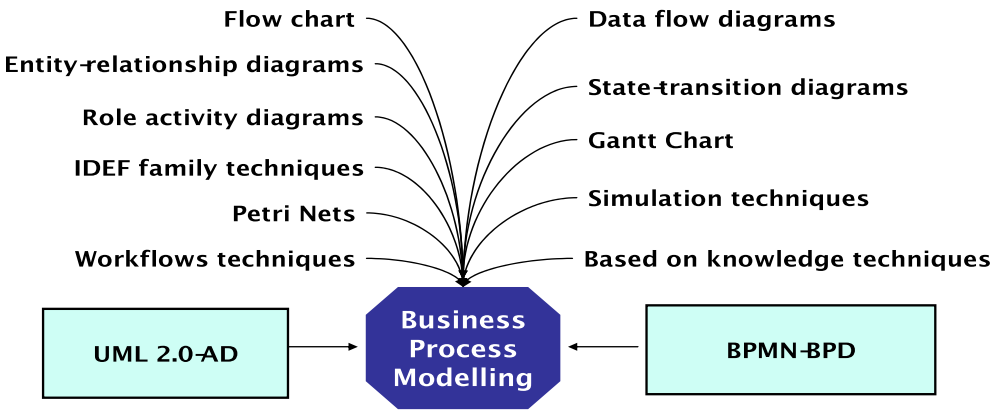


Fig. 1. Business process modelling techniques.

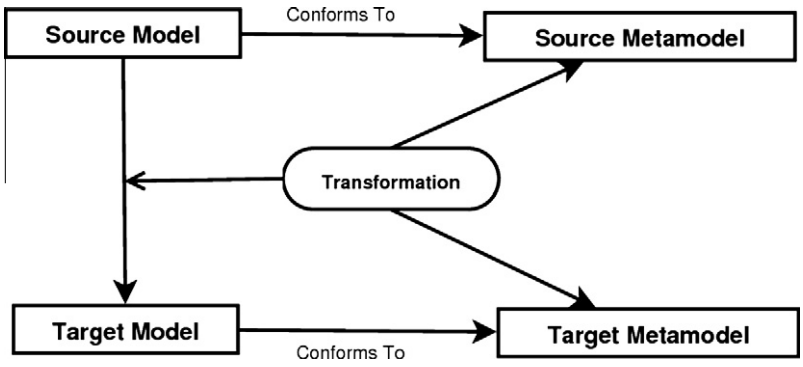


Fig. 2. General model transformation schema.

[68]. In addition, owing to the heavy dependence of computer network-based applications on various software and software controlled systems, the consequences of a security breach in these applications may range from extensive financial losses to dangers to human life, meaning that the threat of technology-enabled crime has given rise to a growing demand to devise new response strategies [15], and software security has therefore become an essential issue [73].

Although the importance of modelling security requirements as early as possible in the software development cycle (and therefore within business process models) is widely accepted by the software engineering community as a means to improve the quality of the developed software, until now the business analyst perspective in relation to security has hardly been dealt with. Works related to business process security [6,27,28,39,59,60,64,67] have contributed interesting ideas, but most of the cases do not make it possible to model security needs together with the standard languages for modelling businesses process. Nor do they offer continuity and traceability of these requirements in the rest of the software development process. Consequently, in earlier works, we have defined a manner in which to represent security requirements along with the description of the business processes. Our proposals consist of the definition of an extension, which we have called business process security (BPSec), for UML 2.0-AD [57] and BPMN–BPD [54] which offers the semantic and syntactic tools that allow the business analyst to specify security requirements within business process models. Moreover, we have defined a preliminary strategy (that we complete, extend, improve, formalize and detail in this paper) with which to obtain analysis class diagrams [53] and use case diagrams [58] directly from secure business process models, which represent a very rich requirements source. Obviously, these use case and class models are not definitive, and must be refined, and they can be considered the first version of the analysis model, thus enabling software developers to study capturing and refinement requirements in more depth.

The structure of the remainder of the paper is as follows: in Section 2, we offer a background section which introduces the reader to our multidisciplinary related research. In Section 3, we present an overview of our general approach, and Section 4 contains the main contribution of this paper, i.e., the set of QVT transformations. In Section 5 we present a technological prototype to support our proposal. Section 6 presents the application of our approach to a real application. In Section 7 we discuss related works and, finally Section 8 presents our conclusions.

2. Background

This section presents the main aspects that underlie and justify our proposal. Section 2.1 looks at the main notations for business process modelling, while Section 2.2 tackles the transformation of models from a general perspective.

2.1. Business Process Modelling Notations

Enterprises have been modelling processes for many years, although their models have not always been known as 'Business Process Models'. A business process was given a standard definition at the end of the 1990s, which was: "a set of procedures or activities which collectively pursue a business objective or policy goal" [69]. Until then, available modelling techniques were used to describe '*how*' organizations performed their work. The models obtained were used as follows: first, to train new employees, second as aids in re-engineering processes, third, to develop simulations to test the processes on inputs that had not occurred in real life and, finally, to develop systems with which to automate the processes they modelled [18].

The increasing popularity of business process orientation has yielded a rapidly growing number of methodologies, modelling techniques and tools to support it [2,18,22]. Generic techniques such as data flow diagrams, entity-relationship diagrams and Gantt Charts (see Fig. 1) have been used for business process modelling. However, the inflexion point in the area of business process modelling has been brought about by the definition of two techniques for the explicit modelling of business process models: the new UML 2.0-AD and BPMN–BPD. The improvements made to UML 2.0-AD and the creation of BPMN–BPD constitute an important change because both notations satisfy a double purpose which is, on the one hand, to be useful for business specialists since they can use them to view the business process, and on the other hand, to serve as a starting point for the creation of software that supports the business activities that have been described [51].

The most important features of these notations include the possibility of representing organization areas or external agents that perform activities in a business process. Although the relationship between these two notations is dealt with in more detail in [70], the equivalences that we have found make it possible to establish an existing relationship between the concepts from the viewpoint of the similarity of the business process objects that each one represents. The proximity of both notations became evident when, in June 2005, the group responsible for BPMN merged with the OMG group, responsible for the UML standard, making the possibility of convergence more likely [31].

2.2. Model transformations: a general view

Models are representations of a piece of the real world, and it is therefore very important to determine whether a model is correct or not. Metamodels are the mechanism used to create correct models in order to model a certain system. A metamodel "is, in effect, an abstract language for some kind of metadata" [45]. Therefore, a precise metamodel is a prerequisite for performing automated model transformations [41].

According to the MDA standard, a model transformation is the process of converting one model into another model from the same system, and it is considered to be a central part of MDE/MDD [3]. Model transformations are composed of fine-grained rules that transform elements defined in a specific source metamodel into other elements of the target metamodel (see Fig. 2). Furthermore, in the field of MDA, transformations can be carried out in many stages, such as software development, refinements, the administration of models representing different concerns, iterative development and system evolution [34].

Model transformations are defined at the metamodel level; the set of rules and constraints (to obtain the target model from the source model) refers to elements of the metamodels (see Fig. 2). Consequently, transformations can be applied to any source model conforming to the source metamodel and produce (following the rules of the transformation) a target model conforming to the target metamodel.

The concept of 'transformation' has been widely used in several contexts of software engineering. Nonetheless, according to [41], every kind of transformation can be expressed as a model transformation, and can be classified according to a set of characteristics:

- *Number of source and target models*: refers to the possibility of considering more than one source model as well as an output.
- *Technological space*: depending on the technological space of the source and target models, the transformation must be defined to deal with different kinds of representations of the involved models (e.g. technological spaces of UML or XML).

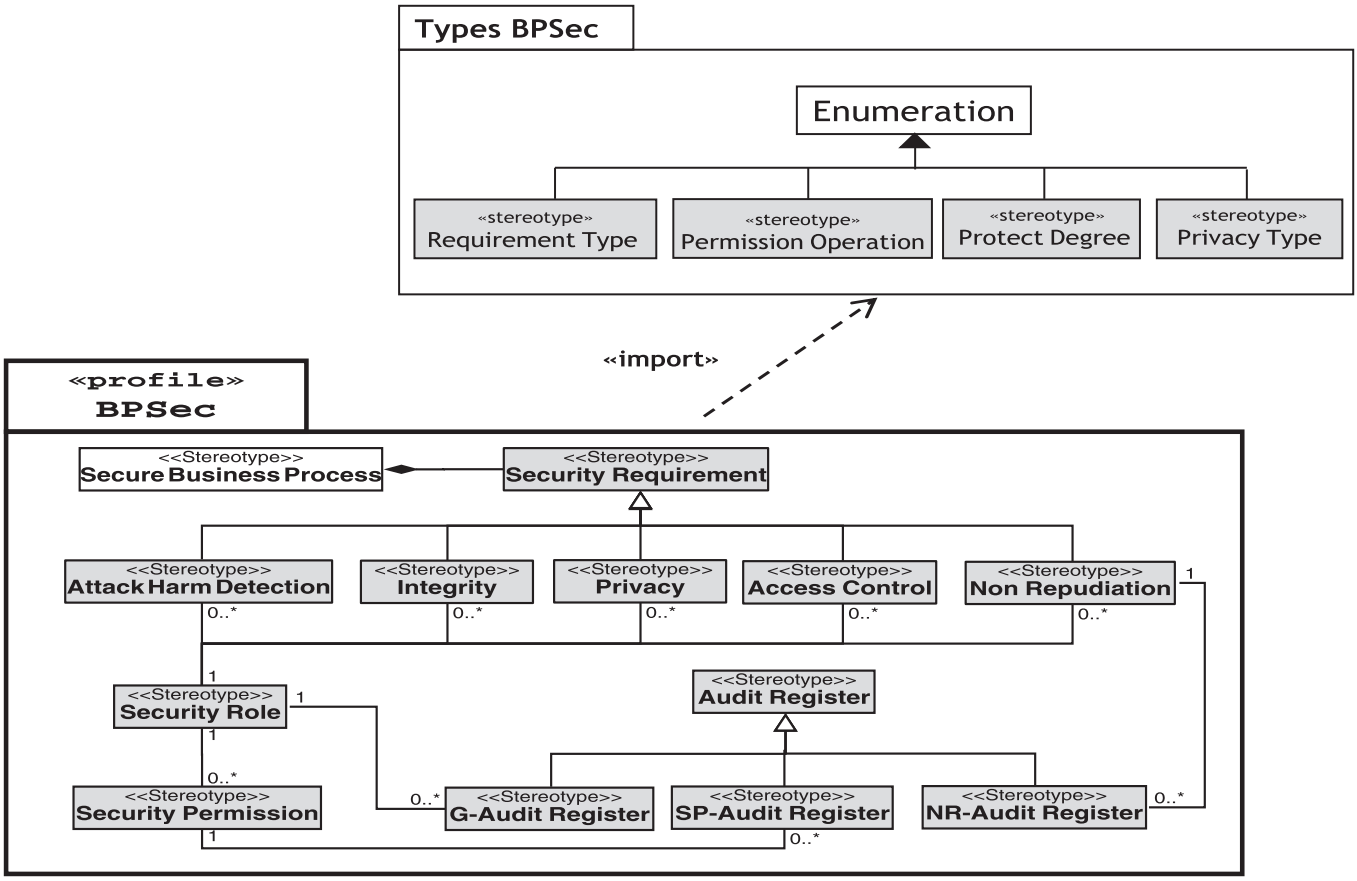


Fig. 3. BPSec-profile.

Fig. 4. Icons to represent security requirements in BPSec.

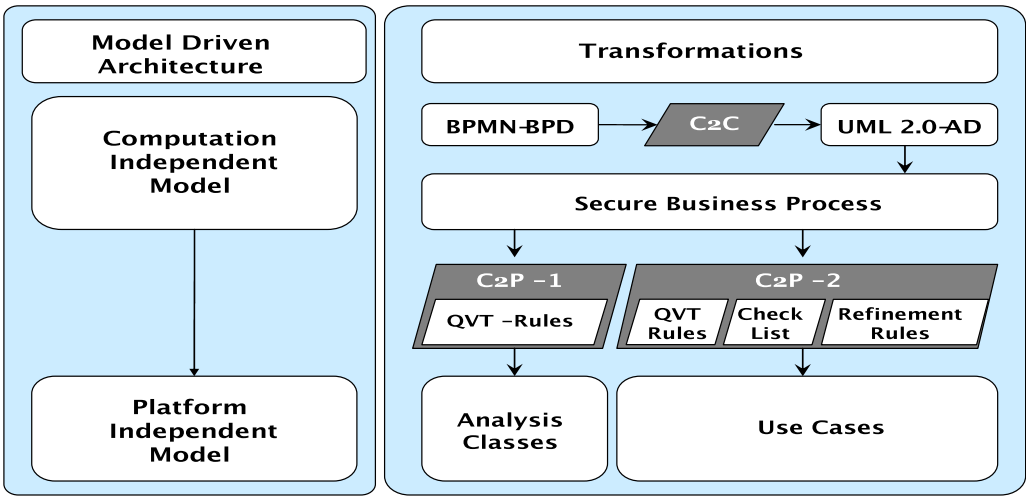


Fig. 5. General Transformation Architecture.

Table 1
Description of the QVT transformation to obtain UML secured artefacts.

	C2C	C2P-1		C2P-2	
		AD2CD[a]	BPSec2CD[b]	AD2UCD[c]	BPSec2UCD[d]
Endogenous					
Exogenous	X	X	X	X	X
Horizontal	X				
Vertical		X	X	X	X
Syntactical					
Semantical	X	X	X	X	X
Technological space	MDA	MDA	MDA	MDA	MDA
Input/output models	1/1	1/1	2/1	1/1	2/1
Description	Generates a UML Activity Diagram from BPMN business process specification	Generates UML Analysis Classes from Secure Business Process models		Generates UML Use Cases from Secure Business Process models	

[a] Contraction for ActivityDiagram to ClassDiagram.
[b] Contraction for BPSec to ClassDiagram.
[c] Contraction for ActivityDiagram2UseCaseDiagram.
[d] Contraction for BPSec2UseCaseDiagram.

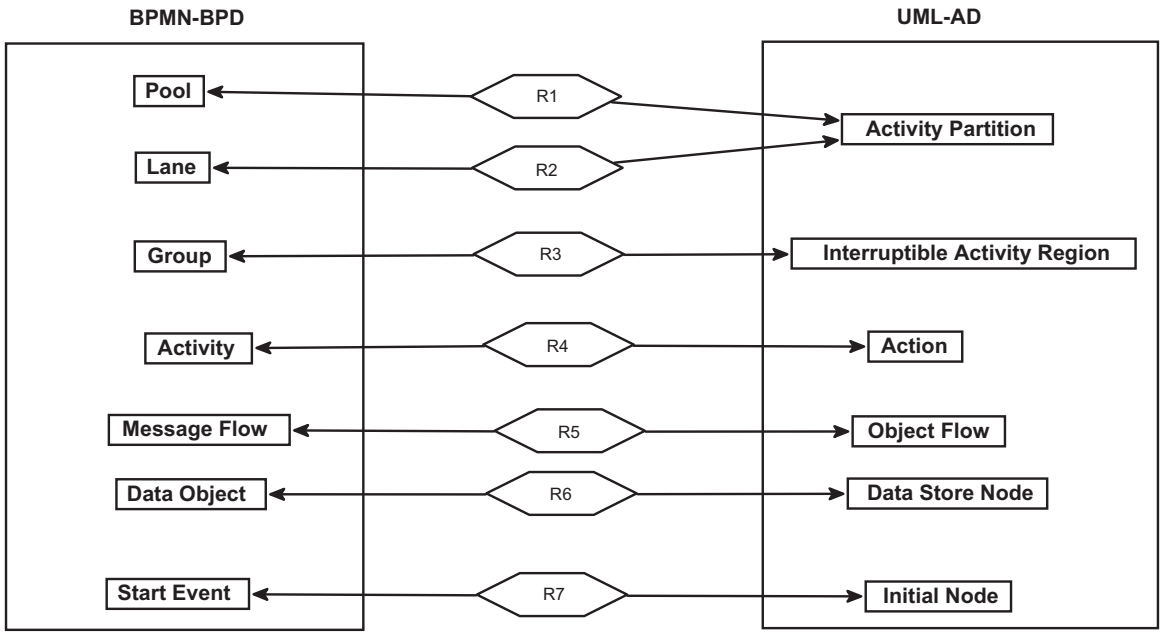


Fig. 6. C2C: BPMN–BPD2UML-AD transformation.

- *Endogenous and exogenous transformations*: when source and target models share the same metamodel, the transformation is called endogenous; otherwise, the transformation is considered to be exogenous.
- *Horizontal and vertical transformations*: a transformation is considered horizontal when source and target models reside in the same abstraction model, whereas the transformation is vertical when source and target models reside at different levels of abstraction.
- *Syntactical and semantic transformations*: it is possible to find/develop transformations that take the semantic of the source model into account.

One important topic related to transformations is the language used to write and describe them. Transformations are specified with suitable languages which are intended to deal with models. Thus, a model transformation language is (in summary) a language that: (i) takes a model as input and (ii) according to a set of rules, and (iii) produces an output model. It is currently possible to find many model transformation languages such as BOTL [13], SiTra [3], Kermeta[1] [19], GReAT [1], ATL [4,11,29,30], and QVT [45], among.

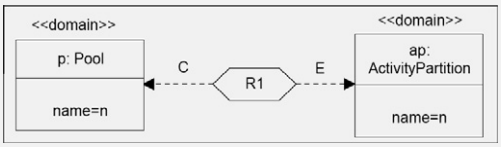
Despite the great variety of model transformation languages, QVT is the only proposal from the Object Management Group (OMG). The current QVT version depends on the specifications of MOF 2.0 [48]and OCL 2.0 [49]. QVT has a dual hybrid nature, since QVT transformations can be written declaratively (declarative syntax) or imperatively (imperative syntax with the operational mappings). The most extended syntax is that which is declarative. QVT transformations written with the declarative syntax are divided into several parts or relations which must be held to complete the transformation. Each relation specifies how one (or various) element(s) from the source model(s) are transformed into one (or various) element(s) of the target model. Relations are specified using a declarative syntax.

Several tool proposals exist to deal with QVT. One example is 'UMT-QVT' [65], an environment designed to perform

[1] http://www.kermeta.org/.

Table 2
Textual view of the BPMN-BPD2UML-AD transformation. Graphical example of the first relation.

Top Relation R1: This *relation* transforms each *Pool* into an *ActivityPartition* with the same name



Top Relation R2: This *relation* transforms each *Lane* into an *ActivityPartition* with the same name
Top Relation R3: This *relation* transforms each *Group* into an *InterruptibleActivityRegion* with the same name
Top Relation R4: This *relation* transforms each *Activity* into an *Action* with the same name
Top Relation R5: This *relation* transforms each *MessageFlow* into an *ObjectFlow* with the same name
Top Relation R6: This *relation* transforms each *DataObject* into an *DataStoreNode* with the same name
Top Relation R7: This *relation* transforms each *StartEvent* into an *InitialNode* with the same name

Table 3
Textual BPMN-BPD2UML-AD transformation.

```
transformation BPMN-BPD2UML-AD
top relation R1
{
  checkonly domain bpmn_BusinessProcessDiagram p:Pool {name=n}
  enforce domain uml_ActivityDiagram ap:ActivityPartition {name=n}
}
top relation R2
{
  checkonly domain bpmn_BusinessProcessDiagram l:Lane {name=n}
  enforce domain uml_ActivityDiagram ap:ActivityPartition {name=n}
}
top relation R3
{
  checkonly domain bpmn_BusinessProcessDiagram g:Group {name=n}
  enforce domain uml_ActivityDiagram ir:InterruptibleActivityRegion
   {name=n}
}
top relation R4
{
  checkonly domain bpmn_BusinessProcessDiagram ac:Activity {name=n}
  enforce domain uml_ActivityDiagram act:Action {name=n}
}
top relation R5
{
  checkonly domain bpmn_BusinessProcessDiagram ms:MessageFlow
   {name=n}
  enforce domain uml_ActivityDiagram of:ObjectFlow {name=n}
}
top relation R6
{
  checkonly domain bpmn_BusinessProcessDiagram do:DataObject
   {name=n}
  enforce domain uml_ActivityDiagram dsn:DataStoreNode {name=n}
}
top relation R7
{
  checkonly domain bpmn_BusinessProcessDiagram s:StarEvent {name=n}
  enforce domain uml_ActivityDiagram in:InitialNode {name=n}
}
```

transformations based on QVT. 'UMT-QVT' is based on UMT, an open tool with a special plug-in to transform models according to the QVT syntax. SmartQVT[2] also implements QVT, but uses the operational QVT syntax (not the declarative syntax). SmartQVT is presented as an open-source plug-in for Eclipse.[3] MOMENT[4] is a generic framework for model management. In addition to the anticipated environments, MOMENT (which is based on MAUDE,[5] a language based on rewriting logic) includes an Eclipse plug-in and implements the declarative syntax of QVT. There are also commercial tools, such as Together Architect 2006 [33] which implements the operational syntax of QVT. MediniQVT[6] is a complete implementation of the QVT OMG standard. This tool for modelling transformation works either as an Eclipse plug-in or a standalone application and accepts models written with UML2 or ECORE.

3. Overview of our MDA approach

The creation of models at high levels of abstraction and their subsequent transformation, based on the specification of semi-formal rules as presented in this paper, makes it possible to capture requirements early, which reduces software creation costs and time. The MDA proposal offers the conceptual framework to carry out this task. In our proposal, transformations are carried out through which it is possible to obtain a set of UML artefacts from a secure business process model. These artefacts, which include security requirements, form part of the specifications that make it possible to describe and characterise an information system.

The model transformations are carried out using a source model and a target model as a base, and the specifications of metamodels must be included in order to describe the transformation rules. The source model that we use in this work corresponds to the metamodel for a secure business process. This metamodel corresponds to the UML 2.0-AD or BPMN–BPD metamodel, for which we have described an extension [54,57] and through which it is possible to incorporate security requirements along with the description of the business process. With the extension, which we have named BPSec, it is possible to represent security requirements both with UML 2.0-AD and BPMN–BPD. These security requirements must be understandable for business analysts and unambiguous for security experts. To this end, we have considered that security requirements must meet the characteristics of: (i) clarity in their definition, (ii) potential importance in the field of business and (iii) definition independence in relation to specific security solutions. To do this, we have used the taxonomy of security requirements proposed by [21] as a reference. We have selected from that taxonomy the security requirements Access Control, Attack-Harm Detection, Security Auditing, Integrity, NonRepudiation and Privacy. However, in this paper we will only describe BPSec and it is assumed that the metamodels for UML 2.0-AD, classes and use cases in [47] and the metamodel for BPMN–BPD, in an earlier version in [54] are known or at least available.

The basic elements of BPSec coincide both for UML 2.0-AD and for BPMN–BPD. For reasons of clarity, they have been described as two UML packages (see Fig. 3). The Type BPSec package includes the types of data that define operation permission, types of Privacy, degrees of protection and types of requirements. These types of

[2] http://smartqvt.elibel.tm.fr/index.html.
[3] http://www.eclipse.org/.
[4] http://moment.dsic.upv.es/component/option,com_frontpage/Itemid,1/.
[5] http://maude.cs.uiuc.edu/.
[6] http://projects.ikv.de/qvt/.

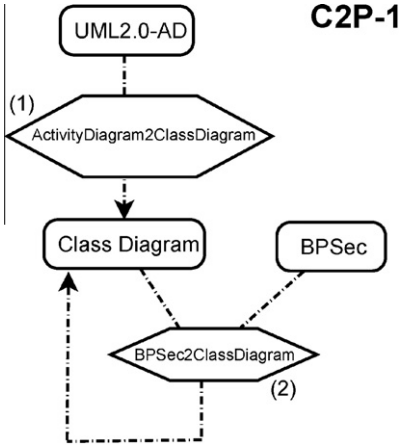


Fig. 7. Overview to the C2P-1 transformation.

data form the BPSec space that makes it possible to use them as labelled values in the new defined stereotypes.

The «SecureBusinessProcess» stereotype is the generic name for the stereotype that specialises the Activity (UML 2.0-AD) class and the BMN-BPD BusinessProcessDiagram class. This stereotype thus preserves all the relations of the class that it specialises. For a business process to be secure, the «SecureBusinessProcess» stereotype must be composed of at least one security requirement. This makes it possible to establish the relation of the security requirements with the UML 2.0-AD or BPMN–BPD elements. The «SecurityRequirement» stereotype brings together the «AccessControl», «AttackHarm Detection», «Integrity», «NonRepudiation» and «Privacy» stereotypes that represent the security requirements. Our proposal allows the business analyst to specify audit register related to security requirements. We have defined the «AuditRegister» stereotype which has been specialized depending on the information which needs to be registered, in «G-AuditRegister» (for «AccessControl», «AttackHarm Detection», «Integrity», and «Privacy» requirements), «NR-AuditRegister» (for «NonRepudiation» requirement), and «SP-AuditRegister» (for security roles which have been specified together with «AccessControl» requirement). Additionally, we need both «SecurityRole» and «SecurityPermission» to complement the security specifications. Finally, the «RequirementType», «PermissionOperation», «ProtectionDegree» and «PrivacyType» stereotypes are types of data that make it possible to specify the very characteristics of the security requirements of which the profile is formed. From a graphic point of view, our proposal considers a padlock (see Fig. 4a), a *standard de facto*, to represent security requirements. The same symbol, the padlock, but with a twisted corner (see Fig. 4b) is used to represent a Security Requirement with Audit Register. The set of security requirements is shown in Fig. 4.

These stereotypes will be used to include abstract security requirements (understandable to business analysts) within business process models. A brief description of these stereotypes is shown as follows (a detailed description can be found in [55,57]):

- SecurityRequirement will be a generic security requirement which should be specified through a concrete requirement.
- AuditRegister is specified in order to force an audit register for a specific element of the business process (different types of audit registers can be specified).
- AccessControl is specified to represent the need to control the access to a specific resource (process, data, etc.) of the business process model.
- AttackHarmDetection is necessary to specify the need to detect, register and notify an attempted attack or threat, whether it is successful or not.
- Integrity is specified to represent the need to protect specific components of the business process model from intentional and non-authorized alteration (a level of protection can be specified).
- NonRepudiation represents an abstract security requirement which establishes the need to avoid the denial of any aspect of the interaction (e.g. message, transaction, transmission of data).
- Privacy will specify the need to protect a specific element from non-authorized parties in order to avoid to obtain sensitive information. The type of Privacy can be specified (anonymity or confidentiality).

Our proposal is shown in its complete form in Fig. 5. The first column represents the part of MDA in which this work is framed. In this figure it is possible to distinguish the models that our architecture is composed of: the secure business process model (considered as a CIM model) and the analysis class and the use case models (considered as PIM models). It is important to highlight that business processes are the highest abstraction representation level of the logic of an organization, so according to the definition

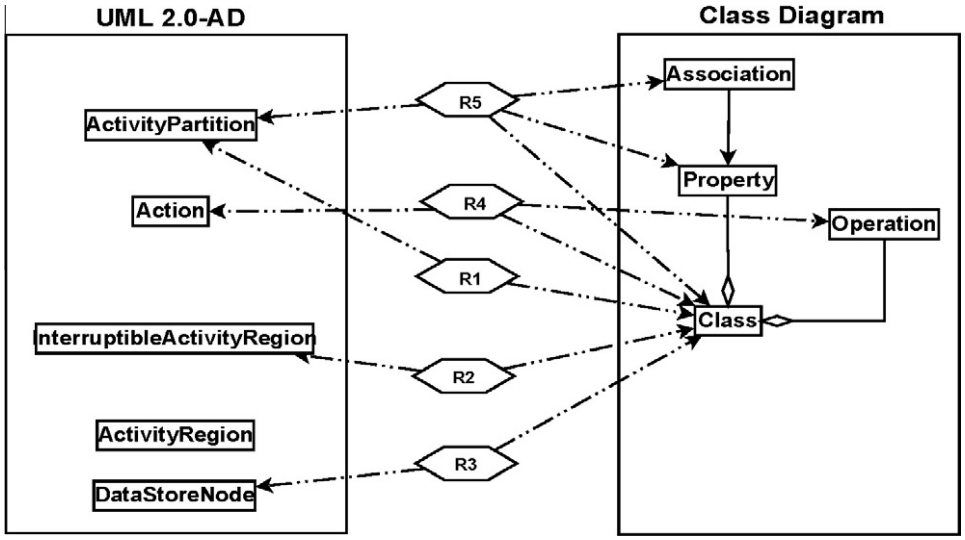
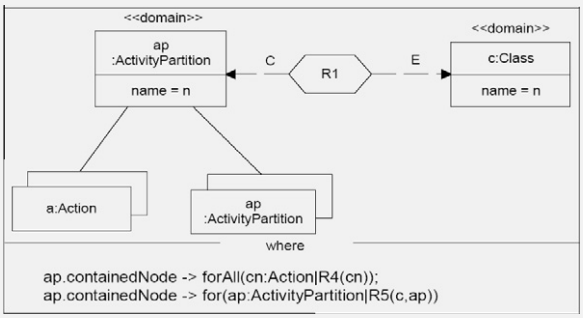
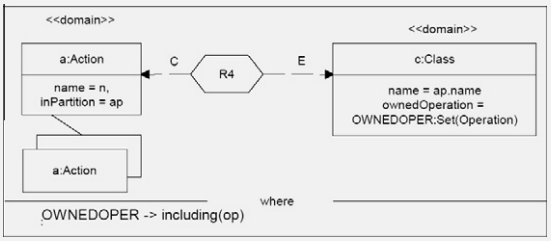
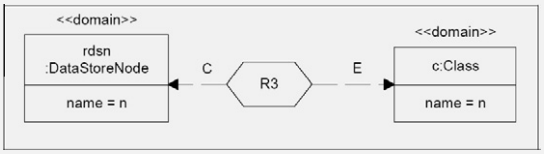
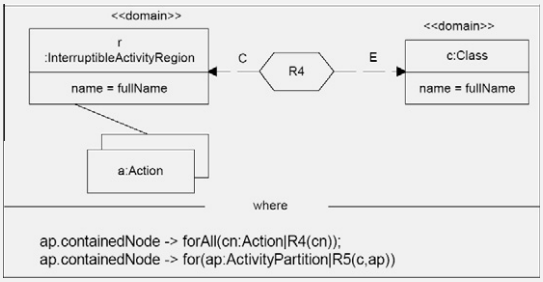
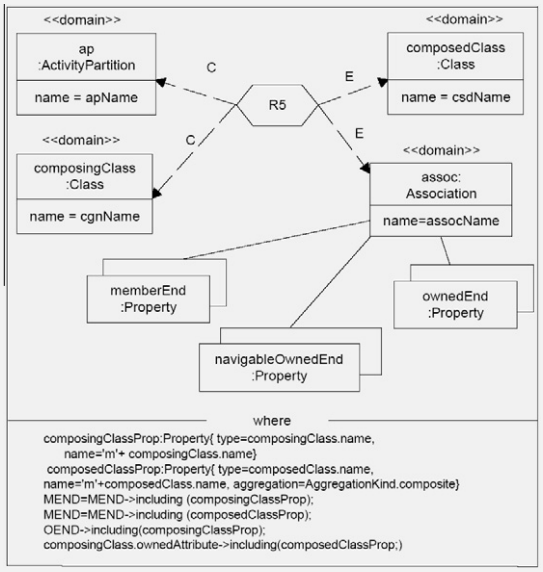


Fig. 8. C2P-1: ActivityDiagram 2ClassDiagram.

Table 4
Detailed view of the transformation ActivityDiagram 2ClassDiagram.

Top Relation R1: A *Partition* is transformed into a *Class* with the same name. All *Actions* inside the *Partition* are identified and transformed into *Operations* (see *Relation R4*). Finally, sub*Partitions* are in turn associated with the *Partitions* container to later create composition relationships (according to the *Relation* R5).	
Top Relation R2: This *relation* transforms a *Region* into an analysis *Class*. *Actions* included in the *Region* are identified to be later transformed into *Operations* (according to the *Relation R4*). Furthermore, partitions contained in the *Region* are verified in order to assign a meaningful name to the *Region* and to the resultant analysis *Class*.	
Top Relation R3: This *relation* transforms a *Data Store* specification into an analysis *Class*.	
Relation R4: In this *relation* the *Actions* are transformed into *Operations*. The latter are attached to the corresponding analysis *Classes* (which have been derived from those *Partitions* or *Regions* containing the *Actions*).	
Relation R5: This *relation* creates the *Aggregation* relationships between those *Classes* which have been derived from *Partitions* containing sub*Partitions*.	

of the MDA standard [44], business processes are considered to be CIM models. On the other hand, the use case and class models are considered to be PIM because both models are in the context of the software development, but it is possible to distinguish different sub-levels in the PIM level, and in our case, use case models would be allocated to a higher level than class models. It would be possible to consider a use case model as a CIM model when no other more abstract representations are considered (e.g. business processes).

The proposed transformations are shown in the second column in a dark colour. We have identified two types of transformations. The first makes it possible to establish equivalence between the business concepts described with BPMN–BPD and the corresponding concept in UML 2.0-AD. This is a transformation between the CIM (C2C)

Table 5
ActivityDiagram 2ClassDiagram QVT Transformation.

```
transformation ActivityDiagram 2ClassDiagram
top relation R1{
  checkonly domain uml_ActivityDiagram ap:ActivityPartition {name=n}
  enforce domain uml_ClassDiagram c:Class {name=n}
    where { ap.containedNode -> forAll(cn:Action|R4(cn));
        ap.containedNode -> for(ap:ActivityPartition|R5(c,ap))}}
top relation R2{
  fullName:String;
  checkonlydomain uml_ActivityDiagram
   r:InterruptibleActivityRegion{name=fullName}
  enforce domain uml_ClassDiagram c:Class{name=fullName}
  where{ r.containedNode -> iterate(a:Action;
  activityRegionsName:Set(String) | activityRegionsName ->
   including(a.name));
  activityRegionsName -> iterate(s:String; fullName |
   fullName=fullName.concat(s));
  r.containedNode -> forAll(cn:Action|R4(cn))
}}
top relation R3{
  checkonly domain uml_ActivityDiagram dsn:DataStoreNode {name=n}
  enforce domain uml_ClassDiagram c:Class {name=n}}
relation R4{
  checkonly domain uml_ActivityDiagram ac:Action {name=n, inPartition=ap}
  enforce domain uml_ClassDiagram c:Class{name=ap.name,
      ownedOperation=OWNEDOPER:Set(Operation)}
  enforce domain uml_ClassDiagram op:Operation {name=n, ownerClass=c}
      where { OWNEDOPER -> including(op)}}
relation R5{
  checkonly domain uml_ClassDiagram composingClass:Class {name=cName}
  checkonly domain uml_ActivityDiagram ap:ActivityPartition
   {name=apName}
  enforce domain uml_ClassDiagram composedClass:Class{name=apName}
  enforce domain uml_ClassDiagram assoc:Association{name=assocName,
    memberEnd = MEND: OrderedSet (Property),
    ownedEnd = OEND: OrderedSet (Property),
    navigableOwnedEnd = NOEND: OrderedSet (Property)}
    where{
      composingClassProp:Property
      {type=composingClass.name, name='m'+ composingClass.name};
      composedClassProp:Property
      {type=composedClass.name, name='m'+composedClass.name
      aggregation=AggregationKind.composite};
      MEND=MEND->including (composingClassProp);
      MEND=MEND->including (composedClassProp);
      OEND->including(composingClassProp);
      maininClass.ownedAttribute->including(secClassProp);
      }}
```

models which seeks to obtain the specification of a secure business process in a single domain, in this case in UML 2.0-AD.

Vertical transformations, which make it possible to move from CIM to PIM (C2P) have been identified as C2P-1 and C2P-2. The specification for transformations from a secure business process model to analysis classes (C2P-1) correspond to a set of QVT rules that make it possible to specify these transformations. Transformations from a secure business process model to use cases (C2P-2) are carried out by applying QVT, a checklist, and refinement rules. All of the transformation rules are specified in detail in Section 4.

4. Model transformation

This section provides details of the specified transformations through which to obtain equivalent business process models (C2C) and UML artefacts (C2P-1 and C2P-2) from the specification of a secure business process.

The QVT transformations developed in the proposed framework to obtain all the analysis artefacts are described and classified in Table 1. All of the transformations can be considered to be *exogenous* since the input and output models conform to different metamodels. Table 1 shows *vertical* transformations such as C2P-1 and C2P-2, where the business process models (CIM level) are transformed into UML models (considered at the PIM level). On the other hand, *horizontal* transformations such as C2C carry out the transformation at the same level of abstraction: the CIM level. With regard to the *semantic* and *syntactical* characteristics of Table 1, all transformations can be considered as being *semantic* transformations. The models in the proposed architecture do not represent syntactical information, but a more abstract type of information. The transformation process abstracts concepts such as security constraints, use cases and analysis classes. Furthermore, transformations such as BPSec2CD and BPSec2UCD improve the input model with the security requirements to obtain the output model, and these transformations could be understood as a kind of refactoring. These transformations are thus considered to be a semantic rather than a syntactical transformation.

The presentation of the transformations is organized as follows: Section 4.1 shows the transformations from CIM to CIM, obtaining an equivalent business process model between the BPMN–BPD and

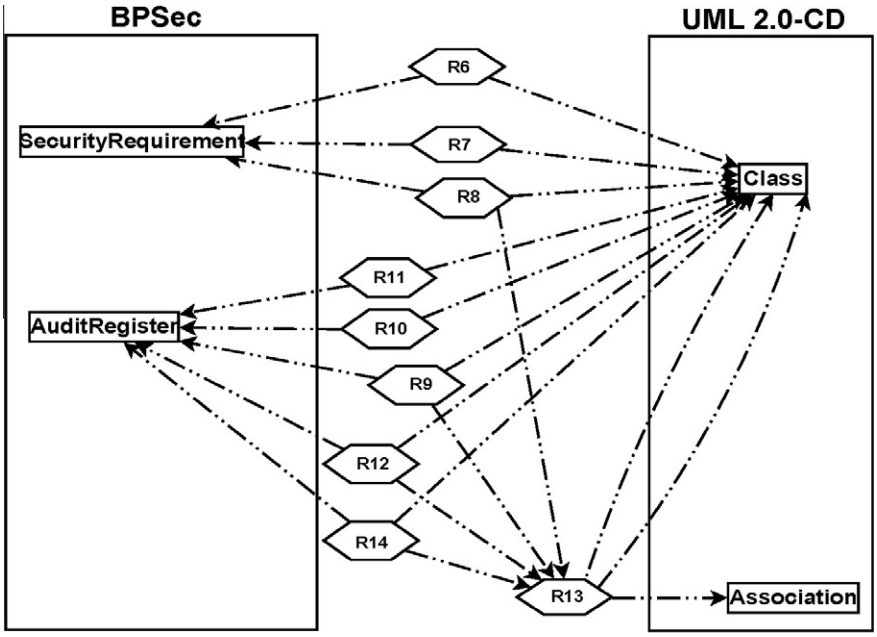
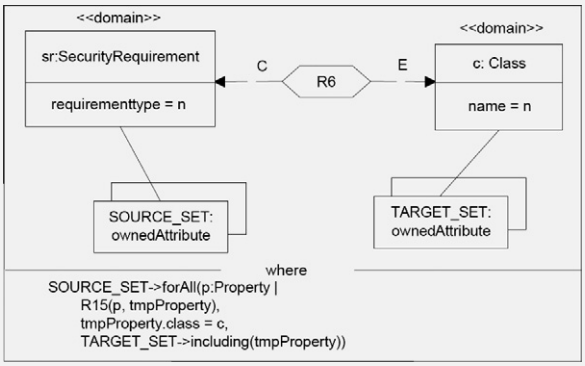


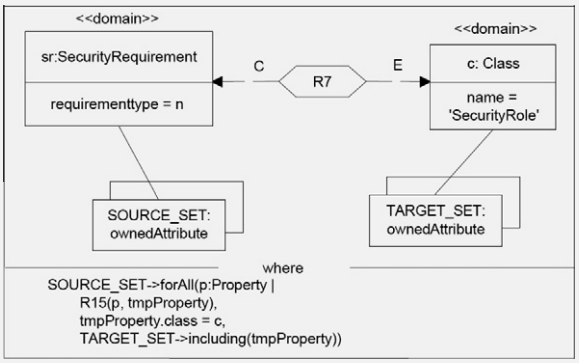
Fig. 9. C2P-1: BPSec2ClassDiagram.

Table 6
Detailed view of the BPSec2ClassDiagram transformation.

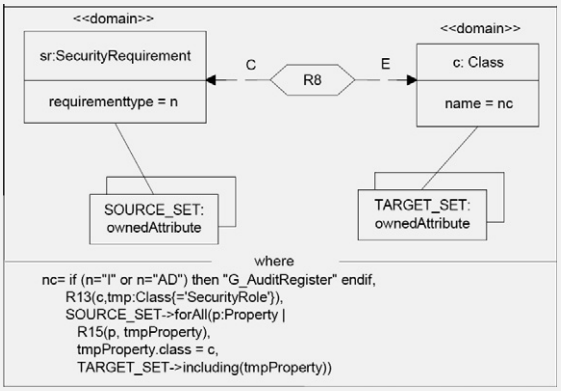
Top Relation R6: A *Class* gets the same name as the specified security requirement. Invokes R15 relation to populate the *Class* with the security related properties.



Top Relation R7: A *Class* named «SecurityRole» is created and associated with the security requirement obtained from the relation R6. The *SecurityRole* class is annotated with the corresponding properties of the SecurityRequirement by means of the R15 relation.



Top Relation R8: A *Class* named «G-AuditRegister» is created (and associated with the security requirement) according to whether the security requirement is an *Integrity* and/or *Attack* and *Threat Detection*, and a set of properties is provided by means of the R15 relation.



Top Relation R9: A *Class* named «G-AuditRegister» is created and associated with an Access Control requirement (annotated with an audit record). By means of R15 the target *Class* is annotated with the properties of the *AuditRegister* element.

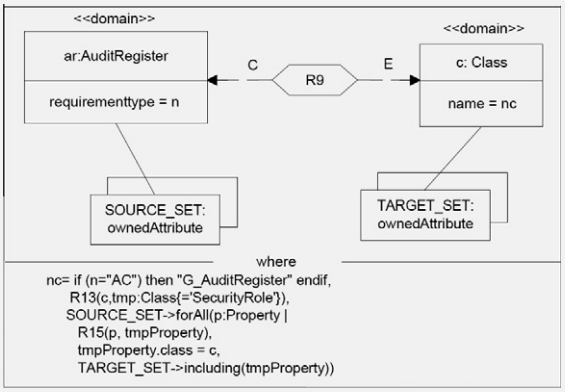
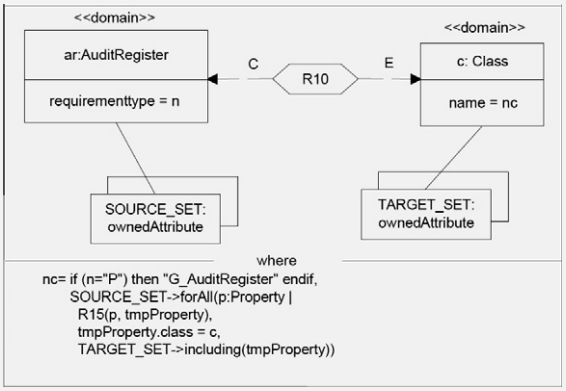
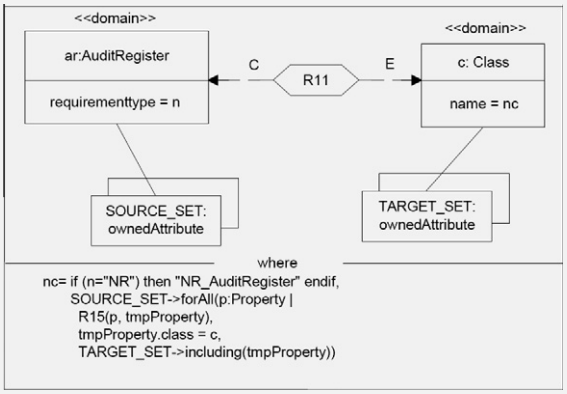


Table 6 (*continued*)

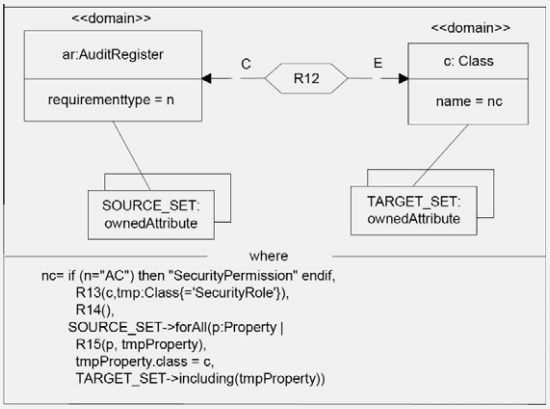
Top Relation R10: A *Class* «G-AuditRegister» is created and associated with a Privacy specification (with an audit register). By means of R15 the target *Class* is annotated with the properties of the *AuditRegister* element.



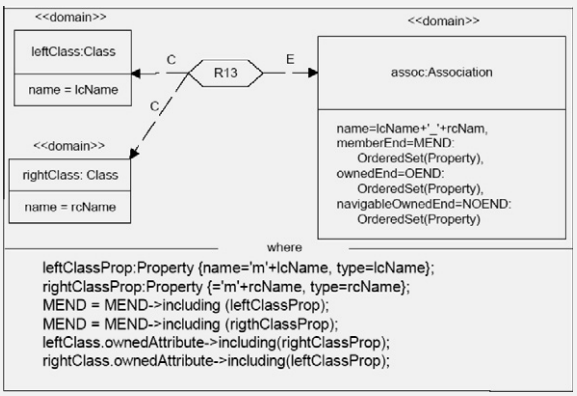
Top Relation R11: A *Class* «NR-AuditRegister» is created and associated with a *NonRepudiation* specification (with an audit register). R15 is invoked to provide the NR-AuditRegister *Class* with the properties of the AuditRegister.



Top Relation R12: A *Class* named «SecurityPermission», related to an *Access Control* specification, is created. Finally the *SecurityPermission* class is provided with the required properties by means of R15.

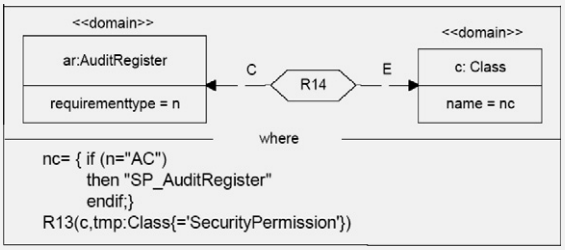
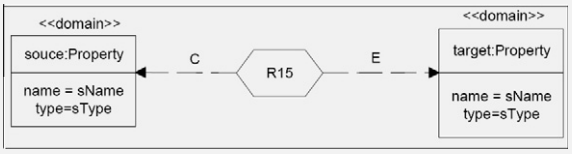


Relation R13: This relation, which is a core relation in this transformation, establishes the relationship (at model level) between the PIM *Classes* and the different security requirements specified in the «SecurityRole» class.



(*continued on next page*)

Table 6 (*continued*)

Relation R14: A *Class* «SP-AuditRegister» is created and associated with an *Access Control* specification (with an audit register).	 <<domain>> ar:AuditRegister requirementtype = n C R14 E <<domain>> c: Class name = nc where nc= { if (n="AC") then "SP_AuditRegister" endif;} R13(c,tmp:Class{='SecurityPermission'})
Relation R15: This is a "support" relation intended to create, for each stereotyped class in the source model, the corresponding properties in the classes created in the target model.	 <<domain>> souce:Property name = sName type=sType C R15 E <<domain>> target:Property name = sName type=sType

UML 2.0-AD specification. Section 4.2 shows the first set of transformations that are made from CIM to PIM which obtain class diagrams from the SBP specification and, finally, this section presents the other group of CIM to PIM transformations that are designed to obtain use cases from the specification of an SBP.

4.1. C2C: from BPMN–BPD to UML 2.0-AD

The first stage for any system under development should be the definition of the business logic that will guide the execution of the system. The definition of the business logic is the most important stage in the process since the business processes establish how the organization works. Thus, the system under development constitutes how this logic is implemented or carried out. According to the MDA abstraction level stack of models, CIMs are the most suitable representation for the logic of an organization since the business process can be considered as the highest representation level.

According to the framework presented here (see Fig. 5), domain experts can specify the systems in two different (but equivalent from the business view point and similar in technical aspects) notations: BPMN–BPD or UML 2.0-AD. The BPMN–BPD and the UML 2.0-AD metamodels allow experts to specify the system at a CIM level.

As Fig. 5 shows, the UML 2.0-AD representation is obtained through the C2C transformation, presented in this section. The C2C transformation takes a BPMN–BPD model (as a source model) and automatically produces a UML 2.0-AD model. The C2C transformation carries out two main tasks: firstly, the structural classes of the UML 2.0-AD model are created (that is, a basic activity diagram); secondly, the transformation includes all the security requirements considered in the BPMN–BPD model in the UML 2.0-AD model. Following this idea, domain experts put the logic of the system into the software development process without taking either technological or computational aspects into account. It is important to note that this is the first transformation executed in the application of the framework presented here.

In particular, Activity Diagrams are considered to represent the SBP since their expressiveness is comparable to that of BPMN–BPD. Thus, also considering the UML 2.0-AD metamodel, the framework presented (i) allows domain experts to describe systems using BPMN–BPD (considering security constraints) at the CIM level and (ii) represents these systems using a UML 2.0. As Fig. 5 shows, the UML 2.0-AD representation is obtained through the C2C transformation, presented in this section. The C2C transformation takes a BPMN–BPD model (as a source model) and automatically produces a UML 2.0-AD model. The C2C transformation carries out two main tasks: firstly, the structural classes of the UML 2.0-AD model are created (that is, a basic activity diagram); secondly, the transformation includes all the security requirements considered in the BPMN–BPD model in the UML 2.0-AD model. Following this idea, domain experts put the logic of the system into the software development process without taking either technological or computational aspects into account. It is important to note that this is the first transformation executed in the application of the framework presented herein.

Fig. 6 shows an integrated view of the C2C QVT transformation (in literature, other authors address the same problem by using other transformation mechanisms, such as XSLT [38]). The left side of the figure shows the elements of the BPMN–BPD metamodel (the source model) involved in the transformation, while the right side represents the elements of the UML 2.0-AD metamodel (target model). The 'equivalence relationship' between both metamodels has been implemented through several *relations,* and each *relation* establishes how an element from the source model is transformed into one (or several) element(s) from the target metamodel. It is thus possible to see in Fig. 6 how, according to *relation* **R2**, each *Lane* element is transformed into an *ActivityPartition* element, and it is also possible to see that the *MessageFlow* element is transformed into the *ObjectFlow* element through the **R5** *relation.* Table 2 has been included in order to provide a more advanced view of the transformation. This table shows the graphical representation for the first relation of the transformation (as an example), along with a textual explanation for all the relation of the relations. The QVT graphical representation in the first row shows the 1:1 relation in which *Pool* element (from the source model) is transformed into an *ActivityPartition* element (in the target model).

In fact, the QVT transformation is a formalization of an existing algorithm which sets up the correspondences between the elements of the source (BPMN–BPD) and the target (UML 2.0-AD) models. In addition to the source and target elements, the rules expressed in the *relations* of the QVT transformation establish the order in which the aforementioned elements of the source model are transformed into the elements of the target model.

In addition to the previous representations of the transformation, the code of the whole transformation has been included in Table 3. This table shows all the relations included in Table 2.

4.2. C2P-1: from secure business process to analysis classes

The UML 2.0-AD model can be used to obtain representations nearer to the system (that is, the PIM). In this step towards the PIM it is necessary to take two kinds of information into account:

Table 7
BPSec2ClassDiagram transformation.

```
transformation BPSec2ClassDiagram{
key Class {name},
key Property {name},
tmpProperty:Property,
top relation R6{
  checkonly domain BPSec sr:SecurityRequirement {
     requirementtype=n,
     ownedAttribute = SOURCE_SET}
  enforce domain uml_ClassDiagram c:Class {
     name=n,
     ownedAttribute = TARGET_SET}
   where{
     SOURCE_SET->forAll(p:Property |
       R15(p, tmpProperty),
       tmpProperty.class = c,
       TARGET_SET->incluging(tmpProperty))
     }}
top relation R7{
  checkonly domain BPSec sr:SecurityRequirement {
     requirementtype=n,
     ownedAttribute = SOURCE_SET}
  enforce domain uml_ClassDiagram c:Class {
     name='SecurityRole',
     ownedAttribute = TARGET_SET}
   where{
     SOURCE_SET->forAll(p:Property |
       R15(p, tmpProperty),
       tmpProperty.class = c,
       TARGET_SET->incluging(tmpProperty))
     }}
top relation R8{
  checkonly domain BPSec sr:SecurityRequirement {
     requirementtype=n,
     ownedAttribute = SOURCE_SET}
  enforce domain uml_ClassDiagram c:Class {
     name=nc,
     ownedAttribute = TARGET_SET}
   where {
     nc= if (n="I" or n="AD") then "G_AuditRegister" endif,
     R13(c,tmp:Class{name='SecurityRole'}),
     SOURCE_SET->forAll(p:Property |
       R15(p, tmpProperty),
       tmpProperty.class = c,
       TARGET_SET->incluging(tmpProperty))
     }}
top relation R9{
  checkonly domain BPSec ar:AuditRegister {
     requirementtype=n,
     ownedAttribute = SOURCE_SET }
  enforce domain uml_ClassDiagram c:Class {
     name=nc,
     ownedAttribute = TARGET_SET }
    where {
      nc= if (n="AC") then "G_AuditRegister" endif,
      R13(c,tmp:Class{name='SecurityRole'}),
     SOURCE_SET->forAll(p:Property |
       R15(p, tmpProperty),
       tmpProperty.class = c,
       TARGET_SET->incluging(tmpProperty))
     }}
top relation R10{
  checkonly domain BPSec ar:AuditRegister {
     requirementtype=n,
     ownedAttribute = SOURCE_SET }
  enforce domain uml_ClassDiagram c:Class {
     name=nc,
    ownedAttribute = TARGET_SET }
    where {
      nc= if (n="P") then "G_AuditRegister" endif,
      SOURCE_SET->forAll(p:Property |
       R15(p, tmpProperty),
       tmpProperty.class = c,
       TARGET_SET->incluging(tmpProperty))
     }}
top relation R11{
  checkonly domain BPSec ar:AuditRegister {
     requirementtype=n,
     ownedAttribute = SOURCE_SET }
  enforce domain uml_ClassDiagram c:Class {
     name=nc,
     ownedAttribute = TARGET_SET }
  where {
     nc= if (n="NR") then "NR_AuditRegister" endif,
      SOURCE_SET->forAll(p:Property |
       R15(p, tmpProperty),
       tmpProperty.class = c,
       TARGET_SET->incluging(tmpProperty))
     }}
top relation R12{
  checkonly domain BPSec ar:AuditRegister {
     requirementtype=n,
     ownedAttribute = SOURCE_SET }
  enforce domain uml_ClassDiagram c:Class {
     name=nc,
     ownedAttribute = TARGET_SET }
   where {
     nc= if (n="AC") then "SecurityPermission" endif,
       R13(c,tmp:Class{name='SecurityRole'}),
       R14(),
      SOURCE_SET->forAll(p:Property |
       R15(p, tmpProperty),
       tmpProperty.class = c,
       TARGET_SET->incluging(tmpProperty))}
    }}
relation R13{
  checkonly domain uml_ClassDiagram leftClass:Class
{name=lcName}
  checkonly domain uml_ClassDiagram
rightClass:Class{name=rcName}
  enforce domain uml_ClassDiagram
assoc:Association{name=lcName+'_'+rc,
    memberEnd = MEND: OrderedSet (Property),
    ownedEnd = OEND: OrderedSet (Property),
    navigableOwnedEnd = NOEND: OrderedSet (Property)}
       where{   leftClassProp:Property {name='m'+lcName,
type=lcName};
                rightClassProp:Property {name='m'+rcName,
type=rcName};
                MEND = MEND->including (leftClassProp);
                MEND = MEND->including (rigthClassProp);
                leftClass.ownedAttribute-
>including(rightClassProp);
                rightClass.ownedAttribute-
>including(leftClassProp);
     }}
relation R14{
  checkonly domain BPSec ar:AuditRegister
{requirementtype=n}
  enforce domain uml_ClassDiagram c:Class {name=nc}
    where { nc= if (n="AC") then "SP_AuditRegister" endif;}
    R13(c,tmp:Class{name='SecurityPermission'})
   }}
Relation R15{
   checkonly domain BPSec source:Property{
     name = sName,
     type = sType}
   enforce domain uml_ClassDiagram target:Property{
     name = sName,
     type = sType}}
```

the structure that represents the activity diagram depicted by the UML 2.0-AD model, and the security information, expressed in the BPSec profile applied to the UML 2.0-AD model. Thus, the UML 2.0-AD constitutes the representation of the BP and the *BPSec* model represents the security requirements to be considered in the BP.

According to Fig. 7, in order to achieve a secure analysis class diagram (hereafter UML 2.0-CD) the C2P-1 transformation is divided into two steps, each of which is implemented in turn by a different transformation. As Fig. 7 shows, the two steps are represented by the following transformations:

(1) ActivityDiagram2ClassDiagram: this transformation is first executed by taking the UML 2.0-AD model as the source model and generating a first version of the UML 2.0-CD (that is, the PIM). This transformation picks up only the structural information from the UML 2.0-AD model.
(2) BPSec2ClassDiagram: This transformation (i) takes the UML 2.0-CD and the *BPSec* models obtained (representing the security requirements) as the source model, and (ii) updates the UML 2.0-CD. In this transformation the inputs are the UML 2.0-CD model and *BPSec* model. Here the UML 2.0-CD

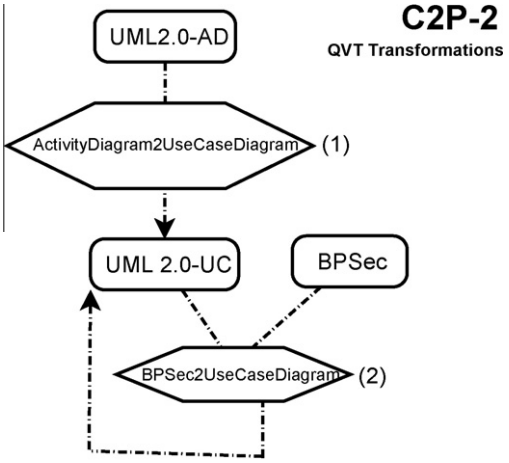


Fig. 10. C2P-2: QVT transformations.

model is also considered to be the output, and the transformation therefore updates the input UML 2.0-CD model with the information from the *BPSec* model (note that in Fig. 7 the arrow that goes from the BPSec2ClassDiagram transformation to the UML 2.0-CD model is annotated with the stereotype "«*Updates*»"). In spite of the fact that the *BPSec* model is shown as a separate model in Fig. 7, it is part of the UML 2.0-AD model. The *BPSec profile* has been drawn as an independent model to improve the understandability of the execution of the C2P-1 transformation.

The following sub-sections will describe each of the transformations of which the C2P-1 global transformation is composed.

4.2.1. From the activity diagram to the class diagram

From the UML 2.0-AD it is possible to obtain more detailed artefacts, such as the analysis class diagram. The class diagram is now considered to be a PIM model, because the system is described without any details of how the system uses the future deployment platforms [5]. The transformation in charge of producing the UML 2.0-CD is called ActivityDiagram2ClassDiagram (see Fig. 7, step 1). Thus, this transformation takes the UML 2.0-AD model as input and produces an analysis UML 2.0-CD model as output. This output model conforms to the UML 2.0-CD metamodel [47].

The ActivityDiagram2ClassDiagram transformation has been outlined in Fig. 8. This transformation is divided into 5 relations. As occurs in other transformations, each relation is in charge of picking up certain elements from the UML 2.0-AD models and transforming them into other elements in the UML 2.0-CD model. The left side of Fig. 8 contains the classes of the UML 2.0-AD metamodel affected by the transformation, and the right side represents those elements from the UML 2.0-CD metamodel involved in the transformation. It is possible to see that the UML 2.0-CD metamodel contains many meta-classes which are not involved in the transformation, but elements such as *Constraint*, *ValueSpecification*, *Interface* and *Parameter* are not yet necessary. These unused elements could, incidentally, be considered in future versions of the transformation to consider other kinds of security issues.

As Fig. 8 shows, there are five relations which specify the algorithm to obtain the analysis UML 2.0-CD model from the input UML 2.0-AD model. Among other things, Fig. 8 states that according to the QVT rule named **R5**, when an *ActivityPartition* is a composed partition, then it is transformed into a set of *Classes* which are associated by means of an *Association* class, and this relationship is implemented by a set of *Properties* of the type of the involved *Classes* (following the semantic of the UML 2.0-CD metamodel established by the standard [50]). A more advanced view of this relation can be observed in Table 4 (which includes the complete QVT code for the implementation of the ActivityDiagram2ClassDiagram transformation) and in Table 5 (in textual mode), where all the elements affected by the relation are shown, along with the *association* established among them. Following the **R4** relation, each *Action* (which always belongs to an *ActivityPartition*) from the UML 2.0-AD is transformed into an *Operation* which belongs to an existing *Class*. These existing *Classes* have been previously created by the **R1** relation which creates a *Class* in the target model for each *ActivityPartition* in the source model.

4.2.2. BPSec to analysis class diagram

After the execution of the ActivityDiagram2ClassDiagram transformation (see Fig. 8), a first version of the UML 2.0-CD analysis is obtained from the UML 2.0-AD representation of the BP. This representation does not contain any security issues and only regards the structural aspects of the system. According to Fig. 7, the second step (that is, the second transformation) of C2P-1 is called BPSec2ClassDiagram and accepts two models as input parameters: (i) the *BPSec* model (in which all the security issues are represented, see Fig. 3), and (ii) the *UML 2.0-CD* model (obtained in the ActivityDiagram2ClassDiagram transformation previously executed in step (1) and depicted in Fig. 7).

The output of the BPSec2ClassDiagram transformation is also one of the input models, particularly the *UML 2.0-CD* model. In the context of QVT, when a model plays the roles of input and output models in a transformation, the semantic of the transformation is not one of 'creation', but rather of 'updating'. Thus, the meaning of the BPSec2ClassDiagram transformation is that the *UML 2.0-CD* model is updated with the information contained in the *BPSec* model.

Fig. 9 summarizes the BPSec2ClassDiagram transformation, showing which meta-classes from the BPSec metamodel (input

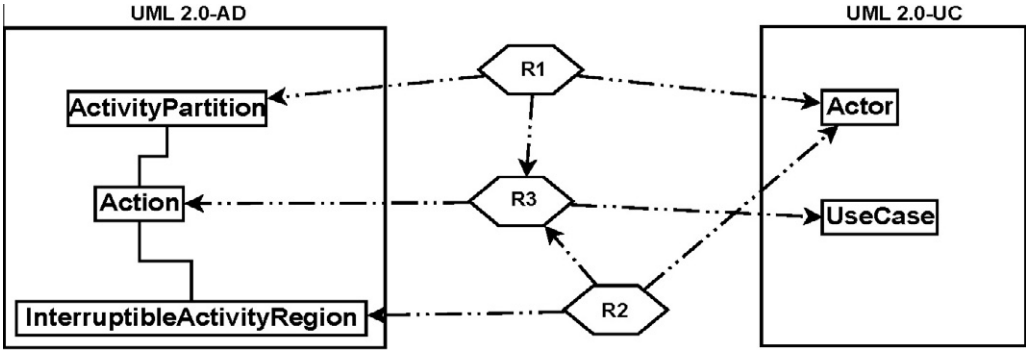


Fig. 11. C2P-2: ActivityDiagram 2UseCaseDiagram.

Table 8
Detailed view of the transformation ActivityDiagram 2UseCaseDiagram.

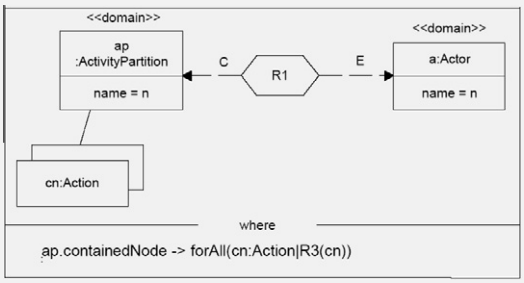
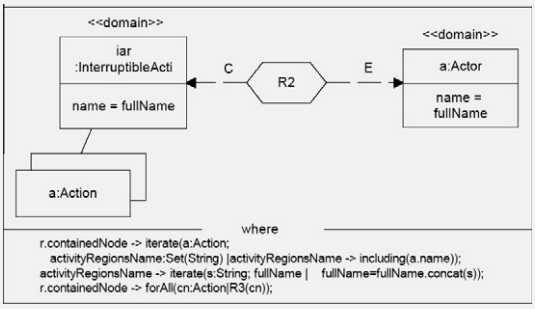
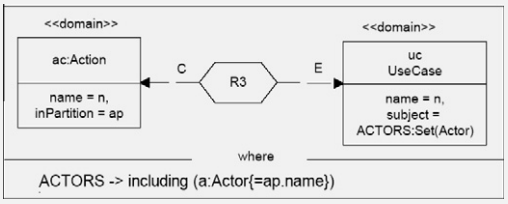
Top Relation R1: Relation in charge of transforming a *Partition* into an *Actor* with the same name. Additionally, this relation establishes the relationships between the *Actor* and the *Use Cases* obtained in relation R3.	
Top Relation R2: A *Region* is transformed into an *Actor*. Additionally, *Partitions* containing the aforementioned *Region* are verified in order to assign a meaningful name both to the *Region* and to the obtained *Actor*.	
Top relation R3: Assigns the *Actions* to the *Use Cases*. Additionally, this relation establishes the relationships between *Actors* and *Use Cases*.	

Table 9
ActivityDiagram 2UseCaseDiagram transformation.

```
transformation ActivityDiagram 2UseCaseDiagram
top relation R1{
  checkonly domain uml_ActivityDiagram ap:ActivityPartition {name=n}
  enforce domain uml_UseCaseDiagram a:Actor{name = n}
    where { ap.containedNode -> forAll(cn:Action|R3(cn))}}
top relation R2{
  fullName:String;
  checkonlydomain uml_ActivityDiagram
   iar:InterruptibleActivityRegion{name=fullName}
  enforce domain uml_UseCaseDiagram a:Actor {name=fullName}
  where{ r.containedNode -> iterate(a:Action;
  activityRegionsName:Set(String) | activityRegionsName ->
   including(a.name));
  activityRegionsName -> iterate(s:String; fullName |
   fullName=fullName.concat(s));
  r.containedNode -> forAll(cn:Action|R3(cn))}}
relation R3{
  checkonly domain uml_ActivityDiagram ac:Action {name=n,
   inPartition=ap}
  enforce domain uml_UseCaseDiagram uc:UseCase {name=n,
   subject=ACTORS:Set(Actor)};
    where {ACTORS -> including (a:Actor{name=ap.name})}}
```

model) are transformed into elements from the UML 2.0-CD metamodel. As can be seen in Fig. 9, the output model is updated with additional classes representing security requirements. The *BPSec* input model includes the security requirements designed by the domain experts in the *SecurityRequirement* and *AuditRegister* classes. While some classes from the input UML 2.0-CD are improved with the security information contained in the *BPSec* model, other classes are created.

From this transformation it is important to note that only three meta-classes from the target metamodel are affected: *Class, Property* and *Association*. The *Class* metaclass is modified or added to represent security issues initially described in the *BPSec* model. The *Property* metaclass is modified or added in order to represent those properties from the *SecurityRequirement* and *AuditRegister* elements (from the soured model) that must be derived in the target model. The *Association* metaclass is created in the transformation to link the new classes to the classes of the input UML 2.0-CD model.

Table 6 provides a representation of each of the relations of which the BPSec2ClassDiagram transformation is composed. This two-column table shows a brief description of each relation in the left-hand column cells, along with the graphic representation of the relations in the right-hand column cells. Table 7 shows the full QVT code for the BPSec2ClassDiagram transformation.

4.3. C2P-2: from secure business process to use cases

When specifying the BP with a UML 2.0-AD model, it is possible to obtain or deduce a set of requirements for the software systems implementing the BP. Obviously, in the context of this work these requirements are specified by means of UML 2.0 *Use Cases* (hereafter UML 2.0-UC model). Since the requirements are obtained from a UML 2.0-AD model, the *Use Cases* obtained would need manual refinement to maximize the correspondence between the SBP and the system requirements expressed in the UML 2.0-UC.

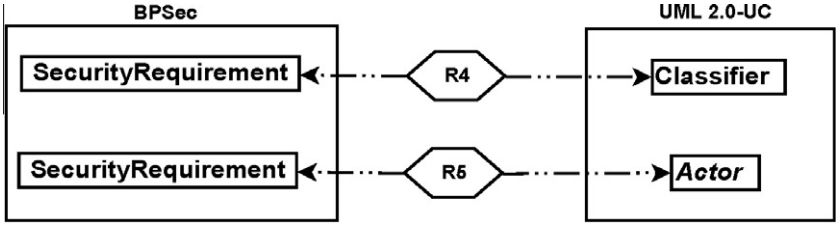


Fig. 12. BPSec2UseCaseDiagram.

Table 10
Detailed view of the BPSec2UseCaseDiagram transformation.

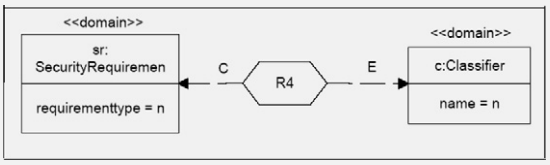
Top relation R4: Creates a *Classifier* with the name of the *Security Requirement* which has been specified.	
Top relation R5: Creates an *Actor* named *Security Staff* associated with each *Security Requirement* which has been specified.	

Table 11
BPSec2UseCaseDiagram transformation.

```
transformation BPSec2UseCaseDiagram
top relation R4{
  checkonly domain BPSec sr:SecurityRequirement {requirementtype = n}
  enforce domain uml_UseCaseDiagram c:UseCase {name=n}}
top relation R5{
  checkonly domain BPSec sr:SecurityRequirement
  enforce domain uml_UseCaseDiagram a:Actor {name="Security Staff"}}
```

In this approach, the C2P-2 transformation is composed of an automated and a manual step. The first step is in turn composed of two QVT transformations intended to generate a UML 2.0-UC model including security aspects. In a second step, this use case diagram is refined manually. Thus the whole C2P-2 transformation can be structured in the following manner:

- *Automatic Step:* The first version of the UML 2.0-UC model is obtained by applying two QVT transformations (see Fig. 10).
  - *ActivityDiagram2UseCaseDiagram transformation*: the UML 2.0-AD model is transformed into a UML 2.0-UC model.
  - *BPSec2UseCaseDiagram transformation:* the security requirements contained in the *BPSec* model are included in the UML 2.0-UC.
- *Manual step* (see Section 4.3.3.): Since the QVT transformations are not sufficient to obtain an accurate and secure UML 2.0-UC model; a manual refinement is required. Thus, for such a refinement the following mechanisms are used:

- *CheckLists*: Using these checklists, additional use cases are included to reflect all the security requirements.
- *Refinement Rules*: Refinement rules are applied when the UML 2.0-UC obtained from the QVT transformation and the checklist is ready. The refinement rules permit the following actions: assign significant names, identify actor hierarchy generalization detection, and delete redundancies.

Fig. 10 shows the two aforementioned transformations (ActivityDiagram 2UseCaseDiagram and BPSec2UseCaseDiagram) which deal with the models involved (UML 2.0-AD, UML 2.0-UC and *BPSec*). The following sub-sections will discuss the QVT transformations responsible for generating the UML 2.0-UC model.

4.3.1. Obtaining the UML 2.0-UC model from the UML 2.0-AD

The ActivityDiagram 2UseCaseDiagram QVT transformation takes the UML 2.0-AD model as input and, by applying an algorithm, produces an equivalent UML 2.0-UC model representing a tentative set of *Use Cases* for the system under development. Fig. 11 shows the elements of both the source (UML 2.0-AD, left side) and the target (UML 2.0-UC, right side) models. As can be seen in Fig. 11, the QVT relations in the middle of the figure relate the meta-classes from the source model to the meta-classes from the target model.

Relations **R1** and **R2** pick up, respectively, the *ActivityPartition* and *InterruptibleActivityRegion* classes from the source model and create *Actor* classes in the target model. On the other hand, the algorithm implemented in the QVT transformation establishes the equivalence between *Action* classes and *UseCase* classes, i.e., for each *Action* in the UML 2.0-AD model the relation **R3** creates a *UseCase* in the target model.

This first version of the UML 2.0-UC model is only based on the structure of the UML 2.0-AD model, and no security concern is included yet. Table 8 shows each relation and provides a basic explanation of them. By focusing on the relation **R2** (from Fig. 11) and the graphic explanation for this relation (in Table 8) it is possible to discover how the *InterruptibleActionRegion* is transformed into an *Actor*. Secondly, **R2** invokes the **R3** relation which in turn associates an *Actor* to a *UseCase*. In addition, Table 9 shows the full QVT code for this transformation. This code corresponds to the graphic representation depicted in Table 8.

4.3.2. Addition of security concerns to the UML 2.0-UC model

As Fig. 11 shows, the output of the ActivityDiagram 2UseCaseDiagram (i.e., the UML 2.0-UC model) is also the input for the

second transformation to produce a secure set of requirements for the system under development. As with the C2P-1 transformation, the C2P-2 QVT transformations produce the secure UML 2.0-UC model in two steps. The second step, in which the security issues are included in the target PIM model, is now carried out by the BPSec2UseCaseDiagram transformation.

As Fig. 12 shows, this transformation is not as complex as the previous ones, but is of no less importance. In this transformation, only two relations are considered. Both relations pick up the same element from the *BPSec* model: the *SecurityRequirement*. Each relation carries out a different interpretation of this metaclass. On the one hand, the **R4** relation creates a *UseCase* in the UML 2.0-UC model; on the other hand the **R5** relation considers that when at least a *SecurityRequirement* is present in the security configuration of the system, then a suitable actor intended to manage security aspects must be present in the requirement specification. The *Actor* class is therefore included in the UML 2.0-UC model. This actor will receive the name of '*Security Staff*'. Following the semantic of the execution of the QVT transformation the **R5** relation will be thrown once for each *SecurityRequirement* present in the source model (that is to say, the UML 2.0-AD model). Since no duplicate classes are desirable in a model, the QVT engine will not create any more instances for this *Actor* (named '*Security Staff*') after the first execution of the **R5** relation.

Tables 10 and 11 show the entire code of this transformation and the graph using the graphical syntax of QVT. As with the other transformations, an explanation of the semantic of each relation is included to improve the comprehension of the QVT code.

4.3.3. Checklists and refinement rules

In this section, we present two additional mechanisms which complement the QVT transformations illustrated in Fig. 10. The transformations previously presented in Sections 4.3.1 and 4.3.2 allow us to obtain certain use cases related to functional and non functional (security) requirements. However, our experience in applying our proposal to several cases has led us to enrich these

Table 12
Security Requirement Checklist specification.

Access Control
«Preconditions» Secure Role, and Permissions over the objects in the secure role scope
«Post-conditions» Secure role validated to access resources, Permissions over the validated objects, and Audit Register (optional)
- Assign secure role to the partition, region or action
- Validate the secure role (this task is complemented with misuse cases described in [20]). This task is divided into:
  - Identify the secure role. This involves recognizing roles before starting the interaction
  - Authenticate the secure role: This task involves the verification of the role identity before starting the interaction
  - Authorize the secure role. This involves assigning privileges to roles that were duly authenticated
- Verify permissions over the objects in the role secure field. This involves a review of the permissions granted to the objects that are within the field of Access Control specification
- If audit register has been specified, then the information related to the security role, the security permissions and the objects in the Access Control specification field must be stored

AttackHarmDetection
«Preconditions» Secure Role
«Post-conditions» Audit Register
- Assign secure role (origin and destination in the case of ObjectFlow).
- Register the type of element over which security requirements and the date and time when access to that element is produced are specified

Integrity
«Preconditions» Secure Role
«Post-conditions» Audit Register
- High-integrity specification involves:
  - Ask for permissions over data store
  - Verify permissions
  - Make security copies (backups)
  - Produce audit register
- Medium-integrity specification involves:
  - Send a warning message related to the data operation
  - Make security copies
  - Produce audit register
- Low-integrity specification involves:
  - Produce audit register

NonRepudiation
«Preconditions» Secure Roles (origin and destination)
«Post-conditions» Valid roles, and Audit Register (optional)
- Assign origin and destination roles
- Validate roles: This task is divided into:
  - Identify the secure role. This involves recognizing roles before starting the interaction
  - Authenticate secure role. This task involves verifying the role identity before starting the interaction
  - Authorize the secure role. This involves assigning privileges to the roles that were duly authenticated

Privacy
«Preconditions» Secure Role
«Post-conditions» Audit Register (optional)
- Assign a secure role (if anonymity was specified, then the role is generic and expires together with the session)
- Validate the role. This task is divided into:
  - Identify the secure role. This involves recognizing the role before starting the interaction
  - Authenticate the secure role. This task involves verifying the role identity before starting the interaction
  - Authorize the secure role. It involves assigning privileges to the role that was duly authenticated
- Verify revelation permissions (anonymity and confidentiality)
- Verify storage permissions (only anonymity)
- Verify audit register specification
- If audit register has been specified, then the information related to the security role must be stored

Table 13
Refinement Rules specification.

[RR1:] Obtain the name of the main subject. This name is the same as the business process name
[RR2:] Obtain the names of the subjects associated with the security requirement specifications. These names are constructed by linking the type of security requirement to the UML 2.0-AD element on which it was specified
[RR3:] Identify the main actor from the presence of a start node in a partition
[RR4:] Establish a generalization relation between the actors obtained from partitions that contain other partitions
[RR5:] Eliminate redundancy in the specifications. This refinement rule is mainly geared at preventing the duplication of specifications that could derive from the generalization and duplication of roles, since these roles can participate both in the general use case and in the security use cases

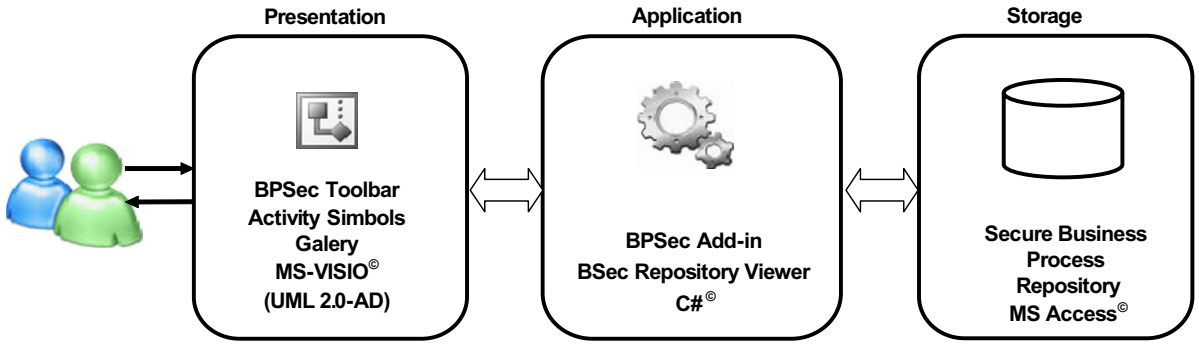


Fig. 13. The components of the BPSec-Tool.

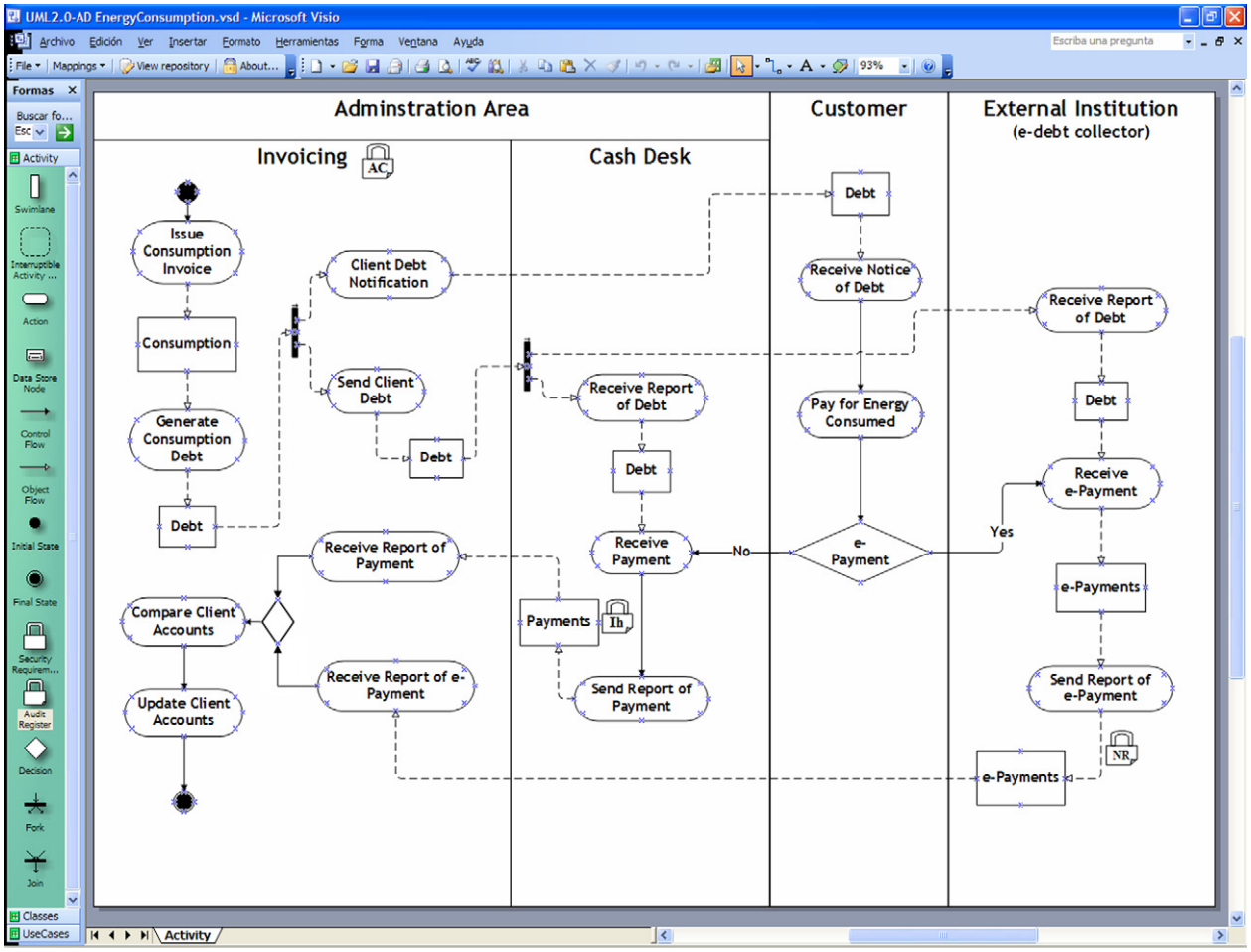


Fig. 14. Secure business process: Payment for Consumption of Electrical Energy.

automatic transformations with a manual step guided by a *Security Requirement Checklist* (which allows use cases to be obtained from specific security requirements) and a set of *Refinement Rules* (which complement the previous use cases with necessary elements that cannot be automatic derived). We have defined our checklist and refinement rules, and have improved and refined them through the characteristic iterative process of the action-research method [9,16]. Therefore, after executing the two transformations shown in Fig. 10, the engineer must first apply the *checklist* from Table 12. By using this *checklist* it is possible to complete the full definition of the Use Cases related to the security specification. This *checklist* is divided into five sections regarding the security requirements with which every SBP must comply (i.e., *Access Control, AttackHarmDetection, Integrity, NonRepudiation* and *Privacy*) (see Table 12).

According to Table 12, for each security requirement a set of pre and post-conditions is established. The pre and post-conditions are checked and when the model representing the Use Cases does not comply with these points, the model is modified in order to fulfil the corresponding security requirement. The activities with which to modify the Use Case model are detailed in this *checklist* and are defined by following the pre and post-conditions. An application of these activities is shown in Section 6.

Once the Use Cases have been generated from the specification of the SBP and the *checklist* has been checked, the engineer must apply the *Refinement Rules* (Table 13). These rules are intended to enrich the model produced by the QVT transformations in Fig. 10 and the *checklist*.

These rules make it possible to assign significant names to the main actors of the Use Case model, identify the main actors (which have not been identified in the process), identify generalization relationships among actors, and remove redundancies related to (1) these generalization relationships and (2) duplicated actors (those actors related to the security use cases). These refinement rules have been described in natural language (see Table 13).

5. Technology prototype implementation to support our proposal

This section presents the *BPSec-Tool*, a prototype that has been developed to support our framework. This prototype allows designers to specify SBPs very easily. Furthermore, the tool executes the aforementioned transformation in order to derive the different models involved in the development process with the security requirements.

Section 4 provides a detailed explanation of the different transformations involved in the architecture. The transformations, written in QVT, formally represent how the elements of the input and output models must be mapped. These transformations can be seen as an algorithm (where the initial or entry point is the *top relation*) that formally depicts all the process. These transformations have thus been considered as the basis for building the *BPSec-Tool*, which supports most of the characteristics of the architecture presented here.

The model transformations presented in this work form part of a broader work which establishes the way in which secure

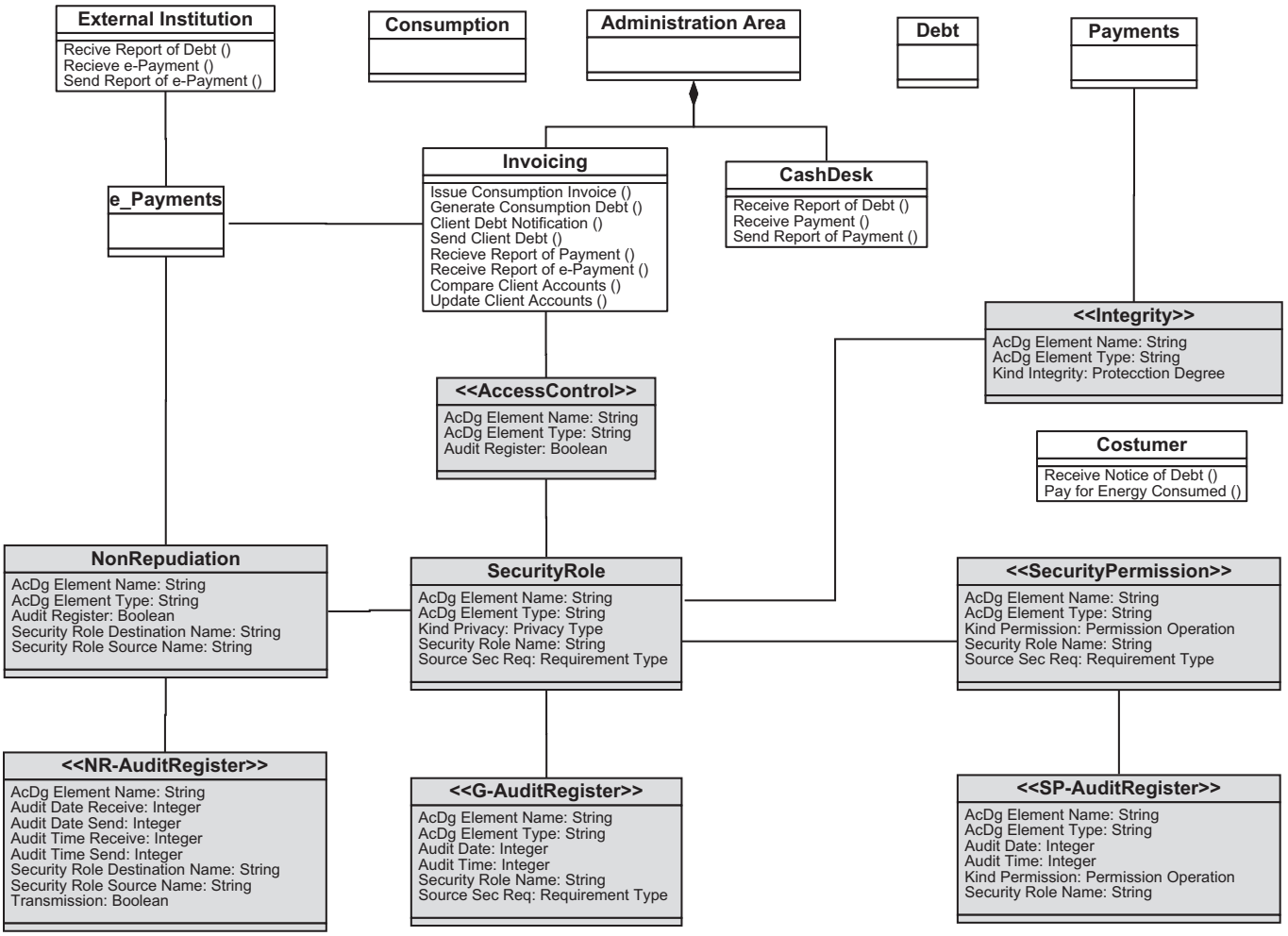


Fig. 15. Class Diagram from 'Payment for Consumption of Electrical Energy'.

business processes can be described, a method that makes it possible to do this task and to construct a prototype that can apply the complete proposal. The prototype, which we have named *BPSec-Tool*, consequently makes it possible to support the tasks related to the construction of a business process, to incorporate security requirements and, through semi-automatic transformations, to obtain analysis classes and use cases.

Three basic criteria were considered in the choice of the technological environment in which *BPSec-Tool* was developed.

- The need for a graphic representation of the business processes. Since this is a basic aspect in UML 2.0-AD and BPMN–BPD, the tool chosen to represent business processes had to provide a graphic environment that accepts templates that make it possible to represent business processes with UML and BPMN. The algorithms described by the transformations are the core of the architecture, but it is essential to offer a graphical representation of the models involved to both analysts and business experts in order to enable them to understand the problem.
- The capacity to adapt the chosen environment. This criterion is very important given that it must be possible to modify the environment to incorporate the graphic representation of the security requirements. Thus, the technological platform must facilitate both the representation of new representation elements and their manipulation.
- Availability of the software. This refers to the ease with which the software can be acquired and installed in the user's environment without entailing a significant financial outlay.

The *BPSec-Tool* prototype was consequently built using a 3-tiered architecture to separate the presentation, application, and storage components. The set of technologies selected to fulfil basic criteria identified are MS-Visio, C#, and MS-Access (see Fig. 13).

The *presentation layer* brings together all the aspects of the prototype relating to interfaces and interaction. These include windows, menus and graphs. The graphic aspect is basic since it interacts with UML and BPMN, thus conferring an extremely important role upon the graphic representation. An add-in was

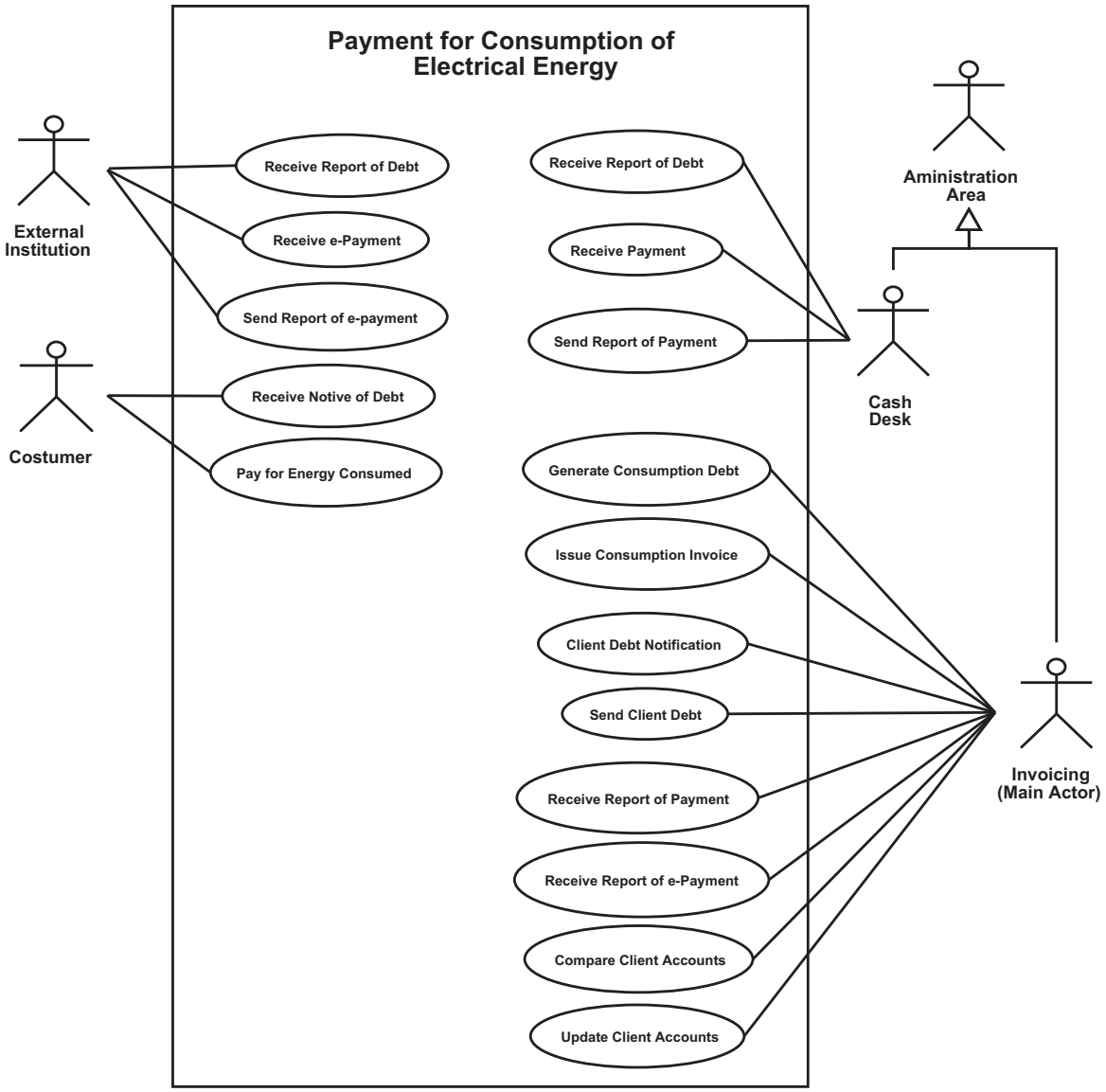


Fig. 16. General use case specification.

therefore used to adapt Microsoft Visio to include additional functions that support the new symbols in the representation of UML and BPMN. The models involved therefore have a graphical representation which eases their manipulation.

In spite of the fact that the *presentation layer* plays an essential role, the *application layer* implements the tasks that must be supported by the prototype:

- The construction of the business process; this is implemented by using a call to operation to standard Microsoft Visio. Business process models can be created from the beginning or imported if the models are available.
- The incorporation of security requirements; here it is necessary to run an add-in constructed over Microsoft Visio that can incorporate, modify and eliminate security requirements. The application makes it possible to choose the type of security requirement at the time that it is added, to define the type of priority associated with the requirement and to specify the associated permissions if an Access Control specification is involved.
- The generation of the repository of secure business processes; to do this, the information is obtained from the secure business process which makes it possible to fill the repository.

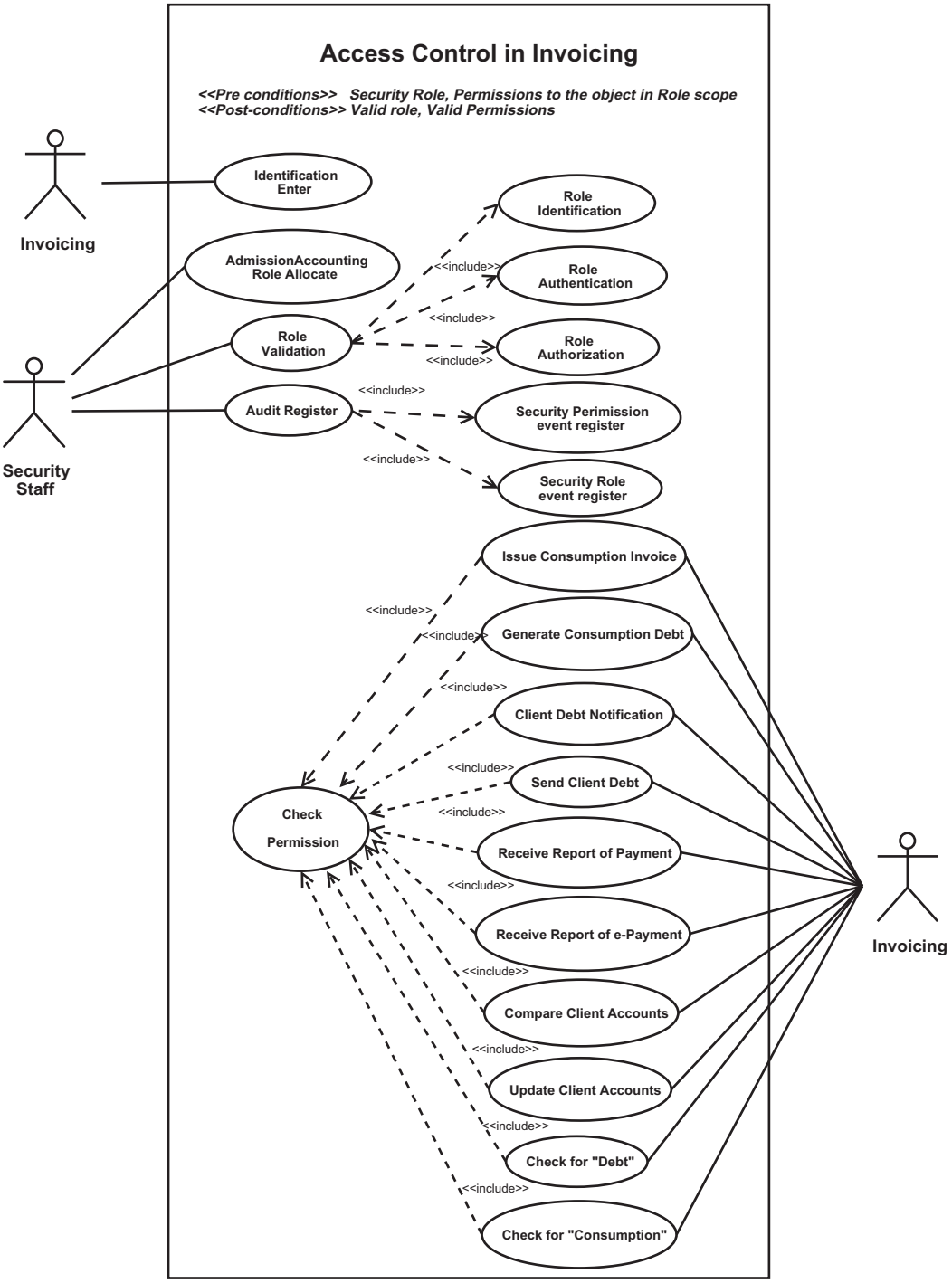


Fig. 17. Access Control use case specification.

– Transformation; this is the implementation of the transformation rules defined with QVT, checklists of refinement rules that were implemented in C#, the application which makes it possible to obtain analysis classes and use cases. All the QVT transformations are considered as algorithms and are thus implemented by using C#. The logic implemented in the application layer corresponds to the semantic of the transformations.

Finally, to manage the persistence of the application, a repository was designed, managed with Microsoft Access, which stores all the information related to secure business process, analysis classes and use cases.

BPSec-Tool does not directly execute the QVT transformations but, as was previously mentioned, it has the transformations coded in C#.NET. The main reason for implementing the transformations in C#.NET rather than executing them directly in an existing QVT engine is simple but important: QVT engines contain important bugs that make it difficult to apply the QVT standard to industrial purposes. The lack of suitable QVT engines is an important problem that has been addressed by other works, such as [38], in which an XSLT strategy has been adopted to carry out model transformations from BPMN models to UML 2.0-AD models. Some of the *plug-ins*/engines/tools that currently implement QVT have been mentioned in Section 2.2. However, these environments are (to date) only intended to write and execute transformations and the integration of these tools with the requited graphical support under the same framework has become quite difficult.

Nonetheless, since the presented transformations follow the QVT standard proposed by the OMG the *BPSec-Tool* will be ready to execute the QVT transformations as soon as a suitable QVT engine is available, which is expected to happen very soon, since QVT is the OMG standard for model transformations.

6. Application of our proposal

In order to illustrate the practical application of our approach, in this section we briefly show the result of an application example we have developed. We base the presentation of this application example on some of the criteria defined in [61], but not from an exhaustive point of view, since the goal of this section is to complement the rest of the paper, rather than to offer a detailed application example. The research methodology we have considered to develop our approach and the application example is action-research [9,16], signifying that we have been able to observe the application of our approach to a real case, and influence and change some elements of this approach (process, metamodels, transformations, etc.). Our purpose in using this research methodology has been (according to the classification defined in [52]) to improve or attempt to improve certain aspects of the development of secure information systems, which is the phenomenon we are studying. We have organized this section into three sub-sections, of which the first introduces the application example, the second applies the transformations defined in this paper, and the third presents the main conclusions of the application example. In order to report the application example, we use a chronological approach, in which we present the transformations at the moment at which our process suggests doing so.

6.1. Application example introduction

The application example was developed in a cooperative whose main business process is the distribution of electricity to rural areas. The Coopelan Ltda. (www.coopelan.cl) cooperative came into being in 1957, and currently maintains 2200 km of electrical lines which are used to supply more than 12,000 clients. Recent years have seen the commercialization of goods and services, both for their clients for electrical energy (who are associates of the cooperative) and for the public in general. From an organizational point of view, the cooperative is made up of a technical area which is related to the distribution of electricity, a commercial area which is in charge of goods and services, and an administrative area. The cooperative has a total of 70 employees.

Because the cooperative's main clients live in rural areas, the way in which it presently receives payment for the consumption of electricity creates two problems: (i) delivery of the invoice upon which the consumption of electrical energy is detailed and (ii) receipt of payment of said debt. Business analysts have used a traditional method to modify the business process associated with the recovery of energy consumption debts, and have incorporated an electronic debt advisor and electronic payment. This complementary method

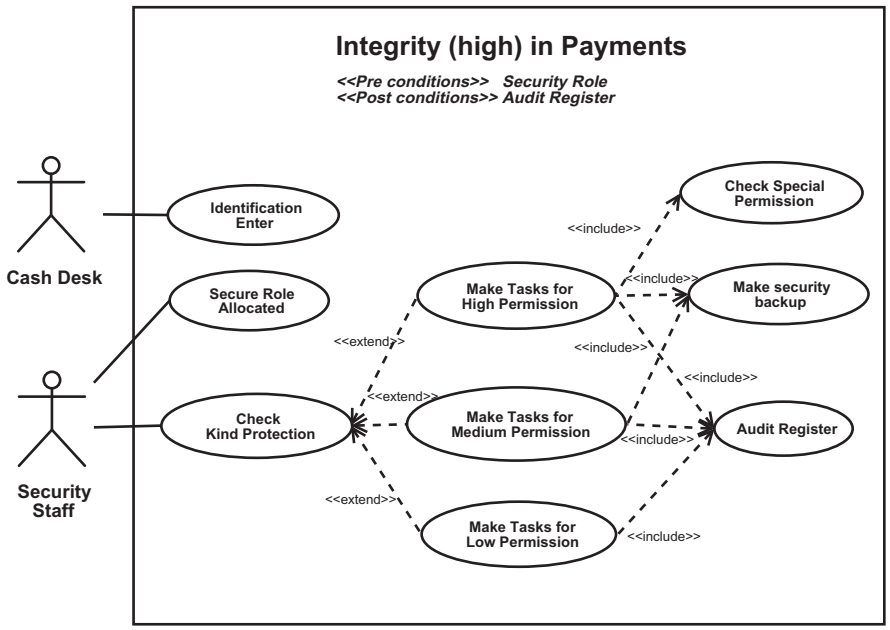


Fig. 18. Integrity in Payments use case specification.

has increased the index of debt recovery. The cooperative has neither the technical nor the operative capacity through which to receive electronic payments (via the Internet) and has therefore decided to employ an external collector to carry out this task.

The business process that we shall describe as a part of our application example concerns payment for the consumption of electrical energy. The application example was carried out with the assistance of the cooperative's business analyst, who is in charge of modelling the business models which are necessary as requirements to develop an information system that this process implements.

6.2. Application example development

We have used our M-BPSec method [56] in the development of this application example. This method makes it possible to elicit the security requirements which form part of a business process description carried out with a UML 2.0-AD or BPMN–BPD. In this case, the SBP was defined with UML 2.0-AD (for which reason C2C was not applied in this application example).

The applications of the M-BPSec process is composed of the following stages:

- The Construction stage, which basically consists of producing a business process, which is performed by the business analyst. In this case, the business process was described using the UML 2.0-AD. The areas identified were the Activity Partitions 'External Institution', 'Customer', and 'Administration Area' which was divided into two central Activity Partitions called 'Invoicing' and 'Cash Desk'. This business process is initiated when the 'Issue Consumption Invoicing' activity is carried out, and it ends with the receipt of payments and an update of the clients' debts.
- In the Security Requirement Incorporation stage, the business analyst identifies which, from his/her point of view, are the vulnerable areas in the business process. A meeting has previously taken place in which the significance of the security requirements considered in the BPSec-Profile are explained. The business analyst identifies vulnerable areas in: (i) the information which is sent from ' External collector' to 'Invoicing', for which NonRepudiation is specified, (ii) the information relating to the payments received in the 'Cash Desk', for which a high level of Integrity is specified, and (iii) the activities and information related to the Invoicing Activity Partition for which Access Control is specified
- The Refinement stage was carried out by the business analyst in conjunction with the security expert. These people analyzed and agreed upon the security requirement specifications and added Audit Register to the NonRepudiation and Access Control specifications.
- Finally, the Transformations stage was applied to the secure business process. This stage was carried out automatically using the BPSec-Tool. The results obtained were the analysis class diagram and a set of use cases related to a general use case for payments for the consumption of electrical energy, an Access Control in Invoicing, an Integrity in payments, and a NonRepudiation for message.

The secure business process seen in Fig. 14 was designed by a business analyst using the BPSec-Tool prototype and was refined by a security expert. The process represented by the Payment for Consumption of Electrical Energy at Coopelan Ltda, is the result of the application of the first three stages of the M-BPSec method. With these three stages, we obtain the representation of a secure business process through a detailed model, and it is ready to be used in order to obtain analysis artefacts which will help us in

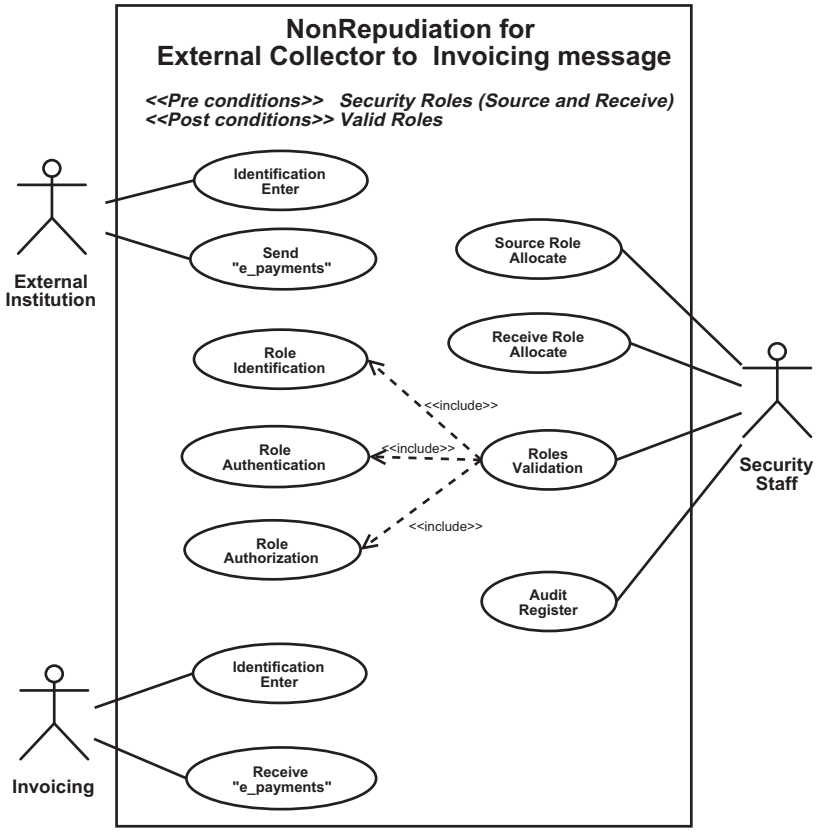


Fig. 19. Integrity and NonRepudiation security use case specification.

the development process in order to implement an information system which will automate the secure business process.

The application of the last stage of the M-BPSec method, Transformations, will be explained in detail below.

In the first place, the way in which the analysis classes were obtained is discovered. To do this, the transformations called C2P-1 (described in Section 4.2) are applied:

- By applying Top Relation R1, it is possible to obtain classes from the specifications of ActivityPartition. These classes are AdministrationArea, Invoicing, CashDesk, Customer, and ExternalIntitution.
- The operations identified in these classes are obtained by the application of Relation R4 (invoked from Top Relation R1). This rule makes it possible to transform the Actions included in an ActivityPartition in class operations. Thus the operations for the AdministrationArea class are:
  - IssueConsumptionInvoice, GenerateConsumptionDebt, ClientDebtNotification, SendClientDebt, ReceiveReportofPayment, ReceiveReportofe-Payments, CompareClientAccounts, UpdateClientAccounts (related to Invoicing partition), ReceiveReportofDebt, ReceivePayment, and SendReportofPayment (related to CashDesk partition).
- The operations identified for the Invoicing class are:
  - IssueConsumptionInvoice, GenerateConsumptionDebt, ClientDebtNotification, SendClientDebt, ReceiveReportofPayment, ReceiveReportofe-Payments, CompareClientAccounts, and UpdateClientAccounts.
- The operations for CashDesk are:
  - ReceiveReportofDebt, ReceivePayment, and SendReportofPayment.
- The operations identified for Customer are:
  - ReceiveNoticeofDebt, and PayforEnergyConsumed.
    - And finally, the operations for the ExternalInstitution class are:
  - ReceiveReportofDebt, Receivee-Payment, and SendReportfoe-Payment.

- The composition of the AdministrationArea class for the Invoicing and CashDesk classes is obtained through the application of Relation R5 invoked from Top Relation R1.
- The classes derived from the data stores are obtained by applying the R3 rule; these are Consumption, Debt, Payments, and e-Payments
- The security specifications contained in the secure business process give rise to security classes that are obtained by the application of Top Relation R6 up to Top Relation R12 and Relation R13 and Relation R14.
- Top Relation R6 gives rise to the AccessControl, Integrity, and NonRepudiation classes;
- Top Relation R7 makes it possible to create a class called SecurityRole;
- Top Relation R9 makes it possible to create the G_AuditRegister class which is associated with the SecurityRole class through Relation R13;
- Top Relation R11 makes it possible to create the NR_AuditRegister class, which corresponds to the audit register associated with the NonRepudation; and
- Finally, Top Relation R12 creates the SecurePermission class and associates it with the SecurityRole and SP_AudirRegister classes through Relation R13 and Relation R14, respectively.

The definition of the attributes of the «AccessControl», «Integrity», «NonRepudiation», «AuditRegister», «NR-AuditRegister», «SP-AuditRegister», «G-AuditRegister», «SecurityRole», «SecurityPermission» classes is carried out in the specification of the BPSec profile (see [54,57]). These attributes are shown in the security-related classes which were obtained from the secure business process model. A detailed diagram of classes that represents the complete transformation of the SBP is shown in Fig. 15.

To obtain use cases, transformation rules, checklists and refinement rules were applied to the secure business process under the designation C2P-2 (Section 4.3). The application of transformation rules is described below:

- Using Top Relation R1, an actor is obtained for each ActivityPartition. In this case, the actors are AdministrationArea, Invoicing, CashDesk, Customer, and ExternalInstitution.
- The use cases associated with the identified actors correspond to the Actions contained in each ActivityPartition. The Actions are transformed into use cases according to Relation R3 invoked from Top Relation R1.
- The *Invoicing* actor is therefore associated with the use cases IssueConsumptionInvoice, GenerateConsumptionDebt, ClientDebtNotification, SendClientDebt, ReceiveReportofPayment, ReceiveReportofe-Payments, CompareClientAccounts, and UpdateClientAccounts.
- The use cases associated with the *CashDesk* actor are ReceiveReportofDebt, ReceivePayment, and SendReportofPayment
- The use cases associated with the *ExternalIntitution* are ReceiveReportofDebt, Receivee-Payment, and SendReportfoe-Payment.

Fig. 16 shows the use cases related to the general business aspects in which security aspects have not been included.

In order to complete the use case shown in Fig. 16, it is necessary to apply the Refinement Rules. According to RR1, it is then possible to obtain a subject name from the business process name. In this case, the subject name is 'Payment for Consumption of Electrical Energy'. With RR3, it is possible to identify the main actor when the StartNode is present in ActivityPartition; in this case the StartNode is in Invoicing ActivityPartition and is therefore related to the Invoicing Actor. Finally, according to RR4, it is possible to obtain a generalization for the Invoicing and CashDesk actors from the AdministrationArea Actor.

The following figures show the use cases in relation to the security specification. These cases are obtained from the description of the checklists presented in Section 4.3.3. Fig. 17 shows the 'Access Control in Invoicing' use case (whose name is obtained according to RR2); the actors involved are Invoicing, obtained from the partition with the same name (see Top Relation R1), and SecurityStaff, obtained from the AccessControl Check List (see Table 12) which has been incorporated in order to carry out the tasks related to the specification of Access Control. The Invoicing actor is related to the use case that is responsible for handling role identification and SecurityStaff is related to the use cases related to accepting the role and validating it. In addition, the permissions associated with the role relating to the objects found in the sphere of Access Control specifications must be validated. Finally, the Security staff actor is related to the use case which must record all the operations performed by the role relating to Access Control for a subsequent control audit.

The Integrity requirement (high) that was specified in the SBP is transformed into the use case shown in Fig. 18. The CashDesk actor (see Top Relation R1) and the SecurityStaff have also been added, and were obtained from the Integrity Check List (see Table 12) which must assign a secure role to CashDesk. Additionally, this actor will have to validate the permissions classes that have been assigned.

Finally, Fig. 19 shows the subject related to the specification of nonrepudiation that is created over the message sent from e-debt

collector to Invoicing. The actors involved here are ExternalInstitution, Invoicing and SecurityStaff. The use cases related to this last actor have to do with assigning roles, validating them and recording events for later audits.

6.3. Application example conclusions

After applying our process and transformation to this application example, we obtained a set of analysis models containing rich information which was directly extracted from the business process, and which specified functional and security requirements. Therefore, both the secure business process and the analysis classes and use cases have been used as input in the software development process which Coopelan Ltda. used to create its software. Obviously these analysis models were not definitive and they needed to be detailed and completed, but the software developed was robust and secure because it integrated crucial requirements throughout all the stages of development.

7. Related works

In this paper we present the transformations of our specialization of the MDA architecture, which considers security in its models. In spite of the existence of other proposals that include security in MDA (such as Model-Driven Security applied in several contexts [8,14,25]), and which offer interesting contributions, our proposal considers the business dimension (CIM level), which is not usually considered by other relevant proposals. In this section we have therefore focused our review on those related works that consider business processes (using UML activity diagrams or BPMN) as a starting model.

In our review of related literature we found proposals with which to initiate the development from BPMN models [71,72], which express authorization and Access Control aspects, and one which uses QVT transformations in order to obtain models near to implementation [24]. In [71] the authors propose a BPMN extension which is oriented towards the representation of authorization constraints, and which evaluates the capabilities of BPMN to express task-based authorization constraints. In [72], the authors define a mapping between the BPMN and XACML metamodels, using the XSLT converter, in order to provide a model driven extraction of security policies from business process models (expressed in a graphical manner). And finally, in [24] the authors present an MOF based metamodel which defines a domain-specific language, a set of MOF–QVT standard rules to define model-to-model transformation. These proposals offer interesting ideas about transforming security concepts into implementation models. However they consider security as an element which is separated from the rest of the system, and they are focused on a very concrete type of security requirements.

We also found works which are related to our transformation to obtain analysis classes from business process. In the first [7], activity diagrams are transformed into analysis classes. This transformation is not performed automatically, and a previous version of UML 2.0 is used. In the second work [63], the software designer studies the business process model described with BPMN by extracting the UML classes which are later refined. The differences between these proposals and ours are firstly, that we use QVT for transformation specifications, and secondly, that we pay special attention to security requirements. Finally, we connect the result of the transformations with a software development process. We can also find works which are related to obtaining use cases from business process specifications. We discovered that [62] suggests the possibility of obtaining use cases from a business process specification made with BPMN, and [36] proposes the automatic attainment of UML artefacts from a BP description that was made using BPMN. The authors extend the BPMN (Extension Level-1) to add information about the sequence and the input and output flows. This allows them to apply rules from which use cases, state diagrams, sequences and collaboration are obtained. [66] discusses a transformation performed from a business process described with UML 2.0-AD to use cases, and finally, in [17], use cases are obtained from business process models that are not represented by activity diagrams. The differences between these proposals and ours are basically the following: (1) even in works in which there are automatic transformations, prior manual intervention is required, (2) transformations are not described using languages specially designed for this purpose (3) the result of transformations does not appear to be linked to a business process development, and finally (4) none of them is related to security aspects.

Moreover, other works such as [23] follow the MDA standard to follow the *Use Case Diagram* to the *Activity Diagram*. The authors propose a QVT-based approach that examines the different execution flows of the existing use cases to derive an *Activity Diagram*. In this work, the starting point in the development process is the functional requirements represented as use cases. The main difference with our work is the starting point, because we propose starting from the business process (represented with both BPMN and activity diagrams), and then deriving the use case diagram in order to initiate the development process. The most remarkable difference is that since we are starting with a representation of the business process enriched with security concerns, the use case diagram is also enriched with such information.

Despite the importance of all the works that we have studied, they do not show transformations from descriptions of secure business processes (in which we have considered both aspects of security and the business itself) to analysis classes and use cases.

8. Conclusions and ongoing work

Being conscious of the vital importance of incorporating security requirements into the software development process as soon as possible, we decided that business process models are those that will be able to include security requirements first. We therefore defined certain techniques to make it possible for business analysts to include security requirements in business process models, using the main notations existing today. Clearly, at this level of abstraction, the security specifications that can be included (by business experts, not security experts) are very abstract and, of course, lack technical nuances, which are obviously incorporated in system models as their development advances.

Since we observed that secure business process models contain information that may directly correspond with elements of more specific models, such as analysis class models and use case models, our goal became to define the necessary transformations that make it possible to construct these models semi-automatically from secure business process models. Once we had constructed these transformations and applied them to several examples and to one application example in a real company, we were able to observe that obtaining these models is very useful and that, although they must be completed and refined, they serve as a good starting point for the development of the software that a set of business processes has to implement. In approaching this research, we considered the higher part of MDA architecture, considering metamodels for both CIM and for PIM and we defined transformations with QVT to obtain elements from target models starting from secure business process models.

To improve the usefulness of the proposal and reduce scaling problems that transformations can cause in business process models with a high amount of content, we developed a prototype for a

tool that provides support both in the graphic modelling of secure business process models (forcing the restrictions defined in metamodels to be complied with) and in automatic transformations of specified QVT rules.

Future work will focus on broadening transformations in order to be able to offer technical solutions to the abstract security requirements defined in business processes, attempting to go even lower in MDA architecture, and to take advantage of this architecture to modernize inverse engineering and models from legacy systems, thus making it possible to carry out abstractions at the business level of applications that have already been constructed and of those that probably do not have access to documentation to extract this highly important information. We shall also analyze the security problems related to the organization of the business processes at different granularity levels, and we shall adapt our proposal to integrate the security implications of composition and decompositions of elements into business process models.

Acknowledgements

We would like to thank Mr. Eduardo Robba, business and systems analyst at Coopelan Ltda., for his valuable collaboration in the development of the application example. This research is part of the following projects: BUSINESS (PET 2008-0136) Project financed by the "Ministerio de Ciencia e Innovación" (Spain), QUASIMODO (PAC08-0157-0668), SISTEMA (PII2I09-0150-3135), and PRALIN (PAC08-0121-1374), both partially supported by the FEDER and the *Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha* (Spain), and MEDUSAS (IDI-20090557) supported by the *Centro para el Desarrollo Tecnológico Industrial. Ministerio de Ciencia e Innovación* (Spain).

References

[1] A. Agrawal, GReAT: a metamodel based model transformation language, in: 18th IEEE International Conference on Automated Software Engineering, Montreal, Canada, 2003.
[2] R.S. Aguilar-Savén, Business process modelling: review and framework, International Journal of Production Economics 90 (2) (2004) 129–149.
[3] D.H. Akehurst, B. Bordbar, M.J. Evans, W.G.J. Howells, K.D. McDonald-Maier, SiTra: simple transformations in Java, in: 9th International Conference on Model Driven Engineering Languages and Systems, Genova, Italy, 2006.
[4] F. Allilaire, T. Idrissi, ADT: eclipse development tools for ATL, in: 2nd European Workshop on MDA, 2004.
[5] G. Antoniol, M. Di penta, M. Zazzara, Understanding web applications through dynamic analysis, in: 12th IEEE International Workshop on Program Comprehension (IWPC), 2004.
[6] M. Backes, B. Pfitzmann, M. Waider, Security in business process engineering, in: International Conference on Business Process Management (BPM), Eindhoven, Netherlands, 2003.
[7] J.P. Barros, L. Gomes, From activity diagrams to class diagrams, in: Workshop Dynamic Behaviour in UML Models: Semantic Questions in Conjunction with Third International Conference on UML, York, UK, 2000.
[8] D. Basin, J. Doser, T. Lodderstedt, Model driven security: from UML models to access control infrastructures, ACM Transactions on Software Engineering and Methodology 15 (1) (2006) 39–91.
[9] R. Baskerville, T. Wood-Harper, A critical perspective on action research as a method for information systems research, Journal of Information Technology 11 (1996) 235–246.
[10] J. Bézivin, Search of a basic principle for model driven engineering, UPGRADE European Journal for the Informatics Professional V (2) (2004) 21–24.
[11] J. Bézivin, F. Jouault, P. Valduriez, An Eclipse-based IDE for the ATL Model Transformation Language, RN, 2005.
[12] BPMN, Business Process Modeling Notation Specification, OMG Final Adopted Specification, dtc/06-02-01, 2006.
[13] P. Braun, F. Marschall, The Bidirectional Object Oriented Transformation Language, RN, 2003.
[14] R. Breu, M. Hafner, F. Innerhofer-Oberperfler, F. Wozak, Model-driven security engineering of service oriented systems, in: Information Systems and e-Business Technologies, 2nd International United Information Systems Conference (UNISCON), 2008.
[15] K.-K.R. Choo, R.G. Smith, R. McCusker, and C.h.w.a.g.a.p.r., Australian Institute of Criminology, future directions in technology-enabled crime: 2007–09, Research and Public Policy Series Edited by Australian Institute of Criminology, vol. 78, 2007.
[16] R.M. Davison, M.G. Martinsons, N. Kock, Principles of canonical action research, Information Systems Journal 14 (2004) 65–86.
[17] R.M. Dijkman, S.M.M. Joosten, An algorithm to derive use cases from business processes, in: 6th International Conference on Software Engineering and Applications (SEA), Boston, USA, 2002.
[18] T. Dufresne, J. Martin, Process Modeling for e-Business, RN, 2003.
[19] J.R. Falleri, M. Huchard, C. Nebut, Towards a traceability framework for model transformations in kermeta, in: European Conference on Model-Driven Architecture Traceability Workshop (ECMDA-TW), Bilbao, Spain, 2006.
[20] D. Firesmith, Security use case, Journal of Object Technology 2 (3) (2003) 53–64.
[21] D. Firesmith, Specifying reusable security requirements, Journal of Object Technology 3 (1) (2004) 61–75.
[22] G.M. Giaglis, A taxonomy of business process modelling and information systems modelling techniques, International Journal of Flexible Manufacturing Systems 13 (2) (2001) 209–228.
[23] J.J. Gutiérrez, C. Nebut, M.J. Escalona, M. Mejías, I.M. Ramos, Visualization of use cases through automatically generated activity diagrams, in: 11th international conference on Model Driven Engineering Languages and Systems, 2008.
[24] M. Hafner, M. Alam, R. Breu, Towards a MOF/QVT-based domain architecture for model driven security, in: Model Driven Engineering Languages and Systems, 2006.
[25] M. Hafner, R. Breu, Security Engineering for Service-Oriented Architectures, Springer, 2009.
[26] P. Harmon, The OMG's model driven architecture and BPM, Business Process Trends 2 (5) (2004).
[27] G. Herrmann, G. Pernul, Viewing business process security from different perspectives, in: 11th International Bled Electronic Commerce Conference, Slovenia, 1998.
[28] P. Herrmann, G. Herrmann, Security requirement analysis of business processes, Electronic Commerce Research 6 (3–4) (2006) 305–335.
[29] F. Jouault, I. Kurtev, On the architectural alignment of ATL and QVT, in: ACM Symposium on Applied Computing – Model Transformation, Dijon, France, 2006.
[30] F. Jouault, I. Kurtev, Transforming models with ATL, in: International Workshop on Model Transformations in Practice (MTiP), 2005.
[31] S.F. King, O.A. Johnson, V. BP, An approach to modelling process variety and best practice, Information and Software Technology 48 (11) (2006) 1104–1114.
[32] A. Kleppe, J. Warmer, W. Bast, MDA Explained: The Model Driven Architecture™: Practice and Promise, Addison Wesley, 2003. p. 192.
[33] P. Kobiakov, MDA and QVT in Together Architect 2006, 2005.
[34] I. Kurtev, State of the art of QVT: a model transformation language standard, in: Applications of Graph Transformations with Industrial Relevance, Third International Symposium (AGTIVE), Kassel, Germany, 2007.
[35] M.B. Kuznetsov, UML model transformation and its application to MDA technology, Programming and Computer Software 33 (1) (2007) 44–53.
[36] P. Liew, P. Kontogiannis, T. Tong, A framework for business model driven development, in: 12 International Workshop on Software Technology and Engineering Practice (STEP), 2004.
[37] B. List, B. Korherr, A UML 2 profile for business process modelling, in: 1st International Workshop on Best Practices of UML (BP-UML) at ER-Conference, Klagenfurt, Austria, 2005.
[38] O. Macek, K. Richta, The BPM to UML activity diagram transformation using XSLT, in: Databases, Texts, Specifications, and Objects (DATESO 2009), 2009.
[39] A. Maña, J.A. Montenegro, C. Rudolph, J.L. Vivas, A business process-driven approach to security engineering, in: 14th. International Workshop on Database and Expert Systems Applications (DEXA), Prague, Czech Republic, 2003.
[40] S.J. Mellor, K. Scott, A. Uhl, D. Weise, MDA Distilled: Principles of Model-Driven Architecture, A. Wesley, 2004. p. 176.
[41] T. Mens, P. Van Gorp, A taxonomy of model transformation, Electronic Notes in Theoretical Computer Science 152 (2006) 125–142.
[42] Object Management Group, Architecture-driven Modernization (ADM): Knowledge Discovery Metamodel (KDM) Specification, Object Management Group, 2006.
[43] Object Management Group, Common Warehouse Metamodel (CWM) Specification, 2003.
[44] Object Management Group, MDA Guide Version 1.0.1, 2003.
[45] Object Management Group, Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, OMG Adopted Specification ptc/05-11-01, 2005, p. 204.
[46] Object Management Group, Unified Modeling Language: Superstructure, Version 2.0, formal/05-07-04, 2005.
[47] Object Management Group, Unified Modeling Language: Superstructure Version 2.1.1 (formal/2007-02-05), 2007.
[48] OMG, Meta Object Facility (MOF) Specification, 2002.
[49] OMG, OCL 2.0 Specification, Version 2.0., RN, 2005
[50] OMG, Unified Modeling Language: Superstructure, Versión 2.0, 2005.
[51] G. Rader, C. Vo, Achieving consistency between business process models and operational guides, in: Rational Software. Enterprise Business Process Documentation (White Paper), 2008.
[52] C. Robson, Real World Research: A Resource for Social Scientists and Practitioner–Researchers, second ed., Blackwell, Oxford, UK, 2002.

[53] A. Rodríguez, E. Fernández-Medina, M. Piattini, Analysis-level classes from secure business processes through models transformations, in: 4th International Conference on Trust, Privacy and Security in Digital Business (TrustBus), Regensburg, Germany, 2007.
[54] A. Rodríguez, E. Fernández-Medina, M. Piattini, A BPMN extension for the modeling of security requirements in business processes, IEICE Transactions on Information and Systems E90-D (4) (2007) 745–752.
[55] A. Rodríguez, E. Fernández-Medina, M. Piattini, Capturing security requirements in business processes through a UML 2.0 activity diagrams profile, in: 2° International Workshop on Best Practices of UML (BP-UML), 2006.
[56] A. Rodríguez, E. Fernández-Medina, M. Piattini, M-BPSec: a method for security requirement elicitation from a UML 2.0 business process specification, in: 3rd International Workshop on Foundations and Practices of UML, Auckland, New Zealand, 2007.
[57] A. Rodríguez, E. Fernández-Medina, M. Piattini. Towards a UML 2.0 extension for the modeling of security requirements in business processes, in: 3rd International Conference on Trust, Privacy and Security in Digital Business (TrustBus), Krakow-Poland, 2006.
[58] A. Rodríguez, E. Fernández-Medina, M. Piattini. Towards CIM to PIM transformation: from Secure Business Processes defined by BPMN to Use Cases. in: 5th International Conference on Business Process Management (BPM), Brisbane, Australia, 2007.
[59] A.W. Röhm, G. Herrmann, G. Pernul, A language for modelling secure business transactions, in: 15th. Annual Computer Security Applications Conference, Phoenix, Arizona, 1999.
[60] A.W. Röhm, G. Pernul, G. Herrmann, Modelling secure and fair electronic commerce, in: 14th Annual Computer Security Applications Conference, Scottsdale, Arizona, 1998.
[61] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering 14 (2) (2009) 131–161.
[62] W. Rungworawut, T. Senivongse, A guideline to mapping business processes to UML class diagrams, WSEAS Transactions on Computers 4 (11) (2005) 1526–1533.
[63] W. Rungworawut, T. Senivongse, Using ontology search in the design of class diagram from business process model, Enformatika, Transactions on Engineering, Computing and Technology 12 (2006) 165–170.
[64] G. Sindre, Mal-activity diagrams for capturing attacks on business processes, in: Requirements Engineering: Foundation for Software Quality, 13th International Working Conference, REFSQ 2007, Trondheim, Norway, 2007.
[65] SOURCEFORGE, UMT-QVT, 2005.
[66] S. Štolfa, I. Vondrák, A description of business process modeling as a tool for definition of requirements specification, in: Systems Integration 12th Annual International Conference, Prague, Czech Republic, 2004.
[67] J.L. Vivas, J.A. Montenegro, J. Lopez, Towards a business process-driven framework for security engineering with the UML, in: Colin Boyd, Wenbo Mao (Eds.), Information Security: 6th International Conference, ISC, Bristol, UK, 2003.
[68] J.P. Walton, Developing a enterprise information security policy, in: Proceedings of the 30th Annual ACM SIGUCCS Conference on User Services, ACM Press, 2002.
[69] WfMC, Workflow Management Coalition: Terminology & Glossary, RN, 1999. <http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf>.
[70] S.A. White, Process Modeling Notations and Workflow Patterns, 2004, BPTrends, March 2004.
[71] C. Wolter, A. Schaad, modeling of task-based authorization constraints in BPMN, in: 5th International Conference on Business Process Management, 2007.
[72] C. Wolter, A. Schaad, C. Meinel, Deriving XACML policies from business process models, in: WISE Workshops, 2007.
[73] M. Zulkernine, S.I. Ahamed, Software security engineering: toward unifying software engineering and security engineering, in: M. Warkentin, R. Vaughn (Eds.), Enterprise Information Systems Assurance and Systems Security: Managerial and Technical Issues, Idea Group, 2006, pp. 215–232.