Decision Support Systems 51 (2011) 446–465

Contents lists available at ScienceDirect

Decision Support Systems

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

Secure business process model specification through a UML 2.0 activity diagram profile

Alfonso Rodríguez [a], Eduardo Fernández-Medina [b,*], Juan Trujillo [c], Mario Piattini [b]

[a] Computer Science and Information Technology Department, University of Bio-Bio. Casilla 447, Chillán, Chile
[b] GSyA Research Group, Information Systems and Technologies Department, University of Castilla-La Mancha. Paseo de la Universidad 4, 13071, Ciudad Real, Spain
[c] LUCENTIA Research Group, Department of Software and Computing Systems, University of Alicante, C/San Vicente S/N, 03690, Alicante, Spain

ARTICLE INFO

Article history:
Received 14 May 2009
Received in revised form 26 October 2010
Accepted 29 January 2011
Available online 5 February 2011

Keywords:
Business process
Security requirement
UML 2.0
Activity diagrams

ABSTRACT

Business processes have become important resources, both for an enterprise's performance and to enable it to maintain its competitiveness. The languages used for business process representation have, in recent years, been improved and new notations have appeared. However, despite the wide acceptance of the importance of business process security, to date the business analyst perspective in relation to security has hardly been dealt with. Moreover, security requirements cannot be represented in modern business process modeling notations.

In this paper, we present an extension of UML 2.0 activity diagrams which will allow security requirements to be specified in business processes. Our proposal, denominated as BPSec (Business Process Security), is Model Driven Architecture compliant since it is possible to obtain a set of UML artifacts (Platform Independent Model-PIM) used in software development from a Secure Business Process model specification (Computation Independent Model-CIM). We also present the application of our approach to an example based on a typical health care institution, in which our M-BPSec method is employed as a framework for the use of our UML extension.

1. Introduction

Business processes have, in recent years, become a fundamental resource through which to achieve and maintain competitive advantages in the market. Enterprises are consequently paying far more attention to the description of their business processes. A business process is defined as a combination of a set of activities within an enterprise with a structure which describes their logical order and dependence whose objective is to produce a desired result [1]. Enterprises create business process models with the purpose of obtaining a simplified view of reality [13]. This realistic description of a business process will allow us to understand and eventually modify a business process with the aim of incorporating improvements into it. This will give enterprises the capacity to adapt to the continuous changes of competitive environments. Since decision support is integrated into business processes and information systems [7], business process models are also used in "What if" scenario simulations in which the organizational behavior must be reflected in the simulation model [17].

Languages that allow us to represent business processes are also becoming increasingly more important owing to the fact that the success of modeling is based both on the ability to express the different needs of the business and on the availability of a notation in which these needs can be described.

For several years, and in accordance with the state of the business process modeling industry [36], it has been possible to identify the Unified Modeling Language (UML) [44] and the Business Process Modeling Notation (BPMN) [8] among the main standards for business process modeling. At present, both notations are accepted throughout the business community and are well-established in research and industry [33].

Security has simultaneously become a fundamental aspect, not only in an enterprise's performance but also in its relationship with its customers. According to Ref. [24] security in the modern business environment necessitates a complete understanding of the continuous events that comprise the way in which businesses organize their activities. Nevertheless, despite the wide acceptance of the importance of security for business processes, its modeling has not been appropriate. Security requirements are generally specified by requirement engineers who have accidentally tended to use architecture specific restrictions rather than security requirements [14]. Moreover, security has been integrated into applications in an ad-hoc manner, often during the actual implementation process [3] or during the system administration phase [35], and the identification of security requirements has been rather confusing owing to the fact that there has, in general, been a tendency to identify functional security requirements. This type of requirements varies according to the kind of application.

* Corresponding author. Tel.: +34 926295300; fax: +34 926295354.
E-mail addresses: alfonso@ubiobio.cl (A. Rodríguez), Eduardo.FdezMedina@uclm.es (E. Fernández-Medina), jtrujillo@dlsi.ua.es (J. Trujillo), Mario.Piattini@uclm.es (M. Piattini).

0167-9236/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2011.01.018

Security requirements, on the other hand, can be specified for every application at the highest level of abstraction and will tend to have the same basic kinds of valuable and potentially vulnerable assets [15].

There is currently an important paradigm shift (from objects to models) in the field of software engineering that may have important consequences in the way in which information systems are built and maintained [6]. This new paradigm is known as Model Driven Engineering, and one of the most standardized approaches is that of Model Driven Architecture (MDA) [42], a paradigm which claims to work at model and metamodel levels. Among the objectives pursued, we can find the separation of business-neutral descriptions and platform dependent implementations, the expression of specific aspects of a system under development with specialized domain-specific languages, the establishment of precise relations between these different languages within a global framework and, in particular, the capability to express operational transformations between them [5]. In this context, Business Process Models are used and defined by business analysts [19] and are considered to be computation independent models. These models, which are defined in a language that is understandable to business people, are a source of requirement for software construction [41], and it is also possible to determine the information processing requirements which are specified in the business processes [12]. Business process models can therefore be used as a starting point in a software development process.

It is clear that the tendency is towards the creation of models, using languages which allows as much expressivity as possible and which can be reused in software development under the model-driven approach. However, the new proposals for business process modeling (UML and BPMN) do not consider the representation of security requirements. This limits the resulting model since it is not possible to obtain the business analyst's viewpoint as regards security. Thus, security aspects cannot be taken into account together with the description of the business process itself. This is a problem if we consider that the early representation of any requirement favors its implementation, thus decreasing costs and time in the following phases of the system's development. Since our proposal is congruent with the MDA approach, it clearly considers the transformation of these specifications into UML artifacts which are oriented towards giving technological support to the performance of the business processes described. Therefore, not only UML but also BPMN proposals must be adapted if their expressivity in relation to security is to be improved. Both notations have mechanisms that permit their adaptation to new domains.

In this paper we present a complete extension of the UML 2.0 activity diagram (UML 2.0-AD) which allows security requirements to be specified in the business process domain. This paper is in the context of a research line which we have been developing for some years, and several papers related to this theme have already been published. For example, in Ref. [50], we presented a preliminary version of our UML extension, which has been improved, detailed and completed in this manuscript, offering the complete specification of stereotypes, tagged values, constraints, associations and new types. Other related publications are Ref. [49], which presents a method that defines a systematic approach for the design of secure business processes and their integration into a software development methodology, or Refs. [51,52] which offer QVT transformations that can be used to obtain analysis class diagrams and use case diagrams from secure business process models. There is a previous related proposal by Jürjens (UMLsec [28]), which is an extension to the UML and serves to integrate security related information into UML models through the specification of a UML Profile. This proposal considers the activity diagrams between their models, and therefore allows security information to be specified in these models, which represent work-flows and more a precise representation of use cases. However, it does not model the specific business view of security as we attempt to do with our approach. In fact, in our opinion, UMLsec and our proposal are perfectly compatible, since UMLsec represents security at the software analyst and designer view (PIM and PSM in MDA terminology), and our proposal represents security at the business analyst views (the CIM).

The structure of the remainder of the paper is as follows: Section 2 shows the main works related to security in business processes. Our general research line is presented in Sections 3, and 4 includes our UML 2.0 extension for a security requirements specification. An illustrative example related to a health care institution is provided in Section 5, and the results of the application of our proposal in a real environment are explained in Section 6. Finally, our conclusions are shown, in Section 7. The technical details of our proposal are included at the end of the paper in three appendices.

2. Related work

In this section, we present works related to security specifications in business processes. These works were selected by carrying out a literature review following the Kitchenham protocol [30], and this allowed us to evaluate and interpret an important set of works related to security specifications in business processes.

One of the works most directly related to our approach is UMLsec [28]. This proposal presents a UML profile which defines a set of stereotypes (e.g. fair exchange, rbac, secrecy, integrity, etc.), taking UML metaclasses (e.g. subsystem, link, dependency, etc.), with their tagged values and constraints, as a base in order to allow the software analyst and designer to express security-related information within the diagrams in a UML system specification. This profile is not focused on a specific UML model, and therefore allows security to be specified in several UML models. In fact, UMLsec [23], employs use case diagrams to capture security requirements, activity diagrams to explain use cases in more detail in the analysis activity, object and sequence diagrams in the design activity, and deployment diagrams in the implementation activity.

UMLsec can therefore be used to specify security within business process models expressed through UML activity diagrams [26]. However, we identify several differences between the UMLsec activity diagrams and our proposal: i) While the UMLsec is a UML profile which extends certain UML general metaclasses, we have defined a profile which extends the specific metaclasses of the UML 2.0 activity diagram. In fact, UMLsec bases the security in activity diagrams on certain tagged values of stereotypes defined from the State metaclass, but we base the security specifications on metaclasses such as Activity, Data Store Node, Activity Partition, Interruptible Activity Region, Object Flow, etc. This allows us to be more precise in the specification of security requirements which are specific to business process elements, and also to exploit the new elements defined by UML 2.0. ii) While UMLsec is focused on the specification of security by the software analyst and designer, in the main software requirements, analysis, design and implementation models, our approach allows the business analyst to specify security at the business level. That is to say, we have selected specific abstract security requirements to be specified in high level business processes, which we consider to be Computational Independent Models. This signifies that both proposals use activity diagrams in two different contexts: UMLsec in a more concrete and software focused context, and ours in a more abstract and business focused context. In fact, UMLsec is based on UML 1.5, which considers activity diagrams focused on flows driven by internal processing [43], while UML 2.0 (as is the case of BPMN) is focused on the modeling of business processes. iii) Both proposals attempt to apply their models within a model driven engineering approach. UMLsec considers techniques based on model-checking to provide the translation of UMLsec models [27], and we use QVT to transform our secure business processes into class diagrams and use case diagrams. In previous works, we have also extended BPMN to the specification of secure business process

models, and BPMN is integrated into our top level architecture, thus offering CIM to CIM and CIM to PIM transformations).

These arguments do not signify that one approach is better than the other. They signify that both proposals are different and compatible. In fact, a possible scenario would be the integration of both proposals: We could use our approach to model the secure business processes, and we would then obtain a first version of the corresponding use case and class models (applying our QVT rules). UMLsec could then be applied for the software analysis, design and implementation.

An approach with which to model security by considering several perspectives is presented in Refs. [20,21,53]. The authors take the five perspectives related to business processes and security: static, functional, dynamic, and the business process (this provides us with an integrated view of all perspectives with a high degree of abstraction). The authors complement the proposal with Commercial Protocols and Service (COPS) as an infrastructure with which to build adaptable electronic markets that emphasize security and equity, and also propose a methodology – Modeling Security Semantics of Business Transactions (MOSS) – with which to analyze and model the semantics of security in business transactions. The authors are very clear as regards aspects related to the need to integrate security from early stages, and emphasize the need to add a new view of security without abandoning the traditional view of security from experts. Unlike our proposal, standard languages for business process modeling and security specification are not used in these works. Moreover, these specifications are not related to a development system information process.

In Ref. [3], security requirements, particularly cryptography, are arbitrarily integrated into the business process development. The authors do this by considering an approach based on refining by stages and extend this by aggregating both security requirement specifications and trust models. This generates a security specification that will later be transformed into refined specifications which already incorporate security. Although this approach clearly establishes the need to specify security requirements at an early stage, unlike our proposal it does not mention either the way in which these security requirements will be specified (notation or technique) or the different roles that will be fulfilled by those involved in a business process.

A business process-driven software development framework based on the UML notation and the integration of security requirements during the early stages of software development is presented in Refs. [37,58]. In these works, UML is used to represent security semantics in an integrated development environment including business processes and systems models. This permits security requirements to be integrated in the same way as other requirements in the context of software development. A method for the systems development managed by business processes in which technological decisions are managed by the business model is also proposed. The need to express security requirements at the level of a business model results from the fact that applications which consider electronic commerce transactions are conceptually similar to non-automated traditional transactions. Both proposals have the advantage of recognizing the need to express security requirements at an early stage, and use them to achieve the implementation of the system. They employ a widely accepted modeling language which facilitates their use and understanding. The main difference between our proposal and these works is that they do not consider business process models as a starting point. The application of security from a business process perspective therefore necessitates carrying out additional work to define the semantics for business process models in an exact manner.

An approach with which to model business processes security is presented in Ref. [22]. The authors show a Modeling Security of Business Process (MoSSBP) framework with which to support domain experts who may not be security experts. In this framework, security requirements can be modeled in a business process based on graphical design concepts from the specification of the business process to its implementation. In this paper, the authors apply the Object-Oriented Security Analysis to the refinement of the business process in order to guarantee security requirements. They also complement this approach with a support tool (SEMBA, based on the ARGO toolset) which facilitates the modeling of the business process and sub-process in the UML object diagrams. The main difference between this proposal and ours lies in the representation of security requirements by business experts. In this proposal the authors use UML notes to graphically represent security. These notes do not have associated restrictions, a detailed description and the relationship established with other UML elements on which these security requirements are specified.

Finally, in Ref. [54], security is modeled through the extension of UML activity diagrams in order to model Mal(icious)-Activity Diagrams. This kind of diagrams uses the same syntax and semantics as the UML-AD standard, but the author adds malicious activities and a malicious actor (both are shown with the same icons but in inverse color). The proposal is focused on the modeling of hostile activities together with legitimate activities in business process models. In this respect, the proposal is similar to the mal-processes proposal because in this work these types of specifications are simply a poorly designed business process that may prove harmful for the customer or stakeholder of the process [45]. The main difference between our approach and this proposal is that we attempt to integrate security requirements related to business process elements, while the Sindre proposal attempts to model hostile activities. It could be said that both proposals are complementary, and in fact they could be used together, since each one is focused on a different problem.

Despite the importance of all the works that we have studied, they do not completely solve the problem of incorporating the business analyst's perspective with regard to security. The business analyst is in charge of defining, modeling and describing a business process. However, to date, s/he has not had a language in which security requirements could be expressed at that level of abstraction. Furthermore, it is widely accepted that the specification of Business Processes requires specific modeling constructors and constraints that are different from those used when modeling the static and dynamic properties of the software systems. Our proposal aims to solve this problem. This has been done by increasing the expressive capacity of a modeling language (UML) by incorporating a set of security requirements into it. Business analysts will be able to use these security requirements to express security aspects related to the business process that they describe intuitively. Our proposal takes into account security aspects which are sufficiently general to make it possible for the business analyst not to be confused by technical security aspects. These specifications are also sufficiently specific to allow security experts to clearly interpret the security requirement and the scope of its specification in relation to future implementation. This allows the specification of a Secure Business Process (SBP) to be considered as part of a software creation process.

3. Secure business process modeling: a model-driven architecture compliant approach

In recent years, software engineering has been influenced by model transformation with the intention of solving problems of time, costs and quality associated with software creation. The way in which to solve these problems is part of the general denomination of engineering driven by models (MDE, Model-Driven Engineering). Engineering driven by models is the software engineering discipline which considers models as first class entities whose purpose is development, maintenance and evolution through the performance of model transformations [40]. The main idea of this approach is that of converting the principle which states that *anything is an object* into the new principle which establishes that *anything is a model* [5].

MDA is an approach which has been defined for software development; its main objective is to permit the creation of models

which can evolve from a perspective that is totally independent of the technological implementation into models which are designed for a specific platform. This approach is composed of: a computation independent perspective that considers a viewpoint of the system environment (CIM, Computation Independent Model); a platform independent perspective that considers a viewpoint of the system operation without specifying platform details (PIM, Platform Independent Model); and a perspective that has to do with a specific platform (PSM, Platform Specific Model) [42].

The model-driven approach is used in our proposal, since it establishes that a business process corresponds to a computation independent model, while UML artifacts such as analysis classes and use cases correspond to platform independent models. Thus, according to the MDA approach, a model transformation from an SBP model to analysis classes and use cases is the transformation from CIM to PIM. Fig. 1, shows all the details of our proposal. The following elements have been colored in dark gray: (i) BPSec; the UML 2.0-AD extension presented in detail in this work; (ii) M-BPSec, a method that has been designed for the ordered and systematic construction of SBP models; (iii) BPSec-Tool, a prototype that supports M-BPSec application; (iv) the SBP model that is obtained from the application of the method supported by the tool and (v) a set of rules described using QVT (Query/View/Transformation) [46] that has been incorporated into the method and the tool, and which describes the transformation from CIM to PIM.

The CIM2PIM transformations shown in Fig. 1 are not the main objective of our work and we shall therefore only present the results obtained from their application in Section 5.4. These transformations permit us to obtain a set of UML artifacts (analysis classes and use cases) that correspond to PIM models from an SBP description (in CIM).

The stages of the unified process [25] are shown in the last column of Fig. 1. Our purpose is to show that not only SBP specifications but also analysis classes and use cases can be used in a complementary manner in a consolidated and successful software development process such as the unified process. Thus, taking into consideration a certain parallelism between our proposal and the unified process, the SBP model will be built in the "Business Model" stage, and the analysis classes and use cases will be defined and refined in the "Requirements" and "Analysis & Design" stages.

The BPSec extension that allows us to extend the expressivity of UML 2.0-AD is defined in this paper. This extension is dealt with in depth in Section 4 and described in Appendices A, B and C. The method (M-BPSec), the tool (BPSec-Tool) and the transformations (CIM2PIM), will not be described in detail since they have only been included with the purpose of providing a context for the use of the BPSec extension.

4. BPSec: activity diagrams UML 2.0 profile

This section shows the UML 2.0-AD extension which a business analyst will be able to use to specify a business process including security requirements. This extension, denominated as BPSec (Business Process Security), allows UML 2.0-AD to be adapted in order to permit security requirements to be specified in the business process domain. These security requirements, which are specified at a high level of abstraction, will be translated into more concrete models and security mechanisms at the same time as the software development process advances.

This section has been divided into five parts to enable a clear and ordered presentation of BPSec. UML extensibility mechanisms and works related to extensions of activity diagrams will be presented in Section 4.1. Section 4.2 will deal with the most relevant aspects related to the security requirements that will be used in our proposal. A description of the way in which the UML 2.0 AD description is related to the new stereotypes will be shown in Section 4.3; Section 4.4 will describe associations between the UML 2.0 AD elements and the new stereotypes, and finally, the M-BPSec method which permits the ordered and systematic application of the BPSec extension will be described in Section 4.5. A detailed description of the new stereotypes, data types and labeled values is presented in Appendices A, B, and C respectively.

4.1. UML extensibility mechanisms

OMG defines two possible approaches for defining domain specific languages. The first of these is based on the definition of a new language (an alternative to UML) by using the mechanisms provided by OMG for the definition of object-based visual languages. First-class extensibility or the "heavyweight extension mechanism" is handled through MOF, where there are no restrictions on what it is permitted to do with a metamodel. It is therefore possible to add and remove metaclasses and relationships whenever necessary, thus creating a completely new language which is significantly different to UML. The second alternative is based on the UML specialization in which some of the language's elements are specialized, thus imposing new restrictions on them while respecting the whole UML metamodel and leaving the original semantics of the UML elements unchanged. The profile mechanism, the "lightweight extension mechanism", is a straightforward mechanism for adapting an existing metamodel and can be better respected by existing modeling tools. There are several reasons why it may be necessary to customize a metamodel: (i) to provide a terminology that is adapted to a particular platform or domain (i.e. capturing EJB terminology such as home interfaces, enterprise java beans, and archives); (ii) to provide a syntax for constructs that do not have a notation (i.e. in the case of actions); (iii) to provide a different notation for already existing symbols (i.e. the ability to use a picture of a computer rather than the ordinary node symbol to represent a computer in a network) or to add semantics which have remained unspecified in the metamodel (i.e. how to deal with priority when receiving signals in a state machine) [44]. Since we wish to incorporate security in the Business process model domain,

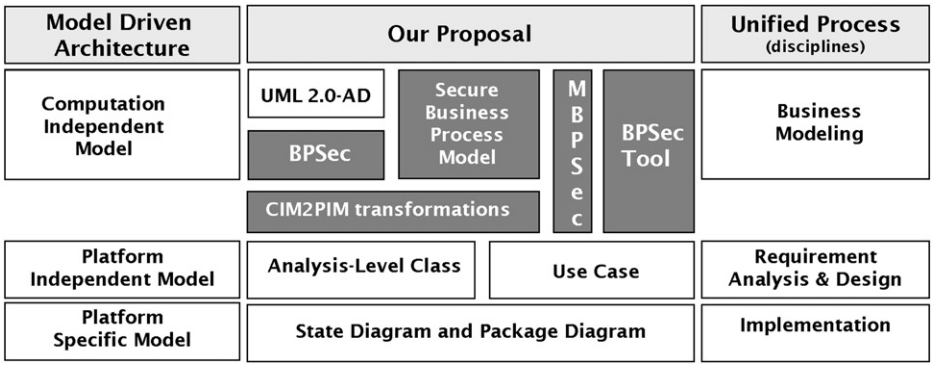


Fig. 1. Overview of our proposal.

we believe that a "lightweight extension mechanism" will allow us to clearly define the BPSec extension.

UML profiles were originally defined in version 1 of UML, although their applicability and widespread use by the software community was limited because they lacked either an unambiguous definition or precise utilization guidelines. The new version of UML (2.0) addresses these issues, providing substantial improvements to the UML profiles of UML version 1 [16]. The profiles consist of Stereotypes, Constraints and Tagged Values. A stereotype is a model element which is defined by its name and by the base class to which it is assigned. Constraints are applied to the stereotype with the purpose of indicating limitations (e.g. pre or post conditions and invariants). They can be expressed in natural language, programming language or through OCL (Object Constraint Language). Tagged values are additional meta-attributes which are assigned to a stereotype, and which are specified as name-value pairs.

Research works related to UML 2.0 extensions and business processes refer to: (i) aspects of the business such as the customer, type of business process, goal, deliverability and measure in Ref. [34]; (ii) extensions of the UML 2 activity diagram with process goals and performance measures to make them conceptually visible and provide a mapping to BPEL to make the measures available for execution and monitoring [31]; (iii) the data warehouse and its relationship with the business process dynamic structures [56]; (iv) the addition of semantics to the activities by considering organizational aspects which allow resource restrictions to be expressed during the execution of an activity [29] and (v) the provision of a more formalized definition of UML 2.0 activity diagram semantics based on the original token flow methodology [57]. No UML 2.0 extension in which security requirements are incorporated into activity diagrams therefore exists in literature.

4.2. Security requirements for business process

An SBP model will contain security requirements which take into account the business analyst's perspective. This perspective had not previously been captured together with the definition of the business process. Although security requirements are widely dealt with in literature, it is difficult to find a definition that is broad enough to describe all their properties and the range that they cover [59]. In our proposal 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.

Security requirements can be understood as needs or restrictions of a user, a stakeholder or the environment with regard to the goal of improving a system's security [59]. Three security objectives have traditionally been identified: confidentiality, integrity and availability. More recently, authentication has been aggregated [18]. If we consider security requirements from the viewpoint of the people and organizations that use computers, they can be expressed as secret, integrity, availability and responsibility [32]. From the point of view of work flows, aspects such as authorization, audit, anonymity and separation of duties can also be added [2].

In this work, the taxonomy proposed by Firesmith in Ref. [15] has been used as a reference in order to identify our security requirements. This author considers that security describes the degree to which valuable assets are protected from significant threats posed by malicious attackers, and decomposes them into a hierarchical taxonomy of security subfactors. These kinds of requirements will tend to have the same basic types of valuable and potentially vulnerable assets, independently of the level of application [15]. In this respect, the Firesmith proposal is more adjusted to the business analyst's viewpoint regarding the "*vulnerable assets*" because technical specific security solutions have not been considered in this abstraction level.

We have selected a subset of security requirements from this taxonomy, taking into account the clarity of their definition, their ease of use by business analysts and finally, whether or not the definition is independent of security specific solutions. This subset of security requirements can be widened if necessary, and is composed of the following elements (as defined by Firesmith): (1) Access Control: this signifies the degree to which the system limits access to its resources solely to authorized external people; (2) Attack harm detection: this is considered to be the degree to which an attack attempt or a successful attack is registered and notified; (3) Non-repudiation: this is the degree to which a party to an interaction (e.g., message, transaction, and transmission of data) is prevented from successfully repudiating (i.e., denying) any aspect of the interaction; (4) Integrity: this is the degree to which components are protected from intentional and unauthorized corruption. Integrity with regard to data refers to the degree to which data components (including communications) are protected from intentional corruption (e.g., via unauthorized creation, modification, deletion, or replay); (5) Privacy: this is the degree to which unauthorized parties are prevented from obtaining sensitive information (we can distinguish between anonymity, which is the degree to which the identity of users is prevented from unauthorized storage or disclosure, and confidentiality, which is the degree to which sensitive information is not disclosed to unauthorized parties (e.g., individuals, programs, processes, devices, or other systems)), and finally (6) Security Audit: this corresponds with the possibility of the security staff collecting, analyzing and giving information about the state and use of security mechanisms.

These requirements are the basis of the UML 2.0-AD extension. Each requirement or set of requirements could be explicitly indicated in a business process. A special case is that of the Audit Security requirement which, since it is basically a register of information that will be used in a security audit process, must be specified as an additional characteristic of other security requirements.

4.3. BPSec: a UML profile for security requirement specification in business processes

The profile is composed of seventeen stereotypes, one of which specializes the Activity class, eleven of which specialize the Element class (from Kernel), four of which specialize the Enumeration class (from BasicBehavior) and the last of which specializes the Actor class (from UseCase). Fig. 2 represents a portion of the UML metamodel in order to show where our stereotypes fit. Only the specialization hierarchies are represented, since the base class that the stereotype specializes is essential to the stereotype description. In this figure, new stereotypes will be colored in gray while classes from the UML metamodel will remain white.

Although each of these stereotypes will be explained in detail in Appendices A and B, a brief description of them is shown here, considering the relationship between them and the elements of the UML 2.0 metamodel.

The relationships between the aforementioned stereotypes and UML 2.0-AD are explained below:

- The «SecureActivity» stereotype, which specializes the Activity class, maintains the relationships between Activity and the other elements of UML 2.0-AD (see Fig. 2). The secure class represented by the «SecureActivity» stereotype is thus related to the ActivityGroup, ActivityNode and ActivityEdge classes.
- The «AccessControl», «AttackHarmDetection», «Integrity», «Non-Repudiation», and «Privacy» stereotypes represent the security requirements and are grouped into the «SecurityRequirement» stereotype.
- The «AuditRegister» stereotype has been specialized in «NR-AuditRegister», «SP-AuditRegister» and «G-AuditRegister». This specialization makes it possible to associate particular characteristics related to audit register with the following requirements: Non Repudiation (related to «NR-AuditRegister»), Access Control and

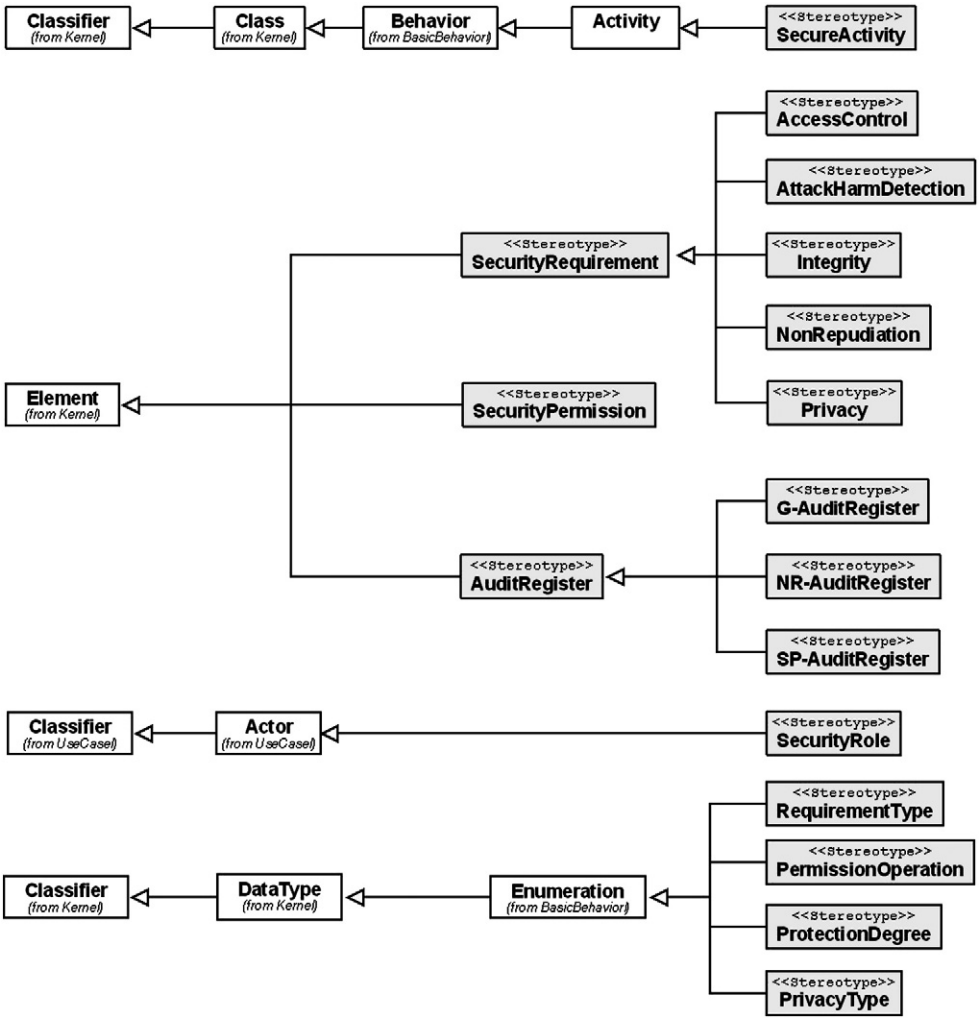


Fig. 2. BPSec stereotypes from UML 2.0 specification.

permission specifications (related to «SP-AuditRegister»), and Attack Harm Detection, Integrity, and Privacy (related to «G-AuditRegister»).
- The «SecurityRole» stereotype, which allows us to complement the security specifications associated with Access Control and Privacy requirements.
- The «SecurityPermission» stereotype, which is used to indicate the existing permissions over objects that are within the scope of an Access Control specification.
- The «RequirementType» strereotype, which corresponds to a data type that represents valid combinations between security requirements.
- «PermissionOperation», «ProtectionDegree» and «Privacy» stereotypes, which correspond to data types that are necessary for the specification of certain characteristics of the security requirements of which the profile is formed.

The stereotypes of which the proposed extension is formed have been grouped into two packages; BPSec and Types BPSec (see Fig. 3). The most important stereotype in the BPSec package is «SecureActivity» which, as has already been stated, specializes the Activity class. The relationship between «SecureActivity» and the «SecurityRequirement» stereotype establishes that there will be a secure activity only if at least one security requirement is specified.

The Types BPsec package contains the definition of new data types which are necessary for BPSec definition. «import» is conceptually equivalent to importing an element to each individual member of the imported namespace. In this case, the «import», relationship established between the packages implies that the importing namespace (Types BPSec) adds the names of the members of the package to the BPSec namespace, thus allowing us to use the new labeled values in the definition of stereotypes.

All the stereotypes of which Types BPSec are composed are inherited from the Enumeration metaclass. The types of data allow us to define:

- RequirementType, which contains the values accepted in the individual or combined specifications of security requirements.
- PermissionOperation, which allows us to specify the permitted values for the operation permissions defined for all the objects that are within the scope of an Access Control Specification.
- ProtectionDegree, which contains a classification of the defined protection together with an Integrity specification.
- PrivacyType, which describes the values associated with a Privacy specification.

Since a business analyst performs the specification of security requirements in our proposal, it is necessary to consider the graphical representation of these requirements. This has been done through the use of a padlock which is considered a *de facto* standard associated with security (we have considered studies [10] that demonstrate the close relationship between the padlock icon and security). In Fig. 4a, we show the basic symbol over which a determined security requirement is specified. In Fig. 4b the same symbol but with one of the edges folded, and associated with the symbol used to represent a

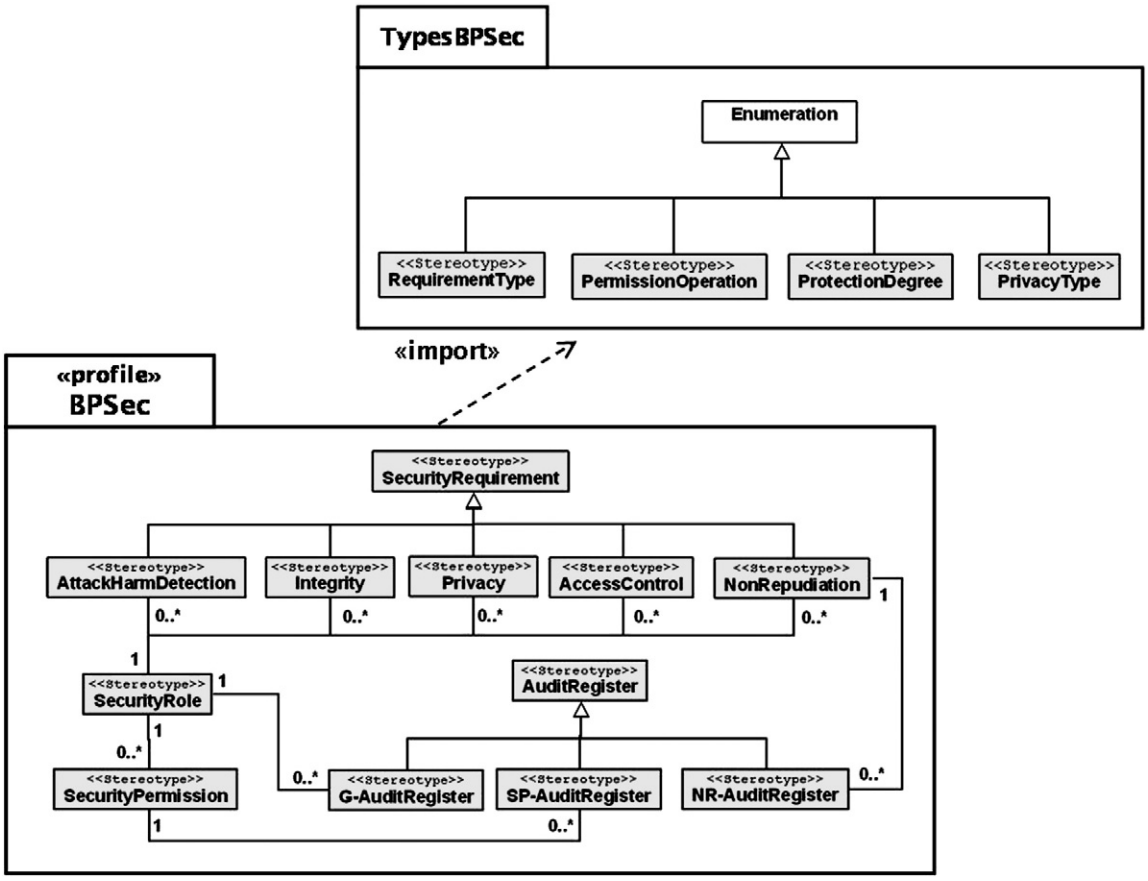


Fig. 3. High level view of BPSec extension and Types BPSec.

comment, annotation or register, will be used to represent a security requirement which also requires audit register.

As the chosen symbol allows us to perform a generic representation of security, it is necessary to indicate precisely which requirement we aim to specify. Each security requirement has therefore been associated with an abbreviation, as is detailed below:

- AC: associated with Access Control. In this case, it is possible to specify either Access Control or Access Control with audit register (see Fig. 4c).
- AD: associated with Attack Harm Detection requirement. This requirement is specified with compulsory audit register (see Fig. 4d).
- Ix: for Integrity security requirement. This requirement must be specified with compulsory audit register. It is also necessary to indicate the desired degree of integrity. The letter x is therefore replaced with the letter w when we wish to specify low integrity, with the letter m to specify medium integrity and with the letter h to indicate high integrity (see Fig. 4e).
- NR: for Non Repudiation requirement. This requirement can alternatively be specified with or without audit register (see Fig. 4f).
- Px for Privacy requirement. This requirement can be specified with or without audit register. We must additionally indicate the type of privacy required. The letter x is therefore replaced with the letter a to indicate anonymity or the letter c to indicate confidentiality (see Fig. 4g).

The stereotypes associated with the BPSec and Types BPSec packages along with the labeled values are described in detail in Appendices A, B, and C respectively.

4.4. BPSec and UML 2.0-AD elements associations

This section provides an explanation of the relationship between the stereotypes defined in the BPSec extension as security requirements and the UML 2.0-AD elements (the remaining new stereotypes complement the security requirements and are not therefore associated with UML 2.0-AD elements). This will be done by focusing on two aspects of this relationship. The first is that of the validity of the relationship while the second concerns the graphical aspects regulating the incorporation of new icons.

The validity of the relationship consists of determining the elements that can be related, along with establishing the characteristics of the relationship. The valid relations are shown in Table 1 in which a symbol (✓) indicates permitted relationships. For example, the «Privacy» security requirement can only be specified in a region and/or a partition.

These basic relationships must be completed with the relationships which exist with the stereotypes that allow the performance of audit register («AuditRegister»), definitions of roles «SecurityRole», and security permissions «SecurityPermission». These stereotypes are

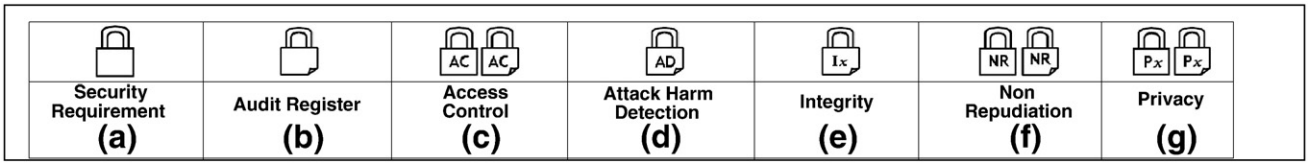


Fig. 4. Icons used to represent security requirement in BPSec.

also part of the BPSec extension. The UML 2.0-AD elements related to the stereotypes of which the BPSec extension is formed (in gray) are shown in Fig. 5.

The second aspect of the relationship between BPSec and the UML 2.0-AD elements concerns the incorporation of the security requirement into the graphical representation of each UML 2.0-AD element. The new symbol (padlock) must be clearly identified among the UML 2.0-AD graphical elements and must not alter the visibility of the icon to which it is related. Table 2 describes the UML 2.0-AD elements over which it will be possible to specify security requirements in a graphic manner. The first column both shows the elements and indicates the location of the padlock in relation to the UML 2.0-AD element. This will be graphically described in the second column.

4.5. M-BPSec: method for security requirement specification in business process

In order to avoid the arbitrary use of BPSec, we have created the M-BPSec method (Method for Business Process Security), which permits the specification of secure business process models. For the sake of completeness in our global proposal, we offer a very high level summary of M-BPSec here, while a more specific description can be found in Ref. [49]. This method is a guide which allows the BPSec extension to be applied in a systematic and unambiguous manner.

M-BPSec, which will be shown in its entirety in Fig. 6, is composed of a set of stages, roles, tools and artifacts that allow us to create SBP models and obtain useful artifacts for software development in an engineering and systematic approach.

The first three stages of M-BPSec are directly related to the definition of SBP models. These can be used to design a business process, add security requirements to it and refine these specifications. The fourth and last stage is a complement through which analysis classes and use cases that can be used as a complement in a software construction process are automatically generated. The use of M-BPSec will thus permit requirements to be captured early, with special emphasis on security requirements. The resulting model (SBP) will be transformed into a set of artifacts which are useful for software construction under a model driven approach. In this respect, M-BPSec is MDA compliant because it allows us to move from CIM to PIM. Our future work will be to obtain code specification through PSM models.

5. Illustrative example

In this section we present an illustrative example which will be developed in accordance with the application of the stages defined in M-BPSec, in other words: (i) business process construction, (ii) security requirement incorporation, (iii) refining, and (iv) transformation.

Although our proposal has already been applied to a real case (concerned with payment for the consumption of electrical energy) [48], in this section we have preferred to present an example in the context of the well-known health sector. This example deals with the classical problem of Patient Admission, which makes it a highly intuitive and pedagogic manner in which to present how our proposal can be used.

The health care institution considered in this illustrative example is a clinic where medical tests, control checks, placement of patients treated for chronic diseases and minor surgery are performed, and where the emergency department and outpatient care have not been considered. The business process described models how a patient receives information about clinical examinations and screenings that are necessary for the placement, implementation of treatments and/or minor surgery. The requirements related to the need to maintain data privacy, control access to these services, and to maintain the integrity of sensitive information are modeled from the security viewpoint. The "Patient Admission" business process in a health institution commences with an admission request which is filled out by a patient. This document, called Admission Request, is sent to the Administration area. In this area, information related to insurance is captured and the existence of a medical history associated with the patient is verified. Once the patient's information has been validated and completed, it is sent to the medical area. The medical evaluation area uses a set of pre-admission tests to determine the patient's medical condition. If necessary, additional examinations will be carried out, and these must be registered from the clinical and economic points of view. Finally, the Medical Evaluation document is completed with information about the patient and is sent to this patient. The business process concludes when the patient has received the medical evaluation.

The application of the M-BPSec stages to the Patient Admissions business process will be presented in the following sections. The M-BPSec method is supported by a prototype BPSec-Tool that permits the construction of a business process, the incorporation of security requirements and the process of obtaining analysis classes and use cases through automatic transformation. BPSec-Tool was built using a 3-tiered architecture to separate the presentation, application, and storage of components, using MS-Visio, C#, and MS-Access technology.

Table 1
Security requirements and activity diagram elements.

Stereotypes for secure activity specification	UML 2.0 element for containment in activity diagrams				
	Action	Data store node	Activity partition	Interruptible activity region	Object flow (data)
Access Control	✓	✓	✓	✓	✓
Attack Harm Detection	–	✓	✓	✓	✓
Integrity	–	✓	–	–	✓
Non repudiation	–	–	–	–	✓
Privacy	–	–	✓	✓	–

5.1. Stage 1: business process construction

The business analyst identified the following Activity Partitions with which to construct the business process using UML 2.0-AD: (i) "Patient", which represents an individual who needs medical attention, care or treatment; (ii) "Administration Area", which is a top partition that is divided into two middle partitions called "Admission" and "Accounting" and represents the area of the organization which is responsible for the information related to payment, insurance and accounts of the treatment and attention received by the patient and (iii) "Medical Area", which is a top partition that is divided into "Medical Evaluation" and "Examinations" middle partitions and represents the area of the organization which is responsible for all information related to the patient's tests, examinations and clinical information. The business analyst also identified an InterruptibleActivityRegion which is within the scope of the Administration Area.

The DataStoreNodes that we have considered are the following: "Admission Request", which contains information relating to patient data and the reason why s/he requires medical attention, "Accounting Data" and "Accounting Information", both of which are dedicated to the storage of information concerning medical insurance, expenses incurred as a result of examination concepts and medical information, "Clinical Data" which includes information about the patient, "Clinical Information" with information about medical supplies, and "Medical Evaluation" which contains the detailed report of the patient's state. There are a total of 15 identified Actions which describe the activities that must be carried out in each of the identified partitions.

Although Fig. 7 shows a business process in which security requirements have already been incorporated, this figure also shows the main aspects of the business process described in this section.

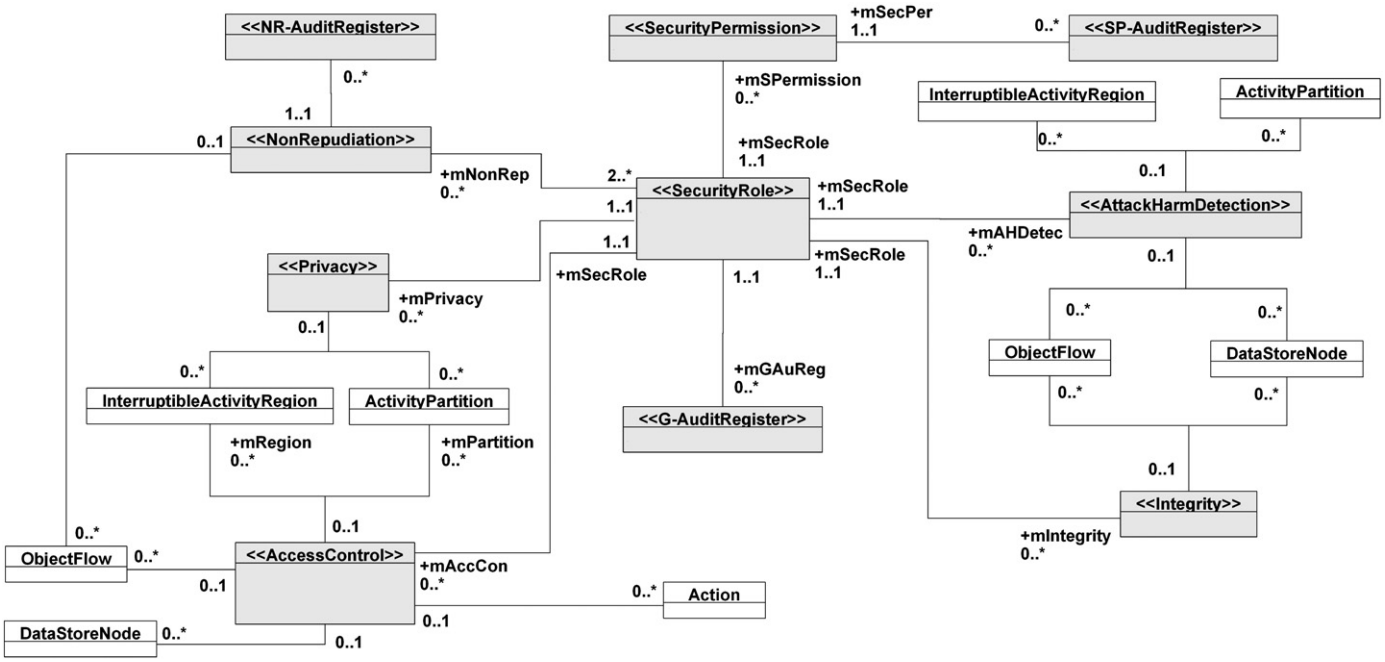


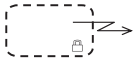
Fig. 5. BPSec stereotypes and activity diagrams element.

5.2. Stage 2: security requirement incorporation

The business analyst must incorporate all the security requirements into the business process definition during this stage. In this case, five security requirements specifications have been made (see Fig. 7); (i) Privacy (top left), (ii) Non Repudiation (right hand side of the "Admission Request" action), (iii) Access Control and Privacy (bottom of Region), (iv) Integrity (right hand side of the "Clinical Information" DataStoreNode), and (v) Attack Harm Detection (left hand side of the "Medical Evaluation" DataStoreNode).

The meaning of the profile of each security requirement specification is explained as follows: (i) a privacy specification, with anonymity, has been created for the "Patient" Activity Partition. The *KindPrivacy* tagged value associated with the privacy specification is *a* (anonymity) with the aim of preventing the storage of sensitive information about Patients; (ii) a non repudiation security requirement has been defined over the out control flow of the "Admission Request" data store node. This control flow is connected to the "Capture Insurance Information" and "Check Clinical Data" actions. According to the definition of this security requirement, the business analyst aims to avoid the denial of the "Admission Request" reception. The values associated with source and destination roles (tagged values SecurityRoleDestinationName and SecurityRoleSourceName) are "Patient" and "Admission" respectively; and (iii) an Access Control and Privacy (with confidentiality) have been defined over the Interruptible Activity Region. Another stereotype, «SecurityRole», must be derived from this specification. Admission/Accounting will be a role. All objects in this region must be considered for permission specification. A padlock with a folded edge has been specified with access control, implying that the *AuditRegister* tagged value is activated. Information about the security role and security permissions must therefore be registered; (iv) an integrity (high) security requirement has been specified for the "Clinical Information" Data Store. The *KindIntegrity* tagged value is *high*, implying the maximum degree of protection for components from intentional and non-authorized alteration. The AuditRegister tagged value is activated. This implies that a G-AuditRegister stereotype is used to register information storage concerning the time, date, and role that the data store uses, protected with an integrity requirement; (v) attack harm detection for the "Medical Evaluation" Data Store Node. This specification implies the maintenance of the register of events (attacks or threats, whether successful or not) which have occurred to potentially vulnerable elements. This requirement can only be specified with the audit register, so a G-AuditRegister stereotype is used to register information storage concerning the time, date, and role.

Table 2
Security requirement and UML 2.0-AD graphic representation.

UML 2.0-AD element	Graphic representation
ActivityPartition: An activity partition normally indicates who or what is responsible for the actions it groups. In UML, the responsible element corresponds with the class supporting the behavior invoked by the actions in a partition. The icon associated with security is drawn in the lower part of the rectangle used for identifying the partition.	
InterruptibleActivityRegion: This consists of a group of activities whose main characteristic is that the whole execution of the region concludes when the term arc leaves the region, independently of the flow of the activities that are grouped in the region. Security is specified in any of the lower edges of the figure.	
DataStoreNode: This fulfills the function of intermediate storage of non-transitory information. This element is new in the UML version 2. Security can be expressed in any of the lower edges of the rectangle used for representing a DataStoreNode.	
ObjectFlow: This is an arc that links actions which may contain the objects or data that pass through it. Security must be specified over the arc itself.	
Action: This is the fundamental unit of the executable functionality. The execution of an action represents certain transformations or processing in the modeled system, whether it is computational or not. Security can be expressed in any of the lower edges of the symbol that represents an action.	

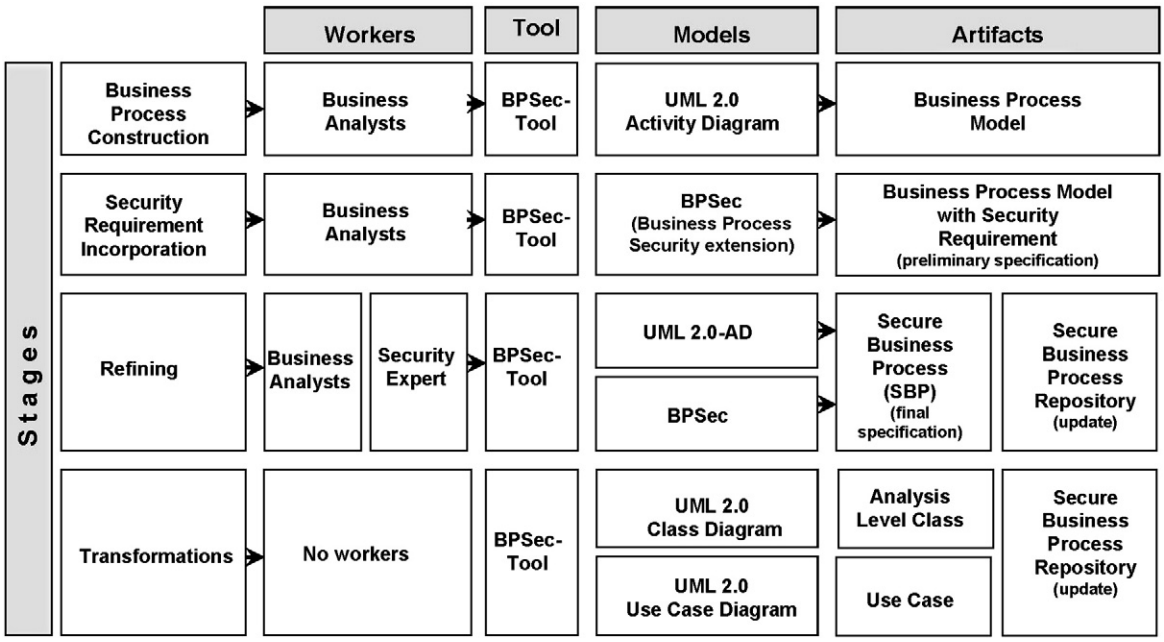


Fig. 6. Complete view of the M-BPSec method.

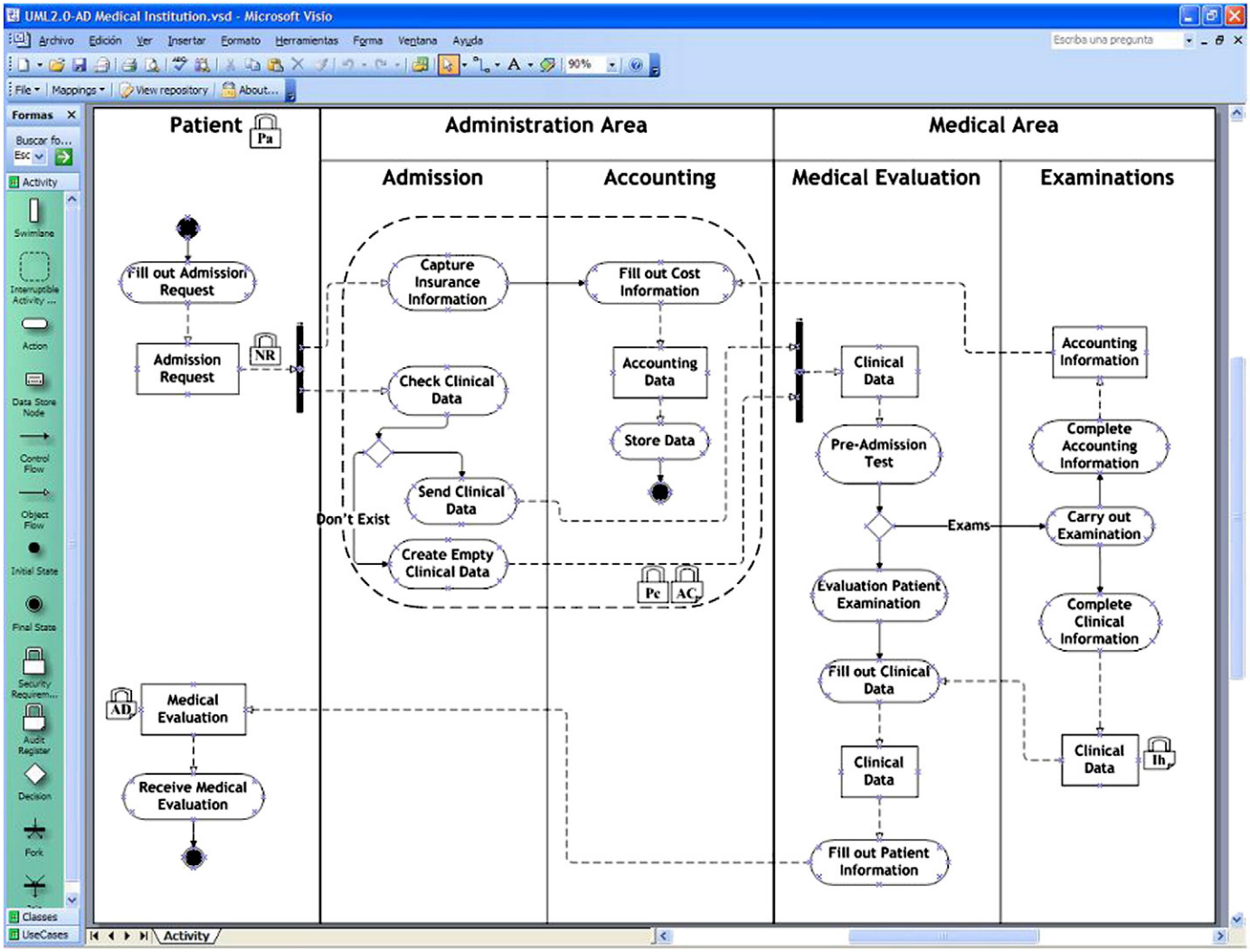


Fig. 7. Admission of patients to a medical institution.

The final task in this stage is to specify the priority for each security requirement defined (order of relative importance for each security requirement according to the business analyst's viewpoint, which must be specified as "*must be*", "*should have*", "*could have*", or "*want to have*") and the security permissions for each of the elements that are within the scope of the Access Control Specification (see final version in Table 3).

5.3. Stage 3: refining

The security requirements specified in the business process description must be reviewed and complemented in this stage. Both the business analyst and the security expert work together, and the specifications that will finally be incorporated into the business process are agreed. The final result of the SBP description that has been built using the BPSec-Tool is shown in Fig. 7. The final result of this stage takes into consideration the refinement of priority specifications for each security requirement and that of the security permission specifications associated with UML 2.0-AD elements that are within the scope of an access control specification. Details of permissions associated with UML 2.0-AD elements will be shown in the specification of the PermissionOperation stereotype in Appendix B.

If an Access Control requirement has been specified in a business process (see Interruptible Activity Region in Fig. 7) then it will be necessary to indicate the type of permission associated with it for each element within the scope of Access Control. Table 4 shows details of the specification of permissions for each element. The UML 2.0-AD element appears in the first column, the type of element we are dealing with is identified in the second column and the type of permission that has been assigned is indicated in the last column. For example, if the business analyst wishes to establish some kind of restriction over a DataStoreNode then this must be specified by indicating one of the following permissions associated with that element (in this case, Update).

Stages 2 and 3 are both directly related to security. The use of BPSec-Tool allows business analysts and security experts to apply and refine the security which has been specified.

5.4. Stage 4: transformation

In this stage, we use BPSec-Tool to obtain a set of analysis classes and use case diagrams. This section shows the analysis classes diagram and some of the use cases that are obtained from the specification of the Patient Admission business process.

Transformations to analysis-level classes require both a set of rules which have been specified in the QVT language and a set of refinement rules [47]. The aim is to obtain an analysis-level class diagram which is representative of the business process, paying special attention to the classes related to security specifications. In Fig. 8 the classes related to security specifications are colored in gray. These classes correspond to the detailed description of each of the stereotypes to which BPSec conforms. Each class diagram contains the security classes which are integrated with the rest of the classes derived from the business process specification. For example, the Integrity specification (with Audit Register) for "Clinical Data" creates a class called Integrity that relates to ClinicalData class and G-AuditRegister class.

Table 3
Security requirement priority.

Security requirement	Priority			
	Must have	Should have	Could have	Wants to have
Privacy (anonymity)	✓			
Nonrepudiation	✓			
AccessControl and AuditRegister		✓		
Privacy (confidentiality)		✓		
Integrity (high)			✓	
AttackHarmDetection				✓

Table 4
Permission in AccessControl specification scope.

UML 2.0-AD element		BPSec element
Name (AcDgElementName)	Type (AcDgElementtype)	Permission (KindPermission)
Capture insurance information	Action	Execution
Fill out cost information	Action	CheckExecution
Check clinical data	Action	Execution
Store data	Action	Execution
Create empty clinical data	Action	Execution
Accounting data	DataStoreNode	Update

With regard to transformations to use cases, it is necessary to specify a set of QVT rules, checklists and a set of refinement rules [51]. When applying these transformations to the Patient Admission process, we have obtained a general use case diagram and five use case diagrams related to the specifications of security requirements.

The general use case diagram will be shown in Fig. 9. Security specifications are not represented in this use case diagram since we will obtain specific use case diagrams related to security. In general terms, the actors represented in Fig. 9 are obtained from the partitions and regions and the use cases are obtained from the actions that are within the scope of each partition or region.

The use case diagrams can be obtained from a set of checklists related to security specifications. Five use case diagrams have been derived from the security requirements specification; Privacy (with confidentiality) and Access Control, Privacy (with anonymity), Non Repudiation, Integrity, and Attack Harm Detection. Fig. 10 shows the use case diagram corresponding to the Privacy (with confidentiality) and Access Control security requirement specifications. In this use case diagram we have identified Admission Accounting and Security Staff actors and the use cases which allow us to assign and validate roles, to verify permissions and to perform audit register.

All the use cases are used as a support element in the process of software creation, as occurs with the class diagram (see Fig. 1). All diagrams, classes and use cases must be refined and completed to ensure that they are effectively useful for the development of the software that will automate the business process.

6. Real application context and learned lessons

In order to demonstrate the useful nature of our proposal, we have conducted a case study in a real organizational environment. This case study has been developed in a cooperative which is dedicated to the distribution of electricity in rural areas. The Coopelan Ltda. (http://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. Because the cooperative's main clients live in rural areas, the way in which it presently receives payment for the consumption of electricity causes 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 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. We have developed a business process which will describe a payment for the consumption of electrical

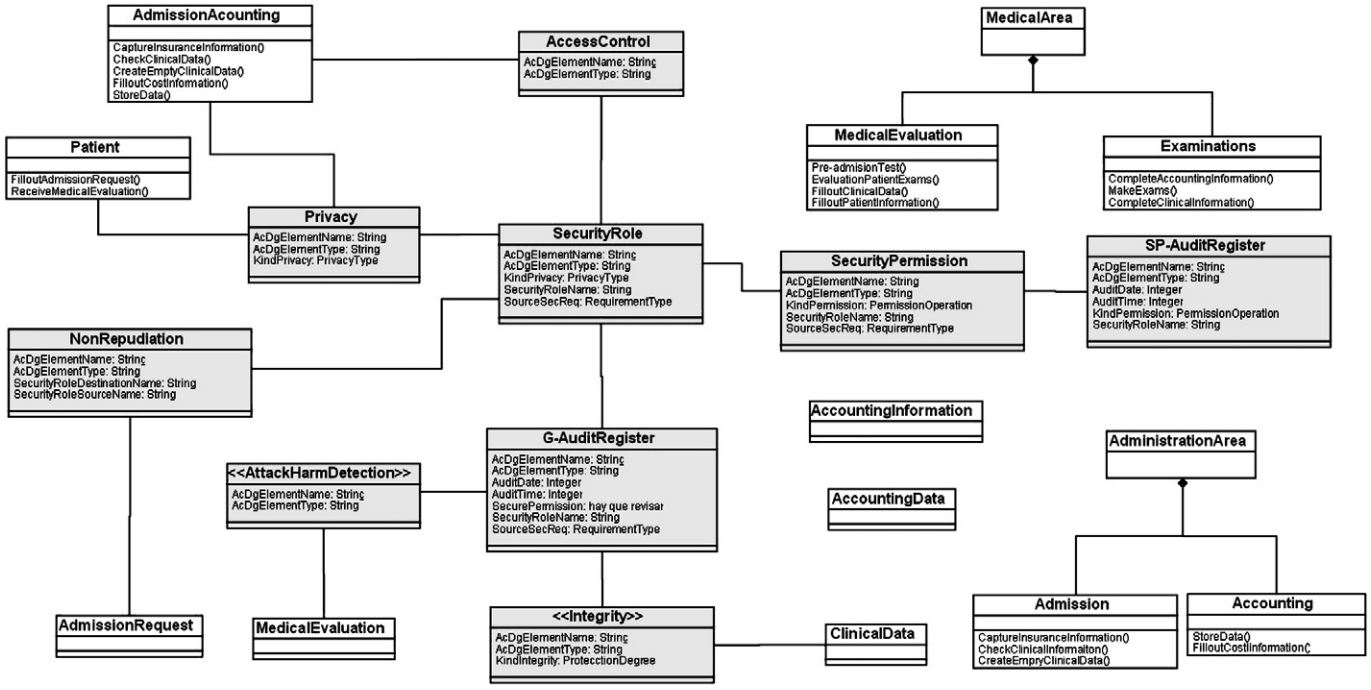


Fig. 8. Analysis-level class from patient admission.

energy. The case study was carried out with the assistance of the cooperative's business analysts.

In order to obtain security requirements we have reviewed works related to the requirements elicitation methods [11,38,39]. However, none of the proposals consider the acquisition of security requirements by using a business process described with UML 2.0-AD as a starting point. We have consequently decided to use M-BPSec in the development of this case study. The results of the application of our proposal are as follows: (i) it has allowed us to improve our UML profile, our M-BPSec method and our BPSec-tool through the characteristic iterative process of the action-research method [4,9] owing to continuous feedback between the researchers and the business analysts; (ii) the analysis artifacts that are automatically generated from the secure business process models are very useful, and can be used as the starting point in the software development process; (iii) we observed that the organization was pleased with the possibility of being able to express security in business process models; (iv) the learning curve associated with the application of the method was acceptable and the use of our BPSec-Tool facilitated the business analysts' work; (v) the use of the M-BPSec method was fundamental to the disciplined development of this case study.

7. Conclusions and ongoing work

In this paper, we have presented an extension of the UML 2.0 activity diagram which will allow business analysts to specify a set of security requirements in business process models. Standard notations for the modeling of business processes do not have the syntactic and semantic capabilities with which to specify security requirements as other elements. Our BPSec extension therefore makes it possible to define a set of security requirements (Access Control, Attack harm detection, Non-repudiation, Integrity, Privacy and Security Audit), thereby improving the expressiveness of the business process models, and allowing the security of the information systems to be developed by taking that requirement source as a base.

The use of business process models in the software engineering field has been considered in our proposal, and it is consequently within the scope of the software development approach based on MDA. In this context, an SBP model is considered to be a computation independent model (CIM). This model is used as a starting point from which to obtain a set of UML artifacts, analysis level classes and use cases, considered as platform independent models (PIM). These artifacts, which contain aspects of security, and which are automatically generated from the business process model, will be used in the requirement, analysis and design stages within the software development process.

Space limitations have hindered us from providing a detailed explanation of the application of our proposal in a real environment. We have, however, carried out a case study which has proved to have several interesting advantages. The learning curve associated with the application of the method was acceptable and the use of the BPSec-Tool facilitated the business analysts' work. It was also possible to improve both the tool and the readability of the artifacts derived from the transformation stage.

Our future work is oriented towards the application of our proposal to more complex scenarios, and also to the integration of our approach into a complete software development methodology. This will allow us to connect the business analyst point of view of security with a much more technical view of security considered within the software development process, considering security requirements, security patterns and security techniques in the moment in which they are necessary and opportune. Finally, we hope to relate security specifications in business processes to the implementation of these specifications in order to obtain information about the effort (cost and time) of transforming specifications, and to develop a measure to verify whether the solutions meet with the original specifications.

Appendix A. Description of stereotypes

In this section, we describe the stereotypes of which the BPSec package is formed. The stereotypes described in this Appendix correspond with the stereotypes of which the BPSec package is formed (see Fig. 5) and will be presented alphabetically in order to facilitate their location.

ACCESSCONTROL

Generalization: Element::SecurityRequirement
Description: This corresponds to the limitation of access to resources to authorized users only. The specification of this requirement by the

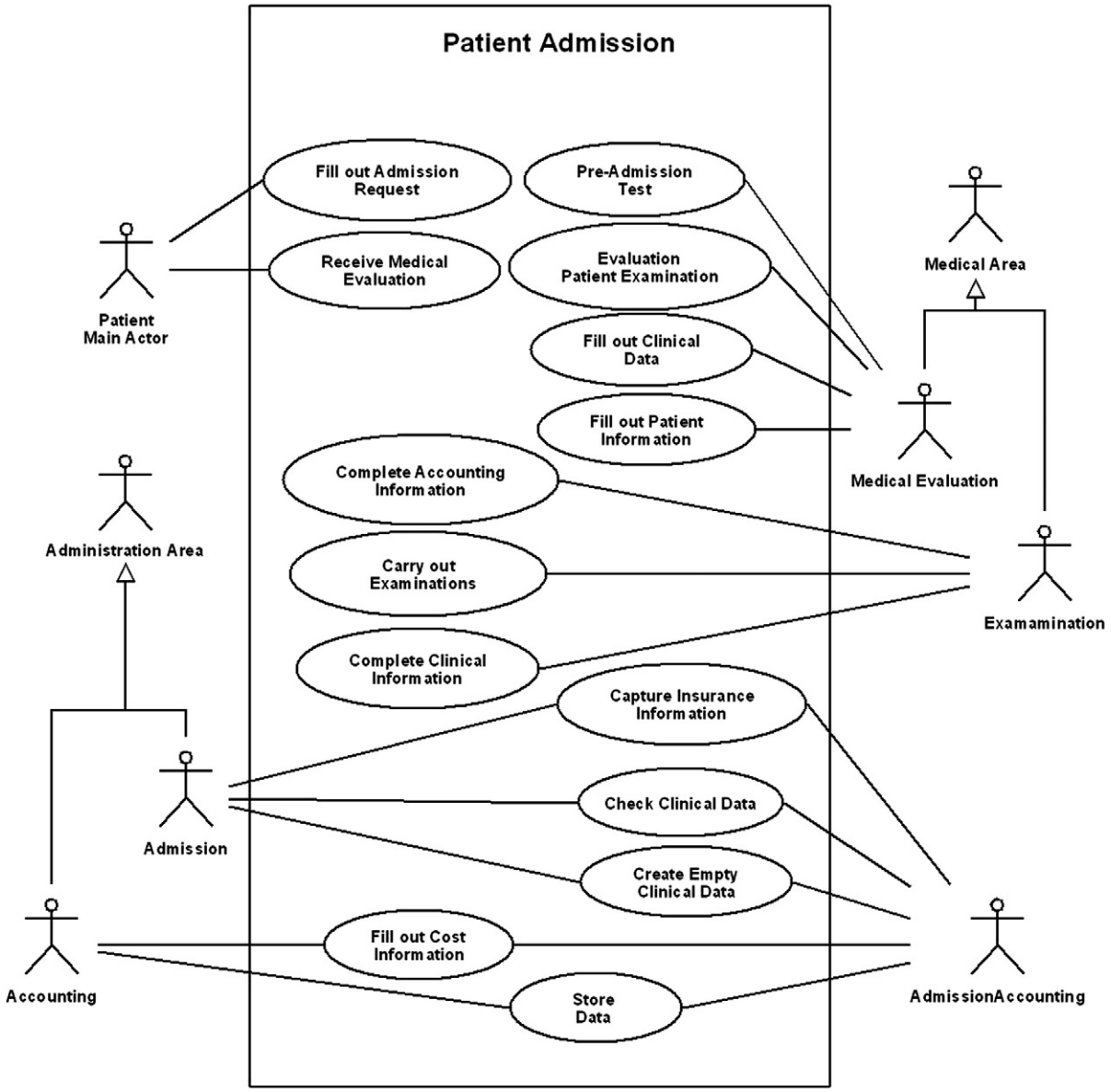


Fig. 9. Patient admission use case specification.

business analyst implies the limitation of access to a set of resources that are considered sufficiently important to be protected in a special manner. From the security perspective, this specification consists of the definition of roles that can be assigned to individuals, entities, programs, devices or other systems, along with the definition of permissions to access objects included in the field of the access control specification. This requirement can also have a specification of audit register. Notation: (Fig. 4c)

Associations: Action[0..*]; ActivityPartition[0..*]; DataStoreNode [0..*]; InterruptibleActivityRegion[0..*]; ObjectFlow[0..*]; SecurityRole [1..1]

Tagged Value: AcDgElementName, AcDgElementType, AuditRegister

Constraints

[1] It can only be specified in the following elements of the activity diagram: Action, ActivityPartition, DataStoreNode, InterruptibleActivityRegion, and ObjectFlow and must be related to a «SecurityRole»

context AccessControl

inv: self.Action→size()>=0

inv: self.ActivityPartition→size()>=0

inv: self.DataStoreNode→size()>=0

inv: self.InterruptibleActivityRegion→size()>=0

inv: self.ObjectFlow→size()>=0

inv: self.SecurityRole→size()=1

[2] The specification of access control gives origin to the «SecurityPermission» stereotype in which the permissions associated with the elements included in the field of access control are situated

context AccessControl

inv: self.mSRole.mSPermission→size()=1

[3] We can indicate the audit register which gives origin to «G-AuditRegister» that is related to «SecurityRole» and to «SP-AuditRegister» which is related to the security permissions that have been specified over the objects included in the field of access control.

context AccessControl

inv: self.AuditRegister=True implies (self.G-AuditRegister→size()>=1 and self.SP-AuditRegister→size()>=1)

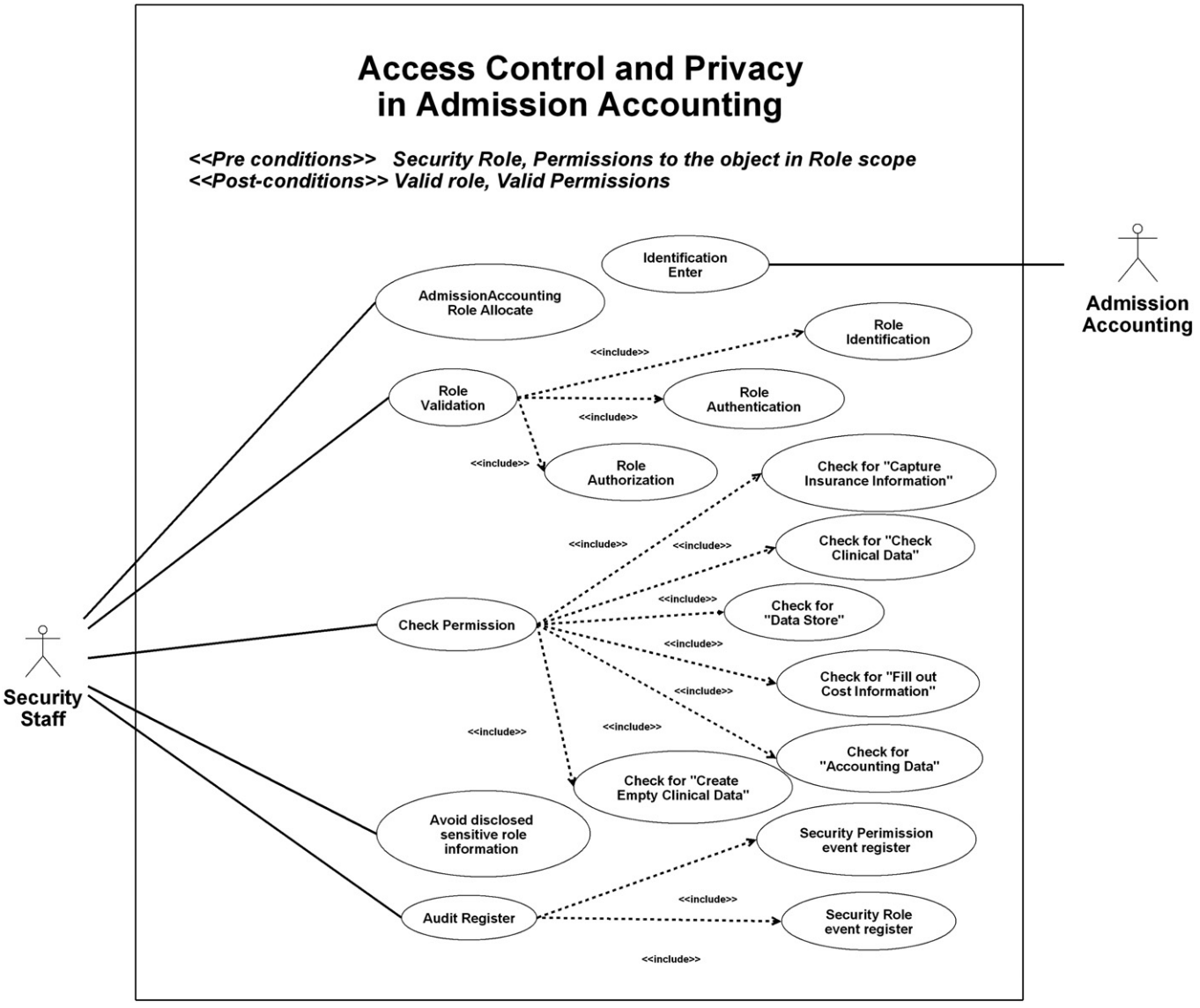


Fig. 10. AccessControl/Privacy use case specification.

[4] The name of the security role generated through an «AccessControl» specification performed over Action, DataStoreNode or ObjectFlow must be the same as the name of the partition or region in which it is contained.

Context AccessControl
inv: if self.Action→size()=1 then
if self.interruptibleRegion→size()>=1 then
self.mSRole.SecurityRoleName=self.interruptibleRegion.Name
endif or else
if self.inPartition→size>=1 then
self.mSRole.SecurityRoleName=self.inPartition.Name
endif
endif
inv: if self.DataStoreNode→size()=1 then
if self.interruptibleRegion→size()>=1 then
self.mSRole.SecurityRoleName=self.interruptibleRegion.Name
endif or else
if self.inPartition→size>=1 then
self.mSRole.SecurityRoleName=self.inPartition.Name
endif
endif
inv: if self.ObjectFlow→size()=1 then
if self.interrups→size()>=1 then
self.mSRole.SecurityRoleName=self.interrups.Name
endif or else
if self.inPartition→size>=1 then
self.mSRole.SecurityRoleName=self.inPartition.Name
endif
endif

[5] When there is a collision between «AccessControl» and «SecurityRole» specifications, this will have the same name as the partition or region which contains the elements over which access control has been specified.

context AccessControl

inv: self.ActivityPartition→size()>1 and
(self.Action→size()=1 or self.DataStoreNode→size()=1 or self.ObjectFlow→size()=1)
implies self.mSRole.SecurityRoleName=self.mPartition.Name
inv: self.InterruptibleActivityRegion→size()>1 and
(self.Action→size()=1 or self.DataStoreNode→size()=1 or self.ObjectFlow→size()=1)
implies self.mSRole.SecurityRoleName=self.mRegion.Name

ATTACKHARMDETECTION

Generalization: Element::SecurityRequirement
Description: This is defined as the detection, registration and notification of an attempted attack or threat, whether it is successful or not. From the business analyst perspective, this requirement represents an attention signal over the elements in which it is indicated. Furthermore, it can be interpreted as a previous step to an access control specification. From the security point of view, this specification implies the maintenance of the register of events (attacks or threats) which have occurred to potentially vulnerable elements. This requirement can only be specified with the audit register Notation: (Fig. 4d)
Associations: ActivityPartition[0..*], DataStoreNode[0..*], InterruptibleActivityRegion[0..*], ObjectFlow[0..*], SecurityRole[1..1]
Tagged Value: AcDgElementName, AcDgElementType
Constraints
[1] A SecurityRole and an audit register must be specified
context AttackHarmDetection
inv: self.SecurityRole→size()=1
inv: self.mSecRole.mGAuReg→size()>=1
[2] When AttackHarmDetection is specified, we must create a security role and an audit register.
context AttackHarmDetection
inv: self.SecurityRole->size()=1
inv: self.mSecRole.mGAuReg->size()>=1
[3] The name of the security register which is generated from an AttackHarmDetection specification carried out over DataStoreNode or ObjectFlow should be the same as the name of the partition or region in which it is contained.
context AttackHarmDetection
inv: if self.DataStoreNode→size()=1 then
self.interruptibleRegion→size()>=1 implies self.mSRole.SecurityRoleName=self.interruptibleRegion.Name
else
self.inPartition→size>=1 implies self.mSRole.SecurityRoleName=self.inPartition.Name
endif
inv: if self.ObjectFlow→size()=1 then
self.interrups→size()>=1 implies self.mSRole.SecurityRoleName=self.interrups.Name
or else
self.inPartition→size>=1 implies self.mSRole.SecurityRoleName=self.inPartition.Name
endif
[4] When there is a collision in the AttackHarmDetection specification, the SecurityRole will have the name of the corresponding partition or region
context AttackHarmDetection
inv: self.ActivityPartition→size()>1 and
(self.Action→size()=1 or self.DataStoreNode→size()=1 or self.ObjectFlow→size()=1)
implies self.mSRole.SecurityRoleName=self.mPartition.Name
inv: self.InterruptibleActivityRegion→size()>1 and
(self.Action→size()=1 or self.DataStoreNode→size()=1 or self.ObjectFlow→size()=1)
implies self.mSRole.SecurityRoleName=self.mRegion.Name

AUDITREGISTER

Generalization: Element (from Kernel)
Description: Abstract class containing audit register specifications related to a security requirement specification. Each audit register type must be indicated in some of its subclasses. Notation: (Fig. 4b) (this corresponds to the symbol associated with security which also combines a comment or annotation in order to represent the fact that a security requirement needs an audit register)
Associations: None
Tagged Value: None
Constraints: None

G-AUDITREGISTER

Generalization: Element::AuditRegister
Description: Contains the audit specifications related to those security requirements which coincide with regard to the information that it is necessary to store. It is directly related to the SecurityRole and only indirectly related to the AccessControl, AttackHarmDetection, Integrity and Privacy requirements, owing to the fact that NonRepudiation has its own audit register. No notation.
Associations: SecurityRole [1..1]
Tagged Value: AcDgElementName, AcDgElementType, AuditDate, AuditTime, SecurityRoleName, SourceSecReq
Constraints
[1] This stereotype must be associated with a «SecurityRole»
context G-AuditRegister
inv: self.SecurityRole→size()=1
[2] The direct relations with the audit register are determined by the values of SourceSecReq, in such a way that the valid values are AC (AccessControl), AD (AttackHarmDetection), I (Integrity) and P (Privacy)
context G-AuditRegister
inv: self.mAccCon→size()=1 implies self.SourceSecReq="AC"
inv: self.mAHDetec→size()=1 implies self.SourceSecReq="AD"
inv: self.mIntegrity→size()=1 implies self.SourceSecReq="I"
inv: self.mPrivacy→size()=1 implies self.SourceSecReq="P"

INTEGRITY

Generalization: Element::SecurityRequirement
Description: This is related to the protection of components from intentional and non-authorized alterations. The integrity specification is valued as low, medium, and high. From the business analyst perspective, an integrity specification (in any degree) is related to the importance of the information contained in the data store or data flow. From the security expert perspective, the integrity specification implies the registration of the involved role,

date and time of access to the data store or data flow. Additionally, security measures are specified according to the degree of integrity. This requirement is always associated with the audit register. Notation: (Fig. 4e)

Associations: DataStoreNode [0..*], ObjectFlow [0..*],SecurityRole [1..1]
Tagged Value: AcDgElementName, AcDgElementType, KindIntegrity
Constraints

[1] It can only be specified in the following elements of the activity diagram: ObjectFlow and DataStoreNode
context Integrity
inv: self.ObjectFlow→size()>=0
inv: self.DataStoreNode→size()>=0

[2] A SecurityRole and an audit register must be specified
context Integrity
inv: self.SecurityRole→size()=1
inv: and self.mSecRole.mGAuReg→size()>=1

[3] The Protection Degree must be specified by adding a lower-case letter according to KindIntegrity. The letter "x" must be replaced with (w) for low, (m) for medium or (h) for high.
context Integrity
inv: self.KindIntegrity→size()=1

NONREPUDIATION

Generalization: Element::SecurityRequirement
Description: This establishes the need to avoid the denial of any aspect of the interaction (e.g. message, transaction, and transmission of data). From the business analyst perspective, Non Repudiation represents the need to protect a determined interaction in such a way that it minimizes potential problems (e.g. legal and liability) in relation to any interaction. From the security perspective, this specification implies the generation of at least two security roles and, alternatively, the audit register. This requirement may additionally have an audit register specification. Notation: (Fig. 4f)

Associations: NR-AuditRegister [0..*] ObjectFlow [0..*], SecurityRole [2..*]
Tagged Value: AcDgElementName, ActionDestinationName, AuditRegister, SecurityRoleDestinationName, SecurityRoleSourceName
Constraints:

[1] It can only be specified in the ObjectFlow element of the activity diagram
context NonRepudiation
inv: self.ObjectFlow→size()>=0

[2] An audit requirement can be indicated for this security requirement
context NonRepudiation
inv: self.AuditRegister=True implies self.NR-AuditRegister→size()>=1

[3] It is related at least to two security roles
context NonRepudiation
inv: self.SecurityRole→size()>=2

NR-AUDITREGISTER

Generalization: Element::AuditRegister
Description: Contains the audit specifications related to the NonRepudiation security requirement. No notation.
Associations: NonRepudiation [1..1]
Tagged Value: AcDgElementName, AuditDateReceive, AuditDateSend, AuditTimeReceive, AuditTimeSend, SecurityRoleDestinationName, SecurityRoleSourceName, Transmission
Constraints

[1] It is valid only if at least one NonRepudiation security requirement specification is specified
context NR-AuditRegister
inv: self.NonRepudiation→size()=1

PRIVACY

Generalization: Element::SecurityRequirement
Description: It is related to information protection conditions concerning a determined individual or entity, and limits access to sensitive information by non-authorized parties. From the point of view of the business analyst, the privacy specification implies the non-revelation (confidentiality) and non-storage (anonymity) of the information regarding a determined role. From the security viewpoint, the specification of privacy with confidentiality implies the protection of the information of a role which must not be revealed to third parties. In the case of privacy with anonymity, it implies that information must not be stored either. This fact implies the creation of generic roles which expire together with the work session. Additionally, this requirement can have a specification of the audit register. Notation: (Fig. 4g).
Associations: ActivityPartition [0..*], InterruptibleActivityRegion [0..*], SecurityRole [1..1]
Tagged Value: AcDgElementName, AcDgElementType, AuditRegister, KindPrivacy
Constraints

[1] It can only be specified in ActivityPartition and InterruptibleActivityRegion activity diagram element
context Privacy
inv: self.ActivityPartition→size()>=0
inv: self.InterruptibleActivityRegion→size()>=0

[2] A privacy requirement has one security role specification
context Privacy
inv: self.SecurityRole→size()=1

[3] The Privacy Type must be specified by adding a lower-case letter according to the PrivacyType tagged value. The letter x, must be replaced with (a) for anonymity or (c) for confidentiality.
context Privacy
inv: self.KindPrivacy–>size()=1

[4] We can indicate the audit register and this gives place to a G-AuditRegister- type class related to the SecurityRole
context Privacy
inv: self.AuditRegister=True implies self.G-AuditRegister→size()>=1

SECUREACTIVITY

Generalization: Classifier::Class::Behavior::Activity
Description: A secure activity contains security specifications related to requirements, role identifications, and permissions. No notation.
Associations: SecurityRequirement [1..*]
Tagged Value: None
Constraints

[1] It must be associated with at least one SecurityRequirement
context SecureActivity
inv: self.SecurityRequirement→size()>=1

SECURITYPERMISSION

Generalization: Element (from Kernel)
Description: Contains permission specifications related to an AccessControl specification. A permission specification must

contain details about the objects and operations involved. No notation.
Associations: SecureRole [1..1], SP-AuditRegister [0..*]
Tagged Value: AcDgElementName, AcDgElementType, KindPermission, SecurityRoleName, SourceSecReq
Constraints
[1] It must be associated with a security role specification
context SecurityPermission
inv: self.SecurityRole→size()=1
[2] A SecurityPermission only exists if Access Control has been specified in some of the UML 2.0 –AD elements
context SecurityPermission
inv: self.mSecRole.SourceSecReq="AC" implies self.SecurityPermission→size()>=1
[3] It can be associated with an audit register specification
context SecurityPermission
inv: self.mSecRole.mAccCon.AuditRegister=True implies self.SP-AuditRegister→size()>=1
[4] The KindPermission must be specified for Actions, DataStoreNode and/or ObjectFlow such as Objects and Operations pairs.
context SecurityPermissions
inv: self.Actions–>size()=1 implies (self.KindPermission="Execution" or self.KindPermission="CheckExecution")
inv: self.DatastoreNode–>size()=1 implies (self.KindPermission="Update" or self.KindPermission="Create" or self.KindPermission="Read" or self.KindPermission="Delete")
inv: self.ObjectFlow–>size()=1 implies (self.KindPermission="SendReceive" or self.KindPermission="CheckSendReceive")

SECURITYREQUIREMENT

Generalization: Element (from Kernel)
Description: Abstract class containing security requirements specifications. Each security requirement type must be indicated in some of its subclasses. Notation (Fig. 4a). This represents the basic symbol over which security requirements are specified and has been adopted because it is considered to be a *de facto* standard associated with security.
Associations: SecureActivity [1..1]
Tagged Value: Criticality
Constraints
[1] A security requirement must be associated with a secure activity
context SecurityRequirement
inv: self.SecureActivity –>size()=1
[2] The notation must be completed in the specification subclass for each security requirement. One security requirement type must be used.

SECURITYROLE

Generalization: Classifier::Actor (from UseCases)
Description: Contains a role specification. This role must be obtained from Access Control and/or Privacy specifications. No notation.
Associations: AccessControl [0..*], G-AuditRegister [0..*], NonRepudiation [0..*], Privacy [0..*], SecurityPermission [0..*]
Tagged Value: AcDgElementName, AcDgElementType, KindPrivacy, SecurityRoleName, SourceSecReq
Constraints
[1] It can be related to the following elements defined in our BPSec extension: AccessControl, G-AuditRegister, NonRepudiation, Privacy and SecurityPermission
context SecurityRole
inv: self.AccessControl→size()>=0
inv: self.G-AuditRegister→size()>=0
inv: self.NonRepudiation→size()>=0
inv: self.Privacy→size()>=0
inv: self.SecuriyPermission→size()>=0
[2] A SecurityRole may be created in an AccessControl, AttackHarmDetection, Integrity, NonRepudiation or Privacy specification, or in a combination of these requirements. The permitted values are shown in the tagged values requirementType description.

SP-AUDITREGISTER

Generalization: Element::AuditRegister
Description: Contains the audit specifications related to the access control security requirement which gives rise to SecurityPermission. No notation.
Associations: SecurityPermission [1..1]
Tagged Value: AcDgElementName, AcDgElementType, AuditDate, AuditTime, KindPermission, SecurityRoleName
Constraints
[1] It must be associated with at least one SecurityPermission
context SP-AuditRegister
inv: self.SecurityPermission–>size()=1
[2] An SP-AuditRegister only exists if Access Control has been specified with the audit register in one of the UML 2.0–AD elements
context SP-AuditRegister
inv: self.SecPer.self.mSecRole.self.mAccCon.AudirRegister=True implies self.SP-AuditRegister→size()>=1

Appendix B. TypeBPSec stereotypes descriptions

Within the context of our BPSec extension, it has been necessary to define data types which allow us to express attributes in the new stereotypes that are related to operation permissions, privacy types, protection degrees and requirement types.

The BPSec extension, which is defined as packages (see Fig. 5), allows us to incorporate these data types into BPSec, making their reuse as attributes of the new stereotypes possible. Data types have been inherited from the UML Enumeration abstract class.

For the specification of BPSec types, we will provide a description corresponding to an explanation of the purpose fulfilled by the data type and its meaning, along with a list of each of the possible values for each data type, including an explanation of their meaning. We additionally indicate the names of the stereotypes by using this data type.

REQUIREMENTTYPE

Description: This stereotype contains the values which are accepted in individual or combined security requirement specifications. It is built as a combination of the AC, AD, I, NR and P abbreviations associated with access control, attack harm detection, integrity, non-repudiation and privacy respectively.
Values: Although the valid combinations are presented in a determined order, for example Access control and Privacy, this order is not compulsory. We must thus verify that the combination of requirements is admissible, regardless of the order of appearance of each requirement.
{AC}: Valid abbreviation for Access Control specification.
{AD}: Valid identification for AttackHarm requirement.
{I}: Value associated with an Integrity specification
{NR}: Valid abbreviation for a Non-repudiation specification.
{P}: Valid Identification for Privacy requirement.

{ACAD}: Combination of values associated with Access Control and Attackharm detection specifications

{ACP}: Values associated with the combination of Access Control and Privacy

{ADP}: Combination of values related to Attackharm detection and Privacy

{ACADP}: Combined values of Access Control, Attack and Threats Detection and Privacy specifications

{ACI}: Valid combination for Access Control and Integrity requirements.

{ADI}: Accepted values for the combination of Attackharm detection and Integrity requirements

{ACADI}: Valid combination for Access Control, Attackharm detection and Integrity

{ACNR}: Accepted values for the combined specification of the Access Control and Non-repudiation requirements.

{ADNR}: Valid combination of Attackharm detection and Non-repudiation

{INR}: Accepted values for the combination of Integrity and Non-repudiation requirements

{ACADNR}: Valid combination for Access Control, Attackharm detection and Non-repudiation requirements

{ADINR}: Accepted values for the combination of Attackharm detection, integrity and Non-repudiation

{ACADINR}: Valid combination for the following requirements: Access Control, Attackharm detection, Integrity and Non-repudiation

Used in: The protection degree is used to define the SourceSecReq labeled values used in G-AuditRegister, SecurityPermission and SecurityRole stereotypes

PERMISSIONOPERATION

Definition: This stereotype contains the permitted values for operation permissions. These permissions are associated with the activity diagrams that are within the scope of an Access Control specification. Operation permissions are defined in relation to the object to which access has been restricted. As the UML 2.0-AD elements considered in an access control specification are actions, data warehouses and object flows, the permissions granted for each one of them are different. For actions, we have considered execution permissions and execution permission checking; for warehouses, we have considered read/write/delete/update permissions and finally, for object flows, we have considered send permissions and send permission checking.

Values

{Execution}: This value is valid for the Action class. The specification of this permission (default value) implies that the action can be executed by the security role associated with a specification to which access control has been restricted.

{CheckExecution}: This value is only valid for the Action class. Its specification implies that we must verify the existence of permissions, for example validation of security role, before executing the action over which this value has been specified.

{Create}: This value is only valid for the DataStoreNode class. Its specification implies that the security role can only create new elements in the data warehouse.

[55]: This value is only valid for the DataStoreNode class. A permission of this kind authorizes the security role only to read the information contained in the data warehouse.

{Delete}: This value is only valid for the DataStoreNode class. The specification of this permission implies that the security role can only eliminate elements that exist in the datawarehouse.

{Update}: This must be defined over the DataStoreNode class. Update is assumed as default value and the security role is permitted to carry out all tasks (reading, writing and elimination) over the datawarehouse.

{SendReceive}: This value is only valid for the ObjectFlow class. The specification of SendReceive (default value) implies that the object flow can be sent or received.

{CheckSendReceive}: This value is only valid for the ObjectFlow class. Its specification implies that the existence of permissions must be verified. For instance, the security role involved in sending or receiving must be validated before sending or accepting in order to receive the object flow.

Used in: The operation permission is used to define the KindPermission labeled values used in the SecurityPermission and SP-AuditRegister stereotypes.

PROTECTIONDEGREE

Description: This stereotype contains a classification of the protection required in an Integrity specification that has been indicated over a data warehouse. This classification is divided into high, medium and low protection.

Values

{h}: This value indicates that High Integrity has been specified. This implies (i) the verification of use permissions, (ii) information support and (iii) the register of events for further audit; all these tasks are associated with the data warehouse over which Integrity has been specified.

{m}: his implies the specification of medium Integrity. The tasks associated with this value are (ii) information support and (iii) register of events for further audit.

{w}: This value expresses the need to protect a data warehouse at a lower degree. We have decided to use the letter w to identify lower degree owing to the similarity between the first letters of the words Integrity and Lower.

Used in: The degree of protection is used to define the KindIntegrity labeled value used in the Integrity stereotype.

PRIVACYTYPE

Description: This stereotype contains the values associated with a Privacy specification. These values allow us to specialize the Privacy specification. The permitted values are anonymity and confidentiality. One value excludes the other.

Values

{a}: This value indicates that Privacy has been specified with anonymity. This is the default value in general terms, and it is interpreted as the need not to reveal information about the security role associated with this specification, along with avoiding its storage.

{c}: This implies the specification of Privacy with confidentiality. This implies that information concerning the security role associated with this specification cannot be revealed.

Used in: This type of privacy is used to define the KindPrivacy labeled value that is used in Privacy and SecurityRole stereotypes.

Appendix C. Tagged value descriptions

In this Appendix, we present (alphabetically) a detailed description of each of the tagged values defined in the stereotypes of which the BPSec extension is formed. This will be done by (i) providing a description corresponding to an explanation of the purpose of the tagged value, (ii) defining the *type* that identifies the characteristics of

the values that can be associated with the tagged value, for example string, integrity, etc., and finally (iii) indicating in which stereotype it *is used*.

AcDgElementName: Contains the name of the activity diagram element. It can have one of the following values: ActivityPartition, InterruptibleActivityRegion, Action, DataStoreNode or ObjectFlow. *Type*: Classifier::DataType::PrimitiveType::String. *Used in*: AccessControl, AttackHarmDetection, G-AuditRegister, Integrity, NonRepudiation, NR-AuditRegister, Privacy, SecurityPermission, SecurityRole, and SP-AuditRegister.

AcDgElementType: Contains the name of the type of the activity diagram element (ActivityPartition, InterruptibleActivityRegion, Action, DataStoreNode or ObjectFlow) over which a security requirement has been specified. *Type*: Classifier::DataType::PrimitiveType::String. *Used in*: AccessControl, AttackHarmDetection, G-AuditRegister, Integrity, NonRepudiation, Privacy, SecurityPermission, SecurityRole, and SP-AuditRegister.

AuditDate: Contains the date on which an event related to an audit requirement specification over «AccessControl», «Privacy», «AttackHarmDetection» or «Integrity» security requirements is registered. *Type:* Classifier::DataType::PrimitiveType::Integer. *Used in*: G-AuditRegister and SP-AuditRegister.

AuditDateSend: Contains the date on which an event related to the sending of an ObjectFlow in a specification of the «NonRepudiation» security requirement is registered. *Type:* Classifier::DataType::PrimitiveType::Integer. *Used in*: NR-AuditRegister.

AuditRegister: Contains a value that indicates whether the security audit register was specified. *Type*: Classifier::DataType::PrimitiveType::Boolean. *Used in*: AccessControl, NonRepudiation, and Privacy

AuditTime: Contains the time at which an event related to the audit requirement specification over «AccessControl», «Privacy», «AttackHarmDetection» or «Integrity» security requirements took place. *Type:* Classifier::DataType::PrimitiveType::Integer. *Used in*: G-AuditRegister and SP-AuditRegister.

AuditTimeReceive: Contains the time at which an event related to the receiving of an ObjectFlow in a specification of the «NonRepudiation» security requirement is registered. *Type*: Classifier::DataType::PrimitiveType::Integer. *Used in*: NR-AuditRegister.

AuditTimeSend: Contains the time at which an event related to the sending of an ObjectFlow in a specification of the «NonRepudiation» security requirement is registered. *Type*: Classifier::DataType::PrimitiveType::Integer. *Used in*: NR-AuditRegister.

Criticality: Contains an identification of the criticality required for all security requirement specifications. *Type*: Classifier::DataType::Enumeration. *Used in*: SecurityRequirement.

KindIntegrity: Specifies the type of integrity (high, medium, lower) indicated in the «Integrity» security requirement *Type*: Classifier::DataType::Enumeration::ProtectionDegree. *Used in*: Integrity.

KindPermission: Contains the type of operation that it is possible to carry out in an object permission pair that is registered when an audit register has been specified over an «AccessControl» security requirement. *Type*: Classifier::DataType::Enumeration::PermissionOperation. *Used in*: SecurityPermission and SP-AuditRegister.

KindPrivacy: Specifies the type of privacy required when a «Privacy» security requirement has been indicated. *Type:* Classifier::DataType::Enumeration::PrivacyType. *Used in*: Privacy and SecurityRole.

SecurityRoleName: Contains the name of the security role which originated from a «Privacy» or «AccessControl» specification. *Type*: Classifier::DataType::PrimitiveType::String. *Used in*: G-AuditRegister, SecurityPermission, SecurityRole and SP-AuditRegister.

SecurityRoleDestinationName: Contains the name of the security role that gives destination to an object flow generated from a «NonRepudiation» specification. *Type*: Classifier::DataType::PrimitiveType::String. *Used in*: NonRepudiation and NR-AuditRegister.

SecurityRoleSourceName: Contains the name of the security role that gives origin to a data flow generated from a «NonRepudiation» specification. *Type*: Classifier::DataType::PrimitiveType::String. *Used in*: NonRepudiation and NR-AuditRegister.

SourceSecReq: Contains the identification of the security requirement from which secure role, security permissions or audit register are generated. *Type*: Classifier::DataType::Enumeration::RequirementType. *Used in*: G-AuditRegister, SecurityPermission and SecurityRole.

Transmission: Contains an indication related to the success or failure of the transmission of an ObjectFlow over which a «NonRepudiation» security requirement with audit register has been specified. *Type*: Classifier::DataType::PrimitiveType::Boolean. *Used in*: NR-AuditRegister.

References

[1] R.S. Aguilar-Savén, Business process modelling: review and framework, International Journal of Production Economics 90 (2) (2004) 129–149.
[2] V. Atluri, Security for workflow systems, Information Security Technical Report 6 (2) (2001) 59–68.
[3] M. Backes, B. Pfitzmann, M. Waider, Security in business process engineering, International Conference on Business Process Management (BPM), Eindhoven, Netherlands, 2003.
[4] 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.
[5] J. Bézivin, In search of a basic principle for model driven engineering, UPGRADE, European Journal for the Informatics Professional V (2) (2004) 21–24.
[6] J. Bézivin, On the unification power of models, Software and Systems Modeling 4 (2) (2005) 171–188.
[7] H.K. Bhargava, D.J. Power, D. Sun, Progress in web-based decision support technologies, Decision Support Systems 43 (2007) 1083–1095.
[8] BPMN, Business Process Modeling Notation Specification, OMG Final Adopted Specification, dtc/06-02-01, 2006.
[9] R.M. Davison, M.G. Martinsons, N. Kock, Principles of canonical action research, Information Systems Journal 14 (2004) 65–86.
[10] Rachna Dhamija, J.D. Tygar, Marti Hearst, Why phishing works, SIGCHI Conference on Human Factors in Computing Systems, ACM, Montreal, Quebec, Canada, 2006.
[11] M.H. Diallo, J. Romero-Mariona, S.E. Sim, T.A. Alspaugh, D.J. Richardson, A comparative evaluation of three approaches to specifying security requirements, 12th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), Luxembourg, 2006.
[12] A. Dutta, S. Heda, Information systems architecture to support managed care business processes, Decision Support Systems 30 (2000) 217–225.
[13] H.-E. Eriksson, M. Penker, Business Modeling with UML.RN, , 2001.
[14] D. Firesmith, Engineering security requirements, Journal of Object Technology 2 (1) (January-February 2003) 53–68.
[15] D. Firesmith, Specifying reusable security requirements, Journal of Object Technology 3 (1) (January–February 2004) 61–75.
[16] L. Fuentes, A. Vallecillo, An introduction to UML profiles, UPGRADE, The European Journal for the Informatics Professional 2 (2) (2004) 6–13.
[17] A. Gregoriades, B. Karakostasb, Unifying business objects and system dynamics as a paradigm for developing decision support systems, Decision Support Systems 37 (2004) 307–311.
[18] C.B. Haley, R.C. Laney, B. Nuseibeh, Deriving security requirements from crosscutting threat descriptions, 3rd International Conference on Aspect-Oriented Software Development (AOSD), Lancaster, UK, 2004.
[19] P. Harmon, The OMG's model driven architecture and BPM, Business Process Trends 2 (5) (2004).
[20] G. Herrmann, G. Pernul, Towards security semantics in workflow management, Thirty-First Annual Hawaii International Conference on System Sciences, Kohala Coast, Hawaii, USA, 1998.
[21] G. Herrmann, G. Pernul, Viewing business process security from different perspectives, 11th International Bled Electronic Commerce Conference, Slovenia, 1998.
[22] P. Herrmann, G. Herrmann, Security requirement analysis of business processes, Electronic Commerce Research 6 (3–4) (2006) 305–335.
[23] S. Houmb, S. Islam, E. Knauss, J. Jürjens, K. Schneider, Eliciting Security Requirements and Tracing them to Design: An Integration of Common Criteria,

Heuristics, and Umlsec, Requirements Engineering (Special Issue — Security Requirements Engineering), 2010(15), 2010, p. 63.
[24] S.-M. Huang, D.C. Yen, Y.-C. Hung, Y.-J. Zhou, J.-S. Hua, A business process gap detecting mechanism between information system process flow and internal control flow. Decision Support Systems 47 (2009) 436–454.
[25] I. Jacobson, G. Booch, J. Rumbaugh, The Unified Software Development Process, 1999, p. 463.
[26] J. Jürjens, Developing Secure Systems with UMLsec — From Business Processes to Implementation in VIS 2001, Vieweg-Verlag, Kiel (Germany), 2001.
[27] J. Jürjens, Model-based security engineering with UML, in: A. Aldini, R. Gorrieri, F. Martinelli (Eds.), Foundations of Security Analysis and Design III, Springer Verlag, 2005.
[28] J. Jürjens, Secure Systems Development with UML, ed. Springer Verlag. 2005.
[29] A. Kalnins, J. Barzdins, E. Celms, UML business modeling profile, Thirteenth International Conference on Information Systems Development, Advances in Theory, Practice and Education, Vilnius, Lithuania, 2004.
[30] B. Kitchenham, Guideline for Performing Systematic Literature Reviews in Software Engineering. Version 2.3. RN, 2007.
[31] B. Korherr, B. List, Extending the UML 2 activity diagram with business process goals and performance measures and the mapping to BPEL, 2nd International Workshop on Best Practices of UML (BP-UML) at ER Conference, Tucson, Arizona, USA, 2006.
[32] B.W. Lampson, Computer security in the real world, IEEE Computer 37 (6) (2004) 37–46.
[33] B. List, B. Korherr, An evaluation of conceptual business process modelling languages, ACM Symposium on Applied Computing (SAC), Dijon, France, 2006.
[34] B. List, B. Korherr, A UML 2 profile for business process modelling, 1st International Workshop on Best Practices of UML (BP-UML) at ER-Conference, Klagenfurt, Austria, 2005.
[35] T. Lodderstedt, D. Basin, J. Doser, SecureUML: a UML-based modeling language for model-driven security, UML, 5th International Conference, Dresden, Germany, 2002.
[36] A. Lonjon, Business Process Modeling and Standardization, BPTrends, 2004, http://www.bptrends.com/.
[37] A. Maña, J.A. Montenegro, C. Rudolph, J.L. Vivas, A business process-driven approach to security engineering, 14th International Workshop on Database and Expert Systems Applications (DEXA), Prague, Czech Republic, 2003.
[38] N.R. Mead, Experiences in eliciting security requirements, CrossTalk: The Journal of Defense Software Engineering, Vol. 19(12), 2006.
[39] D. Mellado, C. Blanco, L.E. Sánchez, E. Fernández-Medina, A systematic review of security requirements engineering, Computer Standards & Interfaces 32 (4) (2010) 153–165.
[40] T. Mens, P. Van Gorp, A taxonomy of model transformation, Electronic Notes in Theoretical Computer Science 152 (2006) 125–142.
[41] A. Oberweis, An integrated approach for the specification of processes and related complex structured objects in business applications, Decision Support Systems 17 (1996) 31–53.
[42] Object Management Group, MDA Guide Version 1.0.1, 2003.
[43] Object Management Group, Unified Modeling Language Specification. Version 1.5, 2003.
[44] Object Management Group, Unified Modeling Language: Superstructure Version 2.1.1 (formal/2007-02-05), 2007.
[45] V. Portougal, D. Sundaram, Business Processes: Operational Solutions for SAP Implementation by 2005, 2005, p. 329.
[46] QVT, Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, ed. OMG Adopted Specification ptc/05-11-01. 2005. 204.
[47] A. Rodríguez, E. Fernández-Medina, M. Piattini, Analysis-level classes from secure business processes through models transformations, 4th International Conference on Trust, Privacy and Security in Digital Business (TrustBus), Regensburg, Germany, 2007.
[48] A. Rodríguez, E. Fernández-Medina, M. Piattini, CIM to PIM transformation: a reality, IFIP International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS), Beijing, China, 2007.
[49] A. Rodríguez, E. Fernández-Medina, M. Piattini, M-BPSec: a method for security requirement elicitation from a UML 2.0 business process specification, 3rd International Workshop on Foundations and Practices of UML, Auckland, New Zealand, 2007.
[50] A. Rodríguez, E. Fernández-Medina, M. Piattini, Towards a UML 2.0 extension for the modeling of security requirements in business processes, 3rd International Conference on Trust, Privacy and Security in Digital Business (TrustBus), Krakow-Poland, 2006.
[51] A. Rodríguez, E. Fernández-Medina, M. Piattini, Towards CIM to PIM transformation: from secure business processes defined by BPMN to use cases, 5th International Conference on Business Process Management (BPM), Brisbane, Australia, 2007.
[52] A. Rodriguez, I. García-Rodríguez de Guzmán, E. Fernández-Medina, M. Piattini, Semi-formal transformation of secure business processes into analysis class and use case models: an MDA approach, Information & Software Technology 52 (9) (2010) 945–971.
[53] A.W. Röhm, G. Pernul, G. Herrmann, Modelling secure and fair electronic commerce, 14th Annual Computer Security Applications Conference, Scottsdale, Arizona, 1998.
[54] G. Sindre, Mal-activity diagrams for capturing attacks on business processes, Requirements Engineering: Foundation for Software Quality, 13th International Working Conference, REFSQ 2007, Trondheim, Norway, 2007.
[55] L. Srinivasan, J. Treadwell, An Overview of Service-Oriented Architecture, Web Services and Grid Computing.RN, http://h71028.www7.hp.com/ERC/downloads/SOA-Grid-HP-WhitePaper.pdf, 2005.
[56] V. Stefanov, B. List, B. Korherr, Extending UML 2 activity diagrams with business intelligence objects, 7th International Conference on Data Warehousing and Knowledge Discovery (DaWaK2005), Copenhagen, Denmark, 2005.
[57] V. Vitolins, A. Kalnins, Semantics of UML 2.0 activity diagram for business modeling by means of virtual machine, Ninth IEEE International Enterprise Distributed Object Computing Conference (EDOC), Enschede, The Netherlands, 2005.
[58] J.L. Vivas, J.A. Montenegro, J. Lopez, Towards a business process-driven framework for security engineering with the UML, in: Boyd Colin, Mao Wenbo (Eds.), Information Security: 6th International Conference, ISC, Bristol, U.K., 2003.
[59] A. Zuccato, Holistic security requirement engineering for electronic commerce, Computers & Security 23 (1) (2004) 63–76.

Alfonso Rodríguez is MBA from the Universidad del Bio-Bio (Chile), and he is PhD in Computer Science from the University of Castilla-La Mancha (Spain). He is Associate Professor at the Computer Science and Auditing Department of the Bio Bio University (Chillán, Chile). His research activities are security in business process and information systems.

Eduardo Fernández-Medina holds a PhD. and an MSc. in Computer Science from the University of Sevilla. His research activity is in the field of security in information systems, and particularly in security in business processes, databases, datawarehouses, and web services. Fernández-Medina is co-editor of several books and chapter books on these subjects, and has papers in international conferences (BPM, UML, ER, ESORICS, TRUSTBUS, etc.). He is author of several manuscripts journals (Decision Support Systems, Information Systems, ACM Sigmod Record, Information Software Technology, Computers & Security, Computer Standards and Interfaces, etc.). He leads the GSyA research group of the Department of Computer Science at the University of Castilla-La Mancha, in Ciudad Real, Spain. He belongs to various professional and research associations (ATI, AEC, AENOR, IFIP WG11.3, etc.).

Juan Trujillo received a Ph.D. in Computer Science from the University of Alicante (Spain) in 2001. His research interests include database modeling, the conceptual design of data warehouses, MD databases, OLAP, as well as object-oriented analysis and design with UML. With papers published in international conferences and journals such as ER, UML, ADBIS, CAiSE, WAIM, Journal of Database Management (JDM) and IEEE Computer, Trujillo has served as a Program Committee member of several workshops and conferences such as ER, DOLAP, DSS, and SCI and has also spent some time as a reviewer for several journals such as JDM, KAIS, ISOFT and JODS.

Mario Piattini has an MSc and a PhD in Computer Science from the Polytechnic University of Madrid. He is a Certified Information System Auditor from the ISACA (Information System Audit and Control Association). The author of several books and papers on databases, software engineering and information systems, Piattini leads the ALARCOS research group of the Department of Computer Science at the University of Castilla–La Mancha. His research interests are: advanced database design, database quality, software metrics, object- oriented metrics and software maintenance.