Refinement-Driven Role-Based Enterprise Workflow Modeling Considering Access Control

Yevheniia Yehorova[1,*,†], Marina Waldén[1,*,†]

[1]Åbo Akademi University, Tuomiokirkontori 3, 20500 Turku, Finland

Abstract

Modeling enterprise workflows has a direct impact on business process management, ensuring the efficient use of resources. Business processes span multiple organizational or system roles that perform tasks to achieve company goals. In this paper, we continue to extend an approach to role-based workflow modeling, based on a refinement that enables stepwise development of workflows from abstract models to detailed implementations. The main idea of our approach is to iteratively refine the model by adding new roles, resources, and sensitivity levels, while maintaining logical consistency and control over access policies. We used an abstract multidimensional matrix for the theoretical visualization of the refinement and the UPPAAL tool for developing and validating the models. Our approach is visualised using a healthcare case study, where hospital workflows are implemented.

Keywords

Role-Based Modeling, Stepwise Refinement, Workflow Modeling, Refinement-Based Modeling, Parallel Process Modeling, Hospital Workflow

1. Introduction

In the modern world, effective management of business processes, particularly each workflow, directly affects the success of the organization as a whole. A workflow is a sequence of tasks, actions, and operations performed by roles within an enterprise to achieve set goals [1, 2]. In this context, workflow modeling plays a main role. It enables the analysis of the sequence of tasks performed, the distribution of roles (human, system, artificial intelligence), and their interaction [1]. With the growing complexity of computing processes, increasing data volumes, and the need to quickly respond to changes, there is a growing need for dynamic scaling, adaptation, and assurance of secure access to resources in workflows. Integrating access control mechanisms into models allows not only to manage tasks, but also to establish who can perform operations in the process and under what conditions.

Traditional methods of manual analysis are no longer sufficient for organizations due to the huge volume of data and tasks. Modeling of workflows have long been essential tools for studying the behavior of people, systems, and processes. This also applies to predicting performance and identifying bottlenecks. However, modeling workflows in actual conditions faces some problems. Modern approaches to business process management should take into account not only the need for formal modeling but also flexibility, adaptability, and the ability to utilize the latest technologies. The issue of access control within processes becomes especially critical, since the execution of operations is often associated with the processing of confidential information and interaction between roles with different levels of responsibility. Access control mechanisms in process models allow enterprises to improve the manageability and transparency of resource use and to minimize the risk of achieving strategic goals.

Designing reliable and manageable workflow systems requires a foundation in access control modeling, layered architecture, and granularity. This paper uses a combination of several fundamental approaches. The transition from an abstract model to specific access policies is carried out in a stepwise

PoEM2025: Companion Proceedings of the 18th IFIP Working Conference on the Practice of Enterprise Modeling: PoEM Forum, Doctoral Consortium, Business Case and Tool Forum, Workshops, December 3-5, 2025, Geneva, Switzerland

*Corresponding author.

†These authors contributed equally.

yevheniia.yehorova@abo.fi (Y. Yehorova); marina.walden@abo.fi (M. Waldén)

0009-0008-5526-4448 (Y. Yehorova); 0000-0001-8703-3179 (M. Waldén)

manner using refinement [3, 4]. This is the process of making the model more precise while maintaining its correctness. Finally, the model is organized as a multi-layered architecture, where each logical representation of the system is structured in layers [1, 5]. Layers allow behavioral and organizational aspects to be separated, ensuring the modularity and manageability of the model.

In our previously proposed enterprise workflow modeling approach [1], the main focus was on the layered structuring of roles, modeling the logic of interactions and separating responsibilities in the system. The Role-Based Access Control (RBAC) was used primarily as a mechanism for structuring internal and external entities. All roles interacted in a single unified model of multi-layered workflow architecture. Aspects related to issues of data access control, resource operations, and refinement of the resources themselves were not considered at that stage.

In this paper, we extend the previously proposed approach by introducing access control components based on a hybrid RBAC and MAC approach. In this approach, roles not only structure the workflow but also have specific permissions for operations (read, execute) on defined resources. Resources, in turn, are classified by sensitivity levels. This extension enables providing the model with strict security policies, especially when it comes to healthcare, space, and military industries [6].

We show the advantages of our method through a healthcare case study where medical staff, patients, and systems represent the roles. The approach is visualized using UPPAAL, and the models are available at the *link*.

2. Related Work

Workflow modeling with access control in mind is rapidly evolving, integrating BPM, access control, and formal methods [7, 8]. Role-based access control models (RBAC) [9] remain the foundation of most approaches to rights management in BPM systems; however, they do not account for the gradual granularity of business logic. Mandatory access control (MAC) [10] provides strict confidentiality but is not adapted to dynamic processes. Current research considers separate approaches, formal verification of BPM models, or combined hybrid RBAC/MAC [11] mechanisms. The gap lies in the fact that most approaches provide an abstract access policy, or it is implemented as an external component separate from the modeling process. The proposed approach fills this gap by formalizing access control as part of the stepwise construction of the model, which ensures its logical integrity.

3. Access Control

Access control is the primary mechanism for ensuring information security. Its task is to limit the actions of subjects (user, system) over objects (data, resources) in working to achieve the goals of the enterprise. An effective access control system allows the prevention of unauthorized access and minimizes the risks of internal abuse. The choice of access control model critically affects the flexibility of management. The most used access control models are role-based and mandatory. In the context of large enterprise companies, role-based access control (RBAC) is most often used. Mandatory access control (MAC) is additionally used in the military, government, and healthcare sectors, where security is a high priority. Given that our case study is within the healthcare area, it is useful to consider both of these access control models.

3.1. Role-Based Access Control

Role-based access control (RBAC) is a standard mechanism for managing the access of users and systems based on their role in an organization [7, 8, 9]. It is used in a variety of settings, from enterprises to mission-critical systems. RBAC is an approach in which roles define user permissions. Each user could be assigned one or more roles, and each role contains a set of permissions to perform actions or operations on resources. It helps to avoid errors in assigning permissions to each user and simplifies access management [9]. The main components of RBAC are following:

- *Users or subjects* are people, systems, or processes that perform actions, operations, or tasks in an organization.
- *Roles* are abstract sets or categories for grouping entities (users or subjects) with similar responsibilities and permissions. The permissions are required to perform specific tasks or actions. A role is often associated with a type of activity and defines a level of access [10].
- *Permissions* are attributes that define which operations a role can perform on resources (e.g., read, write, execute). The allowed operations are defined globally for the entire role without specifics.
- *Objects or resources* are data, files, systems, and services that require access to use the resource within the organization's workflow.

3.2. Mandatory Access Control

Mandatory Access Control (MAC) is an access control model where permissions are defined at the system level rather than at the user level. Restrictions are enforced by access policies and security labels. MAC does not use roles but strictly controls access through security levels.

Security level is a strictly defined characteristic of a subject that determines the category of data to which the user has access. The security level is strictly fixed in the system and consists of two key components: *subject classification* - the user's clearance level, *object classification* - the security level of the resource (class). Each resource is assigned a class, and each subject is assigned a clearance. Access is allowed only if the subject's clearance level is greater than or equal to the object's sensitivity class [10] otherwise it is denied.

$$clearance(subject) \geq class(object) \tag{1}$$

3.3. Combination of RBAC and MAC

MAC and RBAC are role-based and label-based access control models, but each has its characteristics and mechanisms for assigning rights, advantages, and limitations [10]. However, their strict applicability raises the question: What if these key characteristics, operating principles, and application areas of these models are combined? While MAC provides strict controls based on security levels, RBAC is more business process-oriented and provides controls based on roles. In both models, the subject is the users. Their permissions in MAC depend on their clearance levels, and in RBAC, on their roles.

Based on this part of the analysis, we can conclude that in both cases, the user must have a label — a clearance level or role. In addition, MAC also protects objects that have sensitivity levels (classes). Here, we combine the two described models and obtain a hybrid flexible access model, where the role determines access, taking into account the type of sensitivity of objects and the level of access of each role.

As a result, our approach is based on the following:

1. Each user has a role.
2. The access level is assigned to the role rather than to an individual user.
3. Users may be granted different clearances despite having the same role.
4. Each object (or resource) has a sensitivity level (class).
5. Even if users have the same role and clearance, access to certain information may still be restricted based on resource sensitivity.

This is especially important in domains with critical information, such as healthcare, finance, and government systems. Based on this, access control matrices can be built by considering roles, resources, and sensitivity.

3.4. Abstract Multidimensional Access Matrix

The simplest and most formal structure for representing access control is the Access Control Matrix (ACM). The classic ACM is a two-dimensional table whose rows correspond to system subjects and

columns to objects. The matrix cells contain a set of permitted operations that a particular topic can perform on the corresponding object (e.g., read, write, execute) [12]. A simple two-dimensional matrix is insufficient for representing complex access policies and does not scale well, especially in distributed and context-dependent systems. Such systems introduce additional parameters - roles, security levels, execution context - each of which adds a new dimension to the original matrix. In our approach, new dimensions are not defined statically, but are introduced stepwise through refinement of the model at each level in multidimensional access matrices.

4. Stepwise Development

The role-based workflow model is best modeled as a parallel system. Such a system comprises multiple entities, including both people and systems or services, that interact simultaneously, independently, and asynchronously. Given the high complexity of such systems, stepwise development is beneficial for their formalization and synthesis. The proposed approach aims to gradually refine and verify the model, improve manageability, and facilitate a stepwise transition by layers from an abstract representation of the workflow to a specific model suitable for simulation of the process, taking into account the synthesis of roles into a single, common workflow. The abstract model can be visualised using an abstract access matrix with entities and their coordination. In the refinement of the model, this matrix supports the synchronized development of roles, operations, resources, and sensitivity levels in a single model. This approach provides an increased accuracy in access control and resource classification, as well as detailed accounting of sensitivity levels, preserving the behaviour of the model. In addition, it enables the implementation of the principle of least privilege, according to which a subject is granted only the minimum rights necessary to perform their direct tasks [13]. This reduces the risks of errors and internal issues.

4.1. Refinement

Refinement [14], based on Refinement Calculus [15], refers to an iterative process of improving an existing system. The stepwise refinement method [14] consists of a series of correctness-preserving transformations, each of which adds new functional or structural elements to the system without modifying the already proven behaviour stated by the invariant and the correctness of the previous model. In parallel systems, behavior is modeled with events of the form of guarded commands. An event can be executed if its guard evaluates to true. Parallel execution of two events means that they can be executed in any order, achieving the same result.

At each refinement step, superposition refinement [16] is applied, where new variables and new events are added together with an invariant to introduce new features to the system without changing the old behaviour. To prove that a system is a correct superposition refinement of another, more abstract system, the following proof obligations must be satisfied [3, 14]:

1. New variables can be introduced that satisfy the new invariant.
2. Assignments to the new variables that preserve the new invariant can be introduced. Moreover, the guards of the refined events can be strengthened.
3. Each new event in the refined specification should only assign to the new variables and should preserve the new invariant.
4. The new events in the refined specification should not take over the execution, but their guards should eventually become false. Hence, it is ensured that the refined system still allows the old behaviour.
5. Whenever an event in the abstract specification is enabled, then either the corresponding refined event or one of the new events should be enabled.

By discharging all these proof obligations for each refinement step in the system development, we have proven the correctness of the system concerning its specification. This means that the refined

system should satisfy its invariant concerning the new behaviour, preserve the behaviour of the abstract system, and not introduce deadlocks, nor infinite loops that could suppress the old behaviour.

Stepwise superposition refinement can be seen as a layered development, where each refinement step forms a separate layer that adds details on top of the abstract model [5]. This allows for localization of changes, increased modularity of the model, and simplification of its analysis.

4.2. Refinement of model components

Role refinement has been discussed in detail earlier [1], where roles were classified by belonging to the organization (internal/external) and by the task performer (human/system). External roles were refined by adding properties, while internal roles were refined through task decomposition. These classifications directly influence the subsequent selection of resources and sensitivity levels that we are interested in here in this work. Not only are roles specified here, but also resources, classes, and clearances, which together lead to a specification of operations and their admissibility.

Our approach allows us to form the model of access control as a matrix in order to visualize the overall behavior of the workflow. This paper focuses only on internal roles, since they implement the actions that make up the structure and sequence of steps in the workflow. Refinement is carried out in layers from the abstract definition of roles to more specific ones, depending on the tasks performed. Stepwise development of internal roles allows us to refine the workflow structure and to ensure the correct inheritance of access rights. Clarifying roles at the initial stages ensures the level of importance for compliance with the principles of least privilege (PoLP) [13, 17, 18]. PoLP is a fundamental principle of information security, according to which a subject is granted only those rights that are necessary to perform their direct functions. This reduces the risk of internal threats and accidental security breaches.

4.2.1. Refinement of Resources and Security Levels

Resources in the context of workflow are the objects on which operations are performed. They can be data, systems, services, or any other entities that roles interact with. Like roles, resources can be developed with superposition refinement, which is critical for ensuring the consistency of the access model.

In this paper, resource refinement is implemented as a stepwise refinement of an abstract resource into more specific ones. For example, resource "report" can be refined to be of attributes "financial" and "analytical". In addition, new resources can also be introduced in a refinement step. For instance, if at the abstract level the resource was represented only as a data source, then at the refinement step it can be introduced as a separate entity with new operations (e.g., execute) and included in the model as a new resource with access rights. This is considered as superposition and requires consistency with other dimensions (roles and security levels) of the model. Such an expansion of the resource set is driven by the functional requirements of the refined roles, which may require additional services or data to perform tasks. Therefore, resource refinement cannot be considered in isolation. It must be coordinated with other dimensions of the model.

New resources require correct classification by security level (sensitivity), as well as assignment of appropriate permitted operations within the refined access matrix. Sensitivity is a dimension related to access policy. The security level is stepwise refined into specific classes (e.g., sensitive is refined into confidential and secret). To ensure the correctness of the model and the consistency of all dimensions of the access matrix, the refined entities should satisfy the proof obligations in Section 4.1. Additionally, we introduced model development rules (see Fig. 1), which give guidelines for the development process. The combination of model development rules and proof obligations facilitates us to guarantee the correctness of the developed model. All entity refinements are reflected in a multi-dimensional access matrix. The result of stepwise development is a structured, formally refined model that is applicable both for simulating its behavior and for automatic synthesis of access policies. This is especially useful for systems where multiple roles perform actions on the same objects, but with different access rights.

Roles	R1	New sub-roles can inherit some permissions, but not have more permissions than the parent role
	R2	If the original role had rights to certain resources, the set of qualified roles must provide at least the same coverage
	R3	Each qualified role is granted access only to those resources that are necessary to perform its specific tasks.
	R4	Refining roles may require a revision of the associated resources, allowed operations, and security levels
Resources	O1	Resources can be refined to new sub-resources depending on functional necessity for performing the tasks of the roles.
	O2	New resources can be added depending on the functional necessity for performing tasks, where no existing resources are available to cover the new tasks of sub-roles.
	O3	New resources must be consistent with the refined roles and security levels. New resources may require new Strengthened security levels.
Security levels	S1	A security level can be strengthened by adding sub-levels of security, depending on the previously allowed access. Access to abstract data is not a guarantee of access to its more sensitive parts.
	S2	Refining security levels depends on the role and resource refinements.

Figure 1: Workflow model development rules

4.3. UPPAAL

In order for our approach to be more feasible, we need tool support. We use the UPPAAL tool [19] to model not only the operations and the relationships of the roles in the system, but also resources and access to them. UPPAAL is a model-checking tool for the modelling, verification, and validation of real-time systems [19]. A system developed within this tool consists of one or several processes, composed in parallel, and is modeled as networks of timed automata. In this paper, channels are used for binary synchronisation. With UPPAAL, the system can be modeled with a number of refinement steps, and the correctness of the system can be proved. Moreover, both reachability and safety properties can be verified. The UPPAAL simulator is used for the imitation of the general workflow. It shows all instantiated automata and active locations of the current state and allows the user to see every step of every template at the same time.

5. Our Approach - Refinement-Driven Role-Based Modeling

We propose an approach to role-based enterprise workflow modeling driven by refinement and access control. The scenarios take into account several aspects at once - role structure, resource hierarchy, and levels of sensitivity and access. We build on our previous work [2], where a multi-layer role-based workflow was presented, focused on dividing participants into external and internal roles as well as on their interaction. Our approach on refinement-driven role-based modelling involves the following: *adding attribute clearance to roles by MAC; classifying resources by sensitivity; introducing a multidimensional access control matrix that links roles, objects, role clearance levels (clearance), and resource classes (class); stepwise refinement of the model, including role and resource type, and access levels; modeling refined states and transitions in UPPAAL, taking into account the hybrid access model; verification of the critical paths of the model.*

Our approach contains four modeling stages. *At the first stage*, an internal role is defined, for which both a workflow model and an access control model will be developed. This article focuses on a hybrid access control model. Therefore, in addition to the role itself, it is necessary to determine the set of resources that are necessary to perform functional tasks and the minimum set of sensitivity levels that will limit access to confidential information.

The second stage involves stepwise construction of an abstract multidimensional access matrix (see Fig. 2). This structuring is the basis for unified access analysis. The matrix is a four-dimensional tensor with axes for roles, objects, role clearance, and object class. Each point in this multidimensional space represents a specific combination of a role interacting with a specific object, within a specific clearance level and sensitivity class. The abstract multidimensional access matrix can be formally represented as:

$$Opr : (R \times cl(R)) \times (Obj \times class(Obj)) \rightarrow Bool \tag{2}$$

where

- R – multiple roles (e.g., hospital staff);
- $cl(R)$ – clearances of roles (e.g., low, high, restricted);
- Obj – multiple objects or resources (data, systems);
- $class(Obj)$ – classes of objects (e.g., public, confidential);
- Opr – the set of possible operations assigned to each admissible configuration.

The access conditions are checked for each configuration. According to the access rule, an operation is allowed if and only if the clearance of the role is higher than or equal to the class of the object. Operations are formed as a result of applying the access control policy to the combination of role, object, clearance, and class. Thus, the matrix describes not only the permitted combinations but also which operations are explicitly allowed in each context.

		Resources	**Objy**	**Obj(y+1)**	...	**Objm**
		Class	**class s1**	**class s2**	...	**class sl**
Roles	*Clearance*					
Rx	**cl c1**		c1(x)>=s1(y) ∧ Opr x.y	c1(x)>=s2(y+1) ∧ Opr x.(y+1)	...	c1(x)>=sl(m) ∧ Opr x.(m)
	cl c2		c2(x)>=s1(y) ∧ Opr x.y	c2(x)>=s2(y+1) ∧ Opr x.(y+1)	...	c2(x)>=sl(m) ∧ Opr x.(m)
R(x+1)	**cl c3**		c3(x+1)>=s1(y) ∧ Opr (x+1).y	c3(x+1)>=s2(y+1) ∧ Opr (x+1).(y+1)	...	c3(x+1)>=sl(m) ∧ Opr (x+1).(m)
...	**...**		...	...	...	...
Rn	**cl ck**		ck(n)>=s1(y) ∧ Opr n.y	ck(n)>=s2(y+1) ∧ Opr n.(y+1)	...	ck(n)>=sl(m) ∧ Opr n.(m)

Figure 2: Abstract Multidimensional Access Matrix

The main idea of our approach is to iteratively refine the model by adding new roles, resources, and security levels, while maintaining logical consistency and control over access policies. Relationships between roles in the modeling process are implemented through synchronization, which allows interactions between roles within a single workflow. After constructing the abstract matrix, the first layer of the model is developed and visualized in a verification tool (e.g., UPPAAL). Roles are given as templates, resources as states, and security levels as guards.

At the third stage, all components of the model are refined into more specific ones (Section 4): roles by grouping according to functions or responsibilities; resources by refining into new types or possibly adding new systems; security levels by strengthening them depending on roles and resources. Each refined entity is updated in the matrix, and changes in one dimension automatically affect other dimensions (cf. Fig. 2 and Fig. 3).

The fourth stage is the merging of the refined components and the creation of a refined access matrix (*First layer*) (see Fig. 3 as a pattern and the full refined version at the *link*.). The refined matrix is subsequently covered by the model at the next layer and verified accordingly. Comparison of the two layers allows us to assess how the refinement affects the security, consistency, and completeness of the model. This iterative structure makes our approach suitable for scalable enterprise architectures while ensuring controllability, formalization, and verifiability.

			Old Resource	**Objy**		...
			New Resource	**Objy.1**	**Objy.d**	
			Class	**class s1.1**	**class s1.h**	...
Old Role	*New Role*	*Clearance*				
Rx	**Rx.1**	**cl c1.1**		c1.1(x.1)>=s1.1(y.1) ∧ Opr (x.1).(y.1)	c1.1(x.1)>=s1.h(y.d) ∧ Opr (x.1).(y.d)	...
	Rx.a	**cl c2.1**		c2.1(x.a)>=s1.1(y.1) ∧ Opr (x.a).(y.1)	c2.1(x.a)>=s1.h(y.d) ∧ Opr (x.a).(y.d)	...
	Rx.(a+1)	**cl c2.1**		c2.1(x.(a+1))>=s1.1(y.1) ∧ Opr (x.(a+1)).(y.1)	c2.1(x.(a+1))>=s1.h(y.d) ∧ Opr (x.(a+1)).(y.d)	
		cl c2.f		c2.f(x.(a+1))>=s1.1(y.1) ∧ Opr (x.(a+1)).(y.1)	c2.f(x.(a+1))>=s1.h(y.d) ∧ Opr (x.(a+1)).(y.d)	...
...	**...**	**...**		...	...	...

Figure 3: Refined Multidimensional Access Matrix (First Layer)

6. Case Study

Our approach is visualised by a healthcare system, which is a complex environment in which various roles, data with different sensitivity levels, and strict operational constraints interact. The workflow

model considers interaction between medical staff and patients, taking into account role differences and data sensitivity. The process models hospital employees who act within clearly defined sub-roles and access levels corresponding to their authority. The patient is considered an auxiliary external role initiating the request process, while the main behavior of the system is modeled through the operations of internal roles. The goal of this work is to formally describe and stepwise refine the role-based workflow model, taking into account access rules based on data sensitivity. Hence, we formalize the behavior of roles and, on this basis, we visualize them and their interaction using the UPPAAL automata modeling tool, which allows us to simulate the workflow and verify it.

6.1. Overview of Development by Our Approach

In the first stage of modeling, we develop an abstract layer. It includes one abstract role - *Hospital_Staff* [1], and an abstract resource - *Data*, for which a sensitivity classification is specified. To simplify the training model at an abstract level, it was decided to define only patient data (*Data*) as the main resource. Working with confidential information is the most critical in medical systems. This assumption does not limit the applicability of the model, but makes it more compact and visual. The access policy is formalized using two key parameters: *clearance* to determine the role clearance level and *class* to determine the resource security level. We define values for the levels: $clearance \in \{undefined, public, sensitive\}$, and $class \in \{public, sensitive\}$. Clearance *undefined* means no access and is used as the basic default denial rule based on the principle of least privilege. Class does not require a value *undefined* because the resource always has a certain sensitivity. A resource with an access level *public* is available to all roles with clearance *public* or higher. To access sensitive resources (*class = sensitive*), a role must have clearance *sensitive* or higher.

Thus, the matrix can be formally represented as a function:

$$Operation: \ (Roles \times Clearance) \times (Resources \times Class) \ \rightarrow \ \{permit, deny\} \tag{3}$$

where $Roles = \{Hospital\_Staff\}$, $Clearance = \{undefined, public, sensitive\}$, $Resources = \{Data\}$, $Class = \{public, sensitive\}$, and $Operation = \{read\}$.

At this stage, only one operation *read* is considered, since reading is the basic and most universal access operation. We chose *read* as a primary example, allowing us to test the logic of access control for the different values of *clearance* and *class*. The access policy is implemented according to the following logic: *(read : permit)* — if *clearance* is higher or the same as *class*, *(read : deny)* — if *clearance* is lower than *class*. Based on this, an abstract access matrix is developed and presented in Fig. 4 It displays all possible combinations of a role and its clearance level concerning resources of different sensitivity (*class*) for a fixed *read* operation. This matrix is the basis for defining guards in the UPPAAL model.

		Resources	**Data**	
		Class	**public**	**sensitive**
Roles	*Clearance*			
Hospital_Staff	**undefined**		read : deny	read : deny
	public		read : permit	read : deny
	sensitive		read : permit	read : permit

Figure 4: Abstract Access Matrix for the Hospital Staff

6.2. Development and Simulation in UPPAAL

6.2.1. Abstract Specification

After constructing the matrix in Fig. 4, we proceed to modeling in UPPAAL. Roles, resources, and sensitivity levels are modeled as states, transitions, guards, and variables. In the abstract layer, the system is modeled as the interaction of three key entities: *Patient*, *Hospital*, and *Hospital_staff*. These entities are implemented as separate, independent templates in UPPAAL, which are synchronized via

channels using variables to simulate the workflow in a hospital. The *Hospital* acts as a mediator that listens to all events of the roles. All its transitions are abstract and are implemented in *Hospital_staff*, where access levels are defined. The core logic is implemented in the *Hospital_staff* template, which models the role's behavior and access to patient data. According to the access matrix (see Fig. 4), all six permissible combinations of the variables *clearance* and *class* were covered in the model. These variables are directly specified in the global declaration of the model: by default *class* and *clearance* are are of type `int` with the initial value 0, while *permit* and *deny* are of type `bool` and initialized to `false`. The sensitivity levels are encoded as integer constants with the following values: `null = 0`, `undefined = 1`, `public = 2`, and `sensitive = 3`.

Accordingly, the decision is made according to logic in Section 6.1

$$permit = true; \quad \text{if } class \leq clearance \ \wedge \ class \neq null \ \wedge \ clearance \neq null \tag{4}$$

$$deny = true; \quad \text{if } class > clearance \ \wedge \ class \neq null \ \wedge \ clearance \neq null \tag{5}$$

Thus, at each moment, the model covers a specific cell of the access matrix. The values of the variables are specified explicitly, and the simulation is performed for one fixed configuration, which affects the behavior of the entire system.

Initially, the patient arrives at the hospital, and their arrival is confirmed by a notification to the *hospital*(*arrival_notification*). This notification is the starting point for the medical staff workflow. The patient is moved to the state *patient_in_progress*. The hospital staff begins collecting the necessary data and measuring parameters, recording health indicators, and collecting additional *data* (*data_collected*). After completing the information collection, a decision is made on further verification and analysis of the data. At this stage, the hospital staff selects one of the possible scenarios for working with the collected information. The first option is manual verification, in which specialists independently analyse the collected parameters and decide on the patient's further actions. If this option is selected, the staff manually verifies the *data* (*data_checked_manually*), after which the processing is completed.

The second possible option is to use the system with automated verification. If the hospital staff chooses to use the automated system *(auto_data_check_requested)*, the data is sent to it for verification and analysis. Before the check of access, two key parameters are selected. The *clearance* is selected through one of three dedicated channels (*hosp_staff_*_clearance*), and *class* is set through a separate channel, depending on the type of information being processed (*read_*_data*). At this stage, the question of the availability of the system, including its tools, functions, and operations, arises. Since different roles of hospital staff imply different levels of permissions and capabilities, access to the system must be differentiated. However, at an abstract level, we have defined only two possible access levels: access granted (*permit*) or access denied/no access (*deny*).

In case of access restriction (*deny*), the system blocks the use of its capabilities and redirects the hospital staff to manual checking. In this situation, the hospital staff is forced to analyze the data manually since automated tools are unavailable and cannot process the information. If access to the system is allowed (*permit*), the system continues processing the data.

Once all steps are completed, all temporary access parameters are reset (*accesses_reseted*). The patient processing is completed (*process_finished*), and the patient receives the status ready to leave the hospital (*ready_to_leave*). When the patient leaves the hospital (*patient_left*), the hospital confirms exit, and the arrival notification is reset (*arrival_notification = false*).

6.2.2. Layered Development

Although the presented abstract model provides a compact and correct representation of the access control logic, it includes only a single role (*Hospital_Staff*) and resource (*Data*). Therefore, we apply stepwise refinement as described in Section 5. At the First refinement layer, the model is expanded by three dimensions: roles, resources, and security levels.

The abstract role *Hospital_staff* is refined into three new operational roles based on task: *Registrar*, *Med_staff*, and *Physician*. The *Registrar* is responsible for patient registration and admin data man-

agement. The *Med_Staff* performs basic medical procedures and preliminary evaluations, while the *Physician* examines the patient, establishes a diagnosis, and prescribes treatment.

The abstract resource *Data* was refined into two new sub-resources and one completely new resource, taking into account the source and usage of the data. A new resource *Analyzer* was added according to the superposition principle, which involves a new operation *execute*. Consequently, we received three new components of the model at Layer 1: *Personal_data* (patient data, e.g. name, gender, age, date of birth), *Medical_data* (patient medical parameters, e.g. tests, diagnoses, anamnesis), *Analyzer* (system for automated verification and analysis).

While personal data can be both public and sensitive, medical data is sensitive by default [20, 21]. The patient's public personal data includes name, date of birth, gender, and sensitive personal data includes passport data, insurance, and residential address. At the abstract level, clearance is represented by three values: public, sensitive, and undefined (default), while class is represented by public and sensitive values. After the refinement, we define new security levels since all medical data is sensitive by default, but not all of them are accessible to hospital staff. Thus, the sensitive category is refined into two sublevels: confidential and secret. This refinement also applies to the class values. Considering this, Registrator has access to the functions and data that are necessary for registering patients. Role Med_staff has access to personal data and, partially, to the patient's medical data. Role Physician has full access, including secret data and system resources, at this stage of the simulation. Accordingly, the access control mechanism is updated to enforce these distinctions.

As a result of the refinement, the access control matrix is expanded (see Fig. 5) and now includes three new sub-roles, two new sub-resources, and a new resource, as well as a refined set of security levels. The new system Analyser is superposed on to the model by adding new channels, variables, and guards that perform new tasks in such a way that the old behavior of the model is not affected.

6.2.3. Refined Model in UPPAAL

After the refinement, the template *Hospital_Staff* is refined into three separate templates corresponding to the refined roles $Hospital\_Staff \in \{Registrator, Medical\_Staff, Physician\}$. In the new model, each role performs its part of the process, initiating events, setting variables, and performing transitions when guards hold. Each of the new sub-roles now acts within new access levels, and operations on resources are permitted according to the updated First layer matrix (see Fig. 5). Interaction between roles is implemented through synchronization of the corresponding templates. Transitions between states occur when conditions (guards) hold and the values of variables are set by one role, and then read by another role. Data transfer between roles is organized independently and makes it possible to implement multithreading. Thus, the refined model preserves the integrity of the system. The refinement from the abstract level to the refined level can be proved correct, preserving the safety invariants.

			Old Resources	Data				Analyser
			New Resources	PersonalData		MedicalData		
			Class	public	confidential	confidential	secret	confidential
Old Roles	*New Roles*	*Clearance*						
Hospital_Staff	Registrator	undefined		read : deny	read : deny	read : deny	read : deny	execute : deny
		public		read : permit	read : deny	read : deny	read : deny	execute : deny
		confidential		read : permit	read : permit	read : deny	read : deny	execute : deny
	Med_Staff	undefined		read : deny	read : deny	read : deny	read : deny	execute : deny
		public		read : permit	read : deny	read : permit	read : deny	execute : deny
		confidential		read : permit	read : permit	read : permit	read : deny	execute : deny
	Physician	undefined		read : deny	read : deny	read : deny	read : deny	execute : deny
		confidential		read : permit	read : permit	read : permit	read : deny	execute : permit
		secret		read : permit	read : permit	read : permit	read : permit	execute : permit

Figure 5: Refined Multidimensional Access Matrix for the Hospital

6.3. Evaluation of the Rules and Validation of End-to-End Access Control Cases

The model is simulated separately for each refined layer, which allows for stepwise detection and correction of logical errors. At the abstract layer of the model, end-to-end scenarios were simulated to verify the overall access control logic. Scenarios can be executed manually, with stepwise tracking of all transitions and states, or automatically, using the Verifier module and pre-prepared scripts describing the expected behavior of the model. In addition to simulation, a formal check of the critical properties of the model is performed according to the formulas (4) and (5). In particular, the Deadlock-freeness (*A[] not deadlock*) is confirmed, as well as the reachability of correct execution branches depending on the configuration of input parameters. This checks that proof obligation 5 in Section 4.1 is satisfied.

The presented rules were implemented both in the abstract matrix and in a practical case study (see Fig. 6). This confirms that the proposed approach maintains logical integrity: roles are clarified without expanding rights, resources are fragmented or added only when needed, and security levels are strengthened depending on the context.

	Rules	Case Study
R1	New sub-roles not have more permissions than the parent role	Hospital_Staff → Registrator, Registrator cannot gain access above parent.
R2	Combined rights of sub-roles at least same parent.	Registrar, Med_Staff and Physician together cover all Hospital_Staff accesses
R3	Each role gets only task-relevant access.	Registrator ONLY public PersonalData; Physician has access to Data, including confidential MedicalData
R4	Refining roles may revise resources, operations, and security levels	Physician has secret access to MedicalData; Registrator no access
O1	Resources can be split into sub-resources	Data → PersonalData, MedicalData
O2	New resources added if existing do not cover tasks	Resource Analyser is added
O3	New resources must be consistent with the refined roles and security levels	Physician ONLY is allowed to execute Analyser (execute : permit).
S1	Levels can be strengthened with sub-levels	Physician: public → confidential → secret
S2	Refining security levels depends on the role and resource refinements.	Med_Staff has access to confidential but NOT to secret MedicalData.

Figure 6: Formal rules coverage by Abstract model and Case Study

7. Conclusions

The proposed approach combines role structuring, matrix description of access logic, and stepwise refinement to formalize, simulate, and verify workflows in complex distributed systems. The refinement steps enable intuitive design of access control and formal validation. Unlike the classic RBAC model, where access is assumed to be based on a role, in the developed approach, access rights are refined and strengthened. A hybrid combination of RBAC and MAC allows for strict delimitation of access to sensitive data and the necessary flexibility for the daily work of the staff. This makes our approach relevant for medical and other sensitive domains. Using a multidimensional access matrix makes it possible to scale the model with minor effort since adding new roles, resources, or sensitivity levels does not require reworking the core logic. In addition, the approach allows for access refinement based on the context and level of data sensitivity. The proposed approach supports independent stepwise refinement of workflow components with their parallel execution. This creates a basis for modular development and stepwise validation of access policies, applicable both at the design and execution stages. Thus, the proposed approach provides a coherent transformation of an abstract description of access control policies to their executable and verifiable implementation, which makes it a promising tool for building reliable access control systems of high complexity and data criticality.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly only for grammar and spelling checks and take full responsibility for the content.

References

[1] Y. Yehorova, M. Waldén, A layering approach with role-based workflow modelling for the enterprise workflow, in: Proceedings of the 14th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2024), SCITEPRESS, Dijon, France, 2024, pp. 266–273. DOI: 10.5220/0012754900003758.

[2] Workflow Management Coalition, The workflow reference model, 1995.

[3] C. Snook, M. Waldén, Refinement of statemachines using Event B semantics, in: J. Julliand, O. Kouchnarenko (Eds.), B 2007, volume 4355 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2006, pp. 171–185. DOI: 10.1007/11955757_15.

[4] M. Broy, K. Stølen, Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement, Springer, Berlin, 2012.

[5] M. Waldén, Layering distributed algorithms within the B-Method, in: D. Bert (Ed.), B'98, volume 1393 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 1998, pp. 243–260. DOI: 10.1007/BFb0053365.

[6] I. Sommerville, Software Engineering, 10 ed., Pearson, Boston, 2016.

[7] A. Daassa, N. Missaoui, M. Machhout, S. A. Ghannouchi, Access control in BPMSs: A case study for building a privacy-preserving covid-19 process in bonita, Security and Privacy 8 (2025) e70050. DOI: 10.1002/spy2.70050.

[8] A. D. Brucker, I. Hang, G. Lückemeyer, R. Ruparel, SecureBPMN: Modeling and enforcing access control requirements in business processes, in: Proceedings of the 17th ACM Symposium on Access Control Models and Technologies (SACMAT '12), ACM, New York, NY, USA, 2012, pp. 123–126. DOI: 10.1145/2295136.2295160.

[9] V. Attaluri, V. Aragani, Sustainable business models: Role-based access control (RBAC) enhancing security and user management, in: Driving Business Success Through Eco-Friendly Strategies, IGI Global, Hershey, PA, 2025, pp. 341–356. DOI: 10.4018/979-8-3693-9750-3.ch018.

[10] S. Osborn, Mandatory access control and role-based access control revisited, in: Proc. 2nd ACM Workshop on RBAC (RBAC'97), ACM, Fairfax, 1997, pp. 31–40. DOI: 10.1145/266741.266751.

[11] S. Ameer, J. Benson, R. Sandhu, Hybrid approaches (ABAC and RBAC) Toward Secure Access Control in Smart Home IoT, IEEE Transactions on Dependable and Secure Computing vol. 20, no. 5 (2022) pp. 4032–4051. DOI: 10.1109/TDSC.2022.3216297.

[12] M. Heydarian, T. Doyle, R. Samavi, MLCM: multi-label confusion matrix, IEEE Access 10 (2022) 19083–19095. DOI: 10.1109/ACCESS.2022.3151048.

[13] A. Abbas, Maximizing security with the policy of least privilege and segregation of duties in organizations, 2024. DOI: 10.13140/RG.2.2.20183.69287.

[14] R. Back, K. Sere, Stepwise refinement of parallel algorithms, Science of Computer Programming 13 (1990) 133–180. DOI: 10.1016/0167-6423(90)90069-P.

[15] R. Back, J. von Wright, Refinement Calculus: A Systematic Introduction, Springer, New York, 1998. DOI: 10.1007/978-1-4612-1674-2.

[16] S. Katz, A superimposition control construct for distributed systems, ACM Transactions on Programming Languages and Systems 15 (1993) 337–356. DOI: 10.1145/169701.169682.

[17] National Institute of Standards and Technology, NIST SP 800-12 Rev.1, https://csrc.nist.gov/, 2017. Accessed 2025-01-01.

[18] ISO/IEC, ISO/IEC 27002:2022. Information security, cybersecurity, and privacy protection – Code of practice for information security controls, Technical Report, International Organization for Standardization, 2022.

[19] UPPAAL Team, UPPAAL Homepage, https://uppaal.org/, 2024. Accessed 2024-02-28.

[20] European Union, General Data Protection Regulation (GDPR), https://gdpr-info.eu/, 2016. Accessed 2025-07-05.

[21] U.S. Department of Health and Human Services, Office for Civil Rights (OCR), Health information privacy, https://www.hhs.gov/hipaa/index.html, 2021. Accessed 2021-06-09.