

A Layering Approach with Role-based Workflow Modelling for the Enterprise Workflow

Yevheniia Yehorova^a and Marina Waldén^b

Faculty of Science and Engineering, Information Technology, Åbo Akademi University, Tuomiokirkontori, Turku, Finland

Keywords: Business Process Modelling, Role-based Workflow, Formal Modelling, Stepwise Refinement, Layering, Modelling, Simulation.

Abstract: Fast and high-quality workflow management is one of the most important tasks in all domains. Workflow modelling has become a common technique that facilitates and supports the business process or rather its automation. Modelling a workflow always depends on the roles that perform tasks. A formal approach is essential for high-quality system modelling. Besides, visualisation tools are required to represent models and work with them. In this paper, we propose an approach for modelling processes for efficient workflow enterprise management. To make this modelling more fluent and flexible, it should be parallelised based on roles. The core of our approach is the Role-based Workflow method. The Role-based modelling is accomplished using a layered development method based on stepwise refinement. Our approach combines straightforward stepwise modelling with the possibility of a quick assessment of the situation at any stage of the workflow by using the UPPAAL tool for modelling and verification. In this paper, we visualise our approach with a healthcare case study.

1 INTRODUCTION

Assurance of high-quality processes and productivity in any business or enterprise requires highly efficient workflow organisation and management. In this paper, we are looking at workflow design that is driven by a need to provide more efficient business processes and reduce the risk of human errors (Omarov, 2021).


Business process management is the discipline of managing processes as the means for improving business performance outcomes and operational agility. The business process indicates how humans, machines, and systems operationally achieve a goal (Margaria, 2013) by solving “interacting tasks” (ANSI/EIA632A, 2021). As a rule, one business process includes several workflows.


According to the standard, “a workflow consists of a sequence of concatenated steps, where each step follows the precedent without delay or gap and ends just before the subsequent step may begin” (ISO12052, 2017).

Thus, a workflow is an entire mechanism that gives a virtual representation of the actual work. At the core of any workflow are specialists who perform each role and solve each task in their daily activities. Role-based workflow modelling allows fast visualisation and testing of solutions. Additionally, the workflow process based on roles is agile and adaptable to changes.

Optimization-based design in the early stages of architectural design is regarded as an efficient and performance-driven approach (Wang, 2022). The role-based modeling of workflows has been proposed within access control in management systems (Djatcha, 2022). However, an alternative perspective can be explored by shifting the attention from the rights associated with each role to its characteristics, parameters, and tasks.

Because of the workflow complexity, it is appropriate to model it using stepwise refinement that transforms an abstract specification into a deterministic system. New features suggested by the requirements are stepwise added to the specification

^a  <https://orcid.org/0009-0008-5526-4448>

^b  <https://orcid.org/0000-0001-8703-3179>

to avoid handling all the concrete implementation issues at once. (Snook & Waldén, 2006).

This paper proposes a layered approach for the Role-based Workflow modelling method (Liang & Bai, 2006) (Chengjun, 2009) that allows a stepwise refinement of the workflow structure. The objective is to design the structure by using the refinement process to make the characteristics, tasks, and resources of each role at different Business Process Management levels more detailed. To facilitate the development and validation of our model, we rely on the UPPAAL tool (Behrmann, 2006) which is a modelling and verification tool for analysing concurrent processes and their interactions.

We show the advantages of our method with a healthcare case study where medical staff, patients, and systems form the roles. Using our layered approach strengthened by the UPPAAL tool support we can demonstrate how to improve task execution, as well as allocate resources in a hospital, and hence, how to provide better coverage of patient service. All UPPAAL models discussed in the paper are made available (Yehorova, 2024).

2 BUSINESS PROCESS MANAGEMENT

Business Process Modelling (BPM) is used to accurately represent and optimize organizational processes. Process modelling helps to build communication between stakeholders both inside and outside the organization (Alomari, 2018).

BPM allows us to depict the architecture of the organization and the processes in it. Moreover, process models can be developed from different views. Enterprise architecture is an area concerning the organizations, compositions, connections, and relationships of individual elements of an enterprise. These elements can be very different: people, processes, systems, task data, etc. (ISO/IEC/IEEE 42010, 2022)(ISO 15704, 2019).

Researchers argue that existing modelling notation is not capable of covering all points of view, nor all subject areas (Alomari, 2018). There are three levels of abstraction: strategic, tactical, and operational. Multiple levels of abstraction are based on Anthony's model (Joseph Kim-Keung, 2015). This model is the basis for classifying processes.

The operational level is the level where many varying roles each with a separate workflow perform specific tasks. System have not been used as a role in the modelling in the same way as humans.

Roles at the operational level will be internal and external. Internal roles are those who perform tasks in the workflow. External roles are those that will influence the transitions between roles, as well as the choice of the model itself. Graphical visualisation at this level depends on the tasks, roles, and needs of each role.

3 ROLE-BASED WORKFLOW

The Workflow Management Coalition includes the concept of roles, represented as a competency or a set of responsibilities. It is a logical abstraction that depends on the abilities of the participants. It is based on the actions of the roles, not the roles themselves. A role is a set of resources with certain technical capabilities (Liang & Bai, 2006). It does not support dynamic changes in those elements. Using BPM strategies and levels, the roles and their tasks should first be defined before they are used in role-based workflow modelling. In our paper, the roles are defined according to their missions.

Researchers have analysed that a role-based workflow system can better cope with a changing workflow than an activity-based system (Chengjun, 2009). The role-based workflow modelling approach that is presented in our article relies on roles as the trigger and leader for modelling.

The role-based workflow can be considered as a technique or methodology in the field of BPM. Role-based workflows are designed to organize, optimize, and automate processes and tasks. The role-based workflow defines the responsibility of roles and the sequence of activities for each role within specific business processes. This allows the work of various departments to be planned. Role-based workflows typically involve using technology to automate repetitive tasks and visualise the status and progress of tasks and processes.

4 STEPWISE DEVELOPMENT

A role-based workflow can best be modelled as a parallel system. These systems are often complex consisting of different entities and require a stepwise approach to development.

4.1 Refinement

Refinement refers to the process of improving an existing product, project, or system. Stepwise

refinement (Back & Sere, 1990) is one of the main methods for developing high-quality systems that need to be correct by construction. The behaviour of parallel systems in this approach is described with events in the form of guarded commands. An event can be chosen for execution if its guard evaluates to true. The parallel execution of two events means that they can be executed in either order and still perform the same result.

In stepwise refinement, an abstract specification of a system is transformed by correctness-preserving steps into an executable program that has the same behaviour as the original specification. In each refinement step, we can add new variables and new events to introduce new features to the system, while the old behaviour should not be modified, s.c., superposition refinement (Katz, 1993). New invariant properties that this new feature should satisfy are also given. To be able to prove that a system is a correct superposition refinement of another more abstract system the following proof obligations have to be satisfied (Back & Sere, 1990)(Snook & Waldén, 2006):

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 preserving the old behaviour. Moreover, the guards of the refined events can be strengthened.
3. Each new event in the refined specification should only concern 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, the refined system should still allow the old behaviour.
5. Whenever an event in the abstract specification is enabled, then either the refined event or one of the new events should be enabled.

By discharging these proof obligations for each refinement step in the system development we have proven the correctness of the system with respect to its abstract 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.

4.2 Layered Development

Layered Development is a software development methodology that involves structuring a system into distinct layers (or levels), where each layer is a specific component that performs specific functions

and has a clearly defined set of responsibilities as, e.g., data management. Since each layer in a parallel system is responsible for a specific set of features, it interacts with the other layers to provide a complete solution. A layer can only interact with certain other layers, which makes the system as a whole easier to understand, maintain, and develop in a reliable and efficient way.

Superposition refinement using layers (Waldén, 1998) is a special form of Layered Development. The new features are introduced via new variables and new events that modify these variables following the refinement rules above. The layers together with the original, abstract system constitute the final, refined system.

4.3 Model-Checking Tool UPPAAL

In order for our approach to be more feasible we need tool support. We decided to use the UPPAAL tool (UPPAAL, 2001) to model the actions and the relationships of the roles in the system.

UPPAAL is a model-checking tool with a graphical simulator that can be used for the modelling, verification, and validation of real-time systems (UPPAAL, 2001). 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, allowing to specify system properties using a formal language. The UPPAAL simulator is used for imitation of the general workflow (Behrmann, 2006).

5 LAYERED ROLE-BASED WORKFLOW DESIGN APPROACH

We propose a layered, role-based workflow approach for design at the BPM levels. Our approach is carried out step by step applying refinement from an abstract model to a detailed one.

During the *first stage*, the BPM level for which the simulation takes place is determined. In this paper, we focus on the operational level where there are both internal and external roles.

At the *second stage*, the internal and external roles are determined. The role could be a person or a

system. Let us assume that we will have an external role $OutR$, and internal roles $InR1$ and $InR2$, where $InR1$ is the role person, and $InR2$ is the role system. For each role, we define their types and properties.

The main idea of our approach is to stepwise add more details to a model by specifying and adding more variables, events, and states to it. We use roles (see Section 3) as the basis of a hierarchical model, where the characteristics of the roles are modelled by variables and their tasks are described by transitions and states. There are internal roles that perform the workflow and external roles that affect the workflow. These roles will then be refined into new roles with new characteristics and tasks in the form of new variables, transitions, and states. Relationships between roles are modelled as synchronization of events.

The roles are then developed in a layered manner in the *third stage*. How the layer is added to a role, depends on the role's type, attributes, and main characteristics. Our example contains three role types: "external person", "internal person", and "internal system". To simplify the terminology, the external role can be named customer, client, or user, and the internal role can be named staff if it is a person, and tool if it is a system. Systems are most often used to automate processes or support decision-making depending on what the staff needs. Thus, the system could be linked to or used instead of the staff.

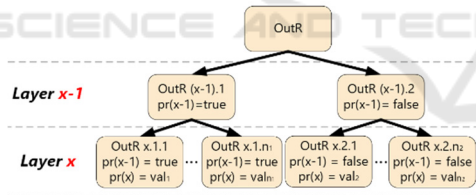


Figure 1: The refinement process of an external role.

External roles are refined by adding parameters called properties. New roles are created by adding new properties to them in the new layer. The new properties are added to the roles in addition to the previous ones. External roles could be related to each role of the staff. Consequently, when adding layers to an external role, it will have new roles with the same base property and different new properties. Figure 1 shows the refinement process of an external role, where:

- x is the number of layers,
- n_i is the number of refined roles on *Layer x*,
- $pr(x)$ is the new property on *Layer x*, and
- val_i is the value of $pr(x)$ of roles $OutRx.1.i$ and $OutRx.2.i$.

In the abstract specification, we have one role $OutR$. On *Layer (x-1)* a new property $pr(x-1)$ of type Boolean has been introduced, and as a result, on *Layer (x-1)*, we have refined $OutR$ to two new person roles: $OutR(x-1).1$ with property ($pr(x-1) = true$) and $OutR(x-1).2$ with property ($pr(x-1) = false$). In the next layer, *Layer x*, property $pr(x)$ with type $Val \subseteq Int$ has been introduced. On *Layer x*, we have refined both the roles on *Layer x-1* to n_1 and n_2 new roles, respectively, depending on the size of Val .

For example, let us assume that our external role in the abstract specification is *Client* and that we want to refine it considering the property *age_group*. Then the first layer consists of the refined roles *Client child*, *Client adult*, and *Client elderly*. The layer has three new roles because the property *age_group* has three values.

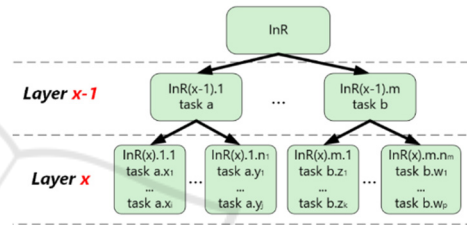


Figure 2: The refinement process of an internal role.

Internal roles are refined by tasks. New roles are created by adding new tasks or refining tasks of the previous role. Internal roles could be related to any external and internal roles. Figure 2 shows the refinement process of an internal role, where

- x is the number of layers,
- m is the number of refined roles on *Layer (x-1)*,
- n_i is the number of refined roles on *Layer x*,
- a, b are the task numbers on *Layer (x-1)*,
- $a.v, a.y, b.z, b.w$ are the task numbers on *Layer x*,
- i is the number of tasks of task $a.v$,
- j is the number of tasks of task $a.y$,
- k is the number of tasks of task $b.z$, and
- p is the number of tasks of task $b.w$.

In the abstract specification, we have one role InR that has been refined to m different roles with separate tasks on *Layer (x-1)*. As a result, we have m new roles with the new tasks a, b , etc. on *Layer (x-1)*. These roles model different persons who work in parallel and whose tasks can be performed in parallel. Each role $InR(x-1).i$ of *Layer (x-1)* has been refined into n_i new roles on *Layer x*, where every role can have a list of new tasks. For example, role $InR(x-1).1$ is refined by roles $InR(x).1.1, \dots, InR(x).1.n_i$ on *Layer x* with a list of i new tasks $a.v_1, \dots, a.v_i$.

For example, let us assume that the internal role is *staff* layered according to its tasks. If the abstract specification has the role *hotel staff* then it can be refined to the roles *receptionist*, *other_staff* etc., on the first layer. The layer contains these new roles because they have separate special tasks.

The *fourth stage* includes the unification and coordination of all stratified roles into a single system. Each role requires careful coordination with the others, resulting in a unified architectural structure. Within this architecture, workflows for each role remain separate, but they all need to be consistent with each other. Hence, during the merging process, the separate layers in the workflow of one role may be changed to better align with the other roles. Layers can be combined or excluded if redundancy is detected.

This process of merging and coordinating roles within the business process results in updated layers. This not only helps to streamline the workflow but also creates the conditions for a culture of adaptation and clear division of roles while ensuring consistency across all layers to achieve high performance and organisational excellence.

6 CASE STUDY - HEALTHCARE SYSTEM

A healthcare system is a good example for role-based workflow modelling, since methods, techniques, and tools from the field of logistics and industry are suitable for managing operations, inventory, and resources in healthcare. The quality of medical care is one of the main objectives of healthcare. Unfortunately, the quality of patient care depends on a number of factors that are not all directly related to medicine. For example, the geographical features of the country or the region, staffing, and workload of medical personnel. Here the question arises, what to do if there is a critical shortage of personnel? Since the workload and fatigue of medical staff affect the quality of care, the need for effective resource management, primarily time and people, emerges.

In the hospital context, a layered architecture can be used to design a role-based workflow focused on the basic role. This workflow can be integrated with existing clinical workflows and systems. In this section, we will deploy our layered development method on the healthcare case study.

6.1 Overview of Development Using Our Approach

As the *first stage* of our layered development approach, we select the level to be the *operational BPM level*. It allows us to use external and internal roles in our case study.

At the *second stage*, the internal and external roles are determined. The external role is *Patient* as a user or customer of medical service. The internal roles are *Hospital staff* as the role person and *System* as the role system.

In the *third stage*, roles are further developed in a layered manner using refinement. Let us start with the role *Patient*. The abstract specification consists of one external role *Patient*. The *degree of urgency* of patient care needs to be determined as the first property of the role *Patient*. Therefore, in *Layer 1* the property *degree of urgency (Emerg)* of type Boolean is defined. As a result, we have two new roles *Patient with emergency (Emerg=1)* and *Patient without emergency (Emerg=0)* in *Layer 1*. In *Layer 2* we add the *degree of hidden urgency, i.e. hidden emergency (HidEmerg)* as an additional property of Boolean type. Using the decision table technique (Copeland, 2004) we select only relevant cases (see Table 1). In the case of a patient with an emergency (*Emerg=1*), the hidden emergency value is not relevant (*HidEmerg=~*), while for a patient without an emergency (*Emerg=0*), both values of hidden emergency are important. Hence, in *Layer 2*, we have three new roles (see Figure 3):

- Patient with an emergency (*Emerg=1 & HidEmerg=~*),
- Patient with a hidden emergency (*Emerg=0 & HidEmerg=1*),
- Patient without any emergency (*Emerg=0 & HidEmerg=0*).

Table 1: (A) Decision Table of all possible conditions of role Patient. (B) Intermediate table. (C) A collapsed Decision Table conditions of role Patient.

(A)			(B)			(C)		
Patient			Patient			Patient		
Emerg	HidEmerg	Hosp	Emerg	HidEmerg	Hosp	Emerg	HidEmerg	Hosp
1	1	1	1	~	1	1	~	1
1	0	1	↓	↓	↓	0	1	1
1	1	0	↓	↓	↓	0	1	0
1	0	0	0	1	1	0	0	0
0	1	1	0	1	0	0	1	0
0	1	0	↓	↓	↓	0	0	↓
0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

In *Layer 3* we introduce the third Boolean property which is *needed for hospitalization (Hosp)*. We proceed from the relevant cases of *Layer 2* that can be found in Table 1(C). For a patient with an emergency, hospital care is always required. Hence, the cases

without hospital care ($Hosp=0$) are ignored (see Table 1(B)). On the other hand, for a patient with a hidden emergency, hospital care is optional, while a patient without any emergency is considered not to need hospital care. Due to this, we have four new roles in *Layer 3* (see Figure 3):

- Patient with an emergency needing care ($Emerg=1 \ \& \ HidEmerg=\sim \ \& \ Hosp=1$),
- Patient with a hidden emergency needing care ($Emerg=0 \ \& \ HidEmerg=1 \ \& \ Hosp=1$),
- Patient with a hidden emergency not needing care ($Emerg=0 \ \& \ HidEmerg=1 \ \& \ Hosp=0$),
- Patient without any emergency ($Emerg=0 \ \& \ HidEmerg=0 \ \& \ Hosp=0$).

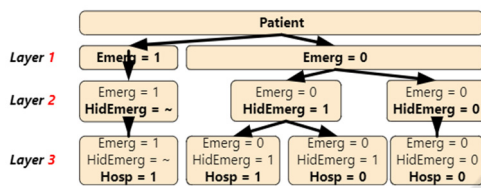


Figure 3: The refinement of the external role Patient.

The next step is the refinement of the internal roles *Hospital Staff* and *System*. In the abstract specification, we have one general role *Hospital Staff*. New internal roles are created by tasks. Considering the large number of possible categories of hospital staff and their tasks we are facing a challenge in selecting tasks for layering. Since internal roles are usually related to external roles, layers for internal roles could be added separately for a single role or in accordance with the needs of external roles. Three main tasks for *Hospital Staff* were defined based on the roles of *Patient*: (1) registration and administration support, (2) preliminary examination, and (3) comprehensive physical examination. Therefore, we have three new roles: *Registrator*, *Medical Staff*, and *Medical Practitioners*.

In the same way as above, the internal role *System* is refined by tasks. These tasks could be defined according to their roles among the hospital staff, *Registrator*, *Medical Staff*, and *Medical Practitioner*. In the implementation (Yehorova, 2024) we only consider *Registrator-focused* system.

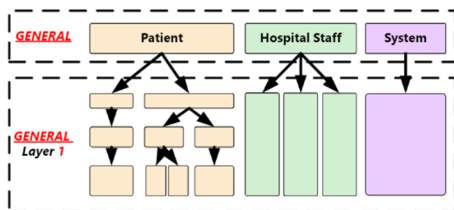


Figure 4: New layers after merging and coordinating roles.

The *fourth stage* of our approach includes the unification and coordination of all roles into one role-based workflow. The roles *Hospital Staff* and *System* have one layer, while role *Patient* has three layers. During the merging process, the separate layers in the workflow of one role may be changed to better align with the other roles (see Section 4). In our case study, three layers of role *Patient* are combined into one to match one single layer of *Hospital Staff* and *System*. The result of this process of merging and coordinating roles is the workflow with updated layers (see Figure 4).

6.2 Development and Simulation in UPPAAL

This paper focuses on the parallel workflow process for different roles, the third and fourth stages of our method. The UPPAAL tool has been used for the simulation of the general workflow. In the abstract model, only one patient can be in the process at one time to highlight potential problems and inaccuracies in the process itself, as well as to provide an opportunity to optimize steps and take into account the characteristics and needs of the patient. Here we only present the abstract specification and one refinement step, *Layer 1*.

Due to the lack of space all UPPAAL models discussed in the paper could be found via the link to our model (Yehorova, 2024).

6.2.1 Abstract Specification

Following the scheme in Figure 4 the abstract specification is defined as three roles: *Patient*, *Hospital Staff*, and *System*. Each role is represented by a separate template. The *Patient* template for the abstract specification contains three locations: *entrance*, *hospital_staff*, and *exit*. The initial location is *entrance*, corresponding to a patient arriving at the hospital. *Hospital Staff* also contains three locations: *new_patient*, *hospital_staff*, and *exit_patient*. Here the initial location is *new_patient* for the patient's arrival. *System* contains two locations: *systemON* and *systemOFF*, where the latter is the initial location.

When a *Patient* arrives at the hospital the process starts with a notification *patient arrived!* sent from *Patient* to *Hospital_staff*. After arriving, the patient is processed by the hospital staff in location *hospital_staff*. Since the patient has arrived (*arrival_notification==true*), but their data have not yet been checked (*check_data==false*) the only allowed transition is through the channel *patient_in_process*, where *Hospital_staff* sends the

request to *Patient*. This indicates that the patient has not yet been checked during this visit. *Hospital staff* checks the patient (*check_patient=true*) and updates the patient data (*check_data=true*).

Next, the hospital staff inputs the patient's data and parameters into the system. Only after the patient's data and patient are checked by the hospital staff using the system, the patient is moved (channel *patient_process_finished*) to location *exit*.

6.2.2 The First Layer

To cover all critical paths of the workflow model each defined role of the patient should be taken into account as defined in Table I (C). The path depends on the selected patient's health condition. When the workflow model is refined new roles are introduced and added as locations. Hence, the location *hospital_staff* is refined by three new locations: *registrator*, *med practitioners*, and *other med staff*.

When a *patient* arrives at the hospital he/she is at the *entrance* location as in the abstract specification. The process starts by deciding the patient's condition (*emergency* or *no-emergency*). In Table I(C) the first case is *emergency*, where the patient *needs hospitalisation* by default. In this case, a request is sent to the hospital staff. An *emergency patient* skips registration and directly arrives at location *other_med_staff*. When the emergency patient has arrived (*emergency==true*) and been processed by *other_med_staff* (*param==true*) he/she directly moves on to the physicians (location *med_practitioners*). Finally, the physicians admit the patient (channel *hospitalisation_of_patient*) to hospital care by specialists (location *other_med_staff*).

The second case (in Table I(C)) concerns a patient with a hidden emergency who needs hospital care. The process starts with the patient notifying the hospital staff. After arriving, the patient is processed by the registrar (in location *Registrator*), since the patient arrived without an emergency (*emergency==false*). The registrar checks the patient data (*data==true*) and inputs the data into the system (*System*) authorised for registrars only. When the patient's data has been checked by the registrar and the system (*data==true* and *sys_data_check==true*), the patient moves to another person on the medical staff (location *other_med_staff*). This person checks the patient's parameters (*param=true*) which could indicate possible earlier hidden emergencies. The presence or absence of a hidden emergency can be selected once after the parameters have been checked (*param==true* and *check_hidden_em==false*). If it is

decided after the check-up that the patient has a hidden emergency (*hidden_emergency==true* and *check_hidden_em==true*) the patient is directed to physicians (location *med_practitioners*) that carry out additional examinations (*examination=true*) (channel *examination_of_patient*) and decide in this case that the patient needs hospital care (*hospitalisation=true*), and allow parameters to be checked again (*param=false*). The patient ends up in the care of a specialist (location *other_med_staff*).

In the other cases (in Table I(C)) it is selected that no hospital care is needed (channel *no_need_hospitalisation*). Medical practitioners end the patient process (channel *finish_patient_process*) and the patient exits the hospital (location *exit* in template *Patient*).

6.3 Verification and Validation

The case study shows an example of our approach on layered development, where we covered different workflows in the hospital depending on the patient's condition. The abstract specification presents the overall picture of the interaction between hospital staff, patients, and systems. The first layer, *Layer 1*, covers four different scenarios for processing patients, depending on whether they are medically urgent and possibly require hospitalisation or not.

The layered development can be proved to be a correct superposition refinement with respect to the abstract specification. This is the case since *Layer 1* introduces new external roles for *Patient* with new characteristics modelled by new variables. These variables and transitions have been added in a controlled manner so that the proof obligations (1)-(5) are satisfied (see Section 4).

Only the new variables are initialised (1) and assigned new values in the transitions (2)-(3) in this layer. The old behaviour remains the same. The guards of the transitions have been strengthened by more precise conditions. The new transitions disable their own guards (4), and hence, they do not start looping. Moreover, the guards of the transitions are constructed so that some transition is always enabled until the exit locations have been reached (5) to guarantee the progress of the model.

The UPPAAL tool has been used to model and analyse the workflows. This allows not only to create formal models and check the correctness of the workflows but also to simulate them. For the case study, all scenarios in Table I(C) have been explored to validate the model.

Modelling using our layered development method can identify problems and improve workflows. This

is the basis for continuous improvement of workflows, innovation, and response to changes in patient needs and healthcare requirements. Our approach could be used as a tool to coordinate the activities of the hospital staff and optimise the use of hospital resources.

7 CONCLUSIONS

The key to clear resource allocation strategies is using well-coordinated workflow models. The model may include all possible results alternatives. In this paper, we propose a new workflow modelling method based on a role-based approach and layered development. We use UPPAAL as our tool support, which allows a concise description and provides means for the analysis of complex systems.

This approach can be applied at any of the BPM levels. The proposed modelling approach consists of four stages for developing models. The first two stages are aimed at defining the subject area, levels of responsibility, and managers' goals, the third stage constitutes the layered development, and the fourth concerns dividing and developing the roles.

The case study shows that the proposed approach allows a structured development of a model stepwise adding details for each role in the workflow. With UPPAAL as our tool support, we can validate our workflow model and achieve increased reliability of the modelling. In conclusion, our proposed development method could provide a powerful tool for analysing and optimising work processes in various industries, which can lead to improved productivity and competitiveness of enterprises.

In the future, we want to elaborate more on how decision support systems could be considered as a role. Moreover, the positive results in this paper open up new prospects for further research in the field of BPM combined with computer science.

ACKNOWLEDGEMENTS

This work has received funding from the Finnish National Agency for Education (EDUFI) via the scholarship OPH-5532-2022, as well as from the Finnish Society of Sciences and Letters.

REFERENCES

Alomari, A., April, A., Monsalve, C., & Gawanmeh, A.

- (2018). Integrating a Decision Tree Perspective at the Operational Level of BPM+. *Computer Systems Science and Engineering*, 33, 219–227.
- ANSI/EIA 632—Processes for Engineering a System | SE Goldmine.
- Back, R. J. R., & Sere, K. (1990). Stepwise refinement of parallel algorithms. *Science of Computer Programming*, 13(2–3), 133–180.
- Behrmann, G., David, A., & Larsen, K. G. (2004). A Tutorial on Uppaal., *Formal Methods for the Design of Real-Time Systems* (Vol. 3185, pp. 200–236). Springer Berlin Heidelberg.
- Chengjun, W. (2009). The Research on the Role-Based Dynamic Workflow Model. *2009 Second International Symposium on Knowledge Acquisition and Modeling*, 2, 358–360.
- Copeland, L. (2004). *A Practitioner's Guide to Software Test Design*. Artech House.
- Djatcha, R. A. D. (2022). A formal approach for role-based modeling of business collaboration processes [Ph.D. thesis, Université de Douala (Cameroun)].
- ISO 12052:2017. ISO.
- ISO 15704:2019. ISO.
- ISO/IEC/IEEE 42010:2022. ISO.
- Joseph Kim-Keung, H. (2015). A review of frameworks for classification of information systems, notably on the Anthony's Triangle. *European Academic Research* Vol. III, Issue 1.
- Katz, S. (1993). A superimposition control construct for distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(2), 337–356.
- Liang, Z., & Bai, S. (2006). Role-Based Workflow Modeling. *2006 IEEE International Conference on Systems, Man and Cybernetics*, 6, 4845–4849.
- Margaria, T., Boßelmann, S., & Kujath, B. (2013). Simple Modeling of Executable Role-Based Workflows: An Application in the Healthcare Domain. *Journal of Integrated Design and Process Science*, 17(3), 25–45.
- Omarov, B., Abdrakhmanov, R., Koishiyeva, T., Bayaly, A., Amirtayev, K., Madaliyeva, G., & Torebay, N. (2021). Smart Hospital: Automation of Business Processes in Medical Centers. *2021 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 106–111.
- Snook, C., & Waldén, M. (2006). Refinement of Statemachines Using Event B Semantics. In J. Julliand & O. Kouchnarenko (Eds.), *B 2007: Formal Specification and Development in B* (pp. 171–185). Springer.
- UPPAAL. <https://uppaal.org/>
- Waldén, M. (1998). Layering distributed algorithms within the B-method. In D. Bert (Ed.), *B'98: Recent Advances in the Development and Use of the B Method* (pp. 243–260). Springer
- Wang, L. (2022). Workflow for applying optimization-based design exploration to early-stage architectural design—Case study based on EvoMass. *International Journal of Architectural Computing*, 20, 41–60.
- Yehorova, Ye. (2024). UPPAAL-models of the healthcare case. <https://goo.su/j9XKf>