Blended Workflow: Organizational Perspective

Guilherme Filipe Pereira Querido Ramos
guilherme.f.p.q.ramos@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2018

Abstract

Workflow systems allow the execution of a process specification with the intent of improving the process efficiency. With the addition of a resource perspective to a workflow system, it is possible to customize the workflow engine to the person that is executing the business process. In this dissertation, I enrich the Blended Workflow with a resource perspective that allows the design and execution of activity and goal specifications that comply to several resource assignments. Blended Workflow is a workflow system that combines executions of both activity and goal specifications that are generated from a data model. It is proposed a set of resource management languages for the Blended Workflow system which has high expressiveness and preserves the consistency between the Blended Workflow models. It is also shown how these languages are implemented and how does the execution engine supports the resource perspective.
Keywords: Workflow system; Model generation; Resource management; Work assignment

1. Introduction

Nowadays, many business processes are translated into an activity workflow in order to improve it's efficiency and interoperability with other processes. However, as time passes by and processes become more complex, they also require more flexibility and swiftness in responding to deviations from the standardized process. This means that we can no longer fully represent a business process in an activity model.

To solve this problem, the Blended Workflow system was presented. This system provides end users with a workflow engine that allows the execution of workflow models. It is possible to execute a workflow using two views, activity and goal-based. The system consistently manages both views, enabling the user to switch the view they are executing accordingly while the work already done is preserved. The generation of these views is based on an initial data model provided to the system, from which the activity and goal models are generated.

During this thesis, I intend to enrich the Blended Workflow system with a resource management perspective that allows the specification of execution constraints based on the organizational structure of the process. The final result should be a design and an execution environments for the Blended Workflow which consistently deals with the resource management aspect, and where can, eventually, be efficiency gains during execution, due to the additional information that is contained in the enriched model.

This document is organized as follows: Section 2 provides a short description of the Blended Workflow. Section 3 presents the related work regarding resource management and how workflow systems usually tackle the problem. Section 4 proposes an approach to implement the concepts presented in the related work in the BW system. Section 5 briefly summarizes the implementation that was made in order to implement the solution proposed in Section 4. Section 6 evaluates the current system according to the parameters defined in the thesis project. Finally, the Section 7 concludes the thesis and proposes the following areas of research.

2. Blended Workflow

Blended Workflow is a workflow system that allows the execution of business processes using two different views, activity-based and goal-based views, both based on the same data model. This system allows end users to choose which view of the process they will follow at a given moment based on the circumstances of the execution. This allows them to better respond to unexpected situations that are not explicitly defined.

The two views that the system provides are based on two equivalent specifications for the business process that are derived from a data model provided to the system.[6] Both views represent the user's current state of the workflow and allow

the users to switch between them, while maintaining the state of the workflow instance. The activity specification follows a BPMN-like specification but enriched with PRE and POST conditions. These conditions define what the data model state be immediately before and after the execution of an activity. The goals specification is organized in a tree that represents each goal's dependencies. A goal is achieved when all of its sub-goals are achieved together with its success condition. Goals have an activation condition, ACT, that needs to be true before the goal can be achieved, an invariant condition, INV, that represents a condition that must hold in order for the goal to be achieved and a success condition, SUC, that represents the state changes produced by the goal. [6]

The transformation of the data model into the two specifications that support activity and goal views is done through a set of semi-automatic steps. There is a Condition Model [6] that is automatically generated and mediates between the data model and the final specifications. This model describes the conditions that need to hold when the data model contains all the required data. It is used to understand what conditions the specifications must maintain in order to keep the integrity between the two models, and it is actually used as the basis of two automatic transformations that generate the activity and goal models. After the models are generated, the designers can apply transformations both in the activity and goal model, by composing and decomposing elements of the model. This is an important feature because it helps the designers adjust the models to the desirable kind of interaction.

Enabling the switch between the two different views means that the activity and goal views need to be consistent in the initial state and during the execution of the workflow.

3. State of the Art

In resource management, there are several concepts that are important to understand before we start diving into the underlying architectures that usually take place in workflow systems. These concepts, based on Muehlen's work[7], will also be used through out the document to describe the solution to the Blended Workflow.

Resource - (or process participant) is an entity that contributes to the execution of business goals and activities, usually having activities assigned to it.

Resource Model - models the resources that contribute to the execution of the workflow model.

Workflow - (or workflow model) is a detailed specification of a process model that is able to be executed by a workflow management system. It is composed by several tasks that are connected in a graph that represent its dependencies. [5]

Workflow Management System - provides a infrastructure for the set-up, execution and analysis of the performance of a workflow.

Role - link between the resource model and the process model. From the point of view of the resource model, a role represents both the capabilities and privileges that a resource has. From the process model, a role represents the capabilities and privileges necessary in order to execute a certain activity.

Capability - propriety of the resource that describe actions that the resource can do regardless of its position. For example, John has a capability that is speaking 5 different languages. This capability is not dependent of its organization position neither is it granted by the organization.

Task duty - type of involvement from a resource to an activity. Usually, task duty represents the different roles in a RASCI table.

Organizational Position - function that a employee has within a organization. It gives the employee some roles and demands the responsibility of achieving the positions goals by the employee.

Organizational Unit - aggregate of multiple organization positions in a logical way as in a practical way in order to improve communication and efficiency between positions. Organizational Unit members work together towards a common goal of the various positions. They are permanent groups of human resources

The **resource model** is a fundamental part of any workflow system that integrates resource management. It is where the information about resources, their roles, responsibilities, how they are structured and related with each other is stored.

A human resource is typically a member of the *organization* and has a specific *position* which gives him *roles*. They can be a member of one or more *organizational units*, or one or more *organizational teams*. Human resources also have relationships with other resources, that represent the organizational hierarchy. A *direct report* is the resource to which the resource reports. A *subordinate* are resources of which the resource is responsible for and to whom they report. [5] They can also have *capabilities* which are useful upon the optimization of assignments of tasks to them.

Having resource management integrated with a workflow system enables the possibility to assign resources to parts of a process specification. Due to having information about what our resources are, what activities the workflow has, and which resource is assigned to which activity, we can automatically extract information about an employee work list, generate shared work lists for groups of employees that need to handle activities together.

The **assignment of tasks** is made at design-time and define the conditions that the resources must fulfill in order to become candidates to perform a work item. Therefore, this specification at design-time must be integrated with the specification of the business process.

It is also possible to have different degrees of involvement to a process's task. RACI matrices allows the specification of different responsibilities in a process for the various members of an organization. [3] They provide a way to plan, organize and coordinate work between resources, while giving resources different degrees of responsibilities in a task.

RACI matrices associate tasks with resources. This association can vary in scope and tasks can either be associated with persons, roles, organizational positions or organizational units. Each cell of the table contains zero or more RACI initials. There are four different functions that a resource can have on a task in RACI tables:

There are also variants of RACI tables where more functions are added. In particular, RASCI tables are particularly interesting to IT organizations, because they provide the function *Support* (S). People with the function S are people that may assist in the completion of the given task.

3.1. Resource Patterns

Russell et al identified in their paper[5] several resource patterns that usually appear on commercial workflow systems. These patterns are divided into 7 categories: (i) Creation, (ii) Push, (iii) Pull, (iv) Detour, (v) Auto-start, (vi) Visibility and (vii) Multiple resource patterns. *Creation patterns* focus on the restrictions that happen at design time that restrict how a work item can be executed. Only the creation patterns are considered because the focus of the thesis is to define a resource management language to be integrated in the Blended workflow Specifications. The creation patterns are:

Direct Distribution - The allocation to individual resources is made at design-time.

Role-Based Distribution - Allocation of a work item to a specific role. This work item will then be made available to every individual resource that has that role.

Deferred Distribution - Design-time specification that the resource assignment will be made at run-time as part of the workflow.

Authorization - Specify the range of privileges that a resource must have to be allocated to a task.

Separation of Duties - Two tasks cannot be executed by the same resource in a given workflow instance.

Case handling - allocation of various work items to the same resource in a given workflow instance.

Retain familiar - Allocation of the work item to a resource that previously executed another work item of the same workflow instance.

Capability-based distribution - Allocation of a work item to a capability. This work item will then be made available to every individual resource that has that capability.

History-based distribution - Allocation of the work item is based on the previous work performed by the resources.

Organizational distribution - Allocation of a work item to a organizational position. This work item will then be made available to every individual resource that has that position.

Automatic Execution - the work item is able to be executed with the intervention of a resource.

3.2. RAL

RAL (Resource Allocation Language) [2] is a language designed to formally specify the resource assignments in a process model while offering a graphical notation (RALph). RAL was designed from the beginning to integrate the specification of a workflow model and a resource model. It allows us to connect these 2 models in an expressive and independent way, enabling the specification and automatic verification of resource assignments. [1] It is a modular language divided into 4 modules each one supporting a specific type of assignments.

RAL is a language that is used as link between the Organization Model and the Business process, and comprises RAL Core, a common set of RAL constructs that is responsible for this connection. This module was designed to cover a subset of the *Workflow Resource Patterns* [5]. RAL Org is an extension to RAL Core with expressions and constraints that allow for the representation of allocations based on organizational structures.

The module RAL AC (Access-Control) extends RAL Core in order to specify run-time constraints,

therefore providing support to the following WRP patterns: Binding of Duties, Case Handling and Retain Familiar. RAL AC also provides a solution to the specification of a task duty. The definition of what is a task duty is open to the organization. In this case, it was defined in order to represent RASCI matrices. *PersonConstraint* is also extended with the following condition so that it is possible to express that the person assigned must be the same related to another activity.

4. Solution

Most of the concepts presented above are applied to an activity-based workflow. However, in the context of the Blended Workflow, it is necessary to apply these concepts to the different models. Firstly to the data model, which is used to generate the activity-based and goal-based specifications. Secondly to the condition model, which is an intermediate model where all the model constraints are explicitly represented to serve as a common basis to the activity and goal models. Finally, to the activity-based and goal-based specifications, which are the final models of the system.

4.1. Specification

The first problem that surfaces while implementing the concepts presented in chapter 3 to the Blended Workflow is the specification of the models. This is an integral part of the problem, because it is with the specification of the data model that the BW models are generated. Therefore, the choice of how to integrate the specification of the resource model and the resource assignments into the data model is of major importance.

And the choice made was to use RAL, both to specify the resource model and for the resource assignments. I chose RAL due to a number of reasons: (i) **expressiveness -** it is possible to represent most of the resource creation patterns using the constructs provided by the language; (ii) **independence** - it is independent from a workflow language and thus it can be integrated with the Blended Workflow Specification.

In spite of RAL being designed to be applied to activities, the assignments and the concepts presented in the RAL work[2] are loosely coupled to activities, and they can be applied to other concepts. In this work, it will be extended to the other models of the Blended Workflow, being applied to entities, attributes, and relations from the data model, and to goals from the goal model.

4.1.1 Resource model

The resource model proposed for the BW Engine is based on the meta-model proposed by RAL (Fig. 1) and plays a integral part in the system. All of BW models are enriched with resource assignments that will reference to the resource model. This resource model is specified by the designer and contains all the information about the organizational structure and its entities that will later be used by the resource assignments.

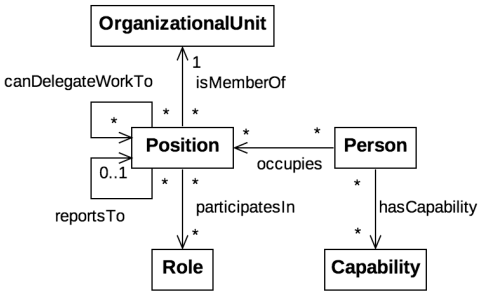

Figure 1: Organizational model proposed by RAL

RAL does not provide a formal specification language for the resource meta-model. The language that allows this specification is proposed in this thesis and complies with the meta-model used by RAL. (Fig. 1) The syntax follows a similar approach to the syntax used to describe BW's data model.

4.1.2 Resource Assignments

The connection between each of the BW models and the resource model is made in the form of resource assignments using the RAL expression language. In the data model, we have data elements that are assigned to the resources present in the resource model. The situation is similar when we analyze the activity model and goal model. An activity and a goal can also be assigned to a resource that complies with their respective data elements.

There are some adaptations to RAL AC module that is responsible for the description of assignments based on process involvement. Taking inspiration in RACI tables, I restricted the RAL AC module to two types of assignments. A resource can either be responsible for or informed about a data element. A data field can have one resource responsible for it and multiple resources informed about it. The resource that is responsible for the data element is the one that defines it. A resource that is informed about the data element can read the information of the data element upon the completion of it.

The final step in the integration of resource management concepts to the specification of BW models is to specify how is the RAL language connected to the data model. There are two parts for the increment that was made to the language. The first part are the rules that allow the designer to specify the process involvement related to a RAL

expression. The second part of the increment to the language addresses how it is applied to the Blended Workflow models.

All of the resource rules are added in the *ResourceRules* block. This approach provides modularity to the specification of a BW model as it separates the data specification from the resource assignments. As a short example, the resource assignments for the creation of the Data entity would be described as follows:

```
ResourceRules {
  Data has responsible HAS ROLE
      ScreeningOfPatients IN UNIT
      OrthopedicUnit\par
  Data.bloodPressure informs IS
      PERSON IN DATA FIELD Data.
      episode.doctor\par
}
```

Although not enforced by the grammar, it is only allowed to have both one rule of the types responsible and inform for each data field. This restriction will be placed in the semantical check that is done to the data model by the engine. This decision was made in order to reduce the probability to errors when writing the resource rules and improve its readability.

I also added a new type of resource rule beside resource assignment. The rule *entityIsPerson* allows the association between the data model entities and resource model persons, i.e, entities that are both data model entities and resource model persons. This is particular useful because, in many cases, there are entities in the data model that represent a person and that same person is present in the resource model.

4.2. Generation of models

The Blended Workflow models go through several stages of transformation during their lives. The first step is to individually develop the data model and the resource model. After the two models have been developed it is necessary to join them together to create a data model that is enriched with resource assignments to resources in the resource model, i.e, write a *ResourceRules* block. The third step is the transformation of this model into the condition model that represents the workflow constraints that the data model implies.

With the condition model generated, it is now possible to generate both the activity model and the goal model with the workflow constraints required by the data model. This generation must also be updated to include the resource assignments and to preserve their constraints in both generated models. This generation is nothing more that trickling down of the resource assignments that each data element has to the respective activity and goal.

Finally, the operations that are supported on both the activity model (6) and goal model (7) also have new constraints that must be taken into account due to the addition of the resource assignments. The generation of both the activity model (4) and the goal model (5) preserve the assignments in the original data model. However, the operations to the generated models, can have a different effect on the assignments: maintain, relax or restrict the models.

4.3. Operations

An important part of the solution are the operations that are made on goals and activities, on the goal model and activity model, respectively. These operations allow the mutation of the workflow and must be analyzed regarding the resource management. With the new resource assignments, it is necessary to define what are the intended semantics of the operations, i.e, preserve, relax or strict the model.

To preserve the semantics, it is necessary that the model produced after the operation generates the same instances that could be generated by the model before the operation. The relaxed semantic tells us that the model produced by the operation generates a set of instances that is a superset of the instances that the model before the operation could produce. Finally, the restricted semantic means that the produced model generates a set of instance that is a subset of the instances the were previously possible.

There 8 different operations across all models that can be made. Two of them, rename activity and rename goal, are neither workflow or resource related. Merge and split operations in either activity and goal models are workflow related but impact the resource assignments, therefore having a different behavior with the addition of the resource perspective. The remaining add/remove sequence operations, only present in the Activity model, only impact the workflow model with the exception of the resource assignments based on the task duty or the assignment deferral.

To the merge activity and goal operations, we must implement all of the semantics present above, i.e, we allow the user to specify which policy we wants for a given merge operation. Since we can only split activities/goals that have been previously merged by the merge operation, the split operation copies the assignments from the merged activity/-goal to the split activity/goal.

4.4. Execution

After each model has been successfully generated, the Blended Workflow engine provides an ex-

ecution interface in which it is possible to execute activities or goals for a given workflow instance. With the addition of a resource perspective, two improvements must be made to the interface and execution model: (i) User identification and (ii) Control access on activities and goals.

The **user identification** is the identification of the different users that use the execution interface. It is necessary to identify each user because we will need to enforce each resource assignment constraint to the respective activity and goal.

The **control access on activities and goals** is the semantical check that is made at runtime that enforces the resource assignment constraints, that each activity and goal has, upon the user that is identified and only allows the resources that make the RAL expression valid to execute the work item.

The BW engine distributes work by providing a list of all the activities or goals, depending on the view, that can be executed by any resource. With the addition of resource assignments, it no longer makes sense to present every single activity/goal that is possible to be executed, but only the list of activities/goals that the resource is eligible to execute.

5. Implementation

The system's architecture has the following structure: (i) designer component - an Xtext project that parses the data model and calls the APIs on the BW engine to generate the specification; (ii) engine component - a backend server that includes the verification of the correctness of the specifications sent by the designer module and allows for the execution of Blended Workflow instances. This engine is implemented in Java Spring and provides an API interface to design, access and operate the models; (iii) frontend component - a Web Application made in React that implements the human interface to access the BW Engine. This application accesses the BW Engine API in order to display its information in HTML and CSS and performs operations on the models.

We can also organize the Blended Workflow into a module architecture. This view of the system gives us a notion of functional modules that exist despite its component organization. It is composed of two parts: the engine and the designer. The engine modules contain the functionality responsible for the allows the execution of the workflow. It includes an *Execution Manager* that keeps track of specification instance's data and its execution and a *Engine UI module* that implements the execution of the models in the user interface. The designer module is responsible for the design of specifications and includes the *DSL Parser*, the *Design UI Module* and the *Models Manager*. The *DSL Parser* is responsible for parsing the BW DSL (Domain Specific Language) and interacts with the *Models Manager* module. The *Models Manager* stores and enables operations on BW specifications. Finally, the *Design UI Module* is the part of the user interface that allows the user access to the design operations that are allowed to be made on the models.

To implement the resource perspective, the modules of the architecture must be changed as follows: (i) the *DSL Parser* should support the proposed resource language and RAL as well as the resource rules language constructs that provide the connection between the data model and the resource model; (ii) Both Execution Manager and Models Manager must support resource management during the design and execution of Blended Workflows respectively; (iii) update the human interface modules to provide resource management in a meaningful way to the process workflow. With theses changes in mind, the modules diagrams takes the structure presented in figure 2.

There are two new modules, *Resource-enabled executor* and *Resource Models*, that are responsible for the execution of the models with the resource assignment's restrictions and with the storage of the resource models, respectively.

The *Engine UI Module* was also separated in two modules that represent two ways of execution a workflow: (i) the *Workflow UI Executor* that provides the standard way of execution the workflow that is implemented by the core project; (ii) the Dashboard UI Executor, that provides a view that is costumized to the user executing the system and that shows every work item that the person has permission to execute.

The *DSL Parser* module was also separated into two different modules: the Core parser, which provides the parsing for the data model DSL, and the Resource Parser, which implements the languages described in the previous chapter, that enable the specification of a resource model and the resource assignments.

Finally, there is the *Authentication Manager* module. This is a new module that provides the user identification that is needed during the execution of the engine.

The new modules structure is depicted in Figure 2.

5.1. Designer

The designer module comprises the Xtext project, the models manager and the designer UI module. The Xtext project allows the design and specification of the data model language. Xtext is an Eclipse GMT project that allows for the creation of textual DSL. [4] Based on an EBNF-like notation, Xtext is able to provide an syntax-highlighted editor for the language and generate an Abstract Syntax Tree

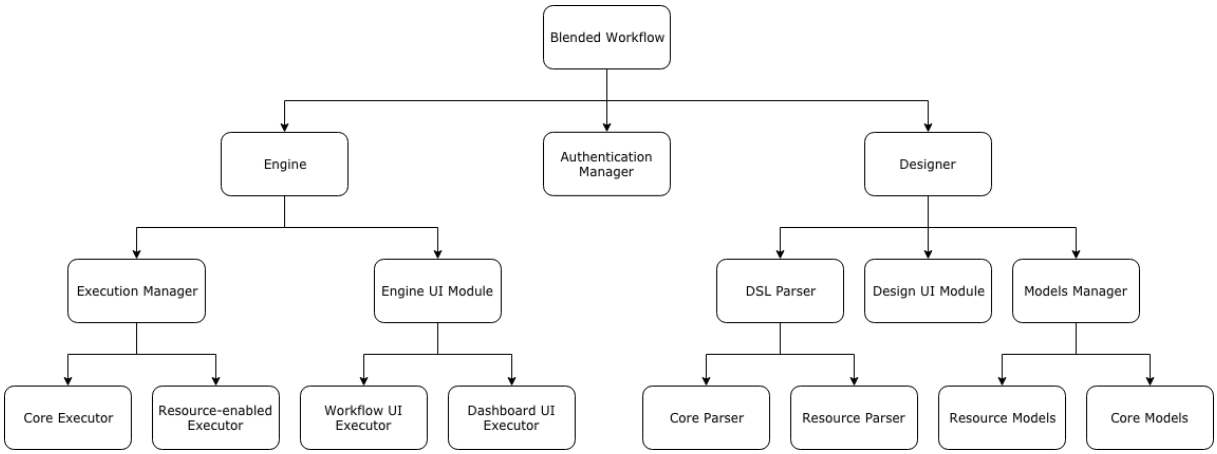

Figure 2: Blended Workflow's module diagram after the addition of the resource perspective

(AST).

The models manager is the part of the design module that is responsible to take the information provided by the parser, evaluate it and store it in the database. It is also responsible for the generation of models and all of the operations that can be made on the models. These modules execute in the Spring Engine component.

The UI module is responsible for the access to the operations that can be made to each model. It is part of the unified interface that is built using React.Js.

The BW languages are denoted in the EBNF-like notation, and each specification is written using the editor provided by Xtext. Upon each save operation in the specification file, the module uses the AST to parse the specification, construct the respective Domain Transfer Objects (DTO) that represent the data model and send the DTOs to the designer to store the specification. The designer API takes the requests, evaluates them, and store them in the database. Upon request by the parser, it also generates the condition, activity and goal models associated with the provided data model. When the engine is executing, the UI module supports the operations that modify the goal and activity models, e.g., merge operations.

Only two parts of the designer module were changed to support the resource perspective. The first is the parsing of the specification, in which I had to enrich the BW language to include the resource languages. The models manager was updated so that it now provides an API to persist resource entities as well as the generation of models. The UI module remained unchanged because it already provided access to the operations that were implemented on the generated model, only throwing new error messages in case they are not allowed.

5.1.1 Storing the resource perspective

The *models manager* is responsible for storing all the resource perspective information, i.e, the resource model, the resource assignments and the enhanced activity and goal models, into a persistent source. This module executes in the engine component that is build using Spring and has as a persistent source a database. The access to the database is made through the Fenix framework.

The module was changed in order to support the design of resource-perpective related entities. To store the entities, it was necessary to create a new fenix framework DML that extends the previous version and add entities for each of the elements of the resource model. It was also necessary to represent the RAL Expression and the hierarchy that was used in the DTOs.

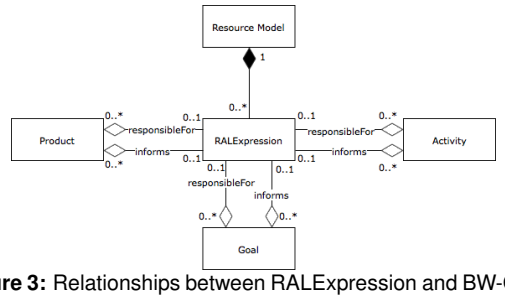

Figure 3: Relationships between RALExpression and BW-Core classes

The UML diagram for RALExpression present in Figure 3 gives an example of a relationship between the core and resource modules. This class is fundamental to enable the resource perspective behaviour. However, to keep the core module low coupled, the classes Product, Activity and Goal have no information of their relation to the class RALExpression. These relations make the resource assignments possible because they associate each of the entity that are used in a workflow execution to a RALExpression.

5.1.2 Specification generation

The generation of models is made in two steps. The first step, in which we generate the models accordingly to workflow restrictions, is already implemented and remain unchanged. The second step is new and implemented in the resource engine. For the activity model, it takes all the entities and attributes present in the data model and for each one, get the activities that have as its post condition the entity in question, and applies to the activity the assignment contained in the entity/attribute. The same is made for the attributes. For the goal model, the methodology is the same with the exception that it will look for goals in which the success condition contains the entity in question. Note that this generation is possible because the generation of the core model is minimal. Each entity and each attribute is assigned to a single activity/-goal, such that the enrichment of the activity and goal core models with resource management does not introduce inconsistencies in relation to the enriched data model.

To support this generation in the database, some relations were added in the DML. These relate the activity and goal entity to a *RALExpression* in a relationship named 'responsible for' and to another *RALExpression* in a relationship named 'informs'. The generation creates these relations, that will later be used by the execution engine to determine whether or not a resource is capable of executing an activity/goal.

5.1.3 Mutating the models

The final part of the designer module that required changes was the operations that can be made to either the activity or goal model. These operations were already implemented in the *core module manager*. However, with the addition of resource assignments, these operations have a different behaviour.

To implement the enhanced behaviour, there were several concerns that must be taken into account: (i) the ability to express which kind of policy we want for a given operation; (ii) ensure that add/remove sequence operations do not break the workflow requirements made by resource assignments; (iii) original resource assignments and the ones resulting from merge operations should be dealt alike; (iv) the merge of resource assignments do not lead into inconsistencies such as (NOT HAS POSITION pos1) AND (HAS POSITION pos1).

Both Rename activity and rename goal remained unchanged since they neither affect the workflow model or the resource model.

The resource perspective does not affect the add/remove sequence operations with the exception of two cases: when one of the activities either has an assignment of types *IsPersonInDataObject* and *IsPersonInTaskDuty* that dependes on the other activity due to the fact that they can introduce implicit circularities in the workflow model. Since the *models manager* in the core project checks for any circularities that may be introduced with the operation, the solution to solve this inconsistency was to add a dependency between the data objects of both activities in order to objectively represent the implicit dependence. This dependence is added by the *DSL Parser* module and is added each time either *IsPersonInDataObject* or *IsPersonInTaskDuty* expressions are parsed. With the addition of this dependence, the core module can now check if the operation will lead to circularities taking into account the resource assignments but without actually using them.

The merge and split operations, for both activity and goal models, had their semantics changed with the addition of a resource perspective. These operations follow the same algorithm for both activity and goal models The merge operation is described as follows:

1. The *Resource models manager* receives a ResourcesMergeOperationDto. This class is an extension of the class MergeOperationDto that was previously used, and adds the information about which type of policy it is supposed to be used in operation.

2. We then extract both activities/goals that are to be merged and get each assignment for the *responsible for* and *informed about* process responsibilities.

3. The designer then proceeds to merge the activity/goal as he has previously done. The philosophy is that we check for incoherences in the merged activity/goal and if any is found, the operation is aborted.

4. The algorithm then tries to create a new RALExpression for the two responsibilities that joins both expressions using the selected policy:

   (a) For the relaxed policy, it is created a RALExprOr that combines both expressions;

   (b) For the restricted policy, it is created a RALExprAnd expression;

5. After the expression is merged, the designer checks if the merged expression has any inconsistency. If it is consistent, the designer assigns the expression to the merged activity/goal.

To check if the merged expression is consistent we use the *SetOfRequiredResources* structure. This structure contains all of the dependencies that the expression has in terms of concrete entities in the resource model. The usage of this structure guarantees to us that the structure give us all of the resource information for chains of expressions that compose with other expressions. The structure contains two sets for: each of the entities of the resource model (positions, persons, capabilities, roles and units); data elements in the form of *task duties*, *in data element* and *history* assignments. One set tells what resource is needed and the other tells what resources we must not have.

The algorithm that checks the consistency of a expression gets the *SetOfRequiredResources* for the expression and traverse the sets in search for negation inconsistencies, i.e, if a resource is in both sets for the category of entity. The expression (HAS ROLE r) AND (NOT (HAS ROLE r)) would generate a *SetOfRequiredResources* structure that would have the role r in the set *roles* and in the set *notRoles*. Since a role is found in both the *roles* and *notRoles* sets, the expression is considered inconsistent. It also checks if the work item products of the merged activity/goal are present in the set that forbids data elements, for example, the work item that has as product the entity e1, and it's RAL expression tells us the person responsible is not in the data field e1.

5.2. Engine

The Engine module is responsible for the execution of Blended Workflow models. It comprises the *Engine UI Module* and the *Execution Manager*. The *Execution Manager* is the model that interacts with the *Models Manager* to enable the execution of the workflow models. The *Engine UI Module* comprises all of the UI that is dedicated to executing specifications.

To enable the resource perspective, a new module, the *Resource-Enabled Executor* module, was added. This module is a sub-module of Execution Manager and provides the same functionality as the *Core Executor* only with a different behaviour that is resource based.

A resource-enhanced execution is a standard execution that obeys the resource assignments that are made to each data element, either though the task duty *responsible for* or *informed about*. Instead of allowing everyone to execute and view every work item in both activity and goal views, a resource-enhanced execution filters work items to the correct persons and provides views that are more useful to each person responsibilities.

Compared to the core execution, a resource-enhanced execution translates into three differences: (i) having pending work item list that only shows the work items that the person executing the system can be responsible for; (ii) restricting the execution of work items to the persons that validate the work item's responsible for *RALExpression*; (iii) the history log of executed work items only shows work items that the person could be responsible for.

Since the resource-enhanced execution is a subset of the core execution and only filters some executions of the core executor, a pipe-and-filter pattern can be applied to the core execution in order to add the new resource execution: we filter instances of the core execution that do not comply with the resource assignments present in the activity/goal work items.

Enabling this type of execution requires one important system feature that was not present until this point: the identification of the person that is executing the system. It is fundamental to be able to determine objectively which person is executing the views, and restrict his operations based on his responsibilities. It is the authentication of users that enables the resource assignment verification.

The method chosen to implement the user authentication was through the Json Web Tokens (JWT) protocol. This protocol was developed to be used for authentications in API applications and works by assigning and passing around an encrypted token in each request that is made to the server. This token helps to identify the logged in user, instead of storing the user in a session on the server and creating a cookie.

The *ExecutionResourceInterface* class provides the functionality for the Resource-enhanced executor module. This is a subclass of the *ExecutionInterface* class, which is the class that implements the Core executor. This class applies the pipe-and-filter pattern to the Execution module through the usage of inheritance. To provide the pipe-and-filter functionality in this inheritance implementation, it is always required that the overridden method, i.e, the method that is being filtered, calls upon it's super method and then applies the filtered intended.

Pending work item set - The first step in executing a workflow instance is to obtain the current list of pending work items. This is where the work item DTOs are obtained to be later passed to the execution of work items. To the standard behaviour of obtaining the pending list of work items, it is important to filter the work items in which only the work items that the logged in user can execute are obtain. This is important to be done in order to present a more simple and useful view for the user executing the system.

Execute a work item - Implementing the resource-enhanced version of the execution of a

work item means that only the person that is logged to the system has permissions to execute it. A person has permission to execute a work item when the work item's responsible for expression has as eligible the logged in person.

History of work items executed - A resource-enhanced version of this function is different in two ways: firstly, instead of returning every single work item executed, only the work items that the logged in user have permission to be informed about it are obtained; secondly, the work items return are of types *ResourceActivityWorkItemDto* and *ResourceGoalWorkItemDto*, with the property *executionUser* set correctly.

5.3. User interface

The final component of the resource perspective is the user interface. This component is divided into two modules: the Engine UI Module and the Design UI Module. The Engine UI Module implements the user interface that allows the execution of workflow instances. The Design UI Module provides the access in the user interface to the graphical representation of the several models, and allows the execution of operations in these models.

The interface is built in ReactJs. ReactJs is an open-source Javascript library that is aimed at building UIs and provides declarative an component-based views and allow web designers to build interfaces that are modular and easy to mantain. The addition of the resource perspective added a new UI module, the Dashboard UI Executor, which is a sub-module of the Engine UI module. This module adds a new view for executing workflow instances that are customized for the specific user.. In the Dashboard view, the logged in user can find all of the work items that he can executed, organized by specifications and workflow instances. This view was only possible, because of the assignment of work items to persons, that allows the system to provide a view that has meaning for the user currently logged.

The motivation behind this view is to offer a simple presentation of all the work items assigned to the user organized by specification and instance. Afterwards, the user can load a chosen instance into the standard executor and start the execution of that instance from there. Previously, a person would have to check every activity/goal view of a workflow instance to know which and how many work items he could execute. Besides providing a global view of which work items the person can execute, it also provides a direct link to the instance executor, that allows the execution of the work items either by activity or goal models.

6. Evaluation

To evaluate if the system meets the expectations and achieves the objective of successfully implementing resource management in the Blended Workflow system it is necessary to measure how many workflow resource patterns does the system support. From the list presented in the thesis, it was concluded that the resource perspective added to the Blended Workflow supports 8 out of 11 resource patterns that have been identified in previous work.

The system provides the execution of workflow instances that respect the resource assignments written in the resource rules, restricting the access to the log of executed work items to the persons that have permission to validate the *informs* responsibility.

The system also handles the consistency of operations in both activity and goal models, ensuring that the old operations do not break any of the resource assignments.

Finally, with the addition of the dashboard view, the UI provides a personalized view for the authenticated person that takes into account the work items that are to be executed and the resource permissions that the person has.

7. Conclusion

The objective was to add a resource perspective that allowed the workflow designers to represent a process resource model as well as making resource assignments between the already implemented workflow models and the resource model. The main difficulty that was faced during this dissertation was how to apply concepts from the resource assignments to three different types of BW elements: data elements, activities and goals. The problem emerged from the fact that all of the concepts presented in the state of the art are usually related to activities gather that the BW elements. The Blended Workflow system is now able to provide both features to workflow designers while adapting all of the concepts presented in the state of the art to the Blended Workflow elements.

Besides providing a platform to specify these enriched activity and goal models, the system also provides an application to execute these models accordingly to the restrictions imposed by the resource assignments. This means that the Blended Workflow engine now provides a control-access to the execution of activities and goals that is very useful for organisations.

References

[1] C. Cabanillas, D. Knuplesch, M. Resinas, M. Reichert, J. Mendling, and A. Ruiz-Cortés. RALph: A graphical notation for resource assignments in business processes. In *Lec-*

ture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), volume 9097, pages 53–68, 2015.

[2] C. Cabanillas, M. Resinas, A. Del-Río-Ortega, and A. Ruiz-Cortés. Specification and automated design-time analysis of the business process human resource perspective. *Information Systems*, 52:55–82, 2015.

[3] C. Cabanillas, M. Resinas, and A. Ruiz-cort. Automated resource assignment in bpmn models using raci matrices. pages 56–73, 2012.

[4] S. Efftinge and M. Völter. oaw xtext: A framework for textual dsls. In *Workshop on Modeling Symposium at Eclipse Summit*, volume 32, page 118, 2006.

[5] N. Russell, A. H. M. Hofstede, and D. Edmond. Workflow resource patterns. *Business*, 3520(5446):13–17, 2004.

[6] A. R. Silva and V. García-Díaz. Integrating activity- and goal-based workflows: A data model based design method. In *Lecture Notes in Business Information Processing*, volume 256, pages 352–363, 2016.

[7] M. Zur Muehlen. Organizational Management in Workflow Applications – Issues and Perspectives. *Information Technology and Management*, 5(3/4):3–4, 2004.