Analyzing the resource perspective of workflow management systems : using a meta modal and constraints

Analyzing the Resource Perspective of Workflow Management Systems: Using a Meta Model and Constraints

M. Pesic and W.M.P. van der Aalst

Department of Technology Management, Eindhoven University of Technology, P.O.Box 513, NL-5600 MB, Eindhoven, The Netherlands.
m.pesic@tm.tue.nl, w.m.p.v.d.aalst@tm.tue.nl

Abstract. Workflow management systems support business processes and are driven by business processes models. These models cover different perspectives including the control-flow, resource, and data perspectives. Contemporary workflow management systems offer a wide variety of mechanisms to support the resource perspective. The resource perspective is essential for the applicability of workflow management systems. This paper focuses on the *resource perspective*, i.e., the way the system distributes work based on the structure of the organization and capabilities/qualifications of people. Our goal is to develop a structured and exact method to serve as aid for the analysis, evaluation and comparison of various workflow management systems. First we develop models of particular workflow management systems (Staffware, FileNet, and FLOWer). Next, we develop a meta model that can be seen as the "least common multiple" of system-specific workflow models. Finally, for each particular workflow management system we develop Object Constraint Language (OCL) expressions that constrain the meta model. Each of the three constrained meta models specifies the resource perspective of a particular system and can be used for further analysis.

1 Introduction

Workflow management systems are process-aware information systems [5, 18]. As process-aware systems, they are used in companies as a means for the computerized structuring and driving of complex business processes. Workflow management systems implement business process models, and use them for driving the flow of work by allocating the right employees to the right tasks at the right times. The system manages the work of employees – it will determine which tasks an employee has to execute and when, which documents will be used, which information will be available during work, etc. Typically, a workflow management system offers several mechanisms to distribute work, and the way the resource perspective is managed varies among different systems. Nevertheless, we believe that existing systems are too limited in this respect [43]. The goal of this paper is not to propose advanced work distribution mechanisms. Instead, we focus on the analysis of existing systems. The goal is to develop a structured method for describing the resource perspective in workflow management systems. This will provide aid for easier analysis, evaluation, comparison and selection of systems.

The work reported in this paper can be seen as an extension of the meta modelling approaches presented in [8, 36–38, 42]. These approaches use static models like, for example, Unified Modelling Language (UML) class diagrams [25] to discuss work distribution concepts. The work reported in this paper starts with developing UML models of different workflow management systems: Staffware 9.0 [48], FileNet Business Process Manager 3.0 [21], and FLOWer 2.05a [39]. Next, a meta model that generalizes models of different systems is developed as "the least common multiple". Although this approach can be found in some earlier work [36], we take a step further and for each of the three systems we develop constraint expressions in the Object Constraint Language (OCL) [26]. These expressions show how the meta model can be constrained to act as a model of each of the workflow management systems we have used. Within the context of the *workflow patterns initiative* [6] (cf. www.workflowpatterns.com) 43 resource patterns [45, 43] have been defined. Workflow management systems can be evaluated using system-independent patterns of work distribution. Unlike patterns, which use a descriptive language to analyze work distribution, OCL constraints that we use offer more structured and more exact description of workflow management systems.

2 Extending the Meta Modelling Approach With Constraints

We use the meta modelling approach as the basis for our work [8, 36–38, 42] and deploy UML class diagrams as a modelling language. Figure 1 shows that we use three steps in our approach. First, we develop *models* of three workflow management systems: Staffware, FileNet, and FLOWer. Second, we build a *meta model* as a generalization of these three workflow management systems. Third, for each of the systems we *constrain* (specialize) the meta model with OCL constraint expressions. The focus of this paper is on the third step, i.e., on the development of system-specific constraints for the meta model. We take the meta model as the starting point in this paper, and shortly describe the models of Staffware, FileNet and FLOWer in the appendix of this paper. We develop and use the models of Staffware, FileNet and FLOWer for the construction of the meta model.

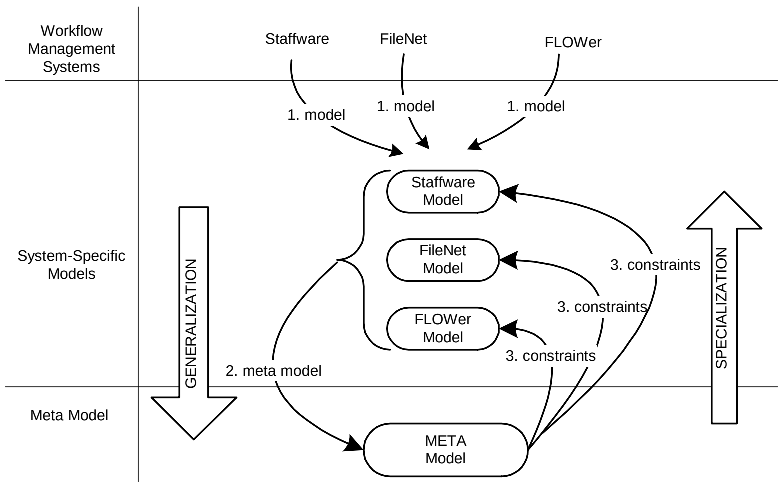


Fig. 1. The Three Steps Approach

Workflow management systems and their models typically contain different concepts/features. Even when some of the concepts are common for various systems, these concepts are often named differently in different systems. Therefore, it is often difficult to analyze and compare systems and their models. By developing a set of OCL constraint expressions, for each of the three workflow management systems, we provide a structured and exact description of particular systems. Figure 2 shows how we extend the meta modelling approach with constraints. First, we use models of three workflow management systems (i.e., Staffware, FileNet and FLOWer) to construct the meta model. This meta model is the “least common multiple” of Staffware, FileNet and FLOWer models - it contains not more than all the features of these three models. Second, for the meta model we develop three sets of OCL constraints. Each set of constraints refers to the scope of the meta model that represents the model of a particular workflow management system. Constraints referring to one workflow management system describe which parts of the meta model refer to that workflow management system, and exclude the parts of the meta model that refer to the other two workflow management systems. While the meta model is a common basis for different workflow management systems, constraints represent the features that are unique for particular systems. This makes it easier to analyze, evaluate and compare *constraints* than different *models* of workflow management systems.

Our focus is on the resource perspective of workflow management systems. Due to the complexity we split the resource perspective model into three UML [25] models:

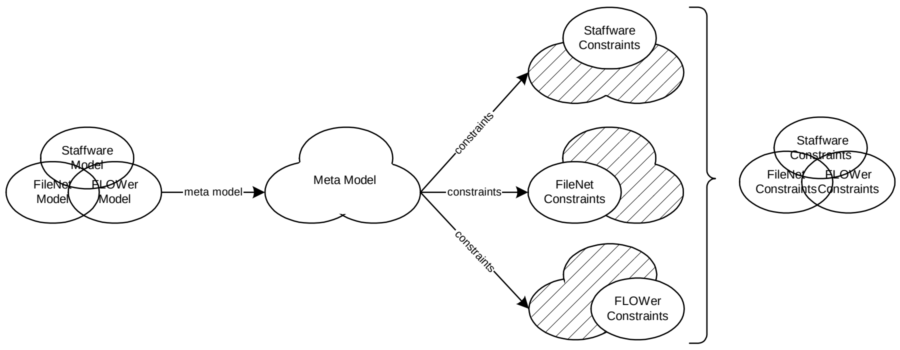


Fig. 2. Constraints - an Extension of the Meta Modelling Approach

The Organization model shows how an organizational structure can be defined in a workflow management system. This can, for example, be done by defining users, their organizational roles and departments;

The Resource Allocation model shows how resource allocation is specified in a process definition in a workflow management system. For every task that has to be executed by a user, the resource allocation mechanism determines the user(s) who can execute the task. The allocation can be specified using the organization model elements (e.g., user-names, roles, departments, etc.);

The Work Distribution module shows how a workflow management system distributes work to employees (resources). The distribution of work to people is done based on the resource allocation specifications for tasks and the current organizational structure.

Figure 3 shows that, at the design-time, the organization model is defined in some administrative tool and resource allocation is specified in the process definition. The workflow engine and work lists handler determine the work distribution of a workflow management system at the run-time.

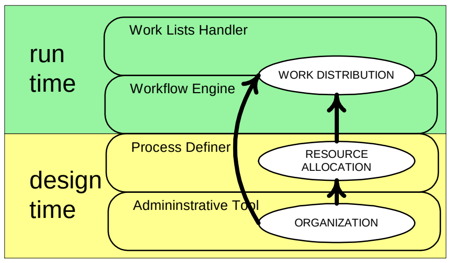


Fig. 3. The Resource Perspective Sub-Models

In line with the structuring the UML models into three sub-models, we we also structure the constraints for every workflow management system into three groups: (1) Organization constraints, (2) Resource Allocation constraints and (3) Work Distribution constraints. Every constraint has a name and unique code. For example, one of the Organization constraints has the name “Group Type”. The constraint code consists of three parts: (1) system name - “SW” for Staffware; “FN” for FileNet; “FW” for FLOWer, (2) sub-model name - “O” for Organization; “A” for Resource Allocation; “D” for Work Distribution, and (3) constraint name. For example, the unique code for the Staffware Organization constraint “Group Type” is “SW-O.GT”.

The remainder of this paper is structured as follows. In Sections 3, 4 and 5 we present the meta model and constraints for Staffware, FileNet and FLOWer. Organization meta model and its constraints are shown in Section 3, Resource Allocation meta model and constraints in Section 4, and Work Distribution meta model and constraints in Section 5. In Section 6 we discuss our results. An overview of related work is given in Section 7. Section 8 concludes the paper. Models of Staffware, FileNet and FLOWer that we have developed and used for the construction of the meta model are shown in the appendix.

3 Organization

We have developed the Organization meta model based on the analysis of Organization models of Staffware, FileNet, and FLOWer. Thus, any organization that could be modelled in any of these three systems can also be modelled in the Organization meta model. In this section we first describe the Organization meta model. Second, we describe a real-life example that illustrates the most important properties of the model. Finally we describe how the proposed Organization meta model can be mapped to models of Staffware, FileNet or FLOWer using OCL constraints.

3.1 Organization Meta Model

Figure 4 shows the Organization meta model for Staffware, FLOWer and FileNet [1]. The structure of an organization can be defined using various groupings of employees. These groupings can be defined based on different aspects of the organizational classification of employees, e.g., an organization can be structured using the hierarchy of departments, roles or other classification criteria. We will refer to the classification criteria as to the *group type* in the Organization meta model. Within each of the classification criteria there is a set of concrete groups, which are represented by the *group object* in the model. The model allows for a detailed description of groups by assigning *attributes* to a *group type*. If an *attribute* is assigned to the *group type*, then it is possible to assign a *value* for that *attribute* to the corresponding *group objects*. *Group objects* are often structured into a hierarchy. This is enabled in the model by creating a relation that allows a *group object* to act as a *master* of other *group objects*. Similarly, a *group object* can also be a subordinate (in the relation *sub*) of other *group objects*. One organizational *position* can be assigned to a number of *group objects* and one *group object* can be referred to by multiple *positions*. A *user* can occupy multiple *positions*. A position indirectly assigns users to various *group objects*. Every *user* can be described in more detail via *qualifications* (representing concrete values) for each of the predefined *qualification types*.

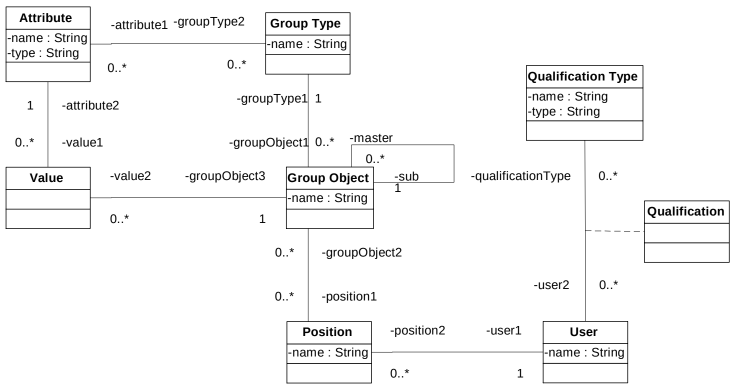


Fig. 4. Organization Meta Model

[1] Organization models of Staffware, FLOWer and FileNet are presented in the appendix (Section 8).

Figure 5 shows an example of how an organization can be modelled using the Organization meta model. In this illustration we focus on one part of the UML model in Figure 4, i.e., we abstract from attributes and their values. In this example, the organization is structured using two classification aspects (group types) – "Department" and "Role". A structure is defined within each of the two group types by creating a hierarchy of group objects: the department of "Technical Sciences" consists of "Mathematics" and "IS" sub-departments; users involved with "Teaching" can be "Professors" or "Assistants", while "Administrative" users can perform a role of "Manager" employees. Two positions within the organization are formed based on the object groups of "Department" and "Role" object types: (1) position "prof A" is related to the "IS" and "Professor" group objects, and (2) position "mngr A" is related to the "IS" and "Manager" group objects. The organization contains two users (employees) – "Wil" and "Jelmer". By assigning them to the two positions, an indirect relation to corresponding group objects is made. This makes it easier to switch positions of users in the organization, by changing the assigned position instead of changing all relating group objects. Another type of qualifications of users requires assigning a concrete value to the relation. An example of this kind of qualifications is the "Birth date", where every user is assigned a specific value for the qualification type.

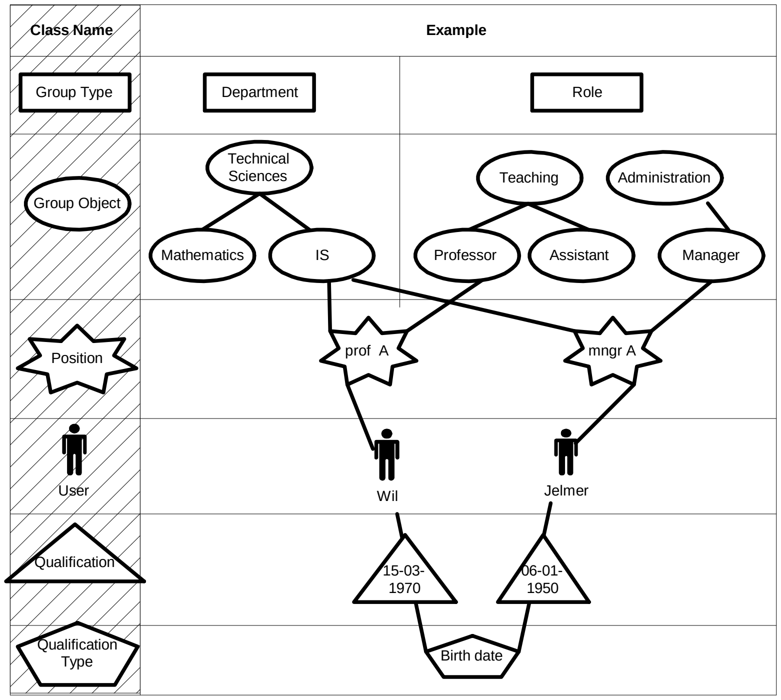


Fig. 5. Example of One Organization

Although the Organization models of Staffware, FileNet and FLOWer have significant differences, some of their elements refer to the same contexts in the Organization meta model. These are shown in Table 1.

3.2 Organization Constraints for Three Workflow Management Systems

In the remainder of this section we will show how the Organization meta model can be mapped to the three system-specific Organization models using OCL constraints expressions. Table 2 shows the constraints for the Organization models of Staffware, FileNet and FLOWer. In the first column we show the context of the constraint, i.e., the name of the meta model class that is constrained.

Table 1. Matching of Concepts in Systems - Organization

Meta Model	**Staffware**	**FileNet**	**FLOWer**
Attribute	Attribute		
Group Object	Group, Role	Workflow Group, Work Queue, OS Group	Role, Function Profile, Work Profile, Context, User Group
Qualification Type	Attribute		
User	User	User	User

The constraint name and unique code are shown in the second and the third column, respectively. The description of the meaning of the constraint is shown in the fourth column.

Table 2. Organization Constraints

context	constraint name	code	description
Group Type	Group Type	O.GT	possible organizational groups, units, etc.;
	Group Qualification	O.GQ	groups can be qualified by assigning group attributes to group types;
Group Object	Group Capacity	O.GC	how many positions can be assigned to a group object;
	Organizational Hierarchy	O.OH	the hierarchical model of the organization;
Position	Organizational Position	O.OP	group objects that one organizational position can be assigned to;
Qualification Type	User Qualification	O.UQ	qualifications of users;

Staffware. Table 3 presents OCL expressions that constraint [2] the Organization meta model to represent the Staffware Organization model. Staffware uses groups and roles to define an organizational structure, by assigning them directly to users. Therefore, constraint (SW-O.GT) allows for two instances of group types – "Group" and "Role". Concepts of groups and roles differ in Staffware. First, groups can have attributes and roles cannot. Second, one group can be assigned to multiple users, while a role can only be assigned to one user. Constraint (SW-O.GQ) makes sure that attributes cannot be assigned to roles and (SW-O.GC) that one role can be assigned to only one position (and indirectly – to only one user). It is only possible to model a flat organizational hierarchy in Staffware, as it is shown in (SW-O.OH). Groups and roles are assigned directly to users in Staffware, and constraint (SW-O.OP) defines that every relation between one user and one group object represents a unique position. Adding corresponding supervising position instances and assigning users to them can model additional Staffware supervision relations between users and groups. In addition to qualifying group types with attributes, Staffware allows qualification of users with the same set of attributes, as it is possible to model this in the Organization meta model via qualification types.

FileNET. Again we use OCL constraints relating to the Organization meta model, as shown in Table 4. FileNet allows for defining organizational structure by creating (and assigning users to) three types of groups: (1) groups of users as they are defined externally, in the operating system, (2) work queues (which act as pools of work items for the members of the queue), and (3) workflow groups to allow for the run-time alteration of the structure of teams. This property is illustrated by constraint (FN.O.GT) while (FN-O.GQ) shows that it is not possible to create attributes for group types. As constraint (FN-O.OH) shows, it is not possible to model hierarchy with any of

[2] In this paper we assume that the reader is familiar with the Object Constraint Language. For a detailed description of the OCL syntax see [26].

Table 3. OCL Constraints for Organization Model in Staffware

context Group Type
inv (SW-O.GT): GroupType.allInstances()->collect(name) = Bag {'Group','Role'}
inv (SW-O.GQ): self.name = 'Role' implies self.attribute1->isEmpty()
context Group Object
inv (SW-O.GC): self.groupType1.name = 'Role' implies self.position1->size() = 1
inv (SW-O.OH): self.master->isEmpty() and self.sub->isEmpty()
context Position
inv (SW-O.OP): self.groupObject2->size() = 1

the group objects in FileNET. Organizational structure in FileNET requires assigning users to individual group objects directly to users, without using positions for indirect assigning of these relations. Because of this, constraint (FN-O.OP) makes sure that a position is created every time a relation between a user and one of the group objects is modelled. Constraint (FN-O.QT) prevents the creation of qualification types for users.

Table 4. OCL Constraints for Organization Model in FileNET

context Group Type
inv (FN-O.GT): GroupType.allInstances()->collect(name) = Bag {'OS Group','Work Queue', 'Workflow Group'}
inv (FN-O.GQ): self.attribute1->isEmpty()
context Group Object
inv (FN-O.OH): self.master->isEmpty() and self.sub->isEmpty()
context Position
inv (FN-O.OP): self.groupObject2->size() = 1
context Qualification Type
inv (FN-O.QT): Qualification Type.allInstances()->isEmpty()

FLOWer. Table 5 shows how the Organization meta model can be constrained in order to represent the way organizations are modelled in FLOWer. First, constraint (FW-O.GT) defines that role, function profile, work profile, user group and context are the four possible types of organizational units. With (FW-O.GQ) we constrain the model and make it not possible to create attributes for group types. FLOWer assumes an organizational structure where only group objects originating form different group types can form hierarchical relations (and group objects originating form the same group type do not form relations), as constraint (FW-O.OH.1) shows. Next, constraints are listed to define the hierarchy structure that group objects can form. Constraint (FW-O.OH.2) allows for roles to be assigned to function profiles. Function profiles are collections of roles, as shown in (FW-O.OH.3). In (FW-O.OH.4) function profiles are assigned to work profiles and are defined for a specified context. Work profiles are collections of function profiles and are defined in a specified context, as shown in (FW-O.OH.5) and (C.FW-O.OH.6). Constraint (FW-O.OH.7) shows how users can be assigned to user groups, which belong to context units. Context is the highest (hierarchy) level organizational unit and other units (function profiles, work profiles and user groups) are defined in specified context units, as shown in constraint (FW-O.OH.8). Every position is assigned to one work profile, as shown in constraints (FW-O.OP.1) and (FW-O.OP.2). Assigning qualifications to users is not possible in FLOWer, as constraint (FW-O.UC) shows.

4 Resource Allocation

In this section we first present the Resource Allocation meta model and then we show how it can be constrained to represent Resource Allocation models of Staffware, FileNet and FLOWer.

Table 5. OCL Constraints for Organization Model in FLOWer

context Group Type	
inv (FW-O.GT):	GroupType.allInstances()->collect(name) = Bag {‘Role’,‘Function Profile’, ‘Work Profile’, ‘Context’, ‘User Group’}
inv (FW-O.GQ):	self.attribute1->isEmpty()
context Group Object	
inv (FW-O.OH.1):	self.master->select(groupType1.name = self.groupType1.name)->isEmpty() and self.sub->select(groupType1.name = self.groupType1.name)->isEmpty()
inv (FW-O.OH.2):	self.groupType1.name = ‘Role’ implies (self.master->select(groupType1.name = ‘Function Profile’)->notEmpty() and self.master->select(groupType1.name <> ‘Function Profile’)->isEmpty())
inv (FW-O.OH.3):	self.groupType1.name = ‘Function Profile’ implies (self.sub->select(groupType1.name = ‘Role’)->notEmpty() and self.sub->select(groupType1.name <> ‘Role’)->isEmpty())
inv (FW-O.OH.4):	self.groupType1.name = ‘Function Profile’ implies (self.master->select(groupType1.name = ‘Context’ or groupType1.name = ‘Work Profile’)->isEmpty() and self.sub->select(groupType1.name <> ‘Context’ and groupType1.name <> ‘Work Profile’)->isEmpty())
inv (FW-O.OH.5):	self.groupType1.name = ‘Work Profile’ implies (self.sub->select(groupType1.name = ‘Function Profile’)->notEmpty() and self.master->select(groupType1.name <> ‘Function Profile’)->isEmpty())
inv (FW-O.OH.6):	self.groupType1.name = ‘Work Profile’ implies (self.master->select(groupType1.name = ‘Context’)->notEmpty() and self.sub->select(groupType1.name <> ‘Context’)->isEmpty())
inv (FW-O.OH.7):	self.groupType1.name = ‘User Group’ implies (self.master->select(groupType1.name = ‘Context’)->notEmpty() and self.master->select(groupType1.name <> ‘Context’)->isEmpty())
inv (FW-O.OH.8):	self.groupType1.name = ‘Context’ implies (self.sub->select(groupType1.name = ‘Function Profile’ or groupType1.name = ‘Work Profile’ or groupType1.name = ‘User Group’)->notEmpty() and self.sub->select(groupType1.name <> ‘Function Profile’ and groupType1.name <> ‘User Group’ and groupType1.name <> ‘Work Profile’)->isEmpty() and self.master->isEmpty())
context Position	
inv (FW-O.OP.1):	self.groupObject2->size() = 1
inv (FW-O.OP.2):	self.groupObject2->select(groupType1.name = ‘Work Profile’)->notEmpty() and self.groupObject2->select(groupType1.name <> ‘Work Profile’)->isEmpty()
context Qualification Type	
inv (FW-O.QT):	Qualification Type.allInstances()->isEmpty()

4.1 Resource Allocation Meta Model

The Resource Allocation meta model is shown in Figure 6. A *process definition* can be defined as a set of *tasks*, which can be of different *task types*. Workflow management systems use various *task types* for modelling *tasks* that are to be executed by *users*, the system, an external application, etc. A *process definition* often uses a set of *data elements*, which are presented to *users* in *tasks* on *forms*. Resource allocation is done on the basis of *authorization* specifications for *tasks* (association-end *authorization1*) and , in some cases, *data elements* (association-end *authorization5*). We refer to these elements that need authorization as to "authorization objects", i.e., tasks and/or data fields are authorization objects. For example, in all three systems we observed, tasks are authorization objects, i.e., it is necessary to assign an authorization object to each task that is meant to be executed by users. Each *authorization* specifies a set of *users* who can perform a specific *action* (e.g., to execute a *task* or to read/write a *data field*). Systems provide for various ways of specifying which *users* are authorized to perform *actions* over authorization objects, i.e., over *tasks* and *data fields*. We refer to an element that can be used to specify an authorization as to an *authorization element*. An *authorization element* can be: (1) a *user*, (2) an organizational unit (a *group object* in the Organization model), (3) a *task*, or (4) a *data element*. When the authorization element specifies a user, then this user is authorized for the action over the authorization object (e.g., to execute a task). Authorization can also refer to an organizational unit (group object), and in this case all users that are members are authorized. It is possible that an authorization refers to another *task* (e.g., the user who executed another task is (or is not) authorized). Another flexible solution for authorization is when an *authorization element* can be determined later, during the execution of the process, using *values* of *data elements* that refer to users, organizational units, etc.

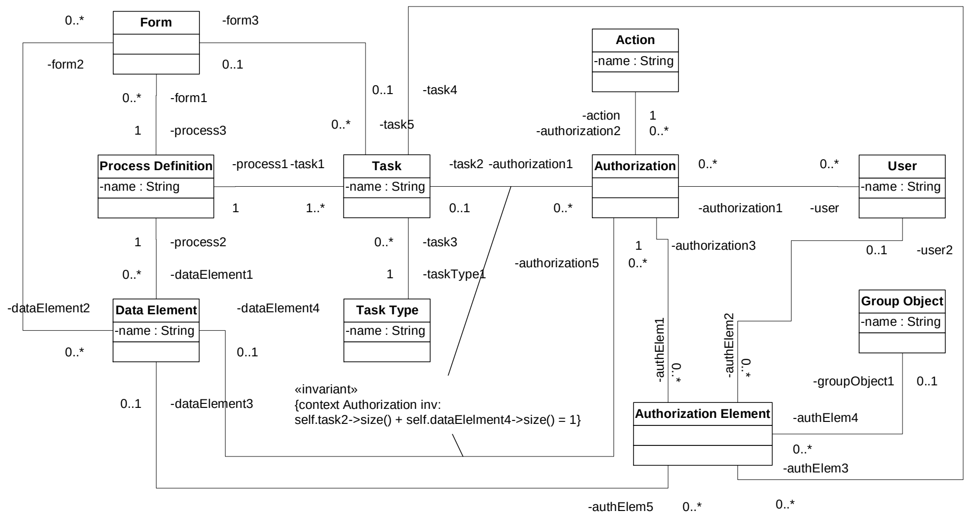


Fig. 6. Resource Allocation Meta Model

Some of these concepts are common for Staffware, FileNet, FLOWer and the meta model. However, they are often named differently, as Table 6 shows. Table 7 shows the constraints for the Resource Allocation models of Staffware, FileNet and FLOWer.

Table 6. Matching of Concepts in Systems - Resource Allocation

Meta Model	**Staffware**	**FileNet**	**FLOWer**
Process Definition	Process	Process	Plan
Data Element	Field	Data Element	Data Element
Task	Task	Step	Step
Form	Form	Form	Form
User		User	

Table 7. Resource Allocation Constraints

context	constraint name	code	description
Task Type	Task Type	A.TT	types of tasks in the system;
Task	User Interface	A.UI	user interface is necessary for tasks that users execute;
	Task Authorization	A.TA	task authorization is necessary for tasks that users execute;
Action	Activity Type	A.AT	execute, open, skip and redo are possible activities;
Authorization	Authorization Object	A.AO	which workflow elements can be authorized (e.g., tasks, data elements, etc.);
Authorization Element	Authorization Subject	A.AS	which workflow elements can be used to define an authorization (e.g., user-names, group objects, data elements, etc.);

4.2 Resource Allocation Constraints for Three Workflow Management Systems

Staffware. Table 8 shows the constraints that map the Staffware Resource Allocation model to the meta model. Constraint (SW-A.TT) defines possible task types. Only if a task is of the type "User Step", then it is assigned a form that will be used for the interaction with the user, as shown in (SW-A.UI). Authorizations are specified in Staffware using address objects that determine which users are allocated for the execution (SW-A.AT) of the task. In the case of a "User Step" task, constraint (SW-A.A) requires at least one address object to be specified. Authorization in Staffware can be created using user-names, group objects, and data fields, as shown in constraint (SW-A.AO). Moreover, authorization is possible only for tasks, and not for data fields, as defined in (SW-A.AS).

Table 8. OCL Constraints for Resource Allocation Model in Staffware

context Task Type	
inv (SW-A.TT):	Task Type.allInstances()->collect(name) = Bag {'Event','User Step', 'Step', 'Automatic Step'}
context Task	
inv (SW-A.UI):	self.taskType1.name <> 'User Step' implies self.form3->isEmpty() and self.taskType1.name = 'User Step' implies self.form3->notEmpty()
inv (SW-A.A):	self.taskType1.name <> 'User Step' implies self.authorization1->forAll(authElem1->isEmpty()) and self.taskType1.name = 'User Step' implies self.authorization1->forAll(authElem1->notEmpty())
context Action	
inv (SW-A.AT):	Action.allInstances()->size() = 1 and Action.allInstances()->forAll(name = 'execute')
context Authorization	
inv (SW-A.AO):	self.dataElement4->isEmpty()
context Authorization Element	
inv (SW-A.AS):	self.task4->isEmpty() and (self.user2->size() + self.groupObject1->size()+ self.dataElement3->size()) = 1

FileNET. Various FileNET concepts impose constraints on the Resource Allocation meta model and they are presented in Table 9. Task types that exist in FileNET are shown in constraint (FN-A.TT). Tasks of the "General Step" task type are offered to users. These taks must be assigned a form for interaction with users and authorization for allocation of users, as shown in (FN-A.UI) and (FN-A.A). The only action users can take is to execute tasks, as constraint

(FN-A.AT) shows. Constraint (FN-A.AO) specifies that authorizations cannot be created for data fields. Group objects and users can be used as authorization elements in FileNET (FN-A.AS).

Table 9. OCL Constraints for Resource Allocation Model in FileNET

context Task Type	
inv (FN-A.TT):	Task Type.allInstances()->collect(name) = Bag {‘System Step’, ‘Launch Step’, ‘Component Step’, ‘Sub-Map Step’, ‘General Step’}
context Task	
inv (FN-A.UI):	self.taskType1.name <> ‘General Step’ implies self.form3->isEmpty() and self.taskType1.name = ‘General Step’ implies self.form3->notEmpty()
inv (FN-A.A):	self.taskType1.name <> ‘General Step’ implies self.authorization1->isEmpty() and self.taskType1.name = ‘General Step’ implies self.authorization1->notEmpty()
context Action	
inv (FN-A.AT):	Action.allInstances()->size() = 1 and Action.allInstances()->forAll(name = ‘execute’)
context Authorization	
inv (FN-A.AO):	self.dataElement4->isEmpty()
context Authorization Element	
inv (FN-A.AS):	self.task4->isEmpty() and self.dataElement3->isEmpty()

FLOWer. FLOWer, being a case-handling system, has some unique features that are not supported by traditional workflow management systems. Some of the features that can be found in the FLOWer Resource Allocation model we will not consider when constraining the Resource Allocation meta model. First, we will not consider the class “Case Type” because this class is used as a reference class for one instance of the FLOWer model – it consists of the process definition, hierarchy of roles etc. Second, we abstract from the various kinds of plans. A *process definition* corresponds to a *plan* in FLOWer. There are different kinds of plans (i.e., “Static Plan”, “Sequential Plan”, “Dynamic”, etc.) and they represent different routing concepts linking sub-processes to tasks. Finally, there is a hierarchy of roles defined in every case type and these roles are used for authorizations. Each of these roles is addressed in the Organization model by “Role” group objects, and here we will assume authorizations to directly refer to the group object, rather than internally defined “Case Role”. After “ruling out” these concepts we present the constraints that are used to describe the FLOWer Resource Allocation model using the Resource Allocation meta model. Constraint (FW-A.TT) defines several task types that are available, while (FW-A.UI) allows a user interface only for two types of tasks. Constraint (FW-A.AT) shows that users can have authorizations to execute, skip, open, redo and read tasks. Constraint (FW-A.AS) requires that authorization elements cannot address directly users, but only group objects of the group type “Role” and allows for referring to other tasks for authorization.

5 Work Distribution

Workflow management systems distribute work to users based on resource allocation rules (which are specified in the process definition) and the organizational structure. Work distribution determines the way in which the work is offered to and executed by users. In this section we present the Work Distribution meta model and constraints that describe t0he work distribution mechanisms of Staffware, FileNet and FLOWer.

Table 10. OCL Constraints for Resource Allocation Model in FLOWer

context Task Type	
inv (FW-A.TT):	Task Type.allInstances()->collect(name) = Bag {‘Plan Element’,‘Form Fill Action’, ‘Action Action’, ‘Operation Action’, ‘Automatic Action’}
context Task	
inv (FW-A.UI):	(self.taskType1.name <> ‘Plan Element’ or self.taskType1.name <> ‘Action Action’) implies self.form3->isEmpty() and (self.taskType1.name = ‘Plan Element’ or self.taskType1.name = ‘Action Action’) implies self.form3->notEmpty()
context Action	
inv (FW-A.AT):	Action.allInstances()->collect(name) = Bag{‘execute’, ‘skip’, ‘open’, ‘redo’, ‘read’ }
context Authorization Element	
inv (FW-A.AS):	self.user2->isEmpty() and self.dataElement3->isEmpty() and self.groupObject1->size() = 1 and self.groupObject1->forAll(groupType1.name = ‘Role’)

5.1 Work Distribution Meta Model

The meta model of Work Distribution is shown in Figure 7. Every *case* initiates a *process definition.* Enabled *tasks* in a case are *work items.* Work items are distributed to *work queues* – pools of work that are accessible to *users.* The three systems we have observed have two general types of work queues: (1) *group queues*, which are created for *group objects* and (2) *user queues*, which are created for individual *users* as their personal queues. Group queues are accessible for all users that are members of the particular group object and user queues are accessible only by a single user. Work items can be *not-processed work items* or *processed work items.* Users can also choose to skip a not-processed work item, and change its state into a processed work item. Not-processed work items that are available for execution are called *enabled work items. Waiting work items* are not-processed items that are not enabled yet. When user opens a waiting work item (s)he explicitly changes its state into an enabled work item. Users can execute only enabled work items. When an enabled work item is executed, its state changes into a processed work item. While users execute enabled work items, they can change *data values* of available *data elements.* A user can choose to reopen a processed work item and change its state into enabled work item.

Table 11 shows which concepts in the Work Distribution meta model can be also found in the Work Distribution models of Staffware, FileNet and FLOWer. In Table 12 we show the constraints that describe the three workflow management systems.

Table 11. Matching of Concepts in Systems - Work Distribution

Meta Model	**Staffware**	**FileNET**	**FLOWer**
Task	User Step	General Step	Step
Work Item	Work Item	Work Item	Work Item
Processed Work Item	Activity	Activity	Processed Work Item
Not-Processed Work Item			Not-Processed Work Item
Waiting Work Item			Waiting Work Item
Enabled Work Item			Enabled Work Item
Work Queue	Work Queue	Queue	
User Queue		User Queue	
Case	Case	Case	Case
Process Definition	Process	Process	Case Type

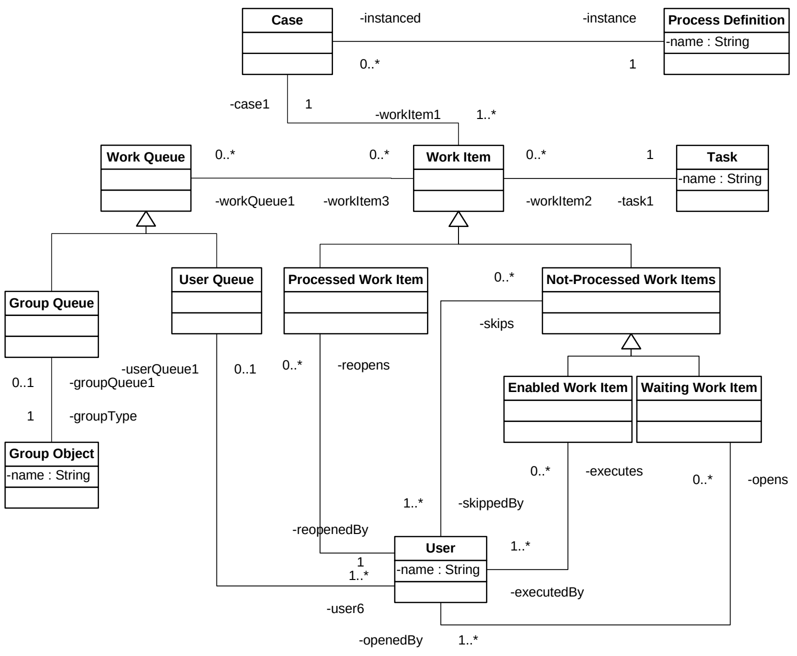


Fig. 7. Work Distribution Model

Table 12. Work Distribution Constraints

context	constraint name	code	description
User	User Action	D.UA	types of activities that users can perform in the system;
	Personal Queue	D.PQ	every personal queue is created for one user;
Group Object	Group Queue	D.GQ	a group queue is created for one group object;

5.2 Work Distribution Constraints for Three Workflow Management Systems

Staffware. In order to describe the Work Distribution model of Staffware using the meta model, we have developed a number of OCL constraints, which are shown in Table 13. While working with work items in Staffware, users have only the possibility to execute enabled items, i.e., it is not possible to skip, open or redo work items. Constraint (SW-D.UA) describes that concepts of skipped, opened and reopened work items do not exist in Staffware. There are two types of work queues in Staffware: (1) constraint (SW-D.PQ) shows that for every user there is a personal work queue, and (2) constraint (SW-D.GQ) states that for every group[3] one work queue is created.

FileNET. Table 14 shows constraints which can be used to describe how Work Distribution model of FileNET can be mapped to the meta model. It is not possible in FileNET to skip, open and reopen work items, as can be seen in constraint (FN-D.UA). Personal work queues are created for every user, as shown in (FN-D.PQ). Constraint (FN-D.GQ) shows that one work queue is created for every group object from the group type “Work Queue”.

FLOWer. Table 15 displays one constraint that maps work distribution model of FLOWer to the meta model. Constraint (FN-D.GQ) shows that work queues are created only for group objects that are of the “Work Profile” group type.

[3] Here we refer to a group object of the group type “Group” in Staffware.

Table 13. OCL Constraints for Work Distribution Model in Staffware

context User	
inv (SW-D.UA):	self.skips->isEmpty() and self.opens->isEmpty() and self.reopens->isEmpty()
inv (SW-D.PQ):	self.userQueue1->size() = 1
context Group Object	
inv (SW-D.GQ):	(self.groupType1.name = ”Group” implies self.groupQueue1->size() = 1) and (self.groupType1.name <> ”Group” implies self.groupQueue1->isEmpty())

Table 14. OCL Constraints for Work Distribution Model in FileNET

context User	
inv (FN-D.UA):	self.skips->isEmpty() and self.opens->isEmpty() and self.reopens->isEmpty()
inv (FN-D.PQ):	self.userQueue1->size() = 1
context Group Object	
inv (FN-D.GQ):	(self.groupType1.name = ”Work Queue” implies self.groupQueue1->size() = 1) and (self.groupType1.name <> ”Work Queue” implies self.groupQueue1->isEmpty())

Table 15. OCL Constraints for Resource Work Distribution Model in FLOWer

context Group Object	
inv (FN-D.GQ):	(self.groupType1.name = ”Work Profile” implies self.groupQueue1->size() = 1) and (self.groupType1.name <> ”Work Profile” implies self.groupQueue1->isEmpty())

6 Discussion

The meta model consists of the Organization, Resource Allocation and Work Distribution meta models. This meta model is developed as the “least common multiple” of models of the three workflow management systems (i.e. Staffware, FileNet, and FLOWer). For each of the three workflow management systems we have also presented constraints and showed how these constraints, when applied to the meta model, capture the functionality offered by each of the three workflow management systems. Table 16 shows all constraints for each of the workflow management systems.

Staffware, FileNet and FLOWer constrain the Organization part of the meta model in the Group Type, Group Object, Position and Qualification Type contexts. All three systems have a predefined limited number of possible Group Types. Constraints show that all three systems have certain limitations when defining Group Qualifications. Staffware is the only system with a constrained Group Capacity (the Group Type “Role” has a limited capacity). The organization has to be modelled using the system-specific Organizational Hierarchy model in all three systems. Organizational Position in all systems has to be defined for every Group Object independently, i.e., every relation between a User and a Group Object is a unique Organizational Position, although FLOWer uses “Work Profile” as a reference for Organizational Positions. Staffware is the only system with no constraints on User Qualification.

Several constraints have been listed for the Resource Allocation part of the meta model. Every system has its own set of possible Task Types. In every system a user interface is provided only for some types of tasks, because some tasks are automatically executed the workflow management system. Only in FLOWer an authorization is specified for all types of tasks, because the whole case is open to a user in the execution time. All three systems have a set of possible Activity Types, but if we look at the constraints, we can see that FLOWer offers the most possibilities. FLOWer is the only system that enables defining authorization rules both for tasks and data elements, while the other two systems give authorizations only for tasks, as the constraints listed under the constraint name Authorization Object show. All three systems have a limited set of elements defined in the Authorization Subjects constraints.

The Work Distribution module has three possible constraints. FLOWer is the only system that does not constraint the User Actions. This was expected because the Activity Type constraints in the Resource Allocation module show the same functionality. Only in FLOWer personal queues are not created for every user automatically by the system. Group Queues are defined for specific Group Objects in all three systems.

Table 16. Staffware, FileNET and FLOWer constraints

context	constraint name	Staffware	FileNET	FLOWer
Organization				
Group Type	Group Type	SW-O.GT	FN-O.GT	FW-O.GT
	Group Qualification	SW-O.GQ	FN-O.GQ	FW-O.GQ
Group Object	Group Capacity	SW-O.GC		
	Organizational Hierarchy	SW-O.OH	FN-O.OH	FW-O.OH.1-8
Position	Organizational Position	SW-O.OP	FN-O.OP	FW-O.OP.1-2
Qualification Type	User Qualification		FN-O.UC	FW-O.UC
Resource Allocation				
Task Type	Task Type	SW-A.TT	FN-A.TT	FW-A.TT
Task	User Interface	SW-A.UI	FN-A.UI	FW-A.UI
	Task Authorization	SW-A.TA	FN-A.TA	
Action	Activity Type	SW-A.AT	FN-A.AT	FW-A.AT
Authorization	Authorization Object	SW-A.AO	FN-A.AO	
	Authorization Subject	SW-A.AS	FN-A.AS	FW-A.AS
Work Distribution				
User	User Action	SW-D.UA	FN-D.UA	
	Personal Queue	SW-D.PQ	FN-D.PQ	
Group Object	Group Queue	SW-D.GQ	FN-D.GQ	FW-D.GQ

The meta model that is used for the comparison of different systems should be designed as the "least common multiple". This meta model should include and generalize common and system-specific aspects of models that are used for the construction of the meta model. Thus, the meta model generalizes over common and includes system specific properties of observed systems.

The meta model includes properties of various systems. To describe which features of the meta model exist in a specific system and which do not exist in the system, we constraint the meta model. By doing this, we "narrow" (limit, constrain) the scope of the meta model to fit the model of a single system. The OCL language proved to be a good basis for defining structured and exact constraint expressions.

Constraints expressions provide for a good overview of the critical concepts in the meta model. These are the concepts that are the most often interpreted differently among different systems. Constraints also show the concepts that some systems limit and others do not.

The expressions themselves precisely describe the way the system interprets concepts from the meta model. However, due to the complexity of systems, some expressions are too long and complex to read and understand.

7 Related Work

Since the early nineties workflow technology has matured [23] and several textbooks have been published, e.g., [5, 18, 29, 33, 37]. During this period many languages for modelling workflows have been proposed, i.e., languages ranging from generic Petri-net-based languages to tailor-made domain-specific languages. The Workflow Management Coalition (WfMC) has tried to standardize workflow languages since 1994 but failed to do so [22]. XPDL, the language proposed by the WfMC, has semantic problems [2] and is rarely used. In a way BPEL [11] succeeded in doing what the WfMC was aiming at. However, both BPEL and XPDL focus on the control-flow rather than the resource perspective.

Despite the central role that resources play in workflow management systems, there is a surprisingly small body of research into resource and organizational modelling in the workflow context [1, 31]. In early work, Bussler and Jablonski [15] identified a number of shortcomings of workflow management systems when modelling organizational and policy issues. In subsequent work [29], they presented one of the first broad attempts to model the various perspectives of workflow management systems in an integrated manner including detailed consideration of the organizational/resource view.

One line of research into resource modelling and enactment in a workflow context has focused on the characterization of resource managers that can manage organizational resources and enforce resource policies. In [17], the design of a resource manager is presented for a workflow management system. This work includes a high-level resource model together with proposals for resource definition, query and policy languages. Similarly, in [32], an abstract resource model is presented in the context of a workflow management system although the focus is more on the efficient management of resources in a workflow context than the specific ways in which work is allocated to them. In [28], a proposal is presented for handling resource policies in a workflow context. Three types of policy – qualification, requirement and substitution – are described together with a means for efficiently implementing them when allocating resources to activities.

Another area of investigation has been into ensuring that only appropriate users are selected to execute a given work item. The RBAC (Role-Based Access Control) model [20] presents an approach for doing this. RBAC models are effective but they tend to focus on security considerations and neglect other organizational aspects such as resource availability.

Flexibility has been a research topic in workflow literature since the late nineties [4, 7, 9, 10, 16, 19, 27, 30, 40, 41, 49]. Flexibility triggers all kinds of interesting research questions, e.g., if a process changes how this should influence the running cases? [7]. Examples of qualitative analysis of flexibility of workflow management system can be found in [13] and [24]. One way of allowing for more flexibility is to use the case handling concept as defined in [3, 9]. FLOWer [12, 39] can be seen as a reference implementation of the case handling concept. Therefore, its resource perspective was modeled in this paper. Besides FLOWer there are few other case handling tools: E.C.H.O. (Electronic Case-Handling for Offices), a predecessor of FLOWer, the Staffware Case Handler [47] and the COSA Activity Manager [46], both based on the generic solution of BPi [14], Vectus [34, 35], and the open-source system con:cern (http://con-cern.org/).

Work distribution of various workflow management systems was investigated within the *workflow patterns initiative* (cf. www.workflowpatterns.com). Besides a variety of control-flow [6] and data [44] patterns, 43 resource patterns [43, 45] have been defined. This paper is the most related to the resource patterns[43, 45]. However, in this paper we use a completely different approach. Instead of patterns, we use meta models to unify functionalities and then specialize using OCL constraints.

Several researchers have developed meta-models, i.e., object models describing the relation between workflow concepts, which include work allocation aspects, cf. [8, 36–38, 42]. The work reported in this paper can be seen as an extension of the meta-modelling approach. We add OCL constraints to meta models that provide aid for understanding, analysis, evaluation and comparison of various workflow management systems.

8 Conclusion

This paper focuses on the *resource perspective*, i.e., the way workflow management systems distribute work based on the structure of the organization and capabilities/qualifications of people. We use UML models and OCL constraint expressions to develop a structured and exact method for analysis, evaluation and comparison of workflow management systems. We start with developing models of particular systems. Based on particular models we develop a generalized meta model. This model is further specialized for every system-specific model using OCL constraint expressions. These expressions show how the model of each of the workflow management systems can be extracted from the meta model. Using OCL expressions we show which concepts are interpreted differently in systems and which concepts are shared. The meta model can be seen as an abstraction of a powerful workflow management system that can be constrained to act like one of the three systems we have observed: Staffware, FileNet, and FLOWer.

Workflow management systems tend to implement many system-specific features. This makes it hard to analyze, evaluate, compare and select an appropriate system. This paper showed that using more exact tools (i.e., UML class diagrams and OCL expressions) to describe different systems provides a structured and exact method to describe various workflow management systems.

References

1. W.M.P. van der Aalst. Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, 18(1):72–76, 2003.
2. W.M.P. van der Aalst. Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 1–65. Springer-Verlag, Berlin, 2004.
3. W.M.P. van der Aalst and P.J.S. Berens. Beyond Workflow Management: Product-Driven Case Handling. In S. Ellis, T. Rodden, and I. Zigurs, editors, *International ACM SIGGROUP Conference on Supporting Group Work (GROUP 2001)*, pages 42–51. ACM Press, New York, 2001.
4. W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors. *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2000.
5. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems.* MIT press, Cambridge, MA, 2002.
6. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
7. W.M.P. van der Aalst and S. Jablonski. Dealing with Workflow Change: Identification of Issues and Solutions. *International Journal of Computer Systems, Science, and Engineering*, 15(5):267–276, 2000.
8. W.M.P. van der Aalst and A. Kumar. Team-Enabled Workflow Management Systems. *Data and Knowledge Engineering*, 38(3):335–363, 2001.
9. W.M.P. van der Aalst, M. Weske, and D. Grünbauer. Case Handling: A New Paradigm for Business Process Support. *Data and Knowledge Engineering*, 53(2):129–162, 2005.
10. A. Agostini and G. De Michelis. Improving Flexibility of Workflow Management Systems. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 218–234. Springer-Verlag, Berlin, 2000.
11. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.1. Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2003.
12. Pallas Athena. *Case Handling with FLOWer: Beyond workflow.* Pallas Athena BV, Apeldoorn, The Netherlands, 2002.
13. J. Bowers, G. Button, and W. Sharrock. Workflow From Within and Without: Technology and Cooperative Work on the Print Industry Shopfloor. In *The Fourth European Conference on Computer-Supported Cooperative Work (ECSCW 95)*, pages 51–66, Stockholm, September 1995. Kluwer Academic Publishers, Dordrecht, The Netherlands.
14. BPi. *Activity Manager: Standard Program - Standard Forms (Version 1.2).* Workflow Management Solutions, Oosterbeek, The Netherlands, 2002.
15. C. Bussler and S. Jablonski. Policy Resolution for Workflow Management Systems. In *Proceedings of the 28th Hawaii International Conference on System Sciences*, page 831. IEEE Computer Society, 1995.
16. F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow Evolution. In *Proceedings of ER '96*, pages 438–455, Cottubus, Germany, Oct 1996.

17. W. Du and M.C. Shan. Enterprise Workflow Resource Management. In *Ninth International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises (RIDE-VE'99)*, pages 108–115, Sydney, Australia, 1999. IEEE Computer Society Press.
18. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems*. Wiley & Sons, 2005.
19. C.A. Ellis and K. Keddara. A Workflow Change Is a Workflow. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 201–217. Springer-Verlag, Berlin, 2000.
20. D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.
21. FileNET. *FileNet Business Process Manager 3.0*. FileNET Corporation, Costa Mesa, CA, USA, June 2004.
22. L. Fischer, editor. *Workflow Handbook 2003, Workflow Management Coalition*. Future Strategies, Lighthouse Point, Florida, 2003.
23. D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.
24. R.E. Grinter. Workflow Systems: Occasions for Success and Failure. *Computer Supported Cooperative Work*, 9(2):189–214, 2000.
25. Object Management Group. *UML 2.0 Infrastructure Final Adopted Specifcation*. Object Management Group, ptc/03-09-15 edition, December 2003.
26. Object Management Group. *UML 2.0 OCL Final Adopted Specification*. Object Management Group, ptc/03-10-14 edition, October 2003.
27. T. Herrmann, M. Hoffmann, K.U. Loser, and K. Moysich. Semistructured models are surprisingly useful for user-centered design. In G. De Michelis, A. Giboin, L. Karsenty, and R. Dieng, editors, *Designing Cooperative Systems (Coop 2000)*, pages 159–174. IOS Press, Amsterdam, 2000.
28. Y.N. Huang and M.C. Shan. Policies in a Resource Manager of Workflow Systems: Modeling, Enforcement and Management. Technical Report HP Tech. Report, HPL-98-156, Palo Alto, CA, USA, 1999. Accessed at http://www.hpl.hp.com/techreports/98/HPL-98-156.pdf on 20 March 2005.
29. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.
30. M. Klein, C. Dellarocas, and A. Bernstein, editors. *Adaptive Workflow Systems*, volume 9 of *Special issue of the journal of Computer Supported Cooperative Work*, 2000.
31. A. Kumar, W.M.P. van der Aalst, and H.M.W. Verbeek. Dynamic Work Distribution in Workflow Management Systems: How to Balance Quality and Performance? *Journal of Management Information Systems*, 18(3):157–193, 2002.
32. B.S. Lerner, A.G. Ninan, L.J. Osterweil, and R.M. Podorozhny. Modeling and Managing Resource Utilization in Process, Workflow, and Activity Coordination. Technical Report UM-CS-2000-058, Department of Computer Science, University of Massachusetts, August 2000. Accessed at http://laser.cs.umass.edu/publications/?category=PROC on 20 March 2005.
33. F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
34. London Bridge Group. *Vectus Application Developer's Guide*. London Bridge Group, Wellesbourne, Warwick, UK, 2001.
35. London Bridge Group. *Vectus Technical Architecture*. London Bridge Group, Wellesbourne, Warwick, UK, 2001.
36. M. Zur Muehlen. Evaluation of Workflow management Systems Using Meta Models. In *Proceedings of the 32nd Hawaii International Conference on System Sciences - HICSS'99*, pages 1–11, 1999.
37. M. Zur Muehlen. *Workflow-based Process Controlling: Foundation, Design and Application of workflow-driven Process Information Systems*. Logos, Berlin, 2004.
38. M. zur Muehlen. Organizational Management in Workflow Applications Issues and Perspectives. *Information Technology and Management*, 5(3–4):271–291, July-October 2004.
39. Pallas Athena. *Flower User Manual*. Pallas Athena BV, Apeldoorn, The Netherlands, 2002.
40. M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
41. S. Rinderle, M. Reichert, and P. Dadam. Correctness Criteria For Dynamic Changes in Workflow Systems: A Survey. *Data and Knowledge Engineering*, 50(1):9–34, 2004.

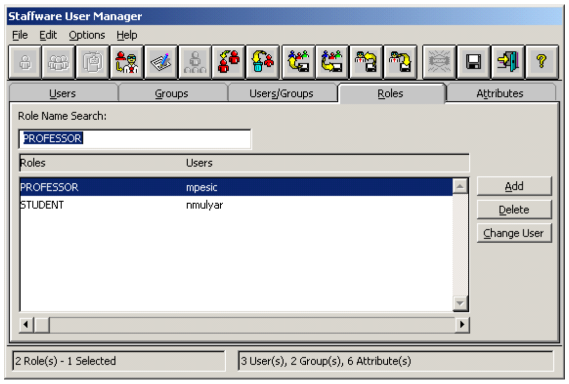
42. M. Rosemann and M. Zur Muehlen. Evaluation of Workflow Management Systems - a Meta Model Approach. *Australian Journal of Information Systems*, 6(1):103–116, 1998.
43. N. Russell, W.M.P.van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Workflow Resource Patterns: Identification, Representation and Tool Support. In O. Pastor and J. Falcao e Cunha, editors, *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05)* , volume 3520 of *Lecture Notes in Computer Science*, pages 216–232. Springer-Verlag, Berlin, 2005.
44. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Data Patterns. QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane, 2004.
45. N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Workflow Resource Patterns. BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, 2004.
46. Software-Ley. *COSA Activity Manager*. Software-Ley GmbH, Pullheim, Germany, 2002.
47. Staffware. *Staffware Case Handler – White Paper*. Staffware PLC, Berkshire, UK, 2000.
48. Staffware. *Using the Staffware Process Client*. Staffware, plc, Berkshire, United Kingdom, May 2002.
49. M. Weske. Formal Foundation and Conceptual Design of Dynamic Adaptations in a Workflow Management System. In R. Sprague, editor, *Proceedings of the Thirty-Fourth Annual Hawaii International Conference on System Science (HICSS-34)*. IEEE Computer Society Press, Los Alamitos, California, 2001.

Appendix: Models of Three Workflow Management Systems

We developed models (UML class diagrams) of work distribution of three commercial workflow management systems: Staffware 9.0 [48], FileNet Business Process Manager 3.0 [21] and FLOWer 2.05a [39]. FileNet and Staffware are examples of two widely used traditional workflow management systems. FLOWer is based on the case-handling paradigm, which can be characterized as "the more flexible approach" [3, 9]. Each of the models we have developed will be described in the remainder of this appendix.

A Staffware

Organization. Staffware Process Administrator contains the User Manager tool (Figure 8), which is used to model the organizational structure in Staffware.



Role Name Search: PROFESSOR

Roles	Users
PROFESSOR	mpesic
STUDENT	nmulyar

2 Role(s) - 1 Selected

3 User(s), 2 Group(s), 6 Attribute(s)

Fig. 8. Staffware - User Manager

Figure 9 shows the Organization model of Staffware. *Organizational entities* can be *users* and *groups*. One group can have multiple users as its members and one user can be a member of multiple groups. Users can supervise other organizational entities. *Attributes* are used to define specific qualifications for organizational entities and one can later assign concrete *values* for different organizational entities, e.g. users can speak different foreign languages. Capabilities of users are represented by their *roles*. Although one user can have many roles, one role can be assigned only to one user.

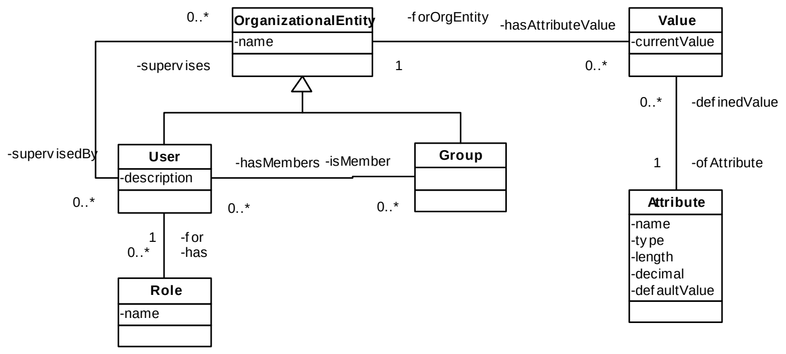
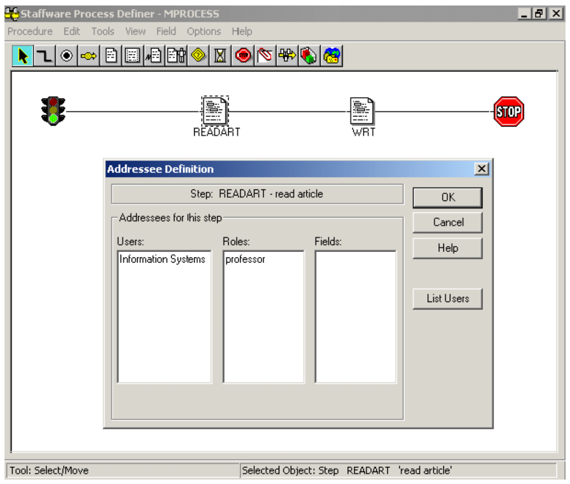


Fig. 9. Staffware - Organization

Resource Allocation. Resource allocation is specified while the process is defined in the tool Process Definer (Figure 10), a part of the Staffware Process Client.


Staffware Process Definer - MPROCESS
Procedure Edit Tools View Field Options Help
READART
WRT
STOP
Addressee Definition
OK
Cancel
Help
List Users
Users:
Roles:
Fields:

Step: READART - read article

Addressees for this step

Users: Information Systems

Roles: professor

Tool: Select/Move

Selected Object: Step READART 'read article'

Fig. 10. Staffware - Process Definer

Figure 11 shows that a *process* is a set of *tasks* in the Resource Allocation model of Staffware. Tasks can be of various types: *event*, *user step*, *step* and *automatic step*. Only the user step has a specified *address* and associated *form*. The address allocates the users who can execute the user step, and the form serves as an interface for the task execution. *Static addresses* specify the *organizational entities* or *roles* of the users who are authorized to execute a task, while *dynamic addresses* refer to one of the *fields* in the process definition. These fields represent variable data values that users can read and alter on forms during the task execution. The relation that an automatic step has with address and form might be surprising because automatic step is a special kind of step which is used to execute a Server process [48]. Although these steps are called "automatic", i.e., should not need an user to execute them, they are assigned an address and a form. In the case of an automatic step, the address should refer to only one user (not a group) under who's authorization the step should be executed. The form is used in the automatic step to provide input and output data for the Server process that should be executed in the automatic step.

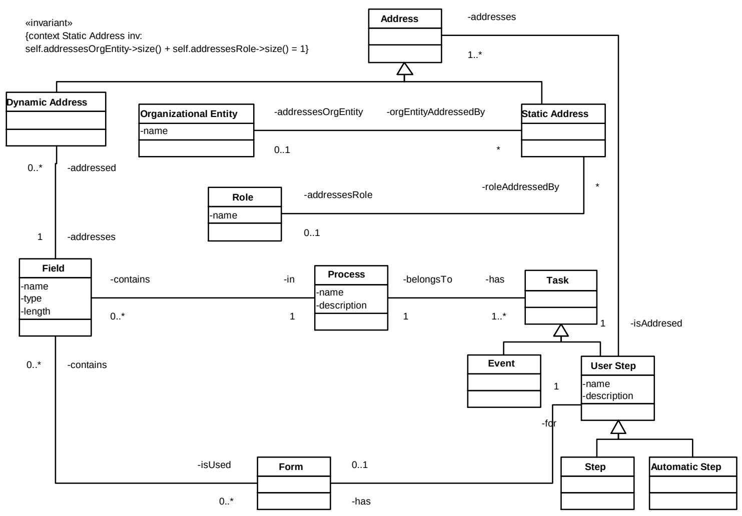
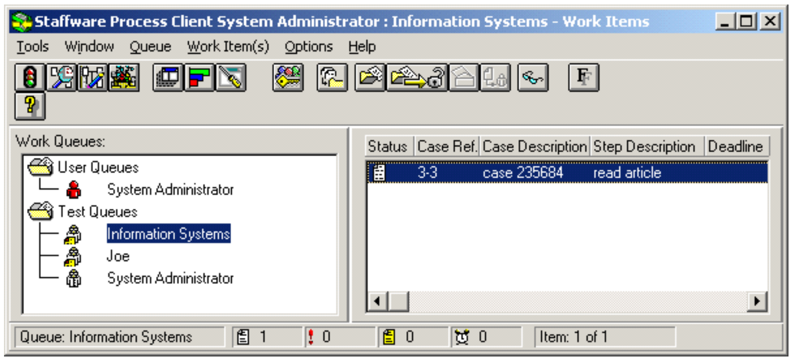


Fig. 11. Staffware - Resource Allocation

Work Distribution. When working with Staffware, users can select and execute their work using the Process Client tool. As Figure 12 shows, work items are distributed to various work queues and made available to the members of these queues.



Status	Case Ref.	Case Description	Step Description	Deadline
	3-3	case 235684	read article	

Fig. 12. Staffware - Process Client

The Staffware Work Distribution model is shown in Figure 13. A *case* is one (executable) instance of a *process*. *Work items* are *user steps* form a case, and are offered to *users* via *work queues*. A work queue is created for every *organizational entity*. Thus, users have access to one personal work queue and to work queues of the groups they are members of. *Activity* is a work item that one user executes. While performing activities users alter *values* of *fields*. Fields that are used in dynamic addresses for resource allocation have values that refer to *roles* or organizational entities.

B FileNet

Organization. The Organizational model of FileNet is presented in Figure 14. *Users* are local users (*OS User*) or groups (*OS Group*) in the running operating system (*OS*). Membership of OS groups are also defined in the operating system, externally form the FileNet system. *Workflow*

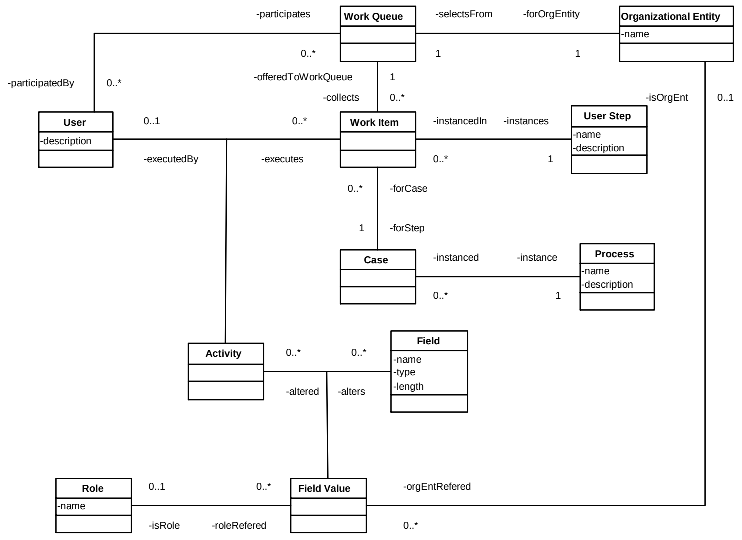


Fig. 13. Staffware - Work Distribution

groups and *work queues* can be defined to represent grouping of users and can have multiple users as members. Users can be members of multiple workflow groups and work queues.

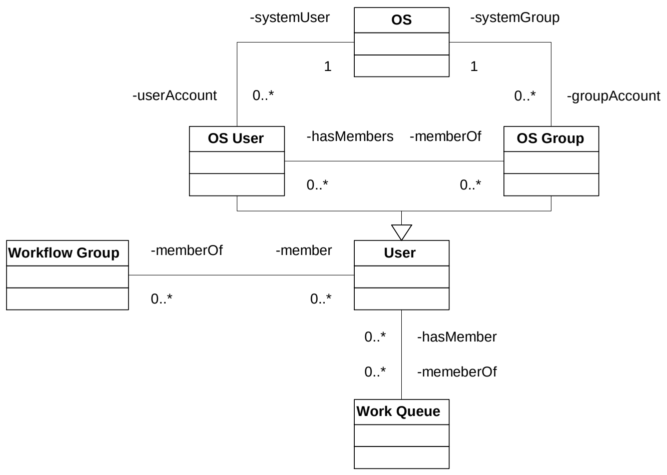
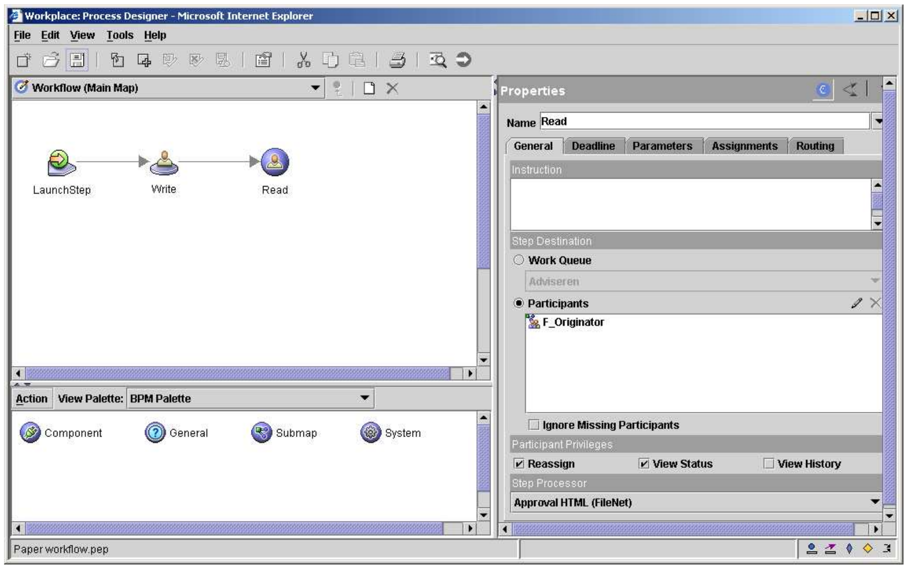


Fig. 14. FileNet - Organization

Resource Allocation. Figure 15 shows the Process Definer tool of FileNet, a tool where within the process definition resources are allocated for tasks.

Figure 18 shows that, in the FileNet Resource Allocation model, a *process* is a set of *steps*. There are various types of steps: *system steps*, *component steps*, *sub-map steps*, *launch steps* and *general steps*. *Workflow groups* (as groups of *users*) are defined in the process definition and they are not valid for any other process. A general step is a step that should be executed by users and it uses forms as an interface. Every general step can have either a *work queue* or a *participant* as the *destination type*. This is used to allocate the task either to a work queue or to users and



Name: Read

Step Destination: Participants — F_Originator

Ignore Missing Participants: ☐

Participant Privileges: ☑ Reassign ☑ View Status ☐ View History

Step Processor: Approval HTML (FileNet)

Fig. 15. FileNet - Process Definer

workflow groups directly. Allocation to participants is used in FileNet to support teamwork where team members can vote for a specified decision.

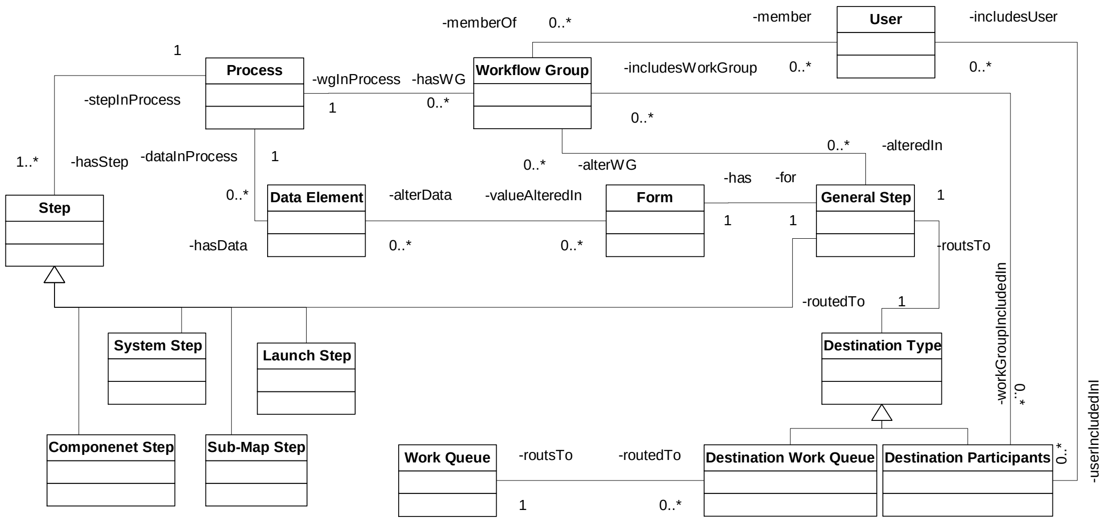
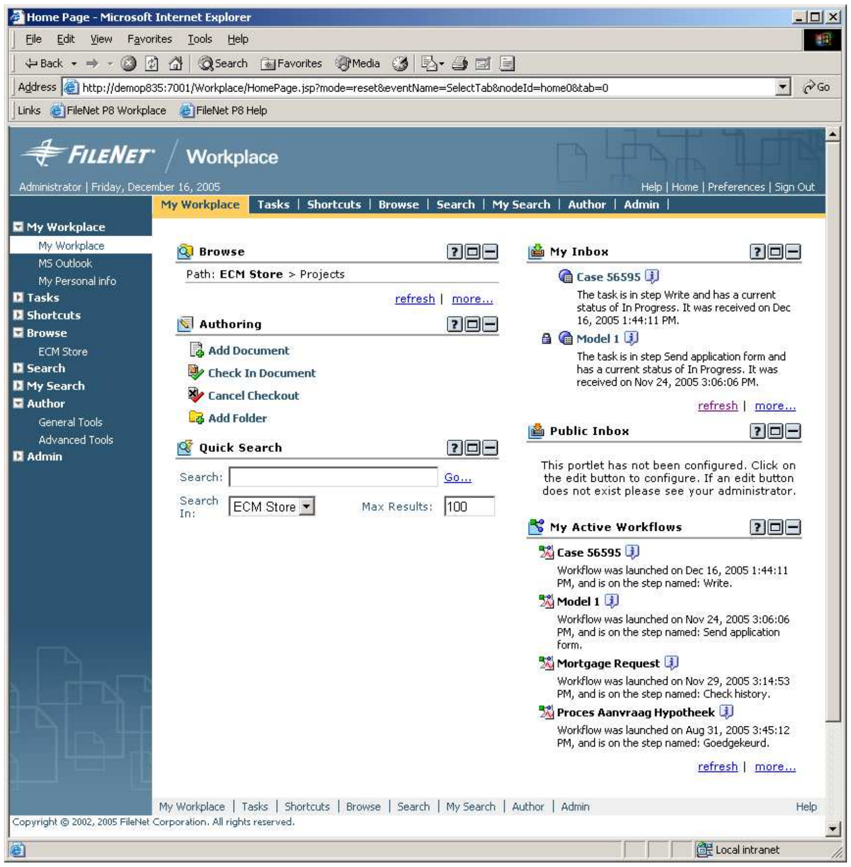


Fig. 16. FileNet - Resource Allocation

Work Distribution. The Workplace is a FileNet tool that presents work items to allocated users. This tool is started in a web browser and it is shown in Figure 17.

Figure 18 shows the model of work distribution in FileNet. *Case* is an *process* instance. *Work item* is a *general step* form a case that is available for *users* via *queues*. *Work queues* are accessible for all members and *user queues* are personal queues for every user. *Activity* is a work item that is executed by a user.


Home Page - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://demop835:7001/Workplace/HomePage.jsp?mode=reset&eventName=SelectTab&nodeId=home0&tab=0

Links FileNet P8 Workplace FileNet P8 Help

FILENET / Workplace

Administrator | Friday, December 16, 2005

Help | Home | Preferences | Sign Out

My Workplace | Tasks | Shortcuts | Browse | Search | My Search | Author | Admin |

- My Workplace
  - My Workplace
  - MS Outlook
  - My Personal info
- Tasks
- Shortcuts
- Browse
  - ECM Store
- Search
- My Search
- Author
  - General Tools
  - Advanced Tools
- Admin

Browse

Path: ECM Store > Projects

refresh | more...

Authoring

Add Document

Check In Document

Cancel Checkout

Add Folder

Quick Search

Search: Go...

Search In: ECM Store Max Results: 100

My Inbox

Case 56595

The task is in step Write and has a current status of In Progress. It was received on Dec 16, 2005 1:44:11 PM.

Model 1

The task is in step Send application form and has a current status of In Progress. It was received on Nov 24, 2005 3:06:06 PM.

refresh | more...

Public Inbox

This portlet has not been configured. Click on the edit button to configure. If an edit button does not exist please see your administrator.

My Active Workflows

Case 56595

Workflow was launched on Dec 16, 2005 1:44:11 PM, and is on the step named: Write.

Model 1

Workflow was launched on Nov 24, 2005 3:06:06 PM, and is on the step named: Send application form.

Mortgage Request

Workflow was launched on Nov 29, 2005 3:14:53 PM, and is on the step named: Check history.

Proces Aanvraag Hypotheek

Workflow was launched on Aug 31, 2005 3:45:12 PM, and is on the step named: Goedgekeurd.

refresh | more...

My Workplace | Tasks | Shortcuts | Browse | Search | My Search | Author | Admin Help

Copyright © 2002, 2005 FileNet Corporation. All rights reserved.

Local intranet

Fig. 17. FileNet - Workplace

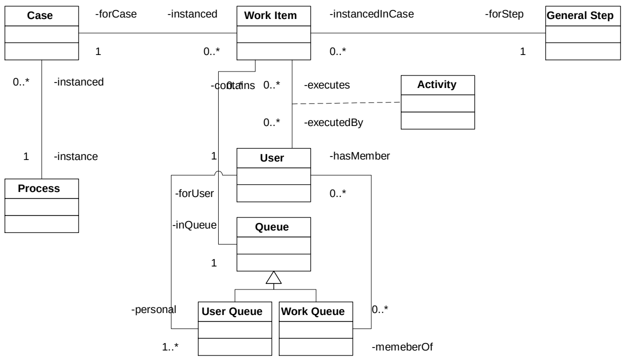


Fig. 18. FileNet - Work Distribution

C FLOWer

Organization. The administrational tool of FLOWer Administrator [4] is shown in Figure 19.

[4] FLOWer system that was used for this paper is installed in Dutch language. The name of the application in Figure 19 is FLOWer *Beheer*, which is Dutch for FLOWer *Administrator*. Further: Rollen is Roles; Functieprofielen is Function Profiles; Werkprofielen is Work Profiles; Distributieprofielen is Distribution Profiles; ZakenZoekers is Case Queries; and Sorteergroepen is Sort Groups in Dutch.

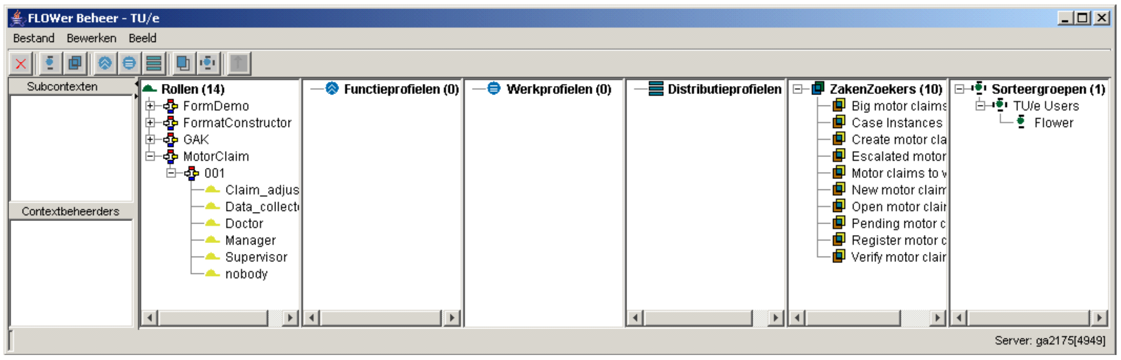


Fig. 19. FLOWer - Administrator

As Figure 20 shows, an organization in FLOWer can be structured into *contexts*. Each context has its *users* which can be members of *user groups* and assigned to *work profiles*. While a user group represents an organizational unit, a work profile represents a group of users with equal distribution rights. A *case type* stores a process definition in FLOWer. Every case type can have multiple *case type releases*. Since a case type has its own definitions of *case roles*, every case type release creates a set of concrete *roles* that correspond to the case type roles. Roles that originate from various case type releases are grouped into *function profiles*, which are further on assigned to work profiles. By using this mechanism we indirectly assign roles to users and define distribution rights in FLOWer.

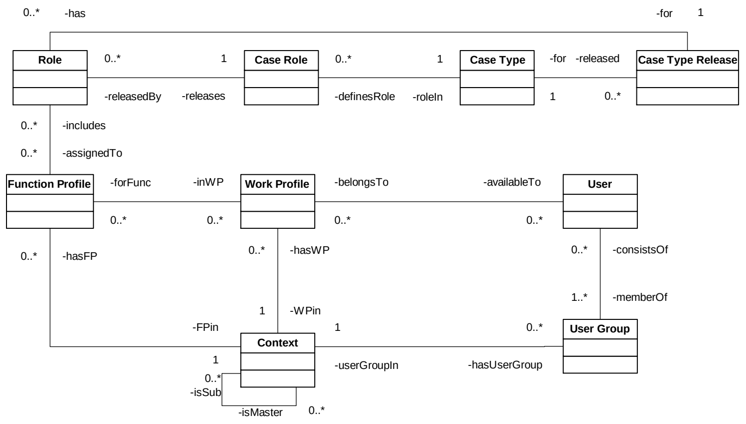
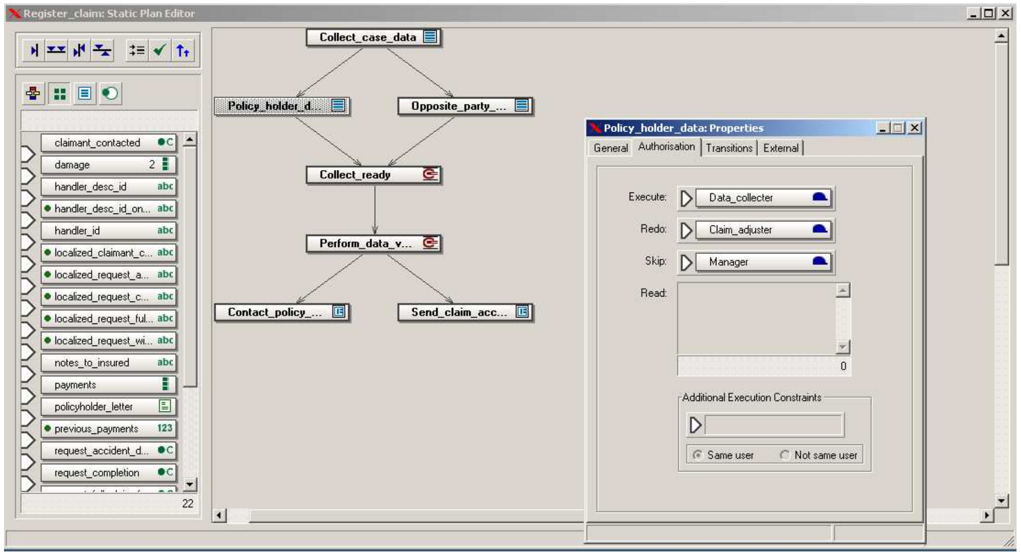


Fig. 20. FLOWer - Organization

Resource Allocation. FLOWer Studio is used to specify a process definition and resource allocation for its tasks. Figure 21 shows how one process definition is open in the FLOWer Studio. Resource allocation is specified for one for the tasks on the “Authorization” tab.

Figure 22 shows the FLOWer Resource Allocation model. A process definition is stored in a *case type*. *Plan* is the building element of a process definition. *Static plans* provide for a sequence in the process, while *sequential* and *dynamic plans* act as iteration elements. *Decision* is used for modelling system and user choices, and it can have at least one *branch*. A *plan element* consists of *steps*, which can be *activity actions*, or can refer to *plan elements*. Activity actions are *form-fill actions* or *external actions*. External *functions* are invoked in *actions*, which are defined globally for the case type. An external action executes one of the case type actions. *Automatic actions* are



Execute: Data_collecter
Redo: Claim_adjuster
Skip: Manager
Read:
0

[✓ Same user] Not same user

Fig. 21. FLOWer - Studio

executed by the FLOWer system, *operation actions* by *users* directly, and *action actions* by users as parts of form-fill actions. A form-fill action is executed by a user using a *form* as an interface. While performing form-fill actions *data elements* from the plan can be viewed and altered by users. Resources are allocated to steps based on the *case roles* and *authorization rules*. Case roles are defined for the case type and they can form a hierarchy among themselves. For every step an authorization defines which case roles are allowed to open, execute, skip, redo and read the step. An *additional constraint* can determine that a step has to be (or must not be) executed by the executor of another step.

Work Distribution. FLOWer, being a case-handling system, distributes whole cases to users. Figure 17 shows that FLOWer Case Query opens the whole case "Motor Claim" is for one user. The user can see and work with various work items that are available in the open case.

A *case* is one instance of a *case type* in FLOWer, as Figure 24 shows. A *work item* is created for *steps* in every case. There are several types of work items, and *users* can perform different actions on them. *Waiting* and *enabled* work items are *not-processed work items*. Only enabled work items can be executed by users. Waiting work items can become enabled when users open them. Users can skip non-processed work items and reopen *processed work items*. If a *case query* uses a case type as a pattern, this means that the case query will present cases (originating from the case type) to users. Case queries are defined for different *contexts* in the organizational system. Administrators can define *interface* that will be presented to users while working with cases in the case query. In a case query it is possible to specify that all cases should start at the specified *action* from the case type. Case types can be presented to users in customized settings using *distribution profiles*. When a distribution profile is assigned to a *work profile* (which is further on assigned to users), a *distribution value* is assigned to the *distribution element* of the distribution profile. The work profile filters the cases on the basis of the specified value of the distribution element. There are two possible types of distribution elements: (1) cases that have the specified value of a *data element* and (2) cases for which the execution process is currently at the specified *process element* (step). Both of these distribution values are specified in the *work profile* and apply for all users that are members of that work profile. Additionally, it is possible to allow users to define *parameter values* if they want to search case queries for predefined *parameters* (data elements).

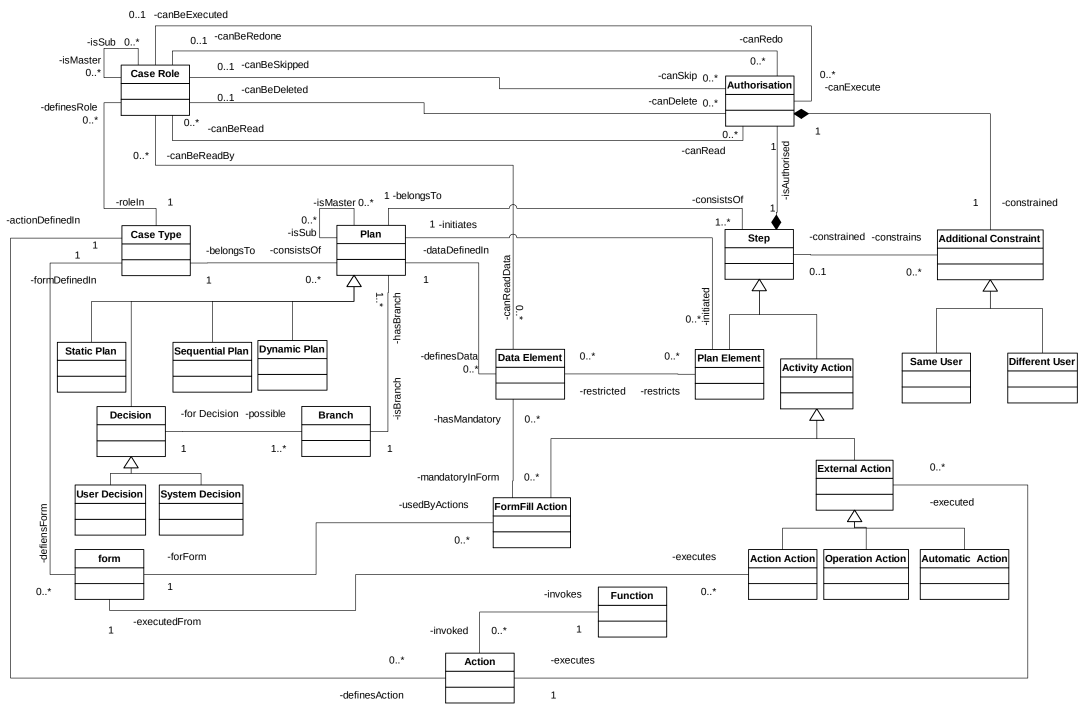
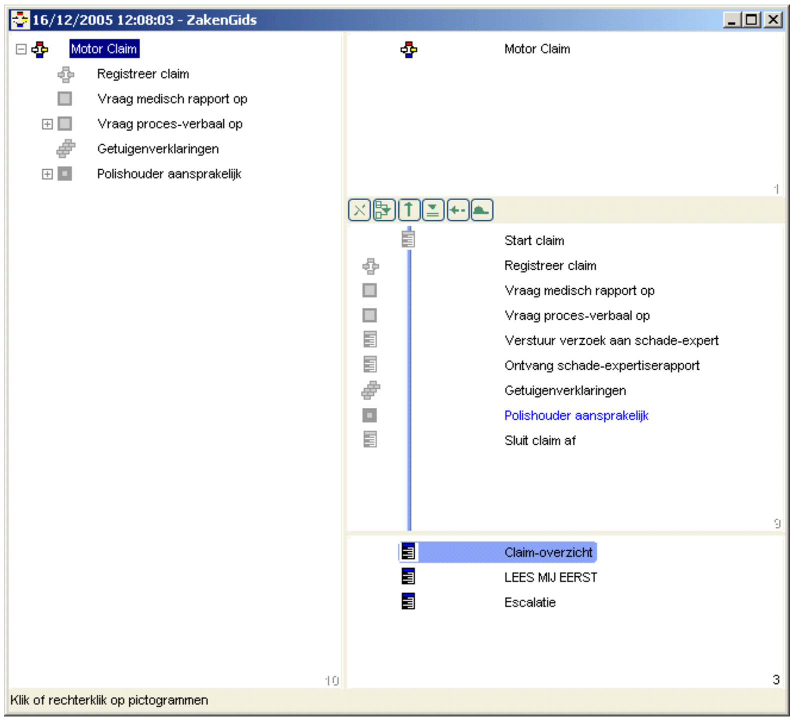


Fig. 22. FLOWer - Resource Allocation



16/12/2005 12:08:03 - ZakenGids

Motor Claim
Registreer claim
Vraag medisch rapport op
Vraag proces-verbaal op
Getuigenverklaringen
Polishouder aansprakelijk

10

Motor Claim

1

Start claim
Registreer claim
Vraag medisch rapport op
Vraag proces-verbaal op
Verstuur verzoek aan schade-expert
Ontvang schade-expertiserapport
Getuigenverklaringen
Polishouder aansprakelijk
Sluit claim af

9

Claim-overzicht
LEES MIJ EERST
Escalatie

3

Klik of rechterklik op pictogrammen

Fig. 23. FLOWer - Case Query

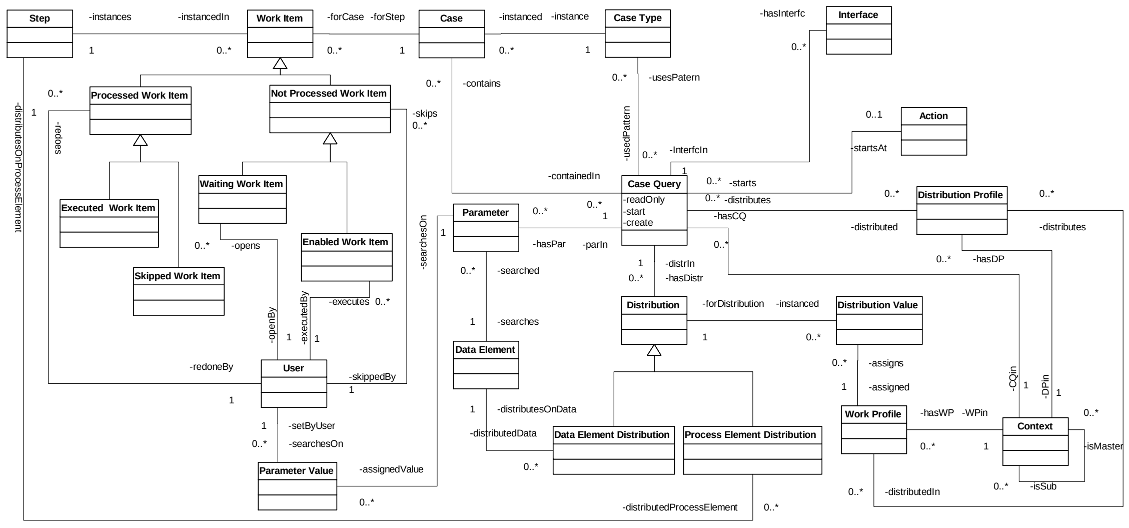


Fig. 24. FLOWer - Work Distribution