Modeling Work Distribution Mechanisms Using Colored Petri Nets

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 their models. These models cover different perspectives including the control-flow, resource, and data perspectives. 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. Contemporary workflow management systems offer a wide variety of mechanisms to support the resource perspective. Because the resource perspective is essential for the applicability of such systems, it is important to better understand the mechanisms and their interactions. Our goal is not to evaluate and compare *what* different systems do, but to understand *how* they do it. We use *Colored Petri Nets* (CPNs) to model work distribution mechanisms. First, we provide a basic model that can be seen as the "greatest common denominator" of existing workflow management systems. This model is then extended for three specific systems (Staffware, FileNet, and FLOWer). Moreover, we show how more advanced work distribution mechanisms, referred to as *resource patterns*, can be modelled and analyzed.

Key words: Work distribution, workflow management, business process management, resource patterns, colored Petri nets.

1 Introduction

Workflow management systems are process-aware information systems [5, 20], which 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. Nevertheless, we believe that existing systems are too limited in this respect. The goal of this paper is not to propose advanced work distribution mechanisms. Instead, we focus on the analysis of functionality in existing systems. The goal is not to evaluate these systems, but to understand *how* they offer specific functionality. Since work distribution defines the quality of work, it is important to consider research from the field of social sciences, e.g., social-technical design [13, 17, 21, 54]. We believe that only by combining both *technical* and *social* approaches, one can truly grasp certain phenomena. A *deeper understanding* of particular aspects of work distribution is essential for developing a new breed of more user-centric systems.

The work reported in this paper can be seen as an extension of the *workflow patterns initiative* [6] (cf. www.workflowpatterns.com). Within the context of this initiative 43 resource patterns [50, 48] have been defined. Using a patterns approach, work distribution is evaluated from the perspective of the end-user as a dynamic property of workflow management systems. The work reported in this paper adds to a better understanding of these mechanisms by providing explicit process models for these patterns, i.e., the descriptive models are augmented with executable models. Most work reported in literature (cf. Section 4) uses static models to describe work distribution. Consider for example the meta modeling approaches presented in [8, 40–42, 47]. These approaches use

static models (e.g., UML class diagrams) to discuss work distribution concepts. This paper takes a truly dynamic model – a *Colored Petri Net* model – as a starting point, thus clearly differentiating our contribution from existing work reported in literature.

Colored Petri Nets (CPNs) [31, 34] are a natural extension of the classical Petri net [45]. There are several reasons for selecting CPNs as the language for modeling work distribution in the context of workflow management. First of all, CPNs have formal semantics and allow for different types of analysis, e.g., state-space analysis and invariants [32]. Second, CPNs are executable and allow for rapid prototyping, gaming, and simulation. Third, CPNs are graphical and their notation is similar to existing workflow languages. Finally, the CPN language is supported by CPN Tools[1] – a graphical environment to model, enact and analyze CPNs.

In this paper, we provide a basic CPN model that can be seen as the "greatest common denominator" of existing workflow management systems. The model will incorporate concepts of a task, case, user, work item, role and group. This model should be seen as a *starting point* towards a more *comprehensive reference model for work distribution*. The basic CPN model is extended and specialized for three specific systems: Staffware [53], FileNet [24], and FLOWer [43]. These three models are used to investigate differences between and similarities among different work distribution mechanisms in order to gain a deeper understanding of these mechanisms. In addition, advanced resource patterns that are not supported by these three systems are modeled by extending the basic CPN model.

The remainder of this paper is organized as follows. Section 2 presents the basic CPN model which should be considered a the "greatest common denominator" of existing workflow management systems. Section 3 extends this model in two directions: (1) Section 3.1 specializes the model for three different systems (i.e., Staffware, FileNet, and FLOWer), and (2) Section 3.2 extends the basic model for selected resource patterns. An overview of related work is given in Section 4. Section 5 discusses our findings and, finally, Section 6 concludes the paper.

2 Basic Model

Different workflow management systems tend to use not only different work distribution concepts, but also completely different terminologies. This makes it difficult to compare these systems. Therefore, we will not start by developing CPN models for different systems and see how these can be unified, but, instead, start with modeling the "greatest common denominator" of existing systems. This model can assist in comparing systems and unifying concepts and terminology. We will use the term *Basic Model* to refer to this "greatest common denominator" and represent it in terms of a CPN model.

In the introduction we already motivated the use of CPNs as a modeling language [31, 34]. A CPN consists of *places* and *transitions* connected by *arcs*. The network structure is static but places can hold *tokens* thus representing the state of the model. The number of tokens per place can vary over time. Moreover, unlike the classical Petri net, tokens can have both a value and a time-stamp. The time-stamps indicate the availability of tokens and can be used to model delays, processing times, timeouts, etc. The value of a token indicates the properties of the object represented by this token. Places (represented by ovals) are typed, i.e., the tokens in a place have values of a particular type (or color in CPN jargon). These types are a subset of the data types in Standard ML such as the primitive types integer and string and compositional types such as tuple, list and record. Each place can hold tokens with values of a certain type. Transitions (represented by rectangles) may consume and produce tokens. Since tokens have values, *arc inscriptions* are needed to specify the input-output relations. Besides the extension with token colors and time-stamps, CPN models allow for hierarchy. Complex models may be decomposed into sub-pages, also referred to as sub-processes or modules, to obtain a layered hierarchical description. A more detailed discussion of the CPN concepts is beyond the scope of this paper. In the remainder, we assume that the reader is familiar with the CPN language and refer to [31, 34] for more details.

[1] CPN Tools can be downloaded from wiki.daimi.au.dk/cpntools/.

The Basic Model represents a workflow management system where the business *process* is defined as a set of *tasks*. Before the process can be initiated and executed, it has to be instantiated. One (executable) instance of a process is referred to as a *case*. Each case traverses the process. If a task is enabled for a specific case, a *work item*, i.e., a concrete piece of work, is created. There is a set of *users* that can execute work items. The users are embedded in the organizational structure on the basis of their *roles*, and the *groups* they belong to. Group is an organizational unit (e.g., sales, purchasing, production, etc.), while role represents a capability of the user (e.g., manager, software developer, accountant, etc.). These concepts are mapped onto CPN types as shown in Table 1. As indicated, CPN uses Standard ML types (e.g., *string* and *int*) and type constructors such as *product* to create pairs and other complex constructs (e.g., *(1,"taskA")* represents a value of type *WI*).

During the work distribution work items change state. The change of state depends on previous and determines the next actions of users and the distribution mechanism. A model of a life cycle of a work item shows how a work item changes states during the work distribution. For more detailed models about life cycle models we refer the reader to literature, e.g., [5, 18, 20, 30, 37, 41]. We have developed and use the life cycle models as aid to describe work distribution mechanisms of each of the workflow systems we have modeled in CPN. The Basic Model uses a simple model of the life cycle of work items and it covers only the general, rather simplified, behavior of workflow management systems (e.g., errors and aborts are not considered). Figure 1 shows the life cycle of a work item of the Basic Model. After the *new* work item has arrived, it is automatically also *enabled* and then taken into distribution (i.e., state *initiated*). Next, the work item is *offered* to the user(s). Once a user *selects* the work item, it is *assigned* to him/her, and (s)he can *start* executing it. After the *execution*, the work item is considered to be *completed*, and the user can begin working on the next work item.

Table 1. Basic Workflow Concepts

```
color Task = string;
color Case = int;
color WI = product Case * Task;
color User = string;
color Role = string;
color Group = string;
```

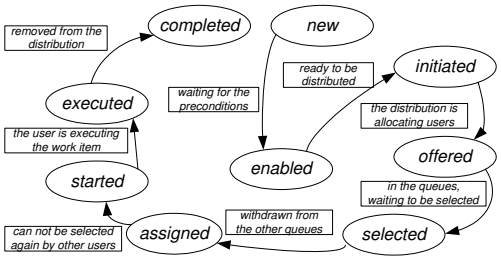


Fig. 1. Basic Model - Work Item Life Cycle

For the simulation (execution) of the work distribution in the model it is necessary to initiate the model by defining *input elements*. Table 2 shows that four elements are required for the simulation of the Basic Model. For every input element, Table 2 shows the element name (i.e., "system users", "new work items", "task maps" and "user maps"). Below the name there are a short description of the element, the color in the CPN model that represents the element and a simple example of the initial element value. In this example, there are two work items available for the case "1": "write article" and "read article" (*new work items*). The authorization (*task maps*) of these two tasks is specified in such a way that the task "write article" is mapped to the user who has the role "student", and is in the group "Information Systems". The task "read article" is mapped to the user with the role "professor", from the group "Information Systems". The organizational structure (*user maps*) contains two *users*. First, there is "Mary" who has the role of "student" in the group "Information Systems". Second, user "Joe" has the role "professor" and he works in the groups "Information Systems" and "Mathematics".

As a model of an abstract workflow management system, we have developed the Basic Model on the basis of predefined assumptions: (1) we abstract from the process perspective (i.e., splits, joins, creation of work items), (2) we only consider the "normal" behavior (i.e., work items are

Table 2. Input For The Basic Model

1. **system users**	
description:	a set of available users;
CPN color:	color Users = list User;
example:	iUser = 1‘”Mary”++1‘”Joe”;
2. **new work items**	
description:	work items that have arrived and are ready to be distributed to users;
CPN color:	color WI = product Case * Task;
example:	iWI = 1‘(1,”write article”)++1‘(1,”read article”);
3. **task maps**	
description:	decision about which work items can be executed by which users is made based on the authorizations given for every task in the process definition;
CPN color:	color TMap = product Task * Role * Group;
example:	iTMaps = [(”write article”,”student”,”Information Systems”), (”read article”,”professor”,”Information Systems”)];
4. **user maps**	
description:	the organizational structure is used to map users to the authorization of tasks;
CPN color:	color UMap = product User * Roles * Groups; (color Roles = list Role; color Groups = list Group;)
example:	iUMaps = [(”Mary”,[”student”],[”Information Systems”]), (”Joe”,[”professor”],[”Mathematics”, ”Information Systems”])];

completed successfully; errors and aborts are not included), and (3) we abstract from the user interface.

The Basic Model model is organized into two sub-systems: the Work Distribution and the Work Lists module. The CPN language allows for the decomposition of complex nets into sub-pages, which are also referred to as sub-systems, sub-processes or modules. By using such modules we obtain a layered hierarchical description. Figure 2 shows the modular structure of the Basic Module. The two sub-modules communicate by exchanging messages via six places. These messages contain information about a user and a work item. Every message place is of the type (i.e., the CPN color) “user work item” (*color UWI = product User * WI*), which is a combination of a user and a work item. Table 3 shows the description of the semantics of each of the messages that can be exchanged in the model.

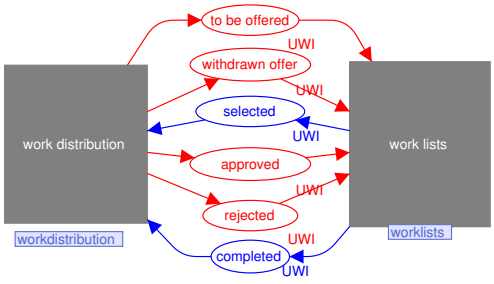


Fig. 2. Basic Model - Main

Table 3. Messages Between Modules

Place	Message
to be offered	A work item is offered to the user.
withdrawn offer	Withdraw the offered work item from the user.
selected	The user requests to select the work item.
approved	Allow the user to select the work item.
rejected	Do not allow the user to select the work item.
completed	The user has completed executing the work item

Work Distribution. The Work Distribution module manages the distribution of work items by managing the process of work execution and making sure that work items are executed correctly. It allocates users to which the *new work items* should be offered, based on authorization (*TMap*) and organization (*UMap*) data. Three (out of four) input elements are placed in this module: *new work items*, *user maps* and *task maps*. The variables used in this module are shown in Table 4.

Table 4. Basic Model - Variables in Work Distribution Module

var tmaps: TMaps;
var umaps: UMaps;
var wi: WI;
var wis:WIs; (color WIs = list WI;)
var uwi: UWI;

Figure 3(a) shows the Work Distribution module. The allocation function *offer* contains allocation rules (allocation algorithm) of the specific distribution mechanism. Work items that are offered to users are stored in the place *offered work items*. After receiving a request from the user to select the work item, the decision is made weather to allow the user to select the item (and thus to execute it), or to reject this request. This decision is made based on the assumption that at one moment, only one user can work on the work item. If the work item has already been selected (i.e., it is not in the place *offered work items*), then the model rejects this request. If nobody has selected the work item yet, the approval is sent to the user and the work item is moved to the place *assigned work items*. A work item that is moved to the place *selected work items* cannot be selected again.

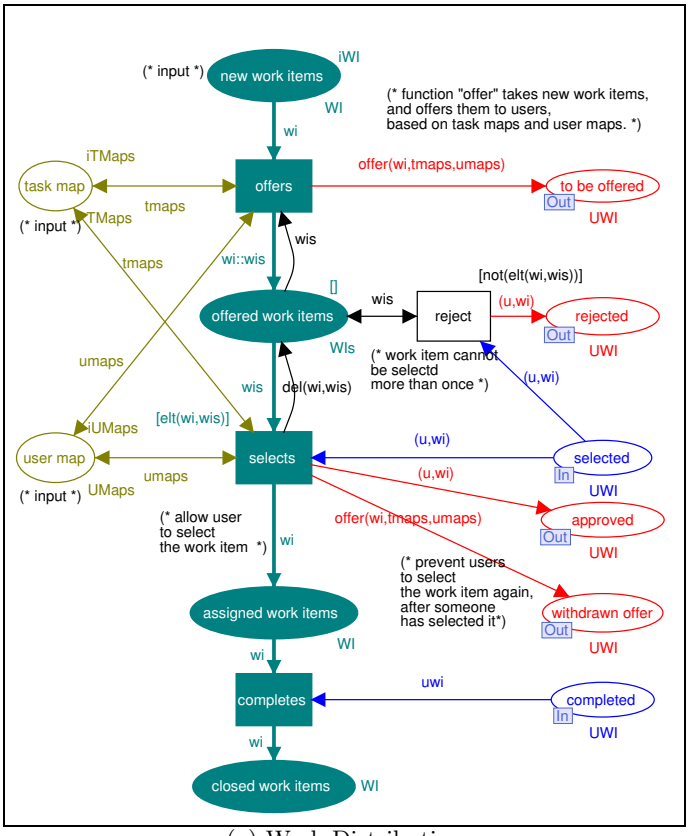
Work Lists. Figure 3(b) shows the Work Lists module. This module receives messages from the Work Distribution module about which work items are to be offered to which users. The Work Lists module further manages events associated with the activities of users. It is decomposed into three units, which correspond to three basic actions users can make: *log on and off* (cf. Figure 3(c)) in the system, *select work* (cf. Figure 3(d)), *start work* (cf. Figure 3(e)), and *stop work* (cf. Figure 3(f)). Once the work item has been *offered* to users, they can *select* it. When a user selects the work item, the *request* is sent to the Work Distribution module. If the request is *rejected*, the action is *aborted*. If the Work Distribution Module *approves* the request, the user can start working on the work item. Once the user has started working, the work item is considered to be *in progress*. Next, the user can stop working, and the work item is *completed*. In order to perform any of these actions, it is necessary that the user is *logged on* in the system.

3 Work Distribution Models

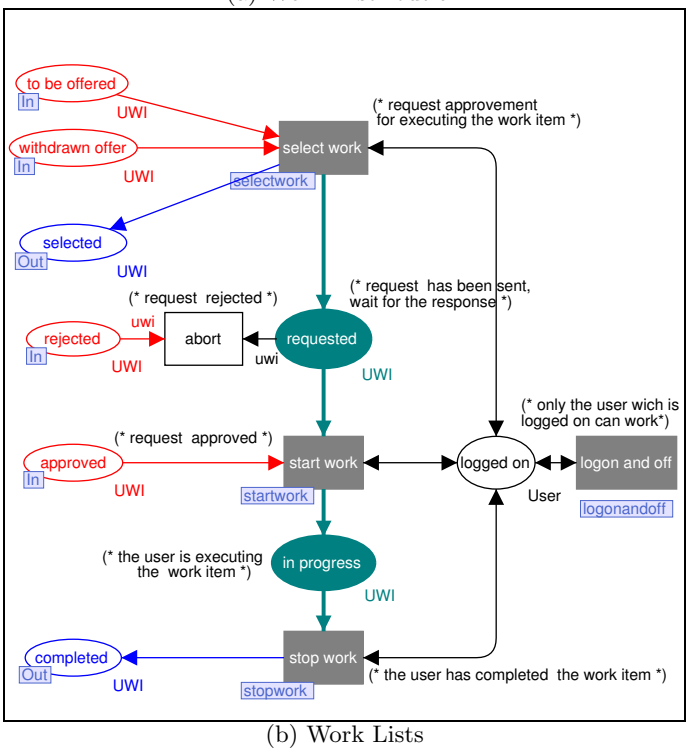
The Basic Model presented in previous section (Section 2) is used as a reference for different extensions and specializations of work distribution. In this section, we first extend and specialize the Basic Model for Staffware, FileNet and FLOWer (Section 3.1). In Section 3.2 we select four of the more advanced resource patterns reported in [48, 50]. These four patterns are not supported by Staffware, FileNet and FLOWer, but we will show that it is easy to extend the Basic Model to adequately address the patterns.

3.1 Workflow Management Systems

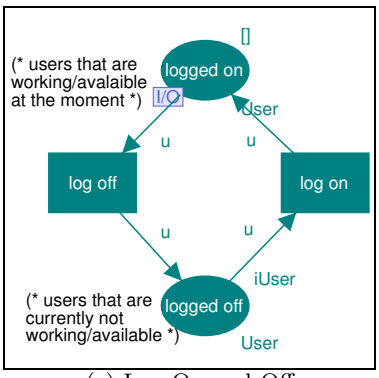
We have modelled the work distribution mechanisms of three commercial workflow management systems: Staffware, FileNet and FLOWer. FileNet and Staffware are examples of two widely used traditional workflow management systems. FLOWer is based on the case-handling paradigm, which can be characterized as "the more flexible approach" [3, 9]. Each of the models we have developed will be described in the remainder of this section.



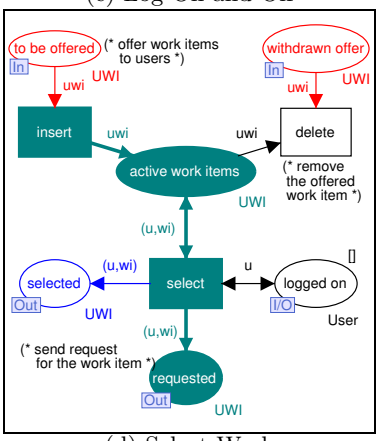
(a) Work Distribution



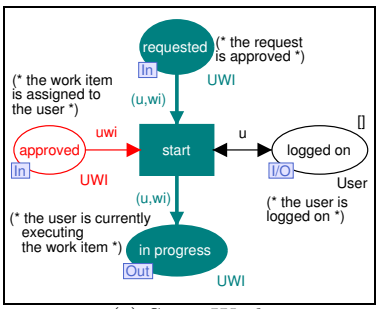
(b) Work Lists



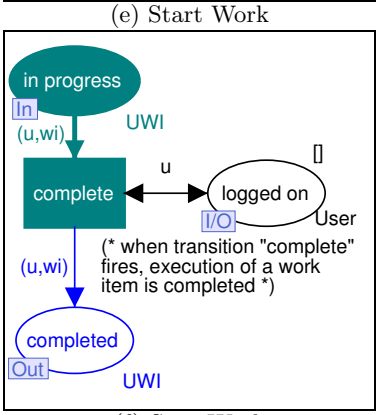
(c) Log On and Off



(d) Select Work



(e) Start Work



(f) Stop Work

Fig. 3. Basic Model

Staffware The Basic Model is upgraded to represent the work distribution of Staffware. The way of modelling the *organizational structure* and *resource allocation algorithm* are changed, while the concept of *work queues* and the possibility of the user to *forward and suspend* a work item are added to the model.

Organizational Structure. Simple organizational structure can be created in Staffware using the notions of *groups* and *roles*. The notion of group is defined as in the Basic Model, i.e., one group can contain several users, and one user can be a member of several groups. However, specific for Staffware is that a role can be defined for only one user. This feature does not require any changes in the model itself. It changes the way the initial value for the *user maps* should be defined – one role should be assigned to only one user.

Work Queues. Groups are used to model a set of users that share common rights. The work item can be allocated to the whole group, instead of listing the names of users that can execute it. Staffware introduces a *work queue* for every group. The work queue is accessible to all members of the group. Single users are also considered to be groups that contain only one member. Thus, one work queue is also created for every user and this personal queue is only accessible by a single user. From the perspective of the user, (s)he has access to the personal work queue and to work queues of all the groups (s)he is a member of. Table 5 shows which colors are added to the model to represent work queues in Staffware. While the Basic Model (Section 2) offers the work item directly to *users*, Staffware offers items in two levels. First, the work item is offered to *work queues* (color WQ). We refer to this kind of work items as to the *queue work item* (color QWI). Second, after a queue work item is offered to a group (work queue) it is offered to each of its members and only one member will execute the queue work item once. We refer to a queue work item that is offered to a member as to the *user work item* (color UWI).

Table 5. Staffware - "Work Queue" Colors

```
color WQ = string;
color QWI = product WI * WQ;
color UWI = product User *QWI;
```

Figures 4 and 5 show that we create two levels in the Work Distribution module to support the two-level distribution of Staffware:

1. In the module itself a new work item is offered to work queues (as a queue work item). The new work item is completed when each of its queue work items is executed. Thus, if a new work item is offered to multiple work queues, it is executed multiple times.
2. In the sub-module *Offering to Users* every queue work item is offered to the queue members (user work item). A queue work item is completed when one of the members executes the user work item.

Resource Allocation. We have changed the the allocation function *offer* to represent the allocation algorithm in Staffware. Just like the Basic Model, Staffware searches for possible users based on *roles* and *groups*. In addition to this, in Staffware users can be allocated by their *user-names* and *data fields* in the process. In *user maps* we use the fields reserved for groups when we want to specify the allocation for user-names. We do this by assuming that every user-name refers to a group with only ne member – the specified user. The second addition in Staffware refers to the fact that resource allocation can be also done at "run-time" on the basis of *data fields* in *task maps* (cf. Table 6). This kind of allocation is referred to as a dynamic work allocation. Every field has a unique name, e.g., *next user*. During the execution of the process, every field is assigned a value, and this value changes (e.g., users can assign values to fields). Staffware assumes that the value of the assigned data field is a group name, a role name or a user name. If the field *next user*

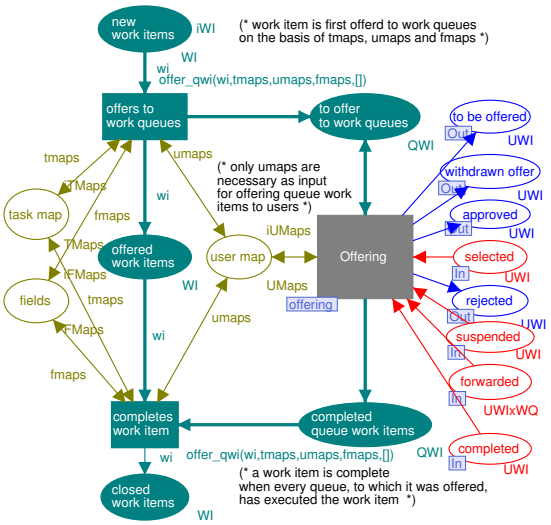


Fig. 4. Staffware - Work Distribution

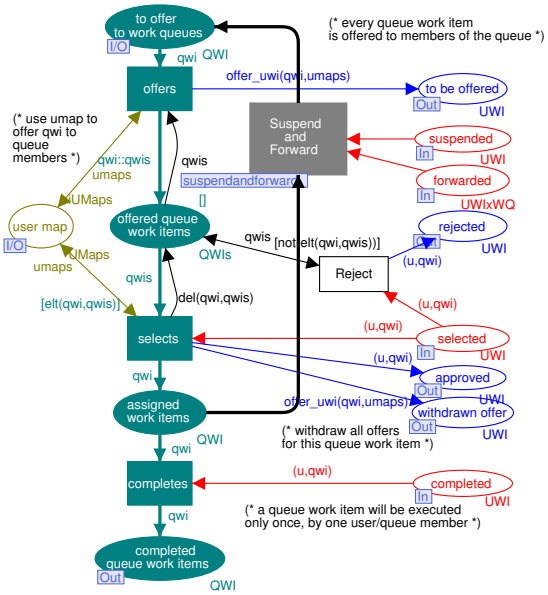


Fig. 5. Staffware - Offering

(which for example has the value of "John Smith" assigned) is specified in the task map of a task, then the actual value of the field is assigned to the task map entry at the moment when the task becomes enabled. Thus, "John Smith" will be used in the allocation.

Table 6. Staffware - Dynamic Work Allocation

```
color Field = string;
color Fields = list Field;
color FValue = string;
color FMap = product Field*FValue;
color FMaps = list FMap;
color TMap = product Task * Users * Roles * Groups * Fields;
```

Table 7 represents the difference between the work distribution and the allocation functions *offer* in the Basic Model and Staffware. Work distribution in Staffware starts with distribution to work queues. Thus, allocation of resources starts with the allocation to work queues: (1) if the allocation refers to group names the work item is allocated to group work queues, and (2) if the allocation refers to user names or roles the work item is allocated to personal work queues. Allocation for every task is specified in the type TMap, as Table 6 shows. If we look at the example from Table 2, we can see that the task "read article" should be allocated to users which are from the group "Information Systems" *and* have the role "professor". Figure 6 shows how this can be done in Staffware. The Basic Model allocates this task to users that are from the group "Information Systems" *and* have the role "professor", i.e., to the user "Joe". Staffware allocates this task to the work queue of the group "Information Systems" and the personal queue of the user who has the role "professor" (cf. Figure 7). In Staffware model, this work item will be executed two times: (1) by one member of the group "Information Systems", i.e., by "Joe" or "Mary", and (2) by the user with the role "professor", i.e., by "Joe". Thus, in Staffware model task "read article" will either be executed (1) twice by "Joe" or (2) once by "Joe" and once by "Mary".

Forward and Suspend. In the Basic Model once the user selects the work item, (s)he can start working on a work item and complete it. Figure 8 shows that Staffware offers a more realistic and

Table 7. Staffware - Distribution of the task "read article"

Users	Work Queue	*TMap* for 'read article' Group *Information Systems*	Role *professor* (assigned to Joe)	Execution 1	Execution 2
Joe	*Information*	#		Joe	
Mary	*Systems*				Mary
Joe	*Joe*		#	Joe	Joe
Mary	*Mary*				

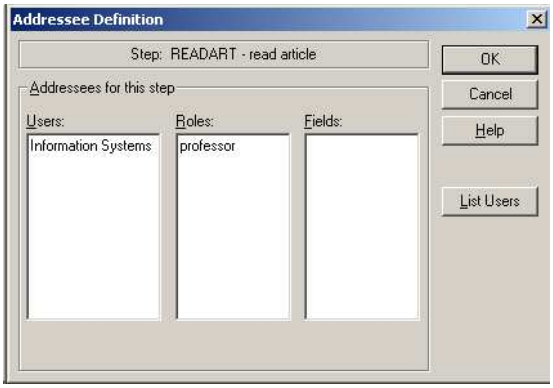


Fig. 6. Staffware - Task Map for "Read Article"

Fig. 7. Staffware - Work Queues for "Read Article"

somewhat more complex model of the life cycle of a work item . After the user *starts* the work item, (s)he can either *execute* it, or *forward* it to another user. Forwarding transfers the work item to the state *offered*, because it is automatically *offered* to the new user. If the user chooses to *execute* the work item, (s)he can *complete* or *suspend* it. When work item is suspended it is transferred back to the state of *initiated*. After this, the system *offers* the work item again, and other users are able to *select* it again.

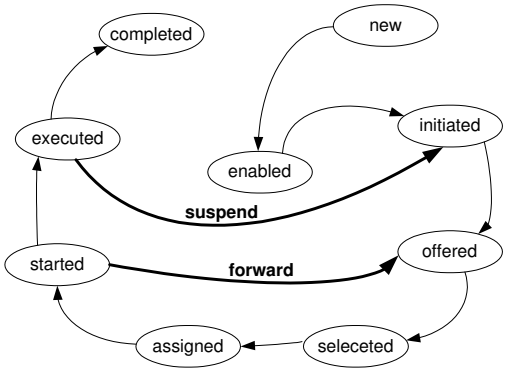
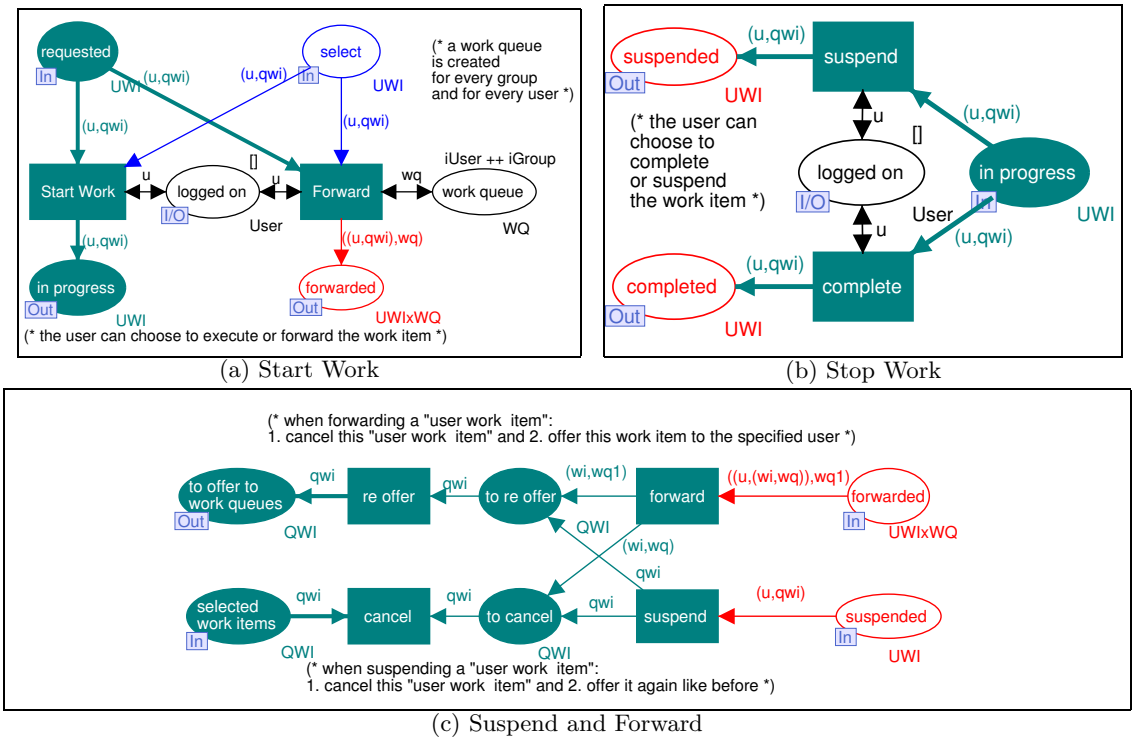


Fig. 8. Staffware - Work Item Life Cycle

Forwarding and suspending of work items adds two messages that are exchanged between Work Distribution and Work Lists modules in Staffware model. Figures 4 and 5 show two new places – *forward* and *suspend*. These two new actions are triggered in the *Work List* module by the user. Figure 9(a) shows that in the module Start Work the user can choose to *select* or *forward* (to another work queue) the work item. Figure 9(b) shows that in the module Stop Work the user can

choose to *complete* or *suspend* the work item. The Work Distribution module handles forwarding and suspending in the Offering to Users sub-module. Figure 9(c) represents how: (1) in case of forwarding the work item is automatically *cancelled* for the current work queue and *offered* to the new work queue, and (2) in case of suspending the work item is *cancelled* for the current work queue and *re-offered* as a new work item.



(a) Start Work

(b) Stop Work

(c) Suspend and Forward

Fig. 9. Staffware - Forward and Suspend

FileNet Like Staffware, FileNet is a widely used traditional process-oriented workflow management system. In this section we will describe the FileNet CPN model that we develop using the Basic Model as a starting reference model.

Organization. The organizational model in FileNet does not allow for modelling *roles*. Table 9 shows which colors are added to the CPN model to represent the two types of organizational groups:

1. Administrators define *work queues* (color *WQ*) and assign their members in the FileNet system. Work queues are valid for every process (workflow) definition.
2. Process modelers can define *workflow groups* (color *WG*) in every process model. Thus, workflow groups are valid only in the process (workflow) model in which they are defined. Workflow groups represent teams in FileNet. While executing a task of a process definition, users have the possibility to change the structure of workflow groups that are defined in that process. Figure 10 shows that users can alter workflow groups (add and remove members) only when a work item is *in progress*. Workflow groups represent teams in FileNet.

Table 8. FileNet - "Work Queue" Colors

```
color WQ =    string;
color WG =    string;
color WQs =   list WQ;
color WGs =   list WG;
color UMap =  product  User
              * WGs* WQs;
```

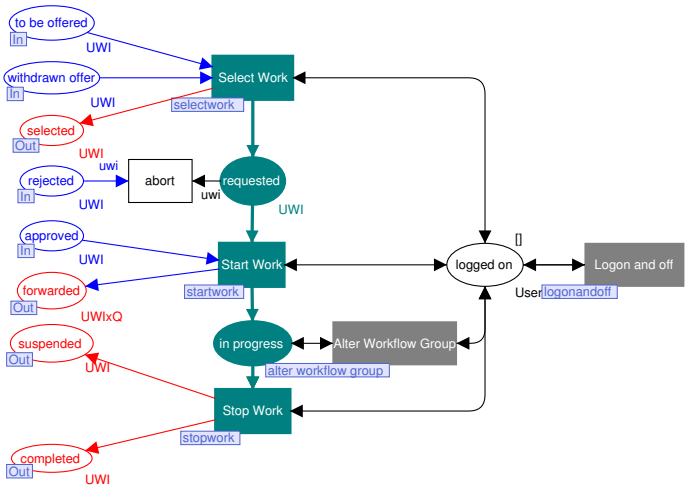


Fig. 10. FileNet - Work Lists

Queues. *Work queues* and *personal queues* are two types of pools from which users can select and execute work items. A work queue can have a number of members while a personal queue has only one member. When a work item is offered to a queue one of the queue members can select and execute the work item. Table 9 shows which colors are added to the FileNet model to represent queues. FileNet distributes work in two levels using queues. First, the work item is offered to queues as a *queue work item* (color *QWI*). Second, the queue work item is offered to the members of the queue as a *user work item* (color *UWI*).

Table 9. FileNet - "Queue Work Item" Colors

```
color Q = string;
color QWI = product WI * Q;
color UWI = product User * QWI;
```

Figures 11 and 12 show that the model of the two-level work distribution in FileNet is similar to the Staffware model. For more detailed description of this kind of distribution we refer the reader to the Staffware description in Section 3.1.

Resource Allocation. FileNet allocates work using work queues and lists of participants. Figure 13 shows that a task in FileNet can be allocated to either a work queue or to a list of participants. Users and workflow groups can be entries or a list of participants. In the FileNet model task maps are defined as a combination of a task, a list of work groups, and a work queue (*color TMap = product Task * WGs * WQ;*). It is necessary to highlight that, when defining the input value for a task map, only work queue or a list of workflow groups should be initiated.

If the task is allocated to a work queue FileNet offers the work item to the work queue. If the task is allocated to a list participants then it is offered to personal queues of all users that are listed as participants or are members in workflow groups that are listed. Allocation via participants is introduced to support team work in FileNet, via so-called "process voting". During the execution of a task, all participants vote for the specified decision. The work distribution mechanism uses their decisions to determine which work items will be executed next. Since our models abstract from the process perspective, we did not model process voting in the FileNet model.

Forward and Suspend. Users can forward and suspend work items in FileNet. When the user selects a work item (s)he can start working on it or forward it to another user. In this case FileNet

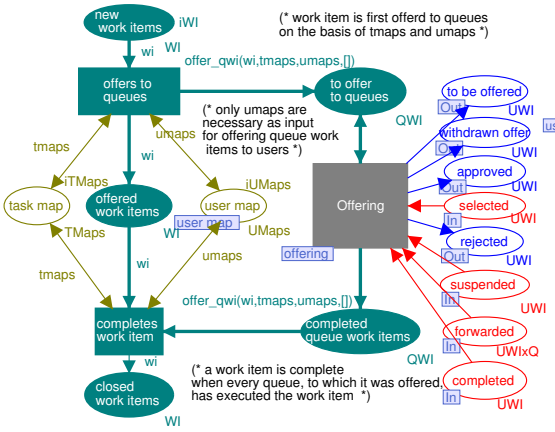


Fig. 11. FileNet - Work Distribution

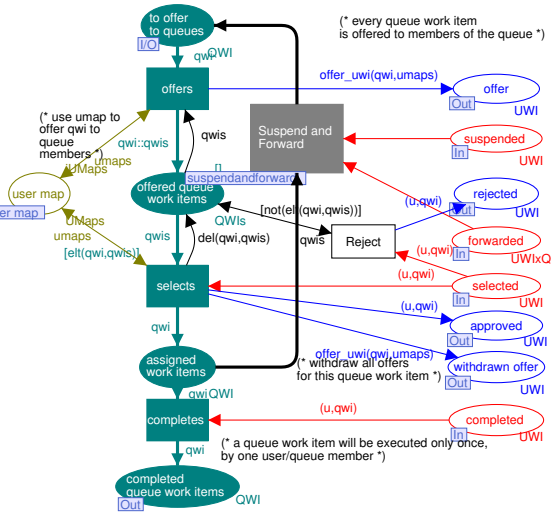


Fig. 12. FileNet - Offering

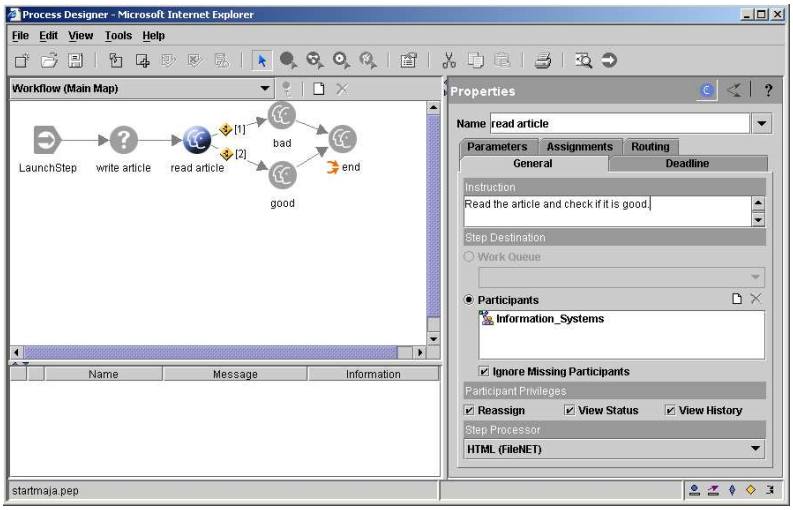


Fig. 13. File Net - Allocation for Work Queues or Participants

automatically offers the work item to the new user. When the user is executing a work item s(he) can complete or suspend the work item. In this case FileNet needs to apply the distribution mechanism again, and offer the work item to all allocated users. Figure 14 shows the life cycle of a work item in FileNet. If the life cycle models of FileNet and Staffware (cf. Section 3.1) are compared it can be seen that they are identical. Therefore, we use the same adjustments in FileNet and Staffware models to implement forwarding and suspension: modules *Start Work* and *Stop Work* are changed and sub-module *Suspend and Forward* is added in the Work Distribution module, as Figure 15 shows. For detailed description we refer the reader to Staffware description in Section 3.1.

FLOWer FLOWer is a case handling system. Case handling systems differ in their perspective from traditional process-oriented workflow management systems because they focus on the case, instead of the process [3, 9]. The user is offered the whole case by offering all available work items

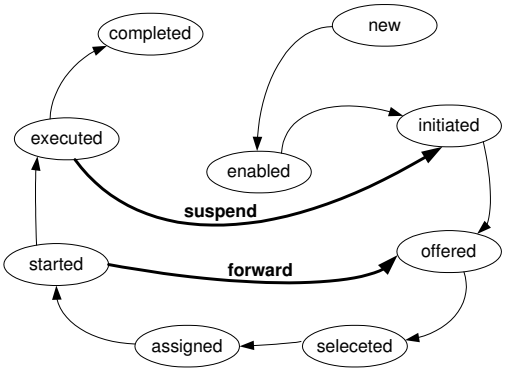
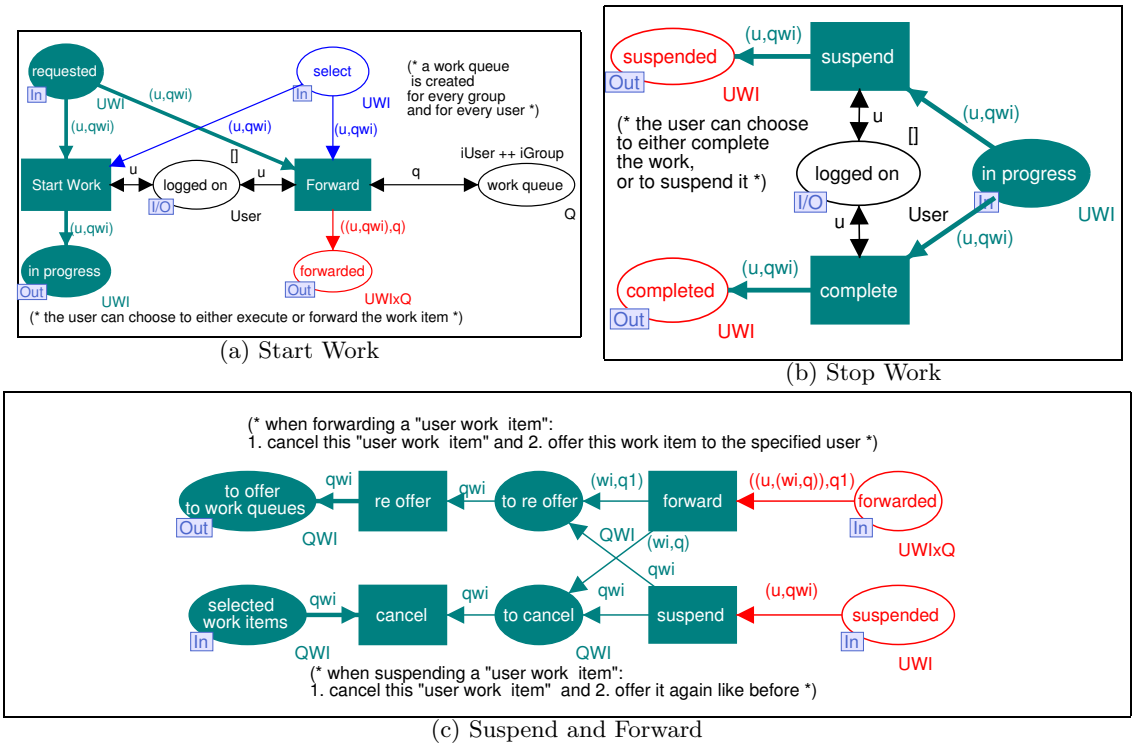


Fig. 14. FileNet - Work Item Life Cycle



(a) Start Work

(b) Stop Work

(c) Suspend and Forward

Fig. 15. FileNet - Forward and Suspend

from the case and s(he) does not have to follow the predefined order of tasks in the process definition.

When modelling FLOWer, we upgraded the Basic Model in such a way that (1) it handles *case-handling distribution* (instead of the process-oriented one), (2) it enables the complex *authorization* and *distribution* specifications that FLOWer has, and (3) it enables users to *execute*, *open*, *skip* and *redo* work items.

Case Handling. Table 10 shows which colors are used to model FLOWer as a case-handling system. Every process definition in FLOWer is referred to as a *case type*. One *case* represents an instance of a case type and is identified by the case identification (color *CaseID*). Figures 16 and 17 show that FLOWer distributes work in two levels:

1. The case is distributed to users (color *UCase*). Only one user can select and open the case at one moment. Figure 16 shows that in the FLOWer Work Distribution model a *case* becomes the object of distribution instead of a *work item*.
2. The selected case is opened for the user in the *Case Distribution* sub-module. Work items from the case are offered to the user, based on the authorization and distribution rules. The user can execute, open, skip and redo work items from the selected case. The *Case Distribution* sub-module and authorization and distribution rules are described in the remainder of this section.

Table 10. FLOWer - Basic Colors

```
color CaseType = string;
color Tasks = list Task;
color Process = product CaseType * Tasks;
color CaseID = INT;
color Case = product CaseID* CaseType;
color WI = product Case * Task;
color UCase = product User * Case;
```

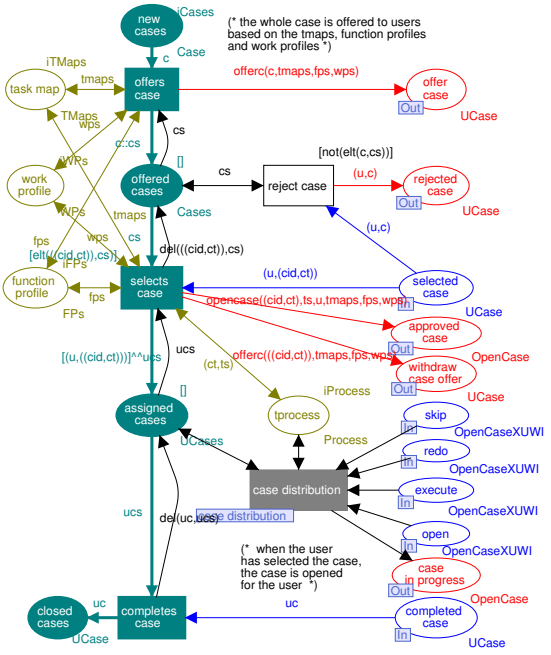


Fig. 16. FLOWer - Work Distribution

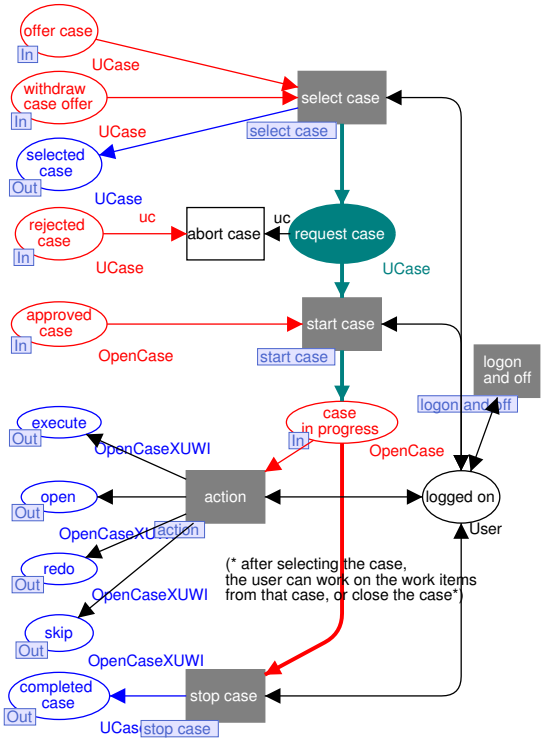


Fig. 17. FLOWer - Work Lists

Authorization Rights. When designing the process for a case type it is necessary to define case-type-specific roles and to assign each role authorization rights for tasks in the process. The authorization rights determine what users *can do*. These rights are applied by the distribution mechanism when opening the case for the user. The user is allowed to work only on tasks for which (s)he has the

authorized roles. Information about the authorization is stored in task maps (i.e., *color TMap = product Task * Role * CaseType*).

Distribution Rights. Distribution rights define what users *should do*. These rights are used to model the organizational structure and to assign authorization rights from the process definitions (case types) to users. *Function profiles* and *work profiles* define distribution rights. Table 11 shows which colors are added to represent these profiles. *Function profile (FP)* assigns authorization roles to users. If, for example, there are two case types (two processes) that both have "secretary" as an authorization role, the function profile "secretary" includes both authorization roles. When we assign the function profile "secretary" to a user, we indirectly assign both authorization roles from two processes. *Work profiles (WP)* assign function profile(s) to users and they can be used to structure organization into groups, departments or units.

Table 11. FLOWer - Distribution Rights in CPN Colors

```
color PRole = product Role * CaseType;
color FN = STRING;
color FP = product FN * PRoles;
color WN = STRING;
color WP = product WN * FNs * Users;
```

Open, Execute, Skip and Redo. Although in a case type tasks in the process definition have the execution order that is suggested to the user, (s)he is not obliged to follow it. When working with an open case in FLOWer, users can: (1) *Execute* the work item which is next in the process definition; (2) *Open* for execution a work item that is still not ready for execution according to the process definition; (3) *Skip* a work item by choosing not to execute the work item which is next according to the process definition, or (4) *Redo* a work item by executing again a work item which has already been executed. Figures 16 and 17 show that four new places are added to the model to represent these four actions. In order to implement these possibilities in the FLOWer model it is necessary keep the information about the *case state*, i.e., about the work items that are (1) *waiting* to be enabled, (2) *active* (i.e. they are enabled and can be executed), (3) *finished* (executed), and (4) *skipped*. Table 12 shows that an open case keeps information about the case state in four lists of work items (waiting, active, finished and skipped).

Table 12. FLOWer - Open Case and Case State

```
color CaseState = product WIs*WIs*WIs*WIs;
color OpenCase = product UCase*CaseState;
```

In FLOWer users work with the interface tool "Wave Front" [43] where they can see the state of the open case. Users can see which work items are waiting, active, finished and skipped. Figure 18 shows one example of an open case in the "Wave Front". The first two tasks ("Claim Start" and "Register Claim") are *finished* work items and they are marked with a "check" symbol. The third work item ("Get Medical Report") was *skipped*, as can be seen from the "arrow" symbol. Thus, finished and skipped work items are presented after the "Wave Front" line. The three *active* work items on the Wave Front are "Get Police Report", "Assign Loss Adjuster" and "Witness Statements". Finally, the two last work items ("Policy Holder Liable" and "Close Case") are *waiting* before the Wave Front to become active.

Figure 19 shows the sub-module *Action* (in the FLOWer Work List module) where we model how user performs the actions to execute, open, skip and redo work items. In FLOWer users can

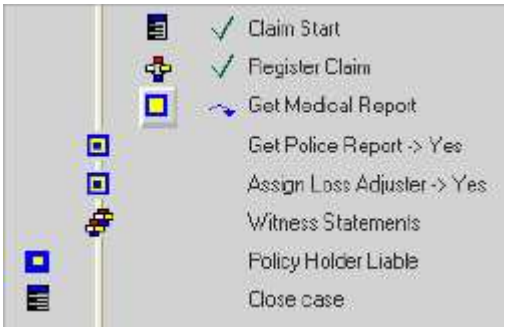


Fig. 18. FLOWer Wave Front

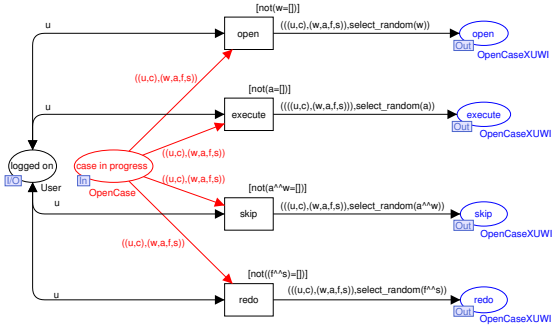


Fig. 19. FLOWer - Action

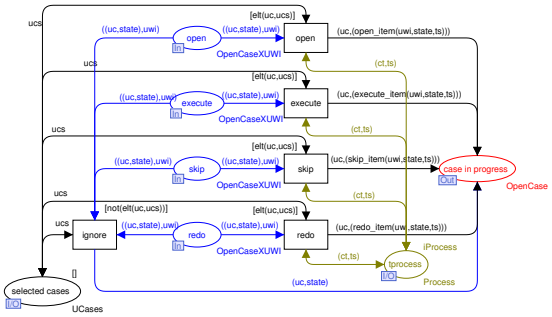


Fig. 20. FLOWer - Case Distribution

choose work items on their own discretion but (due to the complexity of the model) we model this selection as a random function. When the user wants to:

1. **open** an item s(he) selects a work item from the list of *waiting items*;
2. **execute** an item s(he) selects a work item from the list of *active items*;
3. **skip** an item s(he) selects a work item from the lists of *waiting and active items*;
4. **redo** an item s(he) selects a work item from the lists of *finished and skipped items*.

Each of the four actions the user performs changes the state of the open case. For example, opening a work item transfers it to the state active (and, therefore, it is transferred to the list of active items). Figure 20 shows that the Case Distribution module responses in different ways (functions *execute_item*, *open_item*, *skip_item*, and *redo_item*) when each of the four actions is performed. When an action is performed over a work item, the state of the work item changes, as shown in Table 13. The for actions are listed in the column "action". The column "work item becomes" shows how the action changes state of the work item. It often happens that an action performed on a selected work item also effects other items and this is described in the column "side effects".

In FLOWer a work item has a different *life cycle* than in Staffware and FileNet. First, the moment of enabling is different. It is not necessary for the *new* work item to be enabled in order to be *initiated*, *offered*, *selected* and finally *assigned*. Second, FLOWer adds more possibilities for switching between states *assigned*, *enabled*, *started*, *executed* and *completed*. In the model of the FLOWer work item life cycle (cf. Figure 21), additional actions to skip, open and redo are added to the basic life cycle model. A path that is marked with brackets (e.g., "(* redo *)") is a side effect of an action being taken over another work item (e.g., a work item can be directly transferred from the state completed to the state assigned if the action redo is performed on a preceding work item).

Table 13. FLOWer - The Four Actions

action	work item becomes	side effects
open	*active*	Items from *waiting* that preceded become *skipped*.
execute	*finished*	The direct successors in *waiting* become *active*.
skip	*skipped*	Items from *waiting* that preceded become *skipped*. The direct successors in *waiting* become *active*.
redo	*active*	Subsequent items from (*skipped* & *finished*) become *waiting*.

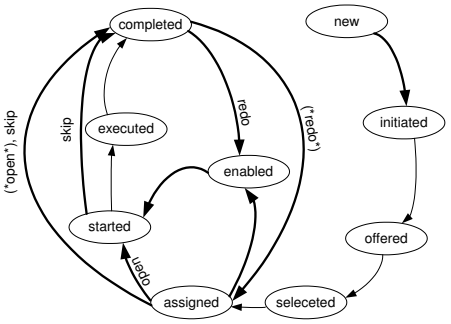


Fig. 21. FLOWer - Work Item Life Cycle

3.2 Resource Patterns

Instead of extending the Basic Model for more systems, we also looked at a more systematic way of work distribution. As indicated, similar concepts are often named and presented differently in different workflow management systems. Therefore, it is interesting to define these concepts in a system-independent manner. We have used 43 documented *resource patterns* [48, 50]. These patterns can be used as representative examples for analyzing, evaluating and comparing different workflow management systems with respect to work distribution. Resource patterns are grouped into a number of categories: *creation patterns*, *push patterns*, *pull patterns*, *detour patterns*, *auto-start patterns*, *visibility patterns*, and *multiple resource patterns*. Each of these patterns can be modeled in terms of a CPN model.

Table 14 shows an overview of the patterns. It also shows whether a pattern is directly supported by the three systems (SW = Staffware, FN = FileNet, FW = FLOWer) and the Basic Model (BM). The Basic Model supports less patterns than any of the three systems. This makes sense since each of the system-specific models can be seen as an extension of the Basic Model. It is interesting to see that existing systems typically support less than half of the patterns directly. This reveals typical limitations of contemporary products. Some of the patterns are considered out-of-scope for our models (marked with "o"). These are typically patterns directly depending on control-flow functionality, while we prefer to focus exclusively on work distribution. Each of the patterns not marked with "o" can easily be added to the Basic Model separately.

We cannot elaborate on each of the patterns, but we will discuss four to illustrate our work. None of the systems supports *Pattern 16: Round Robin*, *Pattern 17: Shortest Queue*, *Pattern 38: Piled Execution*, and *Pattern 39: Chained Execution*. Patterns 16 and 17 are push patterns, i.e., a patterns to push work to a specific user. As auto-start patterns, patterns 38 and 39 enable the automatic start of the execution of the next work item once the previous has been completed.

Round Robin and Shortest Queue. Round Robin and Shortest Queue push the work item to one user of all users that qualify. Round Robin allocates work on a cyclic basis and Shortest Queue to the user with the shortest queue. This implies that each user has a counter to : (1) count the sequence of allocations in Round Robin and (2) count the number of pending work items in Shortest Queue. Tables 15 and 16 show which colors and variables are used to implement counters in models of these two patterns. As Figures 22 and 23 show, these two patterns are implemented in a similar way in the Work Distribution Module. The required changes to the Basic Model are minimal. A counter is introduced for each user (token in place *available*) and functions *round_robin* and *shortest_queue* are used to select one user from the set of possible users based on these counters. Similarly, most of the other patterns can be realized quite easily. The model for Shortest Queue has an additional connection (two *arcs*) that updates the counter when a work item is completed to remove it from the queue.

Table 14. Support for Resource Patterns in 3 Workflow Systems and Basic Model

(+ = direct support, – = no direct support, +/– = partial support, o = out-of-scope)

Nr	Pattern	SW	FN	FW	BM
1	Direct Allocation	+	+	+	+/–
2	Role-based Allocation	+	+/–	+	+
3	Deferred Allocation	+	+	–	–
4	Authorization	–	–	+	–
5	Separation of Duties	–	–	+	–
6	Case Handling	–	–	+	–
7	Retain Familiar	–	–	+	–
8	Capability-based Allocation	–	–	+	–
9	History-based Allocation	–	–	–	–
10	Organizational Allocation	+/–	+/–	+/–	+/–
11	Automatic Execution	+	+	+	o
12	Distribution by Offer – Single Resource	–	–	–	–
13	Distribution by Offer – Multiple Resources	+	+	+	+
14	Distribution by Allocation – Single Resource	+	+	+	–
15	Random Allocation	–	–	–	+
16	Round Robin Allocation	–	–	–	–
17	Shortest Queue	–	–	–	–
18	Early Distribution	–	–	+	–
19	Distribution on Enablement	+	+	+	+
20	Late Distribution	–	–	–	–
21	Resource-Initiated Allocation	–	–	+	+
22	Resource-Initiated Execution – Allocated Work Item	+	+	+	+
23	Resource-Initiated Execution – Offered Work Item	+	+	–	–
24	System-Determined Work List Management	+	+	+	o
25	Resource-Determined Work List Management	+	+	+	o
26	Selection Autonomy	+	+	+	+
27	Delegation	+	+	–	–
28	Escalation	+	+	–	–
29	Deallocation	–	–	–	–
30	Stateful Reallocation	+/–	+	–	–
31	Stateless Reallocation	–	–	–	–
32	Suspension/Resumption	+/–	+/–	–	–
33	Skip	–	–	+	o
34	Redo	–	–	+	o
35	Pre-Do	–	–	+	o
36	Commencement on Creation	–	–	–	–
37	Commencement on Allocation	–	–	–	–
38	Piled Execution	–	–	–	–
39	Chained Execution	–	–	+	–
40	Configurable Unallocated Work Item Visibility	–	–	–	o
41	Configurable Allocated Work Item Visibility	–	–	+	o
42	Simultaneous Execution	+	+	+/–	+
43	Additional Resources	–	–	–	–

Table 15. Round Robin - Colors and Variables

```
color RRCounter = product User*INT;
color RRCounters = list RRCounter;
var count, i: INT;
var rrc: RRCounter;
var rrcs: RRCounters;
```

Table 16. Shortest Queue - Colors and Variables

```
color SQCounter = product User*INT;
color SQCounters = list SQCounter;
var count, i: INT;
var sqc: SQCounter;
var sqcs: SQCounters;
```

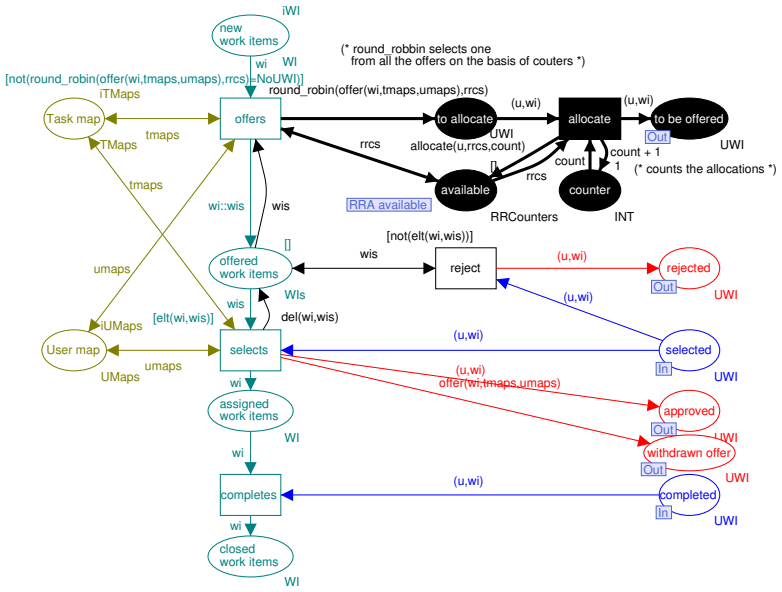


Fig. 22. Push Patterns - Round Robin

Piled and Chained Execution. Piled and Chained Execution are auto-start patterns, i.e., when the user completes execution of current work item the next work item starts automatically. When working in Chained Execution, the next work item will be for the same *case* as the completed one (the user works on different tasks for one case). Similarly, if the user works in Piled Execution the next work item will be for the same *task* as the completed one (the user works on one task for different cases). Figures 24 and 25 show how Piled and Chained Execution are implemented similarly in the Stop Work sub-module. Users can choose to work in the normal mode or in the auto-start mode (which is represented by the token in place *special mode*). The function *select* is implemented to search for the next work item for the same: (1) *task* in Piled Execution and (2) *case* in Chained Execution.

4 Related Work

Since the early nineties workflow technology has matured [26] and several textbooks have been published, e.g., [5, 20, 30, 37, 41]. 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 [25]. 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

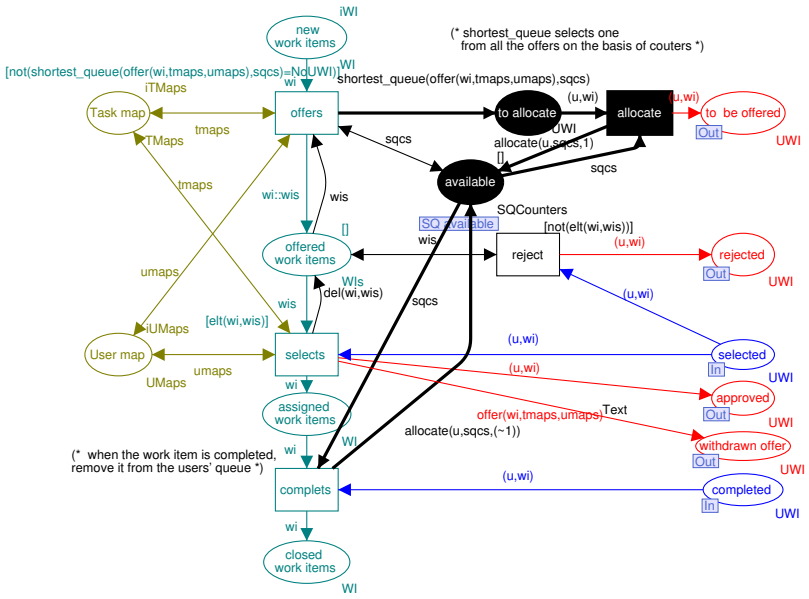


Fig. 23. Push Patterns - Shortest Queue

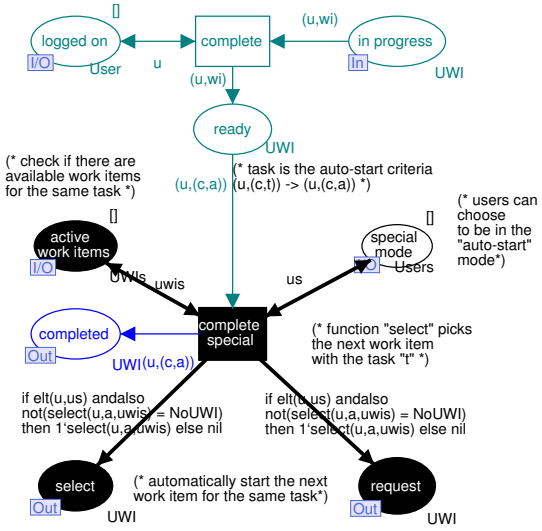


Fig. 24. Piled Execution - Stop Work

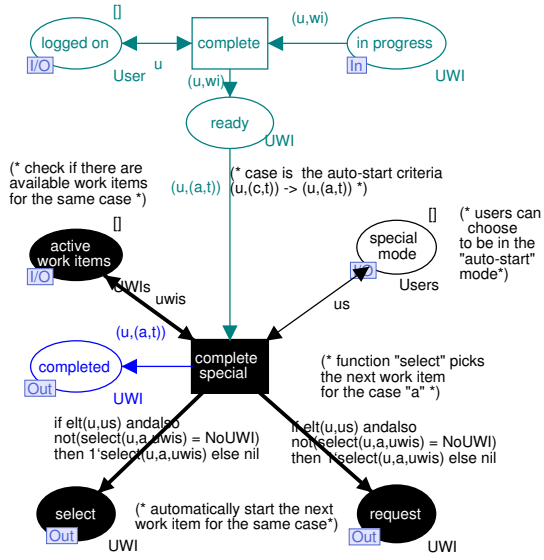


Fig. 25. Chained Execution - Stop Work

[1, 35]. 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 [30], 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 [19], 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 [36], 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 [29], 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 [23] 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.

Several researchers have developed meta-models, i.e., object models describing the relation between workflow concepts, which include work allocation aspects, cf. [8, 40–42, 47]. However, these meta-models tend to focus on the structural description of resource properties and typically do not describe the dynamics aspects of work distribution.

Flexibility has been a research topic in workflow literature since the late nineties [4, 7, 9, 10, 16, 22, 28, 33, 44, 46, 55]. 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 [27]. One way of allowing for more flexibility is to use the case handling concept as defined in [3, 9]. FLOWer [12, 43] 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 [52] and the COSA Activity Manager [51], both based on the generic solution of BPi [14], Vectus [38, 39], and the open-source system con:cern (http://con-cern.org/).

The work reported in this paper can be seen as an extension of the *workflow patterns initiative* (cf. www.workflowpatterns.com). Besides a variety of control-flow [6] and data [49] patterns, 43 resource patterns [48, 50] have been defined. This paper complements the resource patterns [48, 50] by providing executable models for work distribution mechanisms.

5 Discussion

Workflow management systems should provide flexible work distribution mechanisms for users. This will increase the work satisfaction of users and improve their ability to deal with unpredictable situations at work. Therefore, work distribution is investigated as the functionality provided for the user – workflow management systems are tested in laboratories [48, 50] or observed (in empirical research) in companies [13]. This kind of research observes systems *externally* and provides insights into *what* systems do. Analysis of the systems form an *internal* perspective can explain *how* systems provide for different work distribution mechanisms. Due to the complexity of workflow management systems as software products, internal analysis starts with developing a model of the system. Unlike statical models (e.g., UML models), dynamical models (e.g., CPN models) provide for interactive investigation of work distribution as a dynamic feature. CPN models can be used for the investigation of both *what* systems do and *how* they do it.

Workflow management systems often provide for different features or use different naming for the same features. Investigation of work distribution requires analysis, evaluation and comparison of models of several systems. In order for models of different systems to be comparable, it is necessary to start with developing a common framework – a *reference model*. We have developed the Basic Model as a reference model for work distribution mechanisms in workflow management system. The models of Staffware, FileNet, FLOWer and resource patterns are comparable because all models are developed as upgrades of a reference model (the Basic Model).

The model of a workflow system is structured into two modules (sub-models). The Work Distribution module represents the core of the system which is often called the "workflow engine". The Work Lists module represents the so-called "work list handler" of a workflow system and it serves as an interface between the workflow engine and users. The interface between the two modules (i.e., the messages that are exchanged between them) should contain as little information

about the way work items are managed in modules as possible. The way the work items are created, allocated and offered in the Work Distribution module should be abstracted from for the Work Lists module. The reverse also holds: how work items are actually processed by users is implemented in the Work Lists module. Once a proper interface is defined, it is easy to implement various ways of work distribution by adding/removing simple features in either one of the modules. For example, push patterns (Round Robin and Shortest Queue) are implemented in the Work Distribution module and auto-start resource patterns (Chained and Piled Execution) in the Work Lists module.

The flexibility of a work distribution mechanism can be observed through the model of the life cycle of a work item. The Basic Module has a simple, straightforward model and the work item (the user) follows a fixed predefined path. Work items in Staffware and FileNet life cycle models have more freedom to "walk back", thus allowing for implicit cycles in the model (e.g., forwarding and suspending). FLOWer, as the most flexible system, has many alternative paths in the life cycle model. A more complex model of the life cycle of a work item adds messages between the Work Distribution and Work Lists modules. The new messages correspond to new states and paths in the life cycle model.

Both the system-based and the patterns-based CPN models showed that one of the core elements of work distribution is the "allocation algorithm". This algorithm includes the "rules" for work distribution. It is implemented in the Work Distribution module as the function *offer*, which allocates work based on (1) new work items, (2) process definition, and the (3) organizational model. This function should be analyzed further in order to discover an advanced allocation algorithm, which should be more configurable and less system-dependent.

Every system has its own method of modelling organizational structure. Staffware models groups and roles. In FileNet the organizational model includes groups of users and teams, but does not model roles. FLOWer groups users based on a hierarchy of roles, function profiles and work profiles. Thus, each of the system offers a unique predefined type of the organizational structure. Since every allocation mechanism uses elements of the organizational model, limitations of the organizational model can have a negative impact on the work distribution in the system. For example, because in Staffware one role can be assigned to only one user, it is not be possible to offer a work item to a set of "call center operator"-s.

Each of the three models of workflow systems distributes work using two hierarchy levels. Staffware and FileNet use two levels of work distribution: queue work items are first distributed to work queues, and then work items are distributed within each of the work queues. The FLOWer model starts with the case distribution and then distributes work items of the whole case. Although all three systems distribute work at two levels, they have unique *distribution algorithms* (the set of allocation rules implemented in the function *offer*) and *objects of distribution* (work items, queue work items, cases).

Models of resource patterns [48, 50] show that push patterns (Round Robin and Shortest Queue) can be implemented "on top of" the pull mechanism, as a filter. Once the pull mechanism determines the set of allocated users, the "push" allocation function extracts only one user from this set. Auto-start patterns turned out to be remarkable straightforward to model, triggering the question why this is not supported by systems like Staffware and FileNet (FLOWer supports the Chained Execution in a limited form).

6 Conclusions

This paper focused 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. To understand work distribution, we used the CPN language and CPN Tools to model and analyze different work distribution mechanisms. To serve as a reference model, we provided a model that can be seen as the "greatest common denominator" of existing workflow management systems. This model was upgraded for models of three workflow management systems – Staffware, FileNet, and FLOWer. Although the reference model already captures many of the *resource patterns*, we also

modelled four more advanced patterns by extending the reference model. In contrast to existing research that mainly uses static models (e.g., UML class diagrams), we focused on the dynamics of work distribution. Our experiences revealed that it is relatively easy to model and analyze the workflow systems and resource patterns using CPN Tools. This suggests that CPN language and the basic CPN model are a good basis for future research. We plan to test completely new ways of work distribution using the approach presented in this paper. The goal is to design and implement distribution mechanisms that overcome the limitations of existing systems. An important ingredient will be to use insights from socio-technical design [13, 17, 21, 54] as mentioned in the introduction.

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. L. U. de Sitter, J. F. den Hertog, and B. Dankbaar. From complex organiations with simple jobs to simple organizations wiht complex jobs. *Human Relations*, 510(5):497–534, 1997.

18. B. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, *Application and Theory of Petri Nets 2005*, Lecture Notes in Computer Science, pages 444–454. Springer-Verlag, Berlin, 2005.
19. 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.
20. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems*. Wiley & Sons, 2005.
21. F.M. van Eijnatten and A.H. van der Zwaan. The Dutch IOR approach to organisation design. An alternative to business process re-engineering? *Human Relations*, 51(3):289–318, 1998.
22. 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.
23. 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.
24. FileNET. *FileNet Business Process Manager 3.0*. FileNET Corporation, Costa Mesa, CA, USA, June 2004.
25. L. Fischer, editor. *Workflow Handbook 2003, Workflow Management Coalition*. Future Strategies, Lighthouse Point, Florida, 2003.
26. 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.
27. R.E. Grinter. Workflow Systems: Occasions for Success and Failure. *Computer Supported Cooperative Work*, 9(2):189–214, 2000.
28. 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.
29. 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.
30. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.
31. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 1*. EATCS monographs on Theoretical Computer Science. Springer-Verlag, Berlin, 1997.
32. K. Jensen and G. Rozenberg, editors. *High-level Petri Nets: Theory and Application*. Springer-Verlag, Berlin, 1991.
33. 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.
34. L.M. Kristensen, S. Christensen, and K. Jensen. The Practitioner's Guide to Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer*, 2(2):98–132, 1998.
35. 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.
36. 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.
37. F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.
38. London Bridge Group. *Vectus Application Developer's Guide*. London Bridge Group, Wellesbourne, Warwick, UK, 2001.
39. London Bridge Group. *Vectus Technical Architecture*. London Bridge Group, Wellesbourne, Warwick, UK, 2001.
40. 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.
41. M. Zur Muehlen. *Workflow-based Process Controlling: Foundation, Design and Application of workflow-driven Process Information Systems*. Logos, Berlin, 2004.

42. M. zur Muhlen. Organizational Management in Workflow Applications Issues and Perspectives. *Information Technology and Management*, 5(3–4):271–291, July-October 2004.
43. Pallas Athena. *Flower User Manual*. Pallas Athena BV, Apeldoorn, The Netherlands, 2002.
44. M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
45. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.
46. 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.
47. 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.
48. 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.
49. 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.
50. 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.
51. Software-Ley. *COSA Activity Manager*. Software-Ley GmbH, Pullheim, Germany, 2002.
52. Staffware. *Staffware Case Handler – White Paper*. Staffware PLC, Berkshire, UK, 2000.
53. Staffware. *Using the Staffware Process Client*. Staffware, plc, Berkshire, United Kingdom, May 2002.
54. F. M. van Eijnatten. *The Paradigm that Changed the Work Place*. Van Gorcum, Assen, The Netherlands, 1993.
55. 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.