A Resource Model for Cloud-Based Workflow Management Systems
Enabling Access Control, Collaboration and Reuse

Sebastian Görg[1], Ralph Bergmann[1], Sarah Gessinger[1] and Mirjam Minor[2]
[1]*Department of Business Information Systems II, University of Trier, Germany*
[2]*Wirtschaftsinformatik, Goethe University of Frankfurt a. M., Germany*
{*goergs, bergmann, sarah.gessinger*}*@uni-trier.de, minor@informatik.uni-frankfurt.de*

Keywords: Access Control - Workflows - Collaboration - Experience Reuse

Abstract: Cloud-based workflow management systems provide platforms for modeling and execution of workflows while offering the common benefits of cloud computing. Additionally, the opportunity for workflow reuse and collaborative modeling could support agile business, as better workflows can be created according to a particular demand of the business with less effort. This paper addresses the issue of workflow reuse and collaborative modeling within a community of users. We present a new resource model for workflow related resources as well as a proposal for an access control mechanism. The proposed methods are currently under development within the workflow management system CAKE.

1 INTRODUCTION

During the last years the cloud-based computing paradigm emerged. It allows a shift from local IT-systems and information to the internet-cloud. In the cloud, service provider can offer web-based applications which replace local distributed and heterogeneous systems. As high network bandwidths become available, the usage of cloud-based applications delivers the same speed and response times as local applications. Compared to traditional IT-infrastructures cloud computing has attractive features (Liu et al., 2011): It leads to lower maintenance costs, more resources can be shared, they are easier to scale and more reliable.

The Workflow Management Coalition (WfMC) defined a workflow as "The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" (Coalition, 1999). Respectively, a Workflow Management System (WfMS) defines, creates and manages the execution of workflows. In general a WfMS is an information system for the management of business processes.

A cloud-based Workflow Management System provides a web-based platform for the modeling and execution of workflows. There are different trends and professional goals for current cloud-based WfMS. There are WfMS with a focus on a particular market, e.g. cflow[1] on jewelry retailers and health care, nowwecomply.com on compliance, or Taverna[2] on scientific workflows. These systems are easy to understand and to setup for their customers. On the other hand, there are more comprehensive systems like RunMyProcess.com or Visual Workflow by SalesForce[3], which allow to develop applications by modeling a flow of custom software components or web services. Both trends show that cloud-based WfMS are an emerging market.

Nevertheless, today's cloud-based WfMS have in common that they do not take full advantage of the collaboration and experience reuse features the cloud paradigm can provide. They provide the basic security features of classical WfMS, but omit the opportunities of collaborative workflow modeling and of sharing and reuse of the procedural knowledge within the workflows.

An example for the benefit of collaborative workflow modeling in cloud-based WfMS is the opportunity in gaining better workflows. Many enterprises especially small and medium-sized enterprises have only a few employees with the skills to design and maintain business processes and related workflows. This is a big issue when enterprises grow and need to

[1] http://cavintek.com/
[2] http://taverna.org.uk/
[3] http://www.salesforce.com/platform/cloud-platform/workflow.jsp

shift from pure operational business to a more strategic alignment. Workflow modeling is a tedious task which growths with the size of the enterprise. Workflow modelers must have expert knowledge on the business of the enterprise in order to model the workflow. Usually, the workflow modeler cannot have such complete in-depth knowledge of an expert. A better approach for this problem would be that the business-experts are more involved in the modeling while being empowered to model the workflow by themselves. Therefore the workflow modeler can benefit from a cloud-based WfMS which supports collaborative modeling. The workflow modeler can invite an expert by granting access to him/her to the affected workflow. Thereby the expert can use the workflow modeling user interface with an arbitrary browser without installing additional software. The workflow modeler and the expert can then collaboratively model the workflow.
Another example for the benefit of cloud-based WfMS is the possibility to share best-practice workflows. Sharing of workflows means that a workflow modeler allows other modelers or employees to see and to reuse a best-practice workflow for their own purposes. In particular, administrative workflows are very similar throughout different departments, e.g. the deployment of software-updates or escalation workflows. If the knowledge of such internal workflows is easy to obtain and to discover, there is a chance for synergy because the departments are more informed about the internal workflows of other departments and thus could benefit from the shared knowledge.
The remainder of this paper is organized as follows. In the next section the requirements for a cloud-based WfMS are depicted considering different application domains. Section 3 describes the CAKE system, the cloud-based WfMS, and its system architecture. The next two sections describe the resource model and the access control mechanism. Section 6 describes the workflow reuse and collaborative workflow modeling. We conclude by discussing the current state of development.

2 REQUIREMENTS

The presented concept for collaboration and reuse of workflows was developed based on the experience in different application domains for workflows and their according requirements: office/administrative workflows (Minor et al., 2009a), chip design (Minor et al., 2009b), scientific workflows (Minor and Görg, 2011), cooking workflows (Walter et al., 2011), construction workflows and personal/social workflows (Görg et al., 2012). In the focus of the developed resource model are no specific characteristics of these application domains. It is a generic concept which we expect to be applicable in all of these application domains.
In summary, the following general requirements for a cloud-based WfMS have been observed in the above mentioned application domains. Section 5.2 and 6 describe how the resource model fulfils these requirements.

1. Support for modeling of workflows.
2. Support for the execution of workflows including human and automated tasks.
3. Support for collaborative workflow modeling and sharing of resources.

As mentioned in the introduction, normal WfMS are local information systems. Only few people in an organization are committed to model and maintain workflows. In such a case, some accounts for workflow modelers are sufficient. Employees, involved in the execution of tasks, would just need a login to their worklist in order to verify their identity. This corresponds to the classical role distinction of the WfMC between workflow modelers and workflow participants (Coalition, 1999). In a cloud-based WfMS this clear role allocation is blured, because former workflow participants can be proactively involved in the modeling of workflows enabled by the easy accessibility of the technology. The postulated collaboration and sharing requirement for a cloud-based WfMS leads to the development of a resource model for collaboration and reuse.
Collaboration and reuse of workflows have been found useful in all application domains which require extensive and diverse expert knowledge. The sharing feature (see section 6.1) is particularly important in many domains.

3 CAKE THE CLOUD-BASED WFMS

Based on the requirements of section 2 the resource model has been developed as part of the Collaborative Agile Knowledge Engine[4] (CAKE) system. The following sections briefly describe workflows in CAKE and parts of the system architecture.

[4] http://cake.wi2.uni-trier.de

3.1 Workflows in CAKE

CAKE is a prototypical generic software system for integrated process and knowledge management. CAKE integrates selected research results on agile workflows[5], case-based reasoning, and web technologies into a common platform that can be configured to different application domains and needs. Agile workflow technology (Weber and Wild, 2004; Reichert and Dadam, 1998) means that a workflow can be modeled and changed on demand and that even a running workflow can be paused and adapted to new requirements. Thus, agile workflow technology dissolves the clear separation between the build- and runtime of a workflow (Coalition, 1999).

In general, a workflow consists of tasks which are linked by a control flow, and data objects which are linked by a data flow (van Der Aalst et al., 2003). The control flow consists of a sequence of tasks and routing constructs. There are several routing constructs, and of these the parallel execution of sequences (AND-Split and AND-Join), the exclusive choice of sequences (XOR-Split and XOR-Join) or the multi-choice (OR-Split and OR-Join) are often supported by WfMS. Tasks can be executed by humans or automatically by an invoked application. The data flow consists of interlinked data objects. Data objects can be linked amongst others between tasks, between tasks and routing constructs, or between tasks and sub-workflows.

During workflow enactment, tasks can be executed by services (e.g. web services or specific implementations of automated tasks) or certain activities may require human workflow participants (e.g. experts). In Figure 1, a very simple workflow in Cake Flow

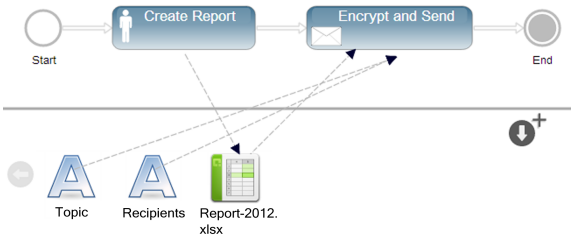


Figure 1: A simple workflow.

Cloud Notation (CFCN) (Minor et al., 2011) is illustrated. CFCN is a workflow modeling language which is derived from UML activity diagrams. We think it will receive more acceptance by the users of the CAKE system. This assumption is based on current HCI research where 'enchantment' is an important aspect and beauty is part of enchantment in using UIs (Wright et al., 2008). Furthermore many HCI researchers state that the visual appeal influences factors as (perceived) reliability, usability, information quality, trustworthiness and usefulness (Lindgaard et al., 2011). In this simple example two subsequent tasks are executed. The first task is delegated to an employee. The employee will create a report and upload it to the workflow. When the data object 'Report-2012.xlsx' becomes available, the execution of the preconfigured automated 'Encrypt and Send' task is triggered. This automated task may send the report to a list of specified recipients.

[5]For more information on agile workflow technology please refer to (Minor et al., 2008).

3.2 System Architecture

In Figure 2, the overall CAKE system architecture is illustrated. The server component consists of a storage layer which handles persistency, an interface layer for the communication with web applications and two central engines, which are briefly described in the following.

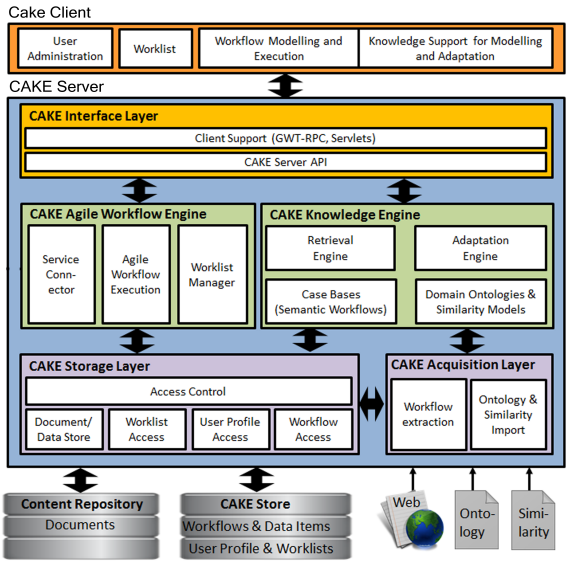


Figure 2: The CAKE system architecture.

The workflow engine is used for the enactment of workflows. Its internal architecture is strongly tied to the reference architecture of the WfMC (Coalition, 1999). Thus, the workflow engine provides interfaces for modeling and execution of workflows (Agile Workflow Execution), for invoking applications (Service Connector), and an interface for the delegation of tasks (Worklist Manager).

The knowledge engine supports process-oriented case-based reasoning (Bergmann et al., 2006; Bergmann and Gil, 2011; Madhusudan et al., 2004;

Chinthaka et al., 2009) and aims at intelligent methods for reusing experiential knowledge. It supports similarity-based retrieval of workflows based on semantic descriptions as well as the automatic adaptation of workflows (Minor et al., 2010). The aim is to support unexperienced workflow modelers with gathered and structured knowledge in a database. This experiential knowledge can then be discovered and reused in order to build or to improve a workflow under construction. The reuse of procedural knowledge is closely linked to the possibility to share and spread workflows. Though a detailed description of this scenario would go beyond the extent of this paper[6].
The interface layer provides communication means which encapsulate the functions of the two engines and it enables the development of complex web applications such as a workflow modeling and execution environment.
The proposed resource model for collaboration and reuse has been integrated by an access control mechanism in the storage layer. This layer has to ensure that any stored resource (a workflow, a task, and any further resources) is accessible and possesses a clear ownership. From an abstract point of view the access control mechanism is a decentralized Discretionary Access Control (dDAC) (Downs et al., 1985) with subject-object relationships specified in Access Control Lists (ACLs). From a detailed point of view it is a workflow specific access control concept which fulfils the given requirements of section 2 with a set of default access rules to ensure operation (see section 5). By decentralized we mean that an user can transfer access rights to another subject. The basic idea is that every resource in the system has a dedicated owner who is allowed to manage the access rights for the resource.

4 THE ACCESS CONTROL MECHANISM

In the following section, the access control mechanism is described. It starts with the resource model in general. Then, the access control mechanism for resources is described. Afterwards logical structures are introduced which facilitate an efficient control over resources and ensure an easier usage of the system.

[6]For more information please refer to (Görg et al., 2012; Bergmann et al., 2006).

4.1 The Resource Model

The resource model is illustrated in the UML class diagram of Figure 3. Everything in the CAKE system is considered as a **Resource** and the access control mechanism aims at specifying the access rights for each resource. A resource has exactly one owner. **Participants** are resources as well, and hence we need to avoid that an **Object**, e.g. a task or a data object, gets access rights to a participant. The subclasses of participant are **Account** and **Participant Group**, which is a group of an arbitrary set of accounts. Accounts have a unique identifier. The resource model does not prohibit that a participant can get access rights to another participant. As a consequence, a participant can have or give access rights to another participant. For instance the membership of an account in a participant group does not necessarily mean that the person who is a member may see the members of the participant group. Only if the person has also got the right to read the content of the participant group the group members become visible to the person.
For the retrieval of resources the enrichment with semantic information is a means to simplify and improve the discovery of resources. The **Semantic Meta Data** can be realized via **Ontologies** which should be specified in a public standard format such as OWL to enable inference mechanisms.
The left four main types of resources are the **Adaptation Case** (for case-based automatic adaptation of workflows (Minor et al., 2010)), **Data**, **Workflow** and **Task**. Due to the nature of workflow management systems, tasks are connected with participants by a specified **Execution**. In the reference model of the WfMC it is the role of the Worklist which controls the **Human Execution** and task delegation. The **Assignment** differentiates between allowed and assigned participants. The modeler may define several allowed participants to execute a task, but only one participant can be assigned in the end. If a participant group is allowed to execute a task, each member may be asked to execute the task but no one is assigned. A participant group can be a set of colleagues or experts who fulfil a predefined role in an organization. Besides the human execution, we also support automated tasks, which can be executed without user interaction. For these tasks, the **Service Execution** is responsible. Automated tasks which use the service execution are custom software components. The automatic sending of an e-mail or the retrieval of data from a database are examples for this.
A concrete workflow consists of a control flow with tasks and data objects linked with tasks. Before

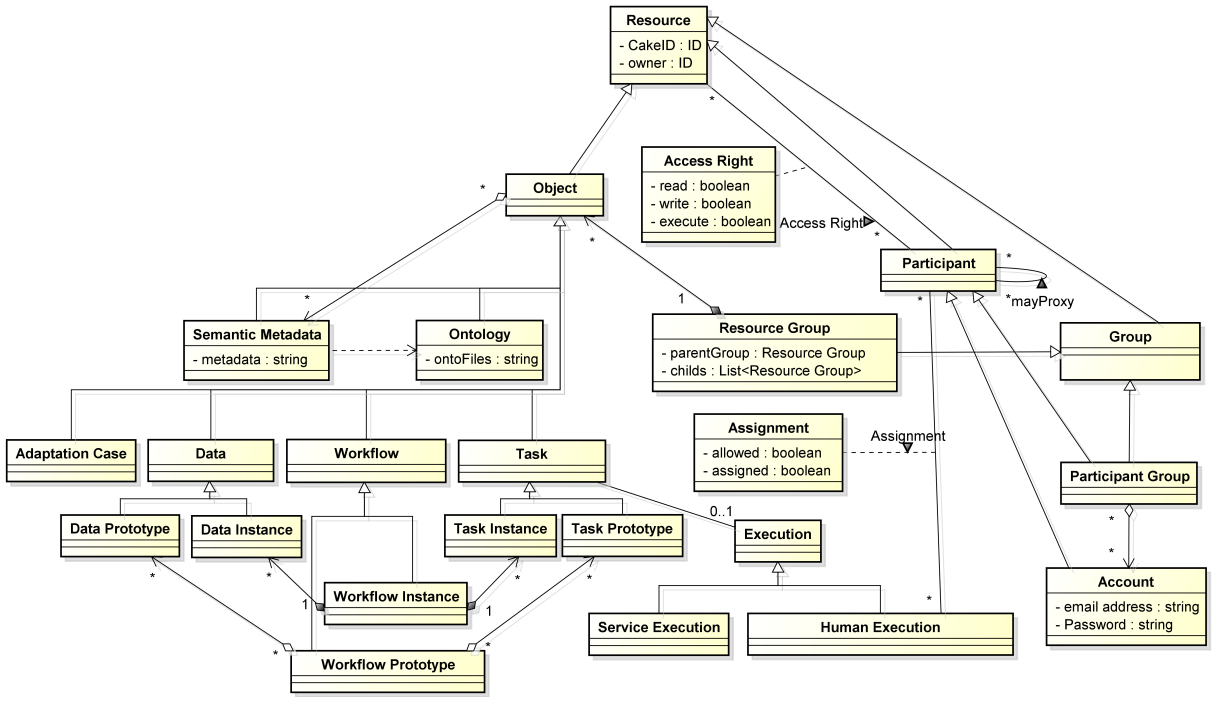


Figure 3: The resource model.

the specializations of the data, workflow and task classes will be explained (see section 5.2), the terms of instance and prototype are defined. These definitions are a generalization of the common workflow-schema/instance separation in WfMS to fit to any resource of the resource model.

- A **Prototype** is a template which cannot be executed. We have prototypes for workflow, task, and data items. We distinguish between **system** prototypes and **derived** prototypes. System prototypes are reusable entities. The usage of a system prototype leads to a derived prototype. A derived prototype is configurable and editable.
- An **Instance** is a copy of a derived prototype. The instantiation creates an executable copy of a derived workflow prototype. Only instances can be executed. Hence, we have instances for workflow, task and data items.

The reasons for the restrictions on derived prototypes as well as the reason why only instances may be used for execution are explained later on in section 5.3. The common workflow artifacts (tasks and data) and their workflow environment are described in more detail in section 5.2 together with some default access rights, which are needed to meet the requirements of section 2.
The last unexplained part of the diagram shows the **Resource Group**. There are two reasons why it is different from the participant group. The first reason is the durability of the group members. If a resource group is deleted, all of its members should also vanish, because they are no longer needed. A group of participants needs another behavior. The deletion of a participant group may never entail the deletion of its group members, i.e. the accounts. The second reason is the need to give a hierarchical structure to resources. In contrast to the participant group, the resource group should be able to form tree structures. Objects might be grouped by participants in order to organize and share personal libraries of resources. For instance, a new secretary shall see all administrative workflows for a certain division of a company. Then, the secretary only needs the read right to the folder where these workflows are.

4.2 Access Control Mechanism

Access Rights are defined between participants and resources. Thus, a set of people (participant group) or a single person (account) can get access rights to an object resource. This also means that every person can control his/her visibility to the public by restricting read rights about himself to a certain person or a group of persons. Access rights regulate three kinds of access: read, write, and execute. The access to a resource is managed by the access rights, but the access rights can only be assigned by the owner. The

resource model only regards positive access rights. It is not possible to explicitly deny a right. Hence, inconsistent access rights due to the membership of participants in different participant groups cannot occur. The access rights that a single account possesses, is a union of all access rights for this account and all participant groups in which the account has a membership. On certain resources, not all access rights are applicable. For instance the execute right is only applicable to workflow instances, but not to individual task or data items.

4.3 Logical Structures

Before the access rights and their default application are described in more detail, some logical structures are explained. These logical structures are based on the resource group and facilitate the organization of owned resources as well as rights management. Furthermore, the logical structures are a means to develop useful and coherent user interfaces. In this section we describe how logical structures are used to ensure the ownership over resources.
In Figure 4, a **Workspace** is illustrated. It is a logical structure which is intended to accommodate an arbitrary set of other logical structures. The workspace is the root for all owned resources of an account. Every account has at least one workspace. The workspace has exactly one owner. All subgroups and their group members, i.e. object resources, get their inherited ownership by the workspace owner. A workflow may only be started if it is part of a subgroup within a workspace and a person has the right to execute it. This ensures that there is always at least one responsible person for its execution, even if the executor is not the owner. To organize resources in a workspace,

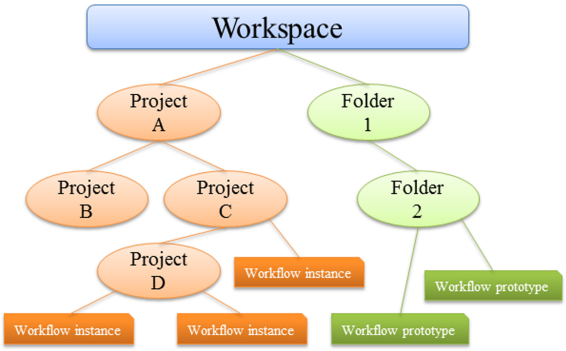


Figure 4: The Workspace - A logical structure.

different logical tree structures can be mounted in the workspace. Mounted means that the workspace resource group gets new members which are also resource groups.

5 APPLICATION OF ACCESS RIGHTS

This section is about access rights and their default application in the context of workflows and resources in general, based on the requirements in section 2. First, the term of ownership is explained and how it is related to a workspace. Then, default access rights to resources are described from different views in order to give a recommendation for a cloud-based WfMS. In the end, the instantiation of derived prototypes is explained in more detail.

5.1 Ownership

Access rights can only be assigned or removed by the resource owner. Ownership can only be taken by a single account. According to the workspace structure all resources in a workspace have the same owner. The concept of access rights could also allow to transfer the ownership of a resource to another participant or group of participants, but this will not be supported for the following reasons.
If there is a workflow on which several people collaboratively work, it is not intended that a user can lose the ability to read, write, or execute his/her formerly owned workflow, because someone else has taken the ownership. In such a case, the owner of the logical structure in which the workflow lies would differ from the workflow owner and this would violate the concept of the workspace. Also non-executable resource objects, such as data (documents), must meet this requirement of the workspace. If a transfer of ownership would be allowed it would cause a resource to switch the workspace.
If the ownership of a resource could be taken by a participant group, then it would be possible that an empty group is the owner of an already running workflow instance, which would also lead to the consequence that no one is responsible for the execution of the workflow and its monitoring. Therefore, only accounts may have a workspace.

5.2 Default Access Rights

After the explanation of the resource model, the access control mechanism in general and its derived logical structures, the data, tasks, and their workflow environment are described by the terms of prototypes and instances.

Data within a workflow can consist of documents, files, or primitive types such as integer or string values. Their system prototypes can be stored in a document management system which uses versioning to

provide the proper release of data objects. By using a data system prototype in the context of a derived workflow prototype / workflow instance, it becomes a derived data prototype / data instance.

Tasks within a workflow can either be automated tasks, which are executed without user interaction or human tasks which needs to be executed by a person. Their system prototypes are specified descriptions of concrete implementations or web-services. By inserting a task system prototype into the control flow of a derived workflow prototype / workflow instance it becomes a derived task prototype / task instance.

Workflows combine tasks by control flow structures and data objects with tasks by data links (van Der Aalst et al., 2003). The system prototype of a workflow is not comparable with the ones for data objects or tasks. For workflows, only one type of system prototype exists. It is an empty container which is able to accommodate data objects and tasks in order to link these to form a logical sequence of activities. By modeling the control flow, it automatically becomes a derived prototype. If we talk of workflow prototypes, it is indeed a derived workflow prototype already.[7]

An operational cloud-based WfMS would need much time to check the access to every resource or even more granular for every attribute of a resource. For this reason this section explains where particular access rights are necessary, considering the modeling and the execution of workflows. The first scenario is described from the perspective of a workflow modeler. Table 1 is a sample instance of the table 'Access Rights' in Figure 3 and explains which rights have to be assigned explicitly. The second scenario is described from the view of a workflow participant who receives a delegated task. It explains why some rights have to be set implicitly for the 'Assignment' of the 'Human Execution' (see Figure 3).

Table 1: Access rights during workflow modeling.

	read	write	execute
System data prototypes	x		
System task prototypes	x		
(Derived) workf. proto.	x	x	
Workflow instances	x	x	x

Modeling of workflows Table 1 shows the access rights which are relevant for workflow modeling. Read rights to system data and task prototypes are required in order to see available data items and tasks which the workflow modeler can use to create the new workflow, i.e. a derived workflow prototype. Hence, read access to these items enable to control which items are available to a certain user for modeling a workflow. These rights may, for example, be specific to a certain participant group. In order to edit the new workflow, the user needs the read and write right to the respective derived workflow prototype being modeled, as the main purpose of modeling is writing a new workflow. These rights are automatically passed to all derived data and task prototypes created as part of this workflow. Hence, a certain right to a derived workflow prototype implies the same right to all of its components. Consequently, there is no need to explicitly specify individual rights for derived data and task prototypes. This simplifies the rights management and avoids incomplete access to a workflow.
Once the workflow prototype is completely modeled, the user has to create a workflow instance as only the workflow instance is executable. In order to actually execute the created instance, the execute right is necessary. The additional write right for the instance specified in the table entitles the user also to modify the executed workflow after it has been started. As for derived workflow prototypes, the access rights for workflow instances are also passed implicitly to the task and data instances. Hence, there is also no need to explicitly specify individual rights for data and task instances.

Execution of human tasks Human task instances are used for the delegation of tasks to participants by the human execution and the specified assignment. Thus the description of tasks must be readable for the assigned or allowed participants as well as provided input data and the ability to write output data. So the assignment of the human execution implies the access rights for task and data instances. Task instances are limited to a read right, because their content and their structure within the control flow is not influenced by the person who is assigned to a human task. During delegation, the task instance will always be in the same position in the control flow and all incoming and outgoing data links will remain unaltered. Data instances need an additional write access for assigned persons if the assigned person has to change a data object or if the creation of a new data object is required for fulfilling the task (see first task in Figure 1).

5.3 Instantiation of Derived Prototypes

A participant needs a read right to a workflow prototype in order to create a workflow instance. The

[7] A derived workflow prototype is equivalent to the common term of a workflow schema/template/definition in WfMS.

owner of the workflow instance is the person who engaged the instantiation process and thereby the workflow instance is part of the workspace of that person. The resource model allows to instantiate all workflow prototypes which are readable for a participant. This is useful if a workflow needs to be executed only one time for a participant and the participant used a workflow prototype from a public repository of workflows. If the participant needs to adapt the workflow instance to his/her needs, the agile workflow technology (see section 3.1) of the CAKE WfMS even allows to change a running workflow instance.
Alternatively, the participant can copy the workflow prototype into his/her own workspace before the instance is created and the workflow is started. The copied workflow prototype is then owned by the participant and thereby she/he also receives read and right access rights to the workflow prototype. This enables him/her to adapt the workflow to his/her particular needs, which is useful in cases multiple equivalent instances of this workflow prototype have to be executed.

6 COLLABORATION AND SHARING

In this section, the processes of sharing resources and the collaborative workflow modeling are briefly described. Afterwards a general issue on copying workflows is discussed in the context of collaboration and sharing.

6.1 Sharing of Resources

Sharing of resources is the reuse of resources by other participants. This means that not only a read right is passed from one participant to another participant but that a resource is copied from one workspace to another. Every resource which is copied this way is recorded in a table which is part of the access control mechanism. This sharing table can be seen as a history for resources and their provenance in a network of participants. Especially, sharing of workflows is of interest because they contain much procedural knowledge. The sharing table (see bottom of Figure 7) allows tracing the provenance of workflows. This serves several purposes. First, in case that workflows contain illegal data or an user misuses the system, it allows us to deactivate all undesired resources and their copied derivations in the cloud-based WfMS. Second, in the scenario of personal/social workflows (Görg et al., 2012) people can see if their workflows are reused. Socializing, besides communication, is the most important motivational factor for using social network sites (Brandtzæg and Heim, 2009) and approval and feedback of other people are part of socializing. The opportunity to see who is not only reading my work but also reusing it, is known as feedback mechanism (Bellotti and Sellen, 1993). This mechanism leverages the access control model to a more transparent model in which a user not only knows to whom resources are passed, but also who is reusing the resources.
In the following the process of sharing a workflow prototype is shown in an example, in which user A wants to share a workflow prototype with user B.

In Figure 5, user A has a workflow prototype in a

Resource	User A Read	User A Write	User B Read	User B Write	Owner
Folder 1	X	X			User A
WF prototype 1	X	X			User A

Figure 5: Sharing a workflow prototype - part I.

Resource	User A Read	User A Write	User B Read	User B Write	Owner
Folder 1	X	X	X		User A
WF prototype 1	X	X	X		User A

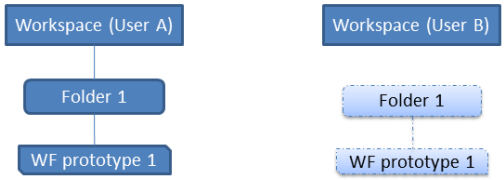
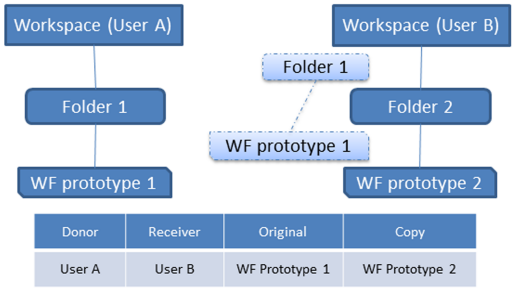


Figure 6: Sharing a workflow prototype - part II.

folder of his/her workspace. Only User A has read and write access to the folder and to the workflow prototype. User B cannot see anything else than his/her own workspace. In Figure 6, user A has given the read right for the workflow prototype and its parent folder to User B. For User B the readable folder and the workflow appear in a space for readable resources. In Figure 7, user B decides to reuse the workflow prototype. Therefore it needs to be copied in his/her own workspace. A new workflow prototype is created, which is owned by User B and which is not visible

Resource	User A Read	User A Write	User B Read	User B Write	Owner
Folder 1	X	X	X		User A
WF prototype 1	X	X	X		User A
Folder 2			X	X	User B
WF prototype 2			X	X	User B



Donor	Receiver	Original	Copy
User A	User B	WF Prototype 1	WF Prototype 2

Figure 7: Sharing a workflow prototype - part III.

for User A. User A gets feedback about this through access to the provenance information.

6.2 Collaborative Workflow Modeling

The access control mechanism allows collaboration on different levels. Not only workflows can be edited collaboratively but also all logical structures which are based on the resource group. So an owner can give write access to a folder to different participants. Then, each of these participants must also get the write right to all workflows which are organized below the concerned logical structure. If such a write-released resource is modified by inserting a new resource, the access rights must also be assigned to the owner of the parent structure and all participants which have write rights to the parent structure.

Resource	User A Read	User A Write	User B Read	User B Write	Owner
Workspace	X	X			User A
Folder 1	X	X	X	X	User A
WF prototype 1	X	X	X	X	User A
WF prototype 2	X	X	X	X	User A

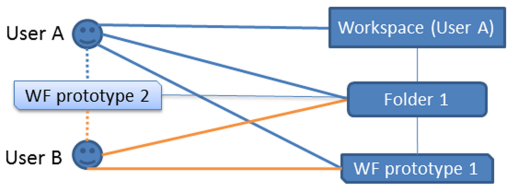


Figure 8: A collaboration example.

Figure 8 shows an example for collaboration. The workspace owner, User A, has granted User B read and write rights to 'Folder 1'. Thus, User B has read and write access to all resources within this folder, i.e. 'workflow prototype 1'. User B may create 'workflow prototype 2' in 'Folder 1' because she/he has a write right. User B is not the owner of the new workflow prototype but she/he has the inherited rights of 'Folder 1' to the new workflow. Then, both users can collaboratively model 'workflow prototype 2'.

6.3 Issues on Copying Workflows

The invitation of participants for a desired collaboration on the design of a workflow and the process of sharing a workflow are quite similar processes up to the point where a new workflow prototype is created. In both situations at least a read right to the workflow is granted to another participant. This read right is the authorization to see all data and tasks in the workflow. However, because of the possible connection of human tasks with participants through the execution assignment, the visibility of allowed or assigned participants must be restricted. This problem is similar to information leakage in Service Oriented Architectures and Cloud-Services. The resource model can handle this issue, because there is a access right relationship between participants (see Figure 3). If a workflow is readable for other participants the assignment on human tasks is only visible for them if there is an explicit read right between these participants and the participants in the assignment.

7 DISCUSSION

Due to the extent of this paper, some issues of the resource model and the access control mechanism cannot be discussed: The modeling and access of hierarchical workflows and the reuse of workflow instances requires a version management for the produced data during the runtime, especially in the context of agile workflows (Minor et al., 2008). Of course, collaborative workflow modeling also requires a lock-mechanism to avoid read-write conflicts.

The access control mechanism described in the paper is fully implemented in the CAKE system; an appropriate graphical user interface that enables to assign rights to resources in the manner described is currently under development. Future work will particularly include an empirical evaluation of the CAKE cloud-based workflow management system, including the sharing and collaboration features.

The problem of access control and security issues is

also widely known in the field of SOA. The access control mechanism in this work is based on the identity of a user and in a first step it cannot meet the requirements to fulfil Service Level Agreements, a trustworthy service invocation or can guarantee automatically compliance to public or internal policy requirements. Though, the field of SOA has a bigger scope. Inter-organizational processes and heterogeneous IT-infrastructures are not yet in the focus of the CAKE system.

ACKNOWLEDGEMENTS

This work is part of the WEDA project. WEDA is funded by Stiftung Rheinland-Pfalz für Innovation, grant no. 974.

REFERENCES

Bellotti, V. and Sellen, A. (1993). Design for privacy in ubiquitous computing environments. In *ECSCW*, pages 75–.

Bergmann, R., Fremann, A., Maximini, K., Maximini, R., and Sauer, T. (2006). Case-based support for collaborative business. In Roth-Berghofer, T., Göker, M. H., and Güvenir, H. A., editors, *Advances in Case-Based Reasoning, 8th European Conference, ECCBR 2006, Fethiye, Turkey, September 4-7, 2006, Proceedings*, volume 4106 of *LNCS*, pages 519–533. Springer.

Bergmann, R. and Gil, Y. (2011). Retrieval of semantic workfows with knowledge intensive similarity measures. In *Case-Based Reasoning. Research and Development, 19th International Conference on Case-Based Reasoning, ICCBR 2011*, volume 6880 of *LNCS*, pages 17—31. Springer.

Brandtzæg, P. and Heim, J. (2009). Why people use social networking sites. *Online Communities and Social Computing*, pages 143–152.

Chinthaka, E., Ekanayake, J., Leake, D., and Plale, B. (2009). Cbr based workflow composition assistant. In *Services - I, 2009 World Conference on*, pages 352 –355.

Coalition, W. M. (1999). Terminology and glossary. `http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf`. [Online; accessed 14-Nov-2012].

Downs, D., Rub, J., Kung, K., and Jordan, C. (1985). Issues in discretionary access control.

Görg, S., Bergmann, R., Minor, M., Gessinger, S., and Islam, S. (2012). Collecting, reusing and executing private workflows on social network platforms. In *WWW'12 Workshop Proceedings*.

Lindgaard, G., Dudek, C., Sen, D., Sumegi, L., and Noonan, P. (2011). An exploration of relations between visual appeal, trustworthiness and perceived usability of homepages. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 18(1):1.

Liu, X., Yuan, D., Zhang, G., Li, W., Cao, D., He, Q., Chen, J., and Yang, Y. (2011). *The Design of Cloud Workflow Systems*. SpringerBriefs in Computer Science. Springer.

Madhusudan, T., Zhao, J., and Marshall, B. (2004). A case-based reasoning framework for workflow model management. *Data and Knowledge Engineering*, 50(1):87 – 115. Advances in business process management.

Minor, M., Bergmann, R., and Görg, S. (2011). Adaptive workflow management in the cloud - towards a novel platform as a service. In *Proceedings of the ICCBR 2011 Workshops*, pages 131—138.

Minor, M., Bergmann, R., Görg, S., and Walter, K. (2010). Towards case-based adaptation of workflows. In Montani, S. and Bichindaritz, I., editors, *Case-Based Reasoning. Research and Development, 18th International Conference on Case-Based Reasoning, ICCBR 2010, Alessandria, Italy, July 19-22, 2010. Proceedings*, LNAI 6176, pages 421–435. Springer.

Minor, M. and Görg, S. (2011). Acquiring adaptation cases for scientific workflows. In *Case-Based Reasoning. Research and Development, 19th International Conference on Case-Based Reasoning, ICCBR 2011*, volume 6880 of *LNCS*, pages 166–180. Springer.

Minor, M., Schmalen, D., Kempin, S., and Kempin, S. (2009a). Demonstration of the agile workflow management system cake ii based on long-term office workflows. In *BPM (Demos)*.

Minor, M., Schmalen, D., and Koldehoff, A. (2009b). Fallstudie zum einsatz agiler, prozessorientierter methoden in der chipindustrie. In Hansen, H. R., Karagiannis, D., and Fill, H.-G., editors, *Business Services: Konzepte, Technologien, Anwendungen, 9. Internationale Tagung Wirtschaftsinformatik, 25. - 27. Februar 2009, Wien*, volume 1, pages 193 – 201. Oesterreichische Computer Gesellschaft.

Minor, M., Tartakovski, A., Schmalen, D., and Bergmann, R. (2008). Agile workflow technology and case-based change reuse for long-term processes. *International Journal of Intelligent Information Technologies*, 4(1):80–98.

Reichert, M. and Dadam, P. (1998). Adept flexsupporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2):93–129.

van Der Aalst, W., Ter Hofstede, A., Kiepuszewski, B., and Barros, A. (2003). Workflow patterns. *Distributed and parallel databases*, 14(1):5–51.

Walter, K., Minor, M., and Bergmann, R. (2011). Workflow extraction from cooking recipes. In Diaz-Agudo, B. and Cordier, A., editors, *Proceedings of the ICCBR 2011 Workshops*, pages 207–216.

Weber, B. and Wild, W. (2004). An agile approach to workflow management. *Proceedings of Modellierung 2004*, pages 187–201.

Wright, P., Wallace, J., and McCarthy, J. (2008). Aesthetics and experience-centered design. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 15(4):18.