Using Workflow Context for Automated Enactment State Tracking

Thomas Sauer[1] and Kerstin Maximini[2]

[1] rjm Business Solutions GmbH
68623 Lampertheim, Germany
t.sauer@rjm.de
[2] University of Trier
Department of Business Information Systems II
54286 Trier, Germany
kerstin.maximini@wi2.uni-trier.de

Abstract. Workflows are getting a more and more pervasive concept for modeling arbitrary activities. The resulting workflow definitions will convey knowledge how and when to apply the right procedures. Whenever innovative, experimental or explorative tasks have to be fulfilled, however, workflows will evolve to previously unknown structures. In order to allow planning and quality assurance nevertheless, the tasks currently enacted within an organization must be known. In this paper, an approach for automatically tracking the active tasks by observing the data produced is presented. Further, it is described how this approach is applied for geographical information management.

1 Introduction

In the last decades, the concept of workflows has evolved from a means to describe the flow of paperwork through an organization to a more abstract and general technique used in many application domains. Nowadays, workflows are most often considered a programmatic structure for designing and executing arbitrary transactions.

Workflows have long been considered useful for storing common and standard procedures, leading to an organizational memory similar to process manuals [1]. However, repeatable methods are not always applicable, and sometimes not even desirable. Whenever innovative, experimental or explorative tasks have to be fulfilled, workflows will evolve to previously unknown structures. These fields have been characterized as knowledge intensive tasks [2] as well as business and time critical processes [3,4]. In order to adhere the challenges in these fields, the *Collaborative Agent-based Knowledge Engine (CAKE)* has been created [5,6]. CAKE offers modeling and enactment support for *agile workflows* or *adaptive workflows* as a flexible solution to express and maintain best practices, lessons learned within the organization, etc.

In this paper, a formal approach for workflow representation and enactment is presented. Furthermore, based on this formalism, a procedure for capturing context information during workflow enactment is introduced, and how to track the currently active task within a workflow using this context information. This allows to learn about the current enactment state or deviations from the original model, at minimal effort for the

team members. For instance, this is useful for manager roles to ensure that the team is focusing on the most appropriate tasks at the moment.

Besides several other application scenarios, CAKE is used for geographical information management [7]. In the next section, this application scenario is presented and the requirements on workflow enactment support are discussed. In section 3, CAKE workflow modeling and the capturing approach are presented. Section 4 shows how to put the latter into practise. In section 5, related and adjacent work is discussed, and finally, a short conclusion and outlook to future work close this paper.

2 Application Scenario

Recently, changes to law according to the regulation of civil works in the German federal state of Hessen does no longer demand owners to seek explicit permission for most refurbishment tasks on buildings. Rather, planners and architects have to ensure themselves that they comply with all regulations that apply to the corresponding land parcels, including monument protection.

In Hessen, each building and site of historic interest (e.g. old marketplaces, or characteristic quarters) is listed in the official monument register. For each monument, its exact location is recorded, as well as a rationale why the monument is subject to protection, and which type of protection applies. For instance, a building may be protected in a whole, or only parts of it may be of historic interest (e.g. the cladding, or an annex). As yet, these facts have been published in print only, leaving many planners and owners unaware of monument protection. This invites inappropriate reconstruction, damage, or even complete demolition of cultural heritage.

To overcome this, rjm business solutions GmbH has been assigned by the Monument Protection Agency of Hessen to conduct the long-term eGovernment project *DenkXweb*. DenkXweb provides a freely accessible Internet service to publicly access the monument register[3]. It presents the protection rationale as described above as well as a detailed map denoting location and dimensions of the protected building or site. As laid down by law [8], the latter is based on the official cadaster as provided by the surveying and mapping authorities. Thus, using DenkXweb, users can look up whether a buildings is a monument by itself, or whether according land parcels are included in site protections (which will have implications on any construction on these parcels).

Besides developing the Internet service, rjm business solutions GmbH and its associate company rjm medienservice GmbH provide data management services as well in order to combine monument register data with the geospatial information provided by the land cadaster. That is, internal tools and procedures are developed to create data displayed by the Internet service most cost-effective, but still as accurate as possible. During everyday work, handling these procedures have proven to be highly knowledge intensive and require the coordination of collaborative activities between the participants.

[3] http://www.denkmalpflege-hessen.de/denkxweb

2.1 Requirements to the Envisaged Solution

Several participants from different organizations will have to be coordinated. Technical roles at rjm will create and maintain the custom software, and will prepare data as required for publication. Technical roles at the monument protection agency will be involved in revising data, and will keep in touch with local authorities, who will in turn notify the agency whenever monuments are demolished, repaired, etc. This applies to all municipalities in Hessen (several hundred), i.e. a large number of interleaved activities will have to be concerted (several thousand) across multiple organizations.

However, these activities cannot be fully planned in advance. For instance, when a project stakeholder demands a specific modification, or when a time-critical issue has to be solved (web server downtime or similar), priorities will be altered, and the original plans will have to be adapted to the new situation. So, in order to deliver support for this application scenario, CAKE has fulfill the following requirements:

1. The various activities and procedures applied have to be defined formally in order to ensure that they are done most cost-effective and most accurate.
2. The currently executed activities have to be revealed in order to allow short-term planning.
3. The role of individuals, tools, etc. has to be exploited for coordination across organization boundaries (e.g. determine whether a change in a custom piece of software affects a customer system).
4. CAKE has to be incorporated within the organization as seamlessly as possible, since users already have to master a plethora of applications, tools and utility programs.

In the next section, the general CAKE architecture is shown with a special focus on workflow modeling and workflow enactment. Afterwards, detecting workflow context is discussed, and how this can help to fulfil the requirements listed above.

3 The CAKE Approach

The approach presented within this paper is built around the *Collaborative Agent-based Knowledge Engine (CAKE)* [3,4,5,6]. CAKE is a general domain-independent system providing a general data and workflow model. The CAKE architecture is illustrated in figure 1. The system consists of three major building blocks, namely the *agent framework*, the *workflow management component*, and the *CBR component*.

The agent framework is a unified interface to couple external knowledge sources as well as user interfaces. It distinguishes between *information agents* and *user agents*. Information agents connect information sources to the CAKE system, enabling access to search engines, databases, groupware calendards, human experts, etc. User agents request knowledge from the CAKE system, and represent interfaces to human system users (e.g. graphical user interfaces or natural language interfaces). For the scope of this paper, only information agents are of particular relevance.

CAKE aims at supporting flexible and changing processes through *adaptive workflow management*. This is realized by sophisticated search facilities for business processes, single tasks, and agents following a a *structural CBR (SCBR)*[9] approach. The

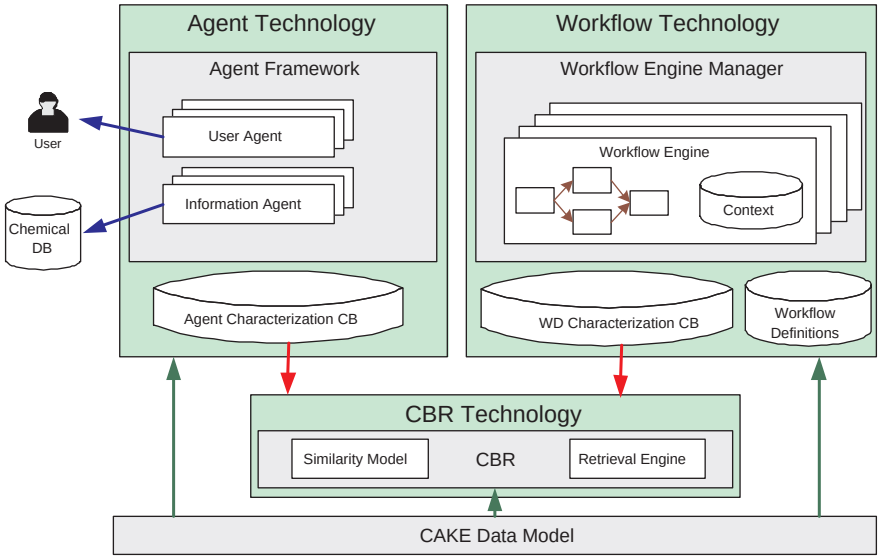

Fig. 1. CAKE System Architecture

latter is working on top of a domain-specific data model and considering the semantics and structural aspects, meets all requirements.

The workflow management component is used for modelling business processes and for specifying collaboration among agents which are following a common goal. In the following section, the workflow management component is presented in detail, and how it can be used to provide dynamic guidelines for planning and enactment of arbitrary activities. Section 3.2 introduces an algorithm how to automatically follow current work progress in the model, and how to detect deviations from the model. The last sections 3.3 and 3.4 focus on technical issues necessary to put the system into practise.

3.1 Workflow Management

The CAKE workflow model approach is based on a data model $\mathbb{D}$ to describe data objects using an object-oriented type hierarchy. The data model is shared across all system components and using this data model, all entities relevant in the organization (e.g. documents, capabilities of humans and machines) can be expressed. The data model contains types to represent the different entities present within the organization, and instances of these types are called *data objects*. More precisely, the data model is an object-oriented model using specialization and aggregation to define the data classes. Available data classes are atomic classes like boolean, integer, etc. as well as compound data classes like aggregates, collections, or intervals.

The following definition describes the representation for any operation taking place in the real world, or are expressed as a machine-executable program:

Definition 1. ***(Task)*** *A* task *describes an arbitrary activity that may be carried out by either a human or a machine by a tuple* $t = (In_t, Out_t)$ *with* In_t, Out_t *two non-empty disjoint sets containing labels denoting the* input ports *and* output ports *of the task.*

A task may cover course-grained activities like "write a report", "send email" as well as fine-grained operations like comparing two alternatives in order to form and-joins, or-splits etc. The task ports are simply labels attached to a task defining the input and output slots for arranging control flow.

Further, $\mathbb{T}$ denotes the set of all tasks available. For simplicity, for two tasks $t, t' \in \mathbb{T}$, it is required that $In_t \cup In_{t'} = \varnothing$ and $Out_t \cup Out_{t'} = \varnothing$. That is, the labels used to denote input and output ports are meant to be unique.

For a set of tasks $T \subseteq \mathbb{T}$, the sets of all input and output ports used are given by $In_T = \bigcup_{t \in T} In_t$ and $Out_T = \bigcup_{t \in T} Out_t$ respectively. Connections between these ports will then express control flow between tasks, as discussed by the next definition:

Definition 2. ***(Workflow Definition)*** *A* workflow definition *is a tuple* $wd = (T, t_s, t_f, conn)$ *comprising a set of tasks* $T \subseteq \mathbb{T}$*, a start task* $t_s \in T$ *with* $|t_{sin}| = 1$*, a final task* $t_f \in T$ *with* $|t_{f_{out}}| = 1$ *and a bijective control flow function* $conn : Out_T \setminus t_{f_{out}} \mapsto In_T \setminus t_{sin}$.

Thus, the control flow between tasks is described by listing connections between output and input ports. For each output port o of the start task t_s, $conn(o)$ points to the input port(s) of the second task; for each output port o' of the second task, $conn(o')$ points to the third task and so on until the final task t_f is reached. Since the number of input and output tasks is not limited for any task besides start and final tasks, selection among alternatives (e.g. "or" split), parallelism (multiplexers) and synchronization (demultiplexers) may be expressed by the tasks.

A workflow definition wd does not establish explicit data flow among tasks. While explicitly expressing data flow would have its advantages for controlling and limiting further evolution (e.g. adding or replacing tasks), it would also have disadvantages due to increased complexity for modeling and connecting tasks. Instead, CAKE establishes data flow implicitly during run time, which is discussed in detail below.

The definitions above are already sufficient to describe the structure of a workflow, i.e. how to reach a goal using a structured approach. In order to put this description into practice, enactment of tasks and workflow descriptions have to be discussed, i.e. how artifacts are produced and consumed. Hence, enactment means to instantiate a concrete control flow within the structure given by the underlying workflow definition, and to describe data flow between the tasks.

Definition 3. ***(Task Enactment)*** *Let* $t \in \mathbb{T}$ *be a task. The* task enactment *of* t *is described by* $enact_t : \mathcal{P}(In_t) \times \mathcal{P}(\mathbb{D}) \mapsto \mathcal{P}(Out_t) \times \mathcal{P}(\mathbb{D})$.

That is, based on a set of "ready" input ports and a set of data available to the task, task enactment chooses a set of output ports and modifies the set of data. For easier reading, for $enact_t(C) = (O, C')$, $C' \subset \mathbb{D}$ and $I \subset In_t$ the projection $result_t(In_t, C) = C'$ is used. Task enactment starts when a sufficient set of input ports are enabled (this is discussed below in detail.) The data available for consumption by a task is also known as its *context*:

Definition 4. ***(Context)*** *A* context $C = \{c_1, c_2, \ldots, c_n\}$ *is a finite set of data objects* $c_i \in \mathbb{D}, 1 \leq i \leq n$.

When enacting a sequence of tasks, their results given by $enact_t$ may be accumulated into a context ready for consumption of the next task. For example, when assuming that data once created will never be altered nor deleted, for the sequence $t_1, t_2, \ldots, t_n$ the respective contexts could be determined as $C_{i+1} = C_i \cup result_t(C_i)$, $0 < i < n$. Accumulating results given by $enact_t$ into a new context is discussed below in more detail. Anyhow, a task may access data present within its context randomly, i.e. data flow between tasks is enabled using the blackboard paradigm (e.g. [10]).

A workflow definition provides structural information how the tasks are interrelated. For concerting their enactment, the data already produced, the input ports currently available and the tasks currently active have to be maintained within an *enactment state*:

Definition 5. ***(Enactment State)*** *Let* $wd = (T, t_s, t_f, conn)$ *be a workflow definition. Let* $I \subset In_T$ *be the set of currently enabled input ports, let* $A \subset T$ *be the set of currently active tasks (i.e. tasks that may be enacted in parallel), and let* C *be a context for these tasks. Then* $state_{wd} = (I, A, C)$ *is called the* enactment state *of* wd.

An enactment state describes the results achieved after having enacted a number of tasks. A sequence of enactment states can be understood as a *workflow instance* of a workflow definition or simply *workflow*:

Definition 6. ***(Workflow Instance, Workflow)*** *Let* wd *be a workflow definition. A* workflow instance *of* wd *or short* wd-workflow *is a sequence* $wf = \{state_{wd1}, state_{wd2}, \ldots\}$ *of enactment states.*

That is, while the workflow definition lays out tasks and their interrelationships, the workflow instance expresses the results achieved by enacting them step-by-step as arranged by the workflow definition. The results achieved during one step is expressed by *workflow enactment*:

Definition 7. ***(Workflow Enactment)*** *Let* $wf = \{state_1, \ldots, state_n, state_{n+1}, \ldots\}$ *be a workflow.* Workflow enactment *of* wf *is described by* $enact_{wf} : \mathcal{P}(In_T) \times \mathcal{P}(T) \times \mathcal{P}(\mathbb{D}) \mapsto \mathcal{P}(In_T) \times \mathcal{P}(T) \times \mathcal{P}(\mathbb{D})$ *so that* $enact_{wf}(state_n) = state_{n+1}$.

Generally, workflow enactment starting at $state_n = (I_n, A_n, C_n)$ will advance to $state_{n+1} = (I_{n+1}, A_{n+1}, C_{n+1})$ as described below:

1. For each $t \in A_n$, $enact_t(I_n, C_n) = (O_{t,n}, C_{t,n})$ is calculated. Thus, $O_{t,n} \subseteq Out_t$ is a set of output ports belonging to task t, and $C_{t,n} \subset \mathbb{D}$ is a set of data objects produced or altered by t.
2. For each output port $o \in O_{t,n}$, the corresponding input ports are retrieved as $Conn_{t,n} = \bigcup_{o \in O_{t,n}} conn(o)$.
3. It is $I_{n+1} = \bigcup_{t \in A_n} Conn_{t,n}$: The set of input ports available in $state_{n+1}$ is the union of all input ports connected to the output ports selected by the enactment of the currently active tasks.

4. It is $A_{n+1} = \{t \in T | I_{n+1} \cap In_t \neq \varnothing\}$: The set of active tasks in $state_{n+1}$ is made up of tasks featuring the input ports found in the previous step.
5. Finally, C_{n+1} is created from merging C_n with $C_{t,n}$, the data objects learned from the tasks $t \in A_n$.

The merging procedure has to be understood as a problem of its own, as conflict mitigation strategies have to be established. For example, one task might try to delete a product while another task still needs access. Issues regarding merging products to create a new context are similar to those discussed in relational database research, and are commonly addressed by transaction management systems. A very simple merging procedure can be given when restricting the context to be monotonous, i.e. that data once created will never be altered nor deleted. Then, it is $C_{n+1} = \bigcup_{t \in A_n} R_t$.

In the above model of workflow enactment, a task is allowed to return an empty set of output ports, indicating that no port can be selected yet (e.g. because the input ports available are not sufficient yet, or because no appropriate data has been found on the context). This may stall a workflow when reaching an enactment state that does compute another state (i.e. $state_n = state_{n+1}$). This has to be handled by the execution environment (e.g. by sending a warning message to the user) as the model itself makes no further assumptions.

As a matter of fact, quite a number of process definition languages and workflow models have emerged during the last decades. CAKE does not aim to replace a specific language, and is not bound to a certain methodology how to perform workflow modeling. Instead, the workflow model has been designed to be as simple as possible in order to fulfill the requirements demanded by the application domains. If required, translation between workflow modeling languages is an option [11] e.g. to leverage specific modeling tools.

When putting the workflow definitions into practice, i.e. when people start enacting them, it is crucial to know about the current enactment state. The set of currently active tasks, for instance, is the foundation of short-term planning, and the vantage point for agile workflow evolution. Traditional workflow-oriented systems follow the plan-do-check-act paradigm. Using a workfow definition created prior to enactment, the set of active tasks is determined and presented as a to-do-list to the user. The user then accepts or rejects the tasks assigned to them, and reports whenever a certain task has been accomplished (or parts of it.) This interaction, however, is often considered tedious work, leading to an acceptance problem: Besides “doing the work”, users feel forced to reason about work progress and their task assignments.

Furthermore, whenever a user executes a workflow spontaneously in order to handle an unforeseeable event (e.g. to handle a customer request), a constructive approach requires him to explicitly represent this workflow enactment as a change of plan in order to add the workflow on the to-do-list. This tends to be cumbersome, especially if the workflow only lasts for a short period in time (e.g. half a day), thus the system gets omitted instead of being used for guidance, and for quality assurance.

To overcome these issues, the idea is to integrate the workflow management system as closely within the organization’s tool chain as possible. That is, information already created using the tools already deployed within the organization is leveraged. Thus,

tasks currently enacted by users are identified not by observing the actual activities done by the users (e.g. "opening E-mail client" or "selecting monument spatial feature"), but the products created during these activities (e.g. the Email sent, or the spatial feature.) Following the definitions given above, this set of products resembles a *context*.

3.2 Deriving enactment state from context

A context resulting from real-world activities contains a rich variety of products ranging from database records, informal and semiformal documents (e.g. emails), formal documents (e.g. CAD drawings), etc. When transforming this collection of real-world information item into a set of data objects represented using the CAKE unified data model, the latter set may be understood as an *observed context* created by enactment of an otherwise hidden workflow definition.

As defined above, a workflow is a sequence of enactment states $wf = \{state_1, state_2, \ldots, state_n, state_{n+1}, \ldots\}$ with the states $state_i$, $i > 1$ resulting from task enactment of the previously active tasks. In turn, task enactment of these predecessor tasks means mapping input ports and input data to output ports and output data. Hence, task enactment may be either characterized *a priori* by the premises (input ports and input data) or *a posteriori* by its results (output ports and output data). For $state_n = (I_n, A_n, C_n)$, the a priori characterization for each of the active tasks $t \in A_n$ is simply $Char_{t,prior} = (I_n, C_n)$. For the a posteriori characterizations the tasks $t \in A_n$ are determined by $Char_{t,post} = enact_t(Char_{t,prior})$. The latter characterizations then are merged and processed to form $state_{n+1}$. Thus, a workflow contains an a priori task characterization within each of its individual states.

When comparing the observed context with the C_i contained in the enactment states, the accompanying set A_i gives the desired set of currently active tasks.

Anyhow, as the actual workflow definition is hidden, the actual workflow is too. Assumed that a case base contains tuples $(Char_{t,prior}, t)$, a reasonable set of active tasks could be determined nevertheless by using structural CBR. Figure 2 illustrates this approach.

3.3 Creating an Observed Context

The CAKE system design introduces *information agents* that represent the various information sources present in the organization [5,6]. Each information agent encapsulates translation rules and procedures how to transform information items available from these sources into CAKE data objects.

Translating information items into data objects basically means to identify "interesting" properties contained within the information item. For instance, an activity record created by a time recording application provides a unique ID number, a work description, the user name who has performed the task, and the duration of the overall operation. While the last three properties may be considered useful by the information agent, the unique ID number is only meaningful to the underlying RDBMS of the time recording application, and is omitted.

For other information items it may not possible to identify distinct properties. For example, a CAD drawing stored in a proprietary data format makes it impossible to

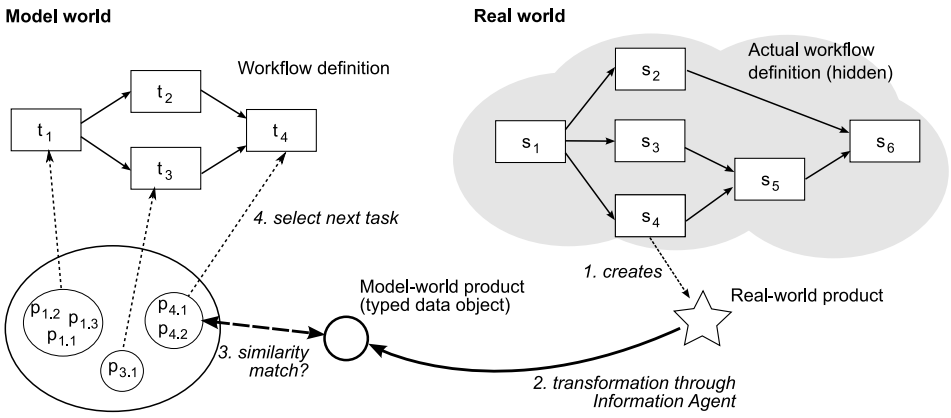

Fig. 2. Deriving the next task by observing context data.

access individual features. However, in relation with another information item knowing that the drawing exists may be sufficient on its own. For instance, if map update files have been loaded before, the drawing is most likely to represent an updated map edition.

In order to leverage this extra information, relationships between the information items are inspected. These structures within the source context are derived by applying *context perspectives*, that put a particular concept in focus. Concepts include, but are not limited to, users, products, tools, events and time as described in the following:

User perspective. The user perspective describes a user centric point of view on the information items contained within the source context. The following relationships between users and the information items are exploited:

- Creator: Who is the original creator of the data object?
- Permissions: Who has access granted for reading or altering particular data?
- Last access: Who did access the product lastly (read/write)?
- Competency: Who has the competencies to create a product?

For each user, the user perspective expresses actual and possible contribution to the source context.

Product perspective. The product perspective is a product-centric view on the observed context, comprising the following properties:

- Type: Which tool was used to create a product?
- Version: Is the product considered a final edition intended for customer use?
- Existence: Does the product exist?

This perspective allows to link an information item with its sources, and provides insight on its intents.

Event perspective. The event perspective puts events in center, and how they have affected an information item within the context:

– Low-level events: Operating system events?
– Utilization: Has the object been utilized or send to someone?
– High-level descriptions: E-mail denoting change request

The event perspective reveals the activities around an information item. This is similar to aspects covered by the user perspective, but not quite the same: For instance, the user perspective will tell whether a particular user has touched an object, but not whether it has been sent to a coworker using an e-mail application.

Time perspective. Time relationships within the context set:

– Creation time: When was the product created?
– Access time: When was the product accessed lastly (read/write)?
– Deprecation: When was the product deleted?

More formally, a perspective is a function $persp = \mathcal{P}(\mathbb{D}) \mapsto \mathcal{P}(\mathbb{D})$, and with $C_{observed}$ the observed context, $persp = (C_{observed})$ is expected to contain data objects listing the slots presented above. While the information agents are free to define what information is translated from the information source, the perspective offers a fixed set of properties shared among all information agents.

This allow to exploit the role and the importance of a certain common organization-specific concept (such as a user role, or a specific tool) for the observed contexts. This eases further design of similarity measures, leading to improvements of similarity matching.

3.4 Designing the case base

As of this writing, the initial case base is about to be created by conducting user interviews. Because of the a priori task characterization style, users can be asked quite intuitive questions e.g. “what tasks are you currently performing”. For each task listed in response, the observed context delivered by the various information agents is stored.

To complete the task characterization, the input ports available are required, too. The input task ports describe the control flow alternatives between tasks, and may be identified if the predecessor task(s) are known (using $conn$). Thus, if a task has been listed that has more than one input port, users are asked about prior tasks, too, in order to identify the alternative taken to start task enactment.

For retrieval, the input ports can help to select the most appropriate tasks by limiting the result set to tasks reachable from the last known alternatives taken by following $conn$ contained in the workflow definition.

So far, only a single workflow has been taken into consideration. In the application scenario, multiple workflows based on the same workflow definitions are enacted in parallel. Thus, besides task characterization, a workflow characterization is required for mapping subsets of the observed contexts to the different workflows. As workflow enactment means concerting task enactment, merging task characterizations to form a workflow characterization appears suitable. In a first retrieval phase, workflows are then identified by these characterizations, which will also determine the partition of observed context that is applicable to the particular workflow. A more detailed definition of workflow characterizations is work in progress.

4 Example

For preparing the data displayed by the DenkXweb Internet service, the workflow definition "import base data" has been created. The workflow definition consists of the tasks "request geospatial information from the authorities" ($t_{request}$), "check whether map and metadata information received matches" (t_{check}), "create CAD files and directory structure" (t_{CAD}) and finally "use CAD application import filter to load map and metadata" (t_{import}). Each task has only one input and only one output port, as workflows based on this workflow definition are expected to be simple process chains.

The workflow definition "import base data" is instantiated whenever data intended to be displayed by the DenkXweb Internet service is going to be prepared. So, for the city of Frankfurt am Main, a workflow has been started, too, and has been proceeded to t_{check}. When employee Alice continues work on Frankfurt, she opens CAD files to hold the data requested before. This results in new files and structures that are laid out in a characteristic structure. Furthermore, she enters her efforts in a time recording application running within the company's Intranet.

Thus, a "file server" information agent picks up the newly created file and directory structure, and conveys it to the data object c_{file}. In addition, the "time recording application" information agent transforms the database record created by Alice into the data object $c_{timerec}$. By looking at the observed context $C_{observed} = \{c_{file}, c_{timerec}\}$ from the product perspective, it becomes clear that the municipality assigned to both data objects is the city of Frankfurt am Main. Using the latter information, the workflow representing ongoing work for this city is retrieved. By similarity matching using the appropriate case base, the task t_{import} is finally retrieved, and the enactment state is updated to $(In_{t_{import}}, \{t_{import}\}, C_{observed})$.

While Alice is still busy continuing workflow enactment, a customer representative calls her. During the phone call, the representative explains that a monument already published has to be revoked as soon as possible, as it had turned out that the original building has been demolished.

Thus, Alice stops her current activities and loads the DenkX database application that maintains the monument protection rationales. She looks up the record in question and deletes it. However, she has not deleted an already published record before, so she logs into the CAKE GUI to find out how to proceed. Using the technique discussed in section 3, the GUI presents that she is currently working on a workflow based on the workflow definition "delete monument". Alice looks up how to proceed, and continues by altering spatial data accordingly and runs a tool to update the web server.

When Manager Bob is asked by another project stakeholder whether the "import base data" workflow for Frankfurt am Main has been already completed, he logs into the CAKE GUI and sees that Alice has started working on it, but is currently busy in completing her ad-hoc workflow. Based on the tasks left to perform in the workflow "delete monument", he estimates when the import is finished, and reports his estimate back to the stakeholder.

5 Related work

In order to model workflows, various concepts have been proposed in the last years. Most of these concepts aim at detecting or avoiding infeasible or suboptimal configurations when modeling workflows. For instance, approaches like MVP-L [12] have been designed specifically for expressing relationships between the various aspects of a development project. Other efforts propose state or UML activity charts as a means of workflow specification and execution [13]. The DYNAMITE [14] system suggests dynamic task nets for expressing baseline plans and for supporting ad-hoc plan decomposition. Recent approaches include XML-based languages like BPEL4WS [15] for concerting tasks carried out by web services [16]. As no single approach has gained broad acceptance yet, configurable process modeling languages have been suggested [11], too. The latter aim at defining a core language to express and maintain general knowledge that can be easily translated back and forth process models used by commercial applications, such as SAP's Event-Driven Process Chains.

Regarding workflow planning and enactment support, several systems have been implemented. The MILOS system [17] is a process-centered environment that aims to support software development activities. MILOS supports ad-hoc changes to plans as well as interleaved planning and enactment, however the system relies on manual user feedback for determining the current enactment state. Workbrain [18] merges Organizational Memory and the workflow concept. For structural planning a CBR application is integrated and can be utilized by a human workflow planner before the respective workflow will be enacted. By providing retrieval on workflow characterizations the CBR application allows users to describe a problem and retrieves similar solutions that can be used to construct new workflows.

Similar to CAKE, CBRFlow [19] combines workflow technology with conversational CBR for coping with changing and unpredictable environments. For realizing the concept of workflow modification during runtime workflow instances can be annotated with cases that consist of a set of question-answer pairs and one action. After the annotation these cases can be directly used as assistance for decision-making processes because of representing additional knowledge, maybe in form of instructions, which can be used during workflow execution. Modifications on the workflow definitions is done by workflow modelers and no search facility on workflow definitions is supported. However, both Workbrain and CBRFlow approaches lack techniques for accessing external information sources.

Capturing workflow enactment information by observing users and their activities is suggested in [20]. Here, the authors capture rather low-level events (opening documents, selecting commands) received from tools deployed within the organization. A similar idea is followed in [21], but the data observed is focused more on the documents viewed and altered than the tools. In [22], a formal and explicit model of knowledge-intensive tasks is introduced, and how tool events can be captured automatically and collated semantically. Similarity matching is used to present information relevant to the current task pro-actively.

6 Conclusion

In this paper, the CAKE workflow model approach has been discussed, and an algorithm has been described how to track current work progress using context information. The latter information allows short-term planning as well as user notifications if deviations from the original plan are detected, allowing quality assurance or discussion of further workflow modifications. Using CBR, the tracking approach will improve as more tasks and their premises are stored in the case base.

While first results are encouraging, modeling workflows and applying the approach presented to the geographical information management application domain is work in progress. As next steps, a student research group at the University of Trier will implement information agents, allowing to determine destination contexts.

Further work includes to enhance progress tracking in order to find deviations from the underlying workflow definition. Then, these deviations could be used to either present warnings to the users or to enhance workflow definitions in terms of process mining by adding alternatives, detecting sub-workflows etc.

References

1. Wargitsch, C., Wewers, T., Theisinger, F.: Workbrain: merging organizational memory and workflow management systems. In: Proceedings of KI'97 Workshop on Knowledge-Based Systems for Knowledge Management in Enterprises, Freiburg, Germany, 21st Annual German Conference on AI'97 (1997)
2. van Elst, L., Aschoff, F.R., Bernardi, A., Maus, H., Schwarz, S.: Weakly-structured workflows for knowledge-intensive tasks: An experimental evaluation. In: Proceedings of the 18th International Workshops on Enabeling Technologies: Infrastructures for collaborative enterprises, Linz, Austria, IEEE Computer Society (2003) 340–345
3. Freßmann, A., Maximini, K., Maximini, R., Sauer, T.: Collaborative agent-based knowledge support for empirical and knowledge-intense processes. In: MATES 2005 / CIA 2005. Volume 3550 of LNAI., Koblenz, Germany, Springer-Verlag (2005)
4. Freßmann, A., Maximini, K., Maximini, R., Sauer, T.: CBR-based execution and planning support for collaborative workflows. In: Workshop "Similarities - Processes - Workflows" on the Sixth International Conference on Case-Based Reasoning (ICCBR 2005), Chicago, Illinois (USA) (2005)
5. Freßmann, A., Maximini, R., Sauer, T.: Towards collaborative agent-based knowledge support for time-critical and business-critical processes. In Althoff, K.D., Dengel, A., Bergmann, R., Nick, M., Roth-Berghofer, T., eds.: Professional Knowledge Management. Volume 3782 of LNAI., Kaiserslautern, Germany, Springer-Verlag (2005) 421–430
6. Freßmann, A., Sauer, T.: Collaboration patterns for adaptive software engineering processes. In: Self-Organization and Autonomic Informatics, IOS Press (2005) 304–312
7. Sauer, T., Maximini, K., Maximini, R., Bergmann, R.: Supporting collaborative business through integration of knowledge distribution and agile process management. In Lehner, F., Nsekabel, H., Kleinschmidt, P., eds.: Multikonferenz Wirtschaftsinformatik 2006 (MKWI 2006), GITO-Verlag Berlin (2006) 349–361
8. Eberhard Fuhr, E.P.: Hessische Verfassungs- und Verwaltungsgesetze (in German). C.H. Beck (2005)
9. Bergmann, R., Breen, S., Göker, M., Manago, M., Wess, S.: Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology. LNAI 1612. Springer (1999)

10. Corkill, D.D.: Collaborating software: Blackboard and multi-agent systems & the future. In: Proceedings of ILC 03 International Lisp Conference, New York, NY (2003)
11. van der Aalst, W., Dreiling, A., Gottschalk, F., Rosemann, M., Jansen-Vulle, M.: Configurable process models as a basis for reference modeling. In: Business Process Management Workshops: BPM 2005. Volume 3812 of LCNS., Springer-Verlag (2006) 512–518
12. Bröckers, A., Lott, C.M., Rombach, H.D., Verlage, M.: MVP-L language report version 2 (1997)
13. Wodtke, D., Weißenfels, J., Weikum, G., Dittrich, A.K., Muth, P.: The mentor workbench for enterprise-wide workflow management. In Peckham, J., ed.: Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, AZ, ACM Press (1997) 576–579
14. Heimann, P., Joeris, G., Krapp, C.A., Westfechtel, B.: DYNAMITE: Dynamic task nets for software process management. In: Proceedings of the 18th International Conference on Software Engineering, IEEE Computer Society (1996) 331–341
15. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business process execution language for web services, version 1.1. specification, BEA Systems, IBM Corp., Microsoft Corp., SAP AG, Siebel Systems (2003)
16. Buhler, P.A., Vidal, J.M.: Towards adaptive workflow enactment using multiagent systems. Volume 6., Springer Science + Business Media (2005) 61–87
17. Maurer, F., Dellen, B., Bendeck, F., Goldmann, S., Holz, H., Ktting, B., Schaaf, M.: Merging project planning and web-enabled dynamic workflow for software development. IEEE Internet Computing **4**(3) (2000) 65–74
18. Wargitsch, C., Wewers, T., Theisinger, F.: An organizational-memory-based approach for an evolutionary workflow management system - concepts and implementation. In: Proceedings of HICSS'98 Thirty-First Annual Hawaii International Conference on System Sciences, Washington, DC, IEEE Computer Society (1998) 174–183
19. Weber, B., Wild, W.: Towards the agile management of business processes. In Althoff, K.D., Dengel, A., Bergmann, R., Roth-Berghofer, T., eds.: WM2005: Professional Knowledge Management Experiences and Visions, Kaiserslautern, Germany, German Research Center for Artificial Intelligence (DFKI GmbH) (2005) 375–382
20. Johnson, P.M., Kou, H., Agustin, J., Chan, C., Moore, C., Miglani, J., Zhen, S., Douane, W.E.: Beyond the personal software process: Metrics collection and analysis for the differently disciplined. In: Proceedings of the 2003 International Conference on Software Engineering, Portland, OR (2003)
21. Fenstermacher, K.: Revealed processes in knowledge management. In Althoff, K.D., Dengel, A., Bergmann, R., Nick, M., Roth-Berghofer, T., eds.: WM2005: Professional Knowledge Management Experiences and Visions, Kaiserslautern, Germany, German Research Center for Artificial Intelligence (DFKI GmbH) (2005) 443–454
22. Schwarz, S.: A context model for personal knowledge management. In: Proceedings of the 2nd International Workshop of Modelling and Retrieval of Context (MRC 2005). (2005)