Proceedings of the 17th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'08)

An Agent-oriented System for Workflow Enactment Tracking

Thomas Sauer
rjm business solutions GmbH
Lampertheim
Germany
t.sauer@rjm.de

Mirjam Minor
Business Information Systems II
University of Trier
Germany
minor@uni-trier.de

Sascha Werno
Dillinger Hütte GTS
Saarbrücken
Germany
saschawerno@web.de

Abstract

The notion of workflows has evolved from a means to describe the flow of paperwork through an organization to a more abstract and general technique used to express best practices and lessons learned in many application domains. When putting workflow definitions into practice, it is important to stay informed which tasks are currently performed, as this allows detecting slipping schedules or unwanted deviations. In this paper, an agent-based approach for automatically tracking the set of active tasks is presented by observing the data produced during enactment. This enables short-term planning and quality control without obliging team members to explicitly document progress.

1. Introduction

In many organizations, workflows are used to describe best practices or lessons learned as algorithmic structures. That is, workflows explain how work results are meant to be achieved and how to avoid problems that have occurred in the past. In order to distribute this experience throughout the organization, workflow management systems (WfMS) have been found useful, e.g. by providing support how to advance from a specific situation to the next steps. To make such support work, anyhow, the WfMS must be aware of the overall progress reached. The latter enables checking whether the workflow can be completed in time, or to assure predefined qualities. Further, by evaluating a series of progress information recorded over time, detailed documentation about project progress can be created, e.g. to fulfill according customer requests.

However, users are often reluctant to manually enter information about work progress into a WfMS. Especially when the human actors involved are experts and the tasks in question have been carried out many times before, the benefits of a WfMS soon appear marginal to people playing technical roles. When still forced to provide feedback on work results, users may start to avoid the system, entering the least amount of information possible, making up data so the workflow appears to be enacted as planned instead documenting deviations, etc. This may prevent workflow evaluation, worsening the acceptance problem when workflows no longer reflect reality.

In this paper, the agent-oriented Progress Information System (PIE) is presented which has been created to overcome this problem by means of an automatic workflow enactment tracking. The system has been developed within the domain of geographical information management [10], which is an integral part of a long-term eGovernment project aiming at creating, maintaining and publishing detailed geographical information for structures of historic interest as demanded by law. Within the project, workflows describe both technical and organizational aspects of everyday work. That is, they express proven techniques how to create artifacts, or represent agreements with the project stakeholders on the order of results. The team members involved are experts utilizing a rich variety of specialized information systems on a daily basis, including a CAD suite, databases, etc. To find out current work progress, the PIE system evaluates digital artifacts produced by the tools already deployed, so users are not obliged to describe their everyday work with an additional system. By following a design of cooperating software agents aligned in well-defined layers, the PIE system is kept modular, flexible and reliable.

Usage of the PIE system is not restricted to geographical information management. Rather, it is applicable to all domains where tasks with observable products occur as routine work in similar processes, even if the tools and information systems involved are not process-aware. The underlying model of workflow definition and enactment supports frequent changes, which is particularly crucial for organizational workflows as these have to be adapted oftenly to respect shifted priorities.

In the next section, the workflow and enactment models serving as the system foundations are described. An example how the models are put into practice is given in sec-

tion 3. In Section 4, the system design is presented in detail, and the concept of software agents is discussed. Section 5 shows recent evaluation results, and Section 6 lists related and adjacent work. A short conclusion closes this paper.

2. Workflow Definition and Enactment

The system presented in this paper uses the Collaborative Agent-based Knowledge Engine (CAKE) [3] to provide modeling and enactment support for agile workflows. That is, the workflow technology offered allows late-modeling and structural adaptation of ongoing workflows. Typical ad-hoc changes are, for instance to re-order some parts of a workflow instance or to insert an additional task. CAKE also offers data modeling facilities to express work results achieved from enacting workflows using a common object-oriented data model. Based on the latter, structural Case Based Reasoning (CBR) [2] is offered.

CAKE uses a light-weight process modeling language for the agile workflows at the user level by extending the UML activity diagram notation by own symbols. An in-depth description of the particular elements of the modeling language can be found in [7]. At the system level, the tasks of a workflow are organized in a block-oriented manner. That means that a workflow is a hierarchy of building blocks like XOR-, AND- or LOOP-blocks. *Trigger tasks*, i.e. sub-workflow placeholder tasks integrate the hierarchy of building blocks with a hierarchy of workflows. Agility of the workflows is restricted in that building blocks cannot be interleaved but they can be nested. Further, before a structural adaptation can take place the concerned area of the workflow instance must be suspended from execution by means of a breakpoint that is set by the user.

When a workflow is about to be enacted, a *workflow instance* is derived from a *workflow definition* (process template). The instantiation of a workflow can be regarded as a kind of parametrization of a process that is performed several times in a very similar way but for different subjects. In turn, the enactment of a task follows the state chart of transitions depicted in Fig. 1. That is, task enactment starts as soon as all data required for processing has become available. While the task is "active", the data available is consumed and new data is produced. The latter forms an a-posteriori *task characterization* [8], as illustrated in Fig. 2.

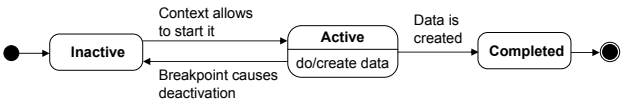

Figure 1. Enactment state transitions of a workflow task.

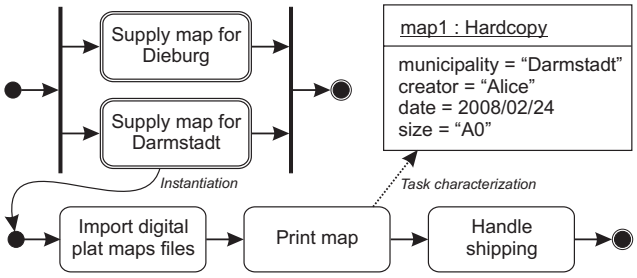

Figure 2. Workflow for providing plat maps.

In the figure, enactment of task "print map" is characterized by the "map1" data object. In turn, the products created or refined by the task may satisfy the pre-conditions of one or more successor tasks, allowing to continue workflow enactment. For a workflow instance, the overall *workflow enactment state* is specified by the aggregation of the task enactment states and task characterizations for all tasks contained in the underlying workflow definition [9].

3. Example

An integral part of geographical information management is to support local experts by providing hardcopies of plat maps printed to scale. Using the maps, experts locate structures of interest to conduct on-site research. Fig. 2 illustrates the workflow "supply map" followed to produce the maps as required. First, the latest land cadastre information is requested from the real estate authorities and transferred into a CAD suite using the "Import digital plat map files" task. After import is completed, the hardcopy is produced, i.e. a detailed map is plotted using a large-format printer. The hardcopy is required to perform the last task "Handle shipping", which comprises trimming, folding and mailing of the physical map to the expert. Thus, the hardcopy characterizes enactment of the "Print map" task and is prerequisite to begin enactment of the "Handle shipping" task. The task characterization is expressed in the means of the common data model, i.e. the physical hardcopy is described by a data object based on an appropriate data class.

The workflow "supply map" may be instantiated in many situations which are described by organizational workflows. The figure shows an organizational workflow to express that two maps may be created in parallel for the cities of "Darmstadt" and "Dieburg".

4. Approach

To learn workflow traces from the information that is present in the organization, PIE has been designed as a multi-agent system (MAS). That is, a set of *software agents*

is employed to cover the different facets of interpreting, understanding and evaluating any data available. Specialized agent types are used for different granularities of perception and decision. As depicted in Fig. 3, the agents are arranged in distinct layers on top of a bottom layer integrating the information systems already deployed at the organization. The information systems are constantly used to create, alter or delete data during everyday work. On the next layer, a network of sensor agents [1] picks up these activities, transforming them into more general data objects for further evaluation. The latter is conducted by further software agents residing on the task layer, allowing to assess whether a certain task enactment state has been reached for any of the tasks contained in a workflow definition. Above, instance agents combine the task states with enactment history previously found for specific instances. The topmost layer hosts workflow agents ensuring consistency of this combination with the workflow definitions. This results in the progress information as desired. Thus, agent types of each layer require to "sense" incomplete, uncertain information from the environment including messages received from the lower layer, to "think" how to interpret this information, and to "act" by sending messages to the next upper layer.

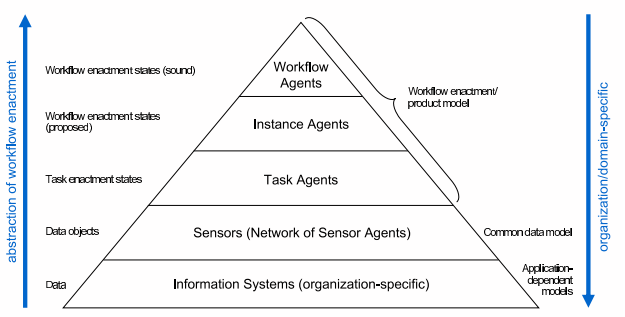

Figure 3. The PIE system layers.

The system uses a directory facilitator (DF) component to keep track of all agents registered with the system. By using the DF, the agents coordinate communication as required during run-time. For maximum flexibility, each agent supports both reactive and pro-active modes of operation. That is, besides responding to specific requests performed by either an agent residing on a higher level or a human actor, each agent is capable in capturing events which occur in a lower level.

4.1. Sensor agents

The sensor agents connect to the information systems deployed throughout the organization with the higher levels of the PIE system. The information systems of interest cover databases, web services, document repositories etc. as accessed and modified during everyday work e.g. by an Intranet application. Each sensor agent is responsible to notice whether specific data has been created, modified or deleted, and to notify the appropriate task agents of these basic events. Furthermore, a sensor agent transforms the system-specific data involved into data objects expressed in terms of the common data model. That is, a sensor agent instantiates appropriate data classes from the common data model, populating it with values available in system-specific form such as database fields.

For instance, when a team member enacts the "print map" task, producing the hard copy is logged by the print server. Among other details the log files expose the actor's name, date printed, and the municipality encoded in the print job name. A print server sensor leverages this information to state that new data has been created, and instantiates a data object of type "Hardcopy" to characterize the task output as shown in Fig. 2.

Sensor agents use an internal state as memory to make sure that the events detected in their environment represent changes in the real world. In particular, persistent data is only reported once, and data is only reported as deleted when the user actively removes it willingly from the information system. In the example above, the print server may hold information about print jobs for up to four weeks. As the physical map does not change after the print job has been completed, the sensor uses the print job log entry only once during this period of time to report a new data object. After four weeks, the log entry is discarded, but the physical hardcopy is still available, so the higher levels do not need to be informed.

To make the PIE system work, the sensor network should cover the information infrastructure completely. Each information source can be covered by as many sensors as required to keep design and programming efforts as low as possible. Each sensor agent may focus on a single subject, but their areas of interest can overlap as well. The more sensors available the better the overall system is able to compensate deficits of single sensor implementations like missing database error handling.

4.2. Task agents

For each task referred by the workflow definitions, a single task agent is used to evaluate sensor data. A task agent tries to find evidence that a particular task has reached a specific enactment state. Such evidence may be given by individual data objects recently found by the sensors, or by a combination thereof. For example, log files may indicate that the "print map" task has reached the enactment state "active", but to become "completed", an additional entry in the accounting system has to be found as well.

Task agents specify the sensors which are capable of delivering results of interest, and use CBR to retrieve appropriate enactment state descriptions for the set of data available. The case base contains pairs of state labels and corresponding task characterizations found during prior enactment. The query is the set of data objects received from the sensor agents which can be seen as a proposed task characterization with unknown state assigned. The best-matching solutions found are then provided to the instance agents residing on the next system layer, using the similarity between problem and solution to indicate confidence in the enactment state transmitted. If there are only poor solutions with low similarity, task agents wait for additional sensory input.

4.3. Instance agents

When enactment of a workflow definition starts, a new instance expressing a distinct thread of execution is created. A workflow definition may be instantiated multiple times in parallel, e.g. when duties have been arranged for different people. To identify that enactment of a specific instance has been continued, instance agents determine a distance between the results reported by task agents and the workflow enactment state already reached. The closest instance is then selected for further inspection. Similar to the concept of sensor agents, the distance is determined in a distributed manner using a network of δ-agents. Each δ-agent uses a unique strategy to calculate a distance between an instance and task enactment as reported by task agents, including (1) distance between enactment states, (2) distance between task characterization and workflow instance parameters and (3) distance between characterizations. Based on the findings of the δ-agents, the instance agent calculates an overall distance as a weighted sum.

Distances between enactment states: The first δ-agent compares the task state descriptions found by the task agents with the task state descriptions contained in the workflow enactment state reached by the workflow instance. As shown in Fig. 1, task enactment is specified through a state chart with the individual states. To determine whether task enactment belongs to a specific workflow instance, the δ-agent looks for the lastly reached tasks contained in the workflow enactment state, i.e. tasks described differently from "inactive" but followed by "inactive" tasks only. The distance sent to the instance agent is the minimal number of transitions required to attain the states found by the task agents from the states of the lastly reached tasks.

For example, when the workflow enactment state for workflow instance "Darmstadt" shown in Fig. 2 lists task "print map" as "active" and the corresponding task agent reports that the task is now "completed", a single step is found as the task state distance to the instance agent. For a second workflow instance which has lastly reached an "active" state of "import files", the distance would be three (one transition to complete "import files", and two transitions to make "print map" first "active", then "completed").

Ideally, for each task its state chart is traversed completely, however task agents may not be capable to report all states in the correct order. Thus, the confidence reported by the task agents is used to rank the results. When it is less likely that the task has reached a certain state in the first place, this raises the distance reported to the instance agent, even if the state descriptions are very close to each other.

Distances between task characterization and instance parameters: The second δ-agent compares the data objects delivered by task agents with structural information contained in workflow definitions that belong to a specific instance. A workflow instance is created by enacting a trigger task contained in a workflow definition. For example, the workflow "supply map" depicted in Fig. 2 lays out the best practices how to prepare a plat map for historians. To actually apply the best practice, an actor has to know for which region file imports are required to start enactment. In the domain of geographical information management, this parametrization is usually provided by the attributes given to the trigger task in its owner workflow definition, e.g. by using the desired municipality's name to label the trigger tasks in an organizational workflow definition. The δ-agent uses the common data model to express the parameters.

To find out whether the products reported by task agents belong to a specific workflow instance, they are compared with the trigger task parametrization. Besides the parameters of the trigger task that has started the instance of interest, the parameters of all trigger tasks belonging to the instantiation hierarchy are considered. For actual comparison, a similarity measure operating on the data model is used for trying to validate the hypothesis that the trigger task parameters influence the outcome of the tasks carried out in the corresponding instance. The higher the similarity, the lower is the distance sent back to the instance agent.

Distances between task characterizations: The third δ-agent determines whether a task characterization reported by task agents is related to a task characterization found during previous workflow enactment. For example, when the workflow "supply map" shown in Fig. 2 is enacted for the city of Darmstadt, the task "import digital plat map files" results in a data object of type "FileSet" referring Darmstadt as the municipality affected. When the task agent detects that the subsequent task "print map" is enacted with the detected "Hardcopy" object referring the same municipality, the newly found task enactment can be assigned to the according workflow instance "Darmstadt", allowing to distinguish it from the instance "Dieburg" running in parallel.

Again, similarity measures are used to compare context and task result. When comparing products with each other, a changing set of attributes may be of interest depending

on the capabilities of the sensors involved. For example, enactment may be found related to a specific instance because they refer to the same location. Another time, the delta agent may find them related because the same human actor was involved or enactment has taken place within a specific time interval.

All task characterizations contained in the workflow enactment state of the instance are taken into consideration, however characterizations found recently are preferred. The similarities determined for each characterization are used to weight the steps required between the respective tasks and the task reported by the task agent. When the workflow instance is freshly started by a trigger task, the data objects available for consumption by the trigger task are used as the characterization for the start task. Thus, even if no products have been created during workflow enactment yet, the δ-agent may find the appropriate instances.

4.4. Workflow agents

Task enactment as determined by the task agents is possibly incomplete, depending on the coverage and reliability of sensors and the extent of the case bases used by the task agents. Thus, the workflow enactment state formed by instance agents may have "gaps", i.e. the workflow state cannot be reached by strictly following the state charts. For each workflow definition, a workflow agent is used to make a decision for a sound workflow enactment state from the findings of the instance agent.

To align the enactment states or to infer states where appropriate data has been missing, the workflow definition blocks are inspected recursively. If the block contains an "active" element, the whole block is found "active". Next, if there is no "completed" element but an "inactive" one, the block counts as "inactive". A block without "inactive" but with "completed" elements is seen as "completed". Further, if all "completed" elements precede "inactive" ones, the state description "active" is used for the block. Otherwise, the state description reported most often is assigned.

For AND- or XOR-blocks, the rules are first applied to determine an individual state for each branch. For AND-blocks, the block is assigned "active" if there is an active branch, otherwise with the state found for the majority of branches. If there are as many "completed" branches as "inactive" ones, the confidence of the task states are used to favor one. The latter is also used to decide for a branch of XOR-blocks.

The workflow enactment states determined by the workflow agents is a snapshot of the current situation in the organization, expressed in terms of the CAKE model of workflow enactment. The information is incorporated into the workflow traces that belong to the respective workflow instances, providing the enactment history as required by the instance agents.

The workflow definitions modeled using the CAKE system are used to automatically create and configure task agents, instance agents, and workflow agents as required. For each task used, a task agent is created and added to the DF, and for each workflow definition, according instance and workflow agents are added to the system. Adding or removing agents from the DF is supported during run time, so when workflow definitions are changed, the configuration can be adapted as required.

5. Evaluation

As of this writing, the PIE system has been evaluated on a specific facet of the geographical information management scenario. The workflows enacted and the artifacts created have been explored by student research groups before [10]. The group identified six data sources, and developed a simple approach to guess work progress based on existence of predefined data. Using the same program code to access the data sources for creating five sensor agents, the PIE system already outperforms the original approach on 77 workflow instances covering three administrative districts (cf. Fig. 4). In more than 79% of all cases, the workflow instance states have been found correctly. When adding two more sensors the success rate raised to 87% because deficiencies of the existing sensors could have been compensated.

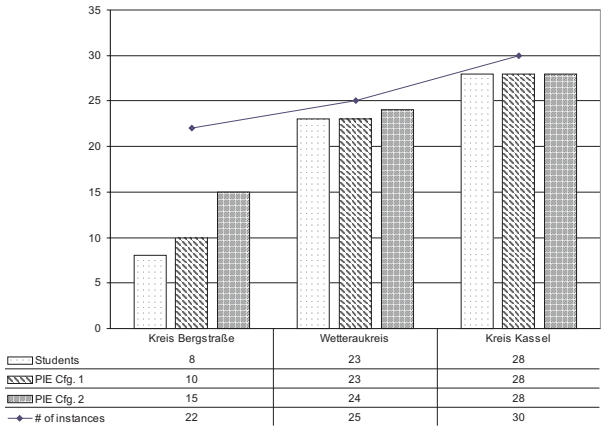

Figure 4. Evaluation results.

6 Related Work

Automated tracking of progress has been discussed in the field of Software Engineering to control the development process [6]. The authors propose a system to collect software metrics through sensor programs attached to

development tools, allowing to conduct analysis and to send alerts to the developer. In the field of business processes management, audit trail analysis has been discussed as workflow monitoring in [13]. The presented system aims at testing workflow history data against predefined qualities.

A system to pro-actively provide support based on the current focus of work is presented in [11]. The system automatically captures activities performed by a user (e.g. open file or view e-mail) and collates them semantically using a formal and explicit model of knowledge-intensive tasks. Using similarity matching, documents or other artifacts relevant in the current situation are presented to the user. A similar approach is discussed in [5] but the observation is focused more on the documents accessed than the tools.

In [4], the authors introduce process query systems to extract meaningful information from large streams of data created by e.g. performance monitors or firewalls. Based on a nondeterministic automaton process model, the suggested system uses a set of hypotheses that can explain whether a certain process state has been reached. Each event received either increases or decreases the number of hypotheses, allowing to detect the execution of a process or parts of it. It is pointed out that these may represent the unwanted steps performed by an attacker, so detecting them automatically can be crucial to perform countermeasures.

To support workflow modeling and evolution, analysis on the actual order of execution determined for tasks performed in an organization has been discussed as workflow mining [12]. The technique presented by the authors aims at discovering workflow definitions from everyday practice.

7. Conclusion

In this paper, the PIE system for agent-oriented workflow enactment tracking has been described. The system allows to leverage the various information systems already deployed in an organization in order to unobtrusively help team members to document their work by tracking enactment of previously defined workflows. This allows to easily assure predefined qualities, helps to detect unwanted deviations from the plan or simplifies workflow evolution. The system follows the common sense, think, act paradigm in retrieving information from the information sources, evaluating them using task and workflow agents, leading to user feedback. A first evaluation led to promising results, as next steps further perspectives of geographical information management are studied using additional sensors. The modular, easily configurable agent-based design of the PIE system helps to keep the required efforts low. It also ensures that the system can be applied to other domains as well.

The workflow enactment information found by the PIE system can be easily integrated with information systems present in the organization, e.g. by automatically adding a list of alternative tasks to a to-do list user interface after a certain part of a workflow instance has been completed.

References

[1] D. Agrawal, R. Biswas, N. Nain, A. Mukherjee, S. Sekhar, and A. Gupta. Sensor systems: State of the art and future challenges. In J. Wu, editor, *Handbook on Theoretical and Algorithmical Aspects of Sensor, Ad Hoc Wireless, and Peer-To-Peer Networks*, pages 317–346, 2006.

[2] R. Bergmann, S. Breen, M. Göker, M. Manago, and S. Wess. *Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology*, volume 1612. Springer, 1999.

[3] R. Bergmann, A. Freßmann, K. Maximini, R. Maximini, and T. Sauer. Case-based support for collaborative business. In *Proceedings of the 8th European Conference on Case-based Reasoning (ECCBR'06)*, volume 4106 of *LNCS*, pages 519–533, 2006.

[4] G. Cybenko and V. H. Berk. Process query systems. *Computer*, 40(1):62–70, 2007.

[5] K. Fenstermacher. Revealed processes in knowledge management. In *Professional Knowledge Management, WM 2005 Revised Selected Papers*, volume 3782 of *LNAI*, pages 443–454, 2005.

[6] P. M. Johnson, H. Kou, J. Agustin, C. Chan, C. Moore, J. Miglani, S. Zhen, and W. E. Douane. Beyond the personal software process: Metrics collection and analysis for the differently disciplined. In *Proceedings of the 2003 International Conference on Software Engineering (ICSE 03)*, pages 641–646, 2003.

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

[8] T. Sauer and K. Maximini. Using workflow context for automated enactment tracking. In *Proceedings of the 1st Workshop on Case-based Reasoning and Context Awareness (CACOA 2006)*, pages 300–314, 2006.

[9] T. Sauer and K. Maximini. Supporting workflow management by automated enactment tracking. In *Proceedings of the 4th German Workshop on Experience Management (GWEM2007)*, pages 109–118, 2007.

[10] T. Sauer, K. Maximini, R. Maximini, and R. Bergmann. Supporting collaborative business through integration of knowledge distribution and agile process management. In *Multikonferenz Wirtschaftsinformatik 2006 (MKWI 2006)*, pages 349–361, 2006.

[11] S. Schwarz. A context model for personal knowledge management. In *Modeling and Retrieval of Context, MRC 2005 Revised Selected Papers*, volume 3946 of *LNCS*, pages 18–33, 2006.

[12] W. M. P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[13] M. zur Muehlen and M. Rosemann. Workflow-based process monitoring and controlling - technical and organizational issues. In *Proceedings of the 33rd Hawaii Int'l Conf. on System Sciences (HICSS-33)*, 2000.