An Agile Approach to Workflow Management

Barbara Weber and Werner Wild

Institute of Value Process Management / Information Systems
Institute of Computer Science / Quality Engineering Research Group
University of Innsbruck, Technikerstraße 25/7, 6020 Innsbruck, Austria
Barbara.Weber@uibk.ac.at

Evolution Consulting
Jahnstraße 26, 6020 Innsbruck, Austria
Werner.Wild@evolution.at

Abstract. In today's dynamic and uncertain business environment workflow management systems must react quickly to change. This paper suggests extending workflow management with case-based reasoning (CBR) to allow just-in-time updates to the predefined workflow model and to provide flexibility by keeping (real) options open rather than freezing them early on. First an initial workflow model is created, covering only the economically justifiable details of a business process. The decision how to model a business process precisely is delayed until the company's needs are more clearly understood and business value can be achieved. When changes become necessary during run-time, the predefined workflow model is extended with additional knowledge in the form of cases. This feedback supports continuous process improvement, resulting in more manageable and more efficient business processes over time. When the knowledge encoded in cases becomes frequently reused, it is refactored into rules and therefore explicitly included in the workflow model.

1 Introduction

Workflow management systems (WFMS) are frequently used to control the execution of business processes and to improve their efficiency and productivity. To date, workflow management systems have been applied to fairly static environments in which the execution of activities follows a highly predictable path. However, today's business is characterized by ever-changing requirements and unpredictable environments (e.g. due to global competition). Existing workflow management systems do not address the needs of the majority of processes [Sh96] and it is widely recognized that more flexibility is needed to overcome these drawbacks (e.g. [SMO00], [Ca98], [Al97], [RD98]).

Due to this limited flexibility companies have often been restricted to quickly respond to changing circumstances and they could not always realize the expected cost savings. When workflow management systems do not allow for changes or handle them in a too

rigid manner, users are forced to circumvent the system to do their work properly. Bypassing the system results in a lack of efficiency and missing traceability. Additionally, the knowledge needed to complete the work is lost as it is not recorded in the system and therefore cannot be reused efficiently when similar problems arise in the future.

The majority of the work done in this area focuses on the modeling period [SK97], i.e. to allow building more flexible workflow models. By providing more powerful and expressive constructs in workflow modeling languages, the need for change can be reduced since new requirements might already be covered by the workflow model. Reference models [Sc98] and workflow patterns [Aa03] have been developed to facilitate workflow modeling and to allow the tailoring of workflows for specific environments or applications.

However, a workflow management system must also be flexible at run-time so that necessary modifications are possible when they arise. In dynamic and uncertain business environments not all eventualities and possible deviations can be considered in advance, as requirements change or evolve over time, and exceptions or ad-hoc events may arise. In order to provide the necessary adaptability, adaptive workflow research suggests exception handling mechanisms (e.g. [Lu00], [Wa98], [KD00] or [Ca98]) and (ad hoc) run-time modifications to the workflow model (e.g. [RD98], [Ca98] or [Aa01]).

Due to the significant modeling time needed, workflow models are often obsolete right after their specification is "completed". Apart from the limited knowledge available at build-time, modeling every single detail in advance can be time-consuming. The cost of building a complete workflow model often exceeds the potential business value. Covering too many details in the up-front process definition involves the risk of modeling rarely needed parts, not yet needed or even unneeded ones.

The value of flexibility under uncertainty is covered by real option theory, the higher uncertainty the higher is the value to wait. Real option approaches are fundamentally risk-reduction strategies, which actually reduce risk by keeping options open rather than freezing decisions early on [PP03]. Options limit downside risk by limiting the cost and time allocated to resolve uncertainty and maximize the upside reward by delaying decisions until more knowledge is available.

In this paper we suggest an agile approach to workflow management by extending workflow execution with case-based reasoning. This supports just-in-time updates to the predefined workflow model fostering the flexibility to keep options open. After providing the backgrounds in section 2, we describe our approach of just-in-time workflow management in section 3. Section 4 outlines a suggested architecture, related work is discussed in section 5, conclusions in section 6 and further studies in section 7.

2 Background

2.1 Workflow Management

Workflow management involves the modeling, the execution and the monitoring of workflows [Ga02]. During *workflow modeling* an abstract representation of a business process is created which specifies which tasks are executed and in what order. A workflow model thus includes functions (i.e. activities), their dependencies, organizational entities that execute these functions and information objects which provide the functions with data input/output. The *execution* and control of the automated parts of a business process are supported by a workflow management system. *Workflow monitoring* provides status information about running workflows; it supports the evaluation of business processes and fosters continuous process improvement.

2.2 Real Options

The term "real options" was coined in 1977 by Stewart C. Myers of Massachusetts Institute of Technology [My77] and was first applied in the oil, gas, copper, and gold industries as a strategy formulation tool. The theory of real options applies financial options to the valuation of investment decisions for non-financial assets (e.g. [DP94], [AK99]). There are several examples of applying the real option theory to software engineering economics (e.g. [Su96], [EF02]).

A financial call option gives the holder the right, without imposing the obligation, to buy a specific amount of a financial asset at a specified price within a certain period of time. For the holder, financial and real options provide an asymmetrical payoff, as the potential loss is limited to the price paid to acquire the option while the upside gain is unlimited; in real options a small investment today gives the investor the future right, but not the obligation, to make a larger investment. Examples for real-options are growth options, flexibility options, learning options, exit options or waiting-to-invest options (see Table 1) [AK99].

Option Type	Description
Growth Options	The Growth Option values the ability to make an investment today and to make follow-up investments later, if the original investment is a success (e.g. undertaking a pilot project to create the option to be a player in an emerging market).
Flexibility Options	The Flexibility Option allows the quick adaptation to changing business or technical conditions and market opportunities (e.g., making an up-front investment in software architecture to better accommodate future changes).

Learning Options	The Learning Option arises from making a small number of investments to test the value of a much larger investment. Learning options can be created through incremental staging of new products or technologies (e.g. developing a prototype before the full application).
Exit Options	The Exit Option gives the holder the right but not the obligation to abandon a project if it is unsuccessful (e.g. stopping a re-engineering project when the project costs exceed expected benefits).
Waiting-To-Invest Options	The Waiting-To-Invest Option gives the holder the right but not the obligation to delay a decision until more information is available and uncertainty can be removed (e.g. waiting until a new technology is established)

Table 1: Types of real options

2.3 Case-based Reasoning and Learning in Workflow Management

Case-based reasoning (CBR) is a contemporary approach to problem solving and learning where new problems are solved by using past experiences and by adapting their solutions to new problem situations [RS89], [Ko93]. Every time a new problem is solved, the information about the problem and its solution is retained and immediately made available for solving future problems.

Case-based reasoning has been applied to workflow management for the configuration of complex core processes [Wa98], to support workflow modeling [KSL02] and exception handling [Lu00].

Like CBR, Organizational Memory Information Systems (OMIS) allow to apply knowledge gained in the past to present situations. According to [SZ95] an OMIS is a system that provides the means by which knowledge from the past is brought to bear on present activities, thus resulting in increased levels of effectiveness for the organization. OMIS are suitable for knowledge-intensive workflows providing additional process information to the users and support them during the execution of activities. However, most OMIS do not include knowledge gained directly during process execution as they define the information needed to successfully complete the work in advance [Go02, p. 66] and therefore do not provide any true learning capabilities. Only very few systems provide approaches to directly reuse the knowledge gained (e.g. WorkBrain [Wa98], WoMIS [Go02]).

3 Just-in-time Workflow Management

In this paper, we build upon the idea of integrating case-based reasoning and workflow management to provide the system with learning capabilities. During workflow

modeling an initial computerized representation of selected business processes is created. To model the control flow between activities business rules are used. At run-time an instance of the workflow model is created and the process is executed as specified in the workflow model. Case-based reasoning is used to support updates to the predefined model during run-time and to keep this information for further reuse.

3.1 Complementary Strengths of Rules and Cases

Extending workflow management with case-based reasoning is motivated by the complementary strengths of rules and cases (see Table 2).

RULES	CASES
Represent general knowledge	Represent specific knowledge
Work best in well understood, narrow domains that are stable over time	Work best in poorly understood, wide domains that are dynamic over time
A lot of knowledge is included from the beginning	Only a limited amount of cases available at the beginning
System is limited to predefined rules	System adapts itself to a new situation

Table 2: Complementary strengths of rules and cases

Rules represent the general knowledge of a domain, while cases are able to utilize specific knowledge of previously experienced concrete problem situations. In general, rules capture broad trends in a domain, while cases are good at covering an underlying rule in more detail and at adding small pockets of knowledge to the rules. The declarative knowledge encoded in rules is adapted to changing environments by cases, without requiring the rules to be rewritten at every new turn of events [Ma02].

Rule-based systems work best in well understood, narrow domains which are stable over time. When principles of a domain are not well understood, rules will be imperfect, but solutions suggested by cases will be more accurate [Wa97]. The knowledge elicitation in rule-bases systems is often difficult and time-consuming ("knowledge elicitation bottleneck") [WM94]. It is extremely difficult to obtain an appropriate set of rules covering all possible eventualities, especially when no underlying domain model exists. It is easier to articulate, examine and evaluate cases than rules [Sl91], because there is not any explicit domain model required and the problem does not have to be fully understood.

Rule-based systems allow the inclusion of valuable knowledge right from the beginning, but they are limited to their predefined rules. If the problem presented to the rule-based system does not match any of its rules, the system cannot respond in a meaningful way [Sl91]. In a pure case-based reasoning system, only a small number of cases is available in the beginning. However, a case-base with a set of representative and well-distributed

cases is crucial for good retrieval accuracy as CBR relies only on the contextual knowledge in the case-base. Whenever a new problem is solved, this information is stored as a case or cases for future reuse, evolving the case-base over time, thus CBR supports incremental and sustained learning. By acquiring new cases, a CBR system adapts itself to new or changing environments.

3.2 Agile Workflow Management

Figure 1 describes the suggested agile approach to workflow management. The integration of workflow management and case-based reasoning relaxes the strict separation between build-time and run-time and allows for easy run-time adaptations to the workflow model. The knowledge about these modifications is immediately reusable, thus adapting the workflow model to changing circumstances during run-time and supporting continuous business process improvement [We03].

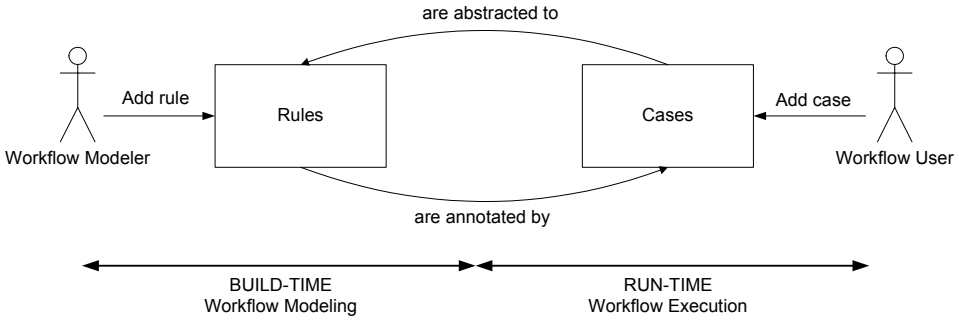

Figure 1: Agile workflow management

At build-time an initial model of an organization's business processes is created. However, it is not always possible to predefine a workflow exactly and in every detail as the business environment is continuously changing, as not all information might be available or as getting this information is prohibitively costly or time-consuming. Therefore workflow modeling should focus on those aspects of a business process that have a clear business benefit or are critical parts of a business process, e.g. legal requirements must be covered in the process model right from the beginning. Process modeling starts with a single business process, which is expected to provide the most business value and continues from there (Waiting-To-Invest Option). This ensures optimal resource allocation, provides the highest initial return-on-investment and fosters early-on feedback for later iterations.

Workflows are evaluated during execution by the workflow user. When run-time changes to the workflow model become necessary due to exceptions or changing requirements, the user annotates the workflow model with context-specific information in the form of cases. This recently gained knowledge is therefore immediately available for reuse without explicitly changing the workflow model (Learning Option).

When the process knowledge encoded in the cases becomes well-established, the workflow modeler should abstract these cases to rules, thus updating the underlying model. Case-based reasoning is used to foster the extraction of rules from the running workflow instances and therefore supports gaining additional domain knowledge. The system and the organization continuously learn how to better handle new situations as more and more experience is gained and the knowledge is readily available for reuse (Learning Option).

The following example illustrates the combination of rule-based reasoning (RBR) and CBR: Strictly enforcing the rule "No vehicles allowed in the park" prohibits the use of any car within the park even in emergencies, e.g. an accident, a fire, a flood, etc. Updating this rule with cases provides the needed flexibility to respond to unforeseen situations. In case of a fire, emergency vehicles must be exempted from the rule. When emergencies occur more frequently, the cases should be abstracted to a rule by the city council ("No vehicles in the park allowed except in emergencies") and thus no longer requiring to make a decision for each emergency case individually.

In our approach rule-based reasoning has precedence over case-based reasoning. CBR is used only when there is no domain knowledge specified in the form of rules, when changes must be performed or when exceptions arise, and to continuously improve the execution of an organization's business processes.

Extending a workflow management system with CBR allows delaying the detailed modeling of a business process until enough knowledge is available and uncertainty can be resolved. Only those aspects of a business process that create clear business benefits and for which enough knowledge is available should be covered in or added to the initial workflow model. The flexibility option enabled by CBR fosters adaptations to the workflow model to respond to changing requirements. Integrating workflow management and case-based reasoning supports short iterations, resolves uncertainty, and permits to rapidly incorporate the results of the learning process into the workflow model.

This approach shares the values of the Agile Manifesto [Ag01]:

- **Individuals and interactions over processes and tools**

  The workflow users are entitled to immediately respond to changes by using CBR and do not have to follow a complicated and time-consuming process.

- **Working system over comprehensive documentation**

  When using this approach it is more important to bring a system up and running as soon as possible and start learning from there, than to create comprehensive documentation first. Feed-back from the workflow users is integrated immediately into the system and thus be made available instantaneously.

- **Customer collaboration over contract negotiation**

  The development of the initial workflow model requires close and informal collaboration between the workflow modelers and the workflow users as they hold the knowledge of how the business processes are actually performed in the company. In the following iterations, permanent changes to the model are also done in close collaboration between the workflow modelers and those users who have entered the cases when deviating from the predefined workflow model.

- **Responding to change over following a plan**

  When a change arises the workflow user can respond to it immediately by entering a case, thus implicitly updating the workflow model. Strictly following a predefined workflow model would be similar to strictly follow a predefined, sometimes outdated plan. Our approach does not enforce the execution of a predefined workflow model as it provides the user with the capability to respond to change.

4. Architecture

4.1 Modeling a Workflow

In the proposed architecture, a workflow is described as a set of nodes and the connections in between them. It consists of a start node, a finite set of routing nodes, a finite set of activity nodes and an end node. At run-time an instance of the workflow definition is created and executed by the Agent. **Agent Instructions** are used to connect the nodes and to control the Agent (i.e. representation of a workflow instance).

The **start node** indicates the beginning of a workflow. The start node is the only node without a predecessor and it is always followed by a routing node (i.e. it contains an Agent Instruction which references the first routing node).

Routing nodes include the routing information and are responsible for routing the Agent from one activity to the next. The decision of where to route an Agent next is made by the routing node using its routing information and the data provided by the Agent (see also section 4.2). A routing node is always followed by one or more activity nodes or by the end node. Each routing node has a Default Agent Instruction and a Rule-Base, as illustrated in Figure 2. The **Rule-Base** contains all business rules that are relevant for this routing node. A business rule is specified in the form "IF Condition THEN Agent Instruction". **Case-Bases** are used at two locations in the process definition: a Default Case-Base is part of a routing node and is used when no rule fires. Additionally, a Case-Base can be attached to any business rule to implicitly update this rule. The **Default Agent Instruction** specifies what to do when no rule fires and no cases are available.

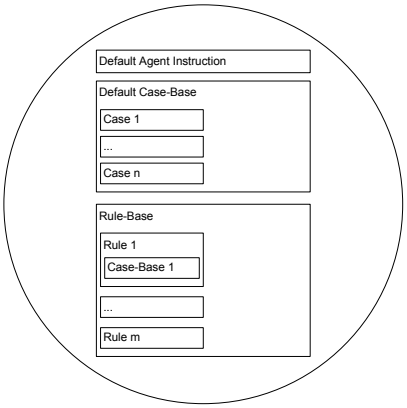

Figure 2: The structure of a routing node

Activity nodes are used to perform an activity, e.g. by using an external application tool. Each activity node contains a reference to an activity tool, a reference to the organizational roles authorized to perform this activity and an Agent Instruction that specifies the next routing node.

Each workflow contains a single **end node**. The end node is the only node without a successor, the predecessor of an end node is always a routing node. This provides a placeholder if cases have to be added.

4.2 Connection between Nodes

The connection between nodes is implemented by Agent Instructions. To move the Agent forward from one node to another, each node except the end node contains information about its successor(s). At each node the Agent is provided with the information needed to decide upon where to go next. In general, this is done in the form of a Default Agent Instruction which contains a reference to the next node and can be interpreted as a simplified ECA-rule (Event-Condition-Action-rule) with an empty condition part. As the Agent is about to leave a node, the Agent Instruction is fetched and the action specified by the Agent Instruction is performed.

The Rule-Base within the routing node facilitates decision-based routing of the Agent and enables selecting one out of several possible succeeding nodes. The Rule-Base contains the relevant business rules for this routing node. During workflow execution, the Agent Instruction of the fired rule is executed.

In contrast to rules, cases are not applied automatically but require user interaction. Cases can be applied by the workflow user when exceptions arise or annotations to the predefined workflow model become necessary.

Figure 3 describes the routing algorithm used to select the appropriate Agent Instruction in a routing node.

Figure 3: The process of routing an Agent

4.3 Example

The process of handling a car insurance claim is composed of the following six tasks (see Figure 4): enter customer request, estimate cost, check cost, perform independent car inspection, authorize and complete repair, and finalize claim. When a customer reports an insurance claim, the request is entered by the call center. After entering the claim report, the garage estimates the cost of repair. The estimate is then checked by the insurance company. If the estimated cost is above a certain threshold, an independent adjuster has to perform an inspection. After a successful car inspection, or if the estimated cost is below a certain threshold, the garage is authorized to fix the car and starts the repair. As the last step in the process, the claim is then finalized by the insurance company.

Each of these tasks is represented by an activity node in the process definition. Routing nodes connect activity nodes to each other. The decision to perform an independent inspection before repairing the car is made at a routing node, based on the business rules in its Rule-Base. The Agent representing the workflow instance and processing this claim follows along the connections between the nodes.

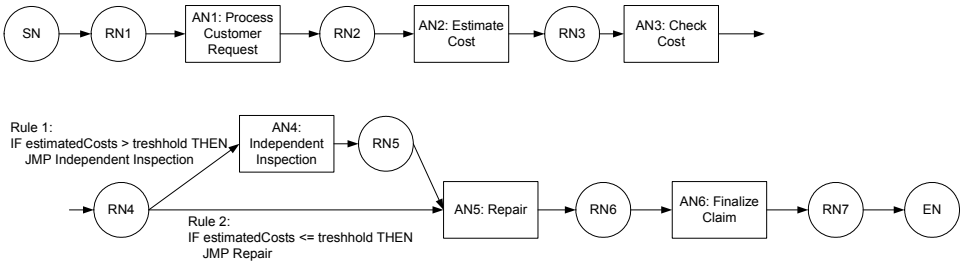

Figure 4: The process of routing an Agent

4.3.1 Adding a new Case

A customer files a new insurance claim. During the processing of the claim problems arise in the "estimate cost" activity and result in a significant delay. The processing continues with the "check cost" activity and after its completion, the Agent enters Routing Node RN4. The Rule-Base of Routing Node RN4 contains two rules, which are analyzed by the Agent (Rule 1 fires if the estimated cost exceeds a specified threshold and an independent car inspection must be performed, otherwise rule 2 fires and the car is repaired without a preceding car inspection).

The user responsible for scheduling and performing the independent car inspection gets a new insurance claim into the To-Do-List on her User Portal. The user selects the item from the To-Do-List and recognizes that the processing of this claim is already significantly delayed and the risk in failing to repair the car on time is unacceptably high. The user knows that on-time repair rates highly in satisfying the customer. As the estimated repair would not be prohibitively expensive, it seems reasonable for the user to skip the independent car inspection. After checking back with her supervisor, the user decides to skip it to reach the company's goals of on-time repairs.

The user creates a new case to define an alternate execution path for this workflow instance and to store this knowledge for future reuse. The user adds a problem description to the new case, explains why an alternative execution path is necessary and includes a set of observations describing the conditions under which the case is applied.

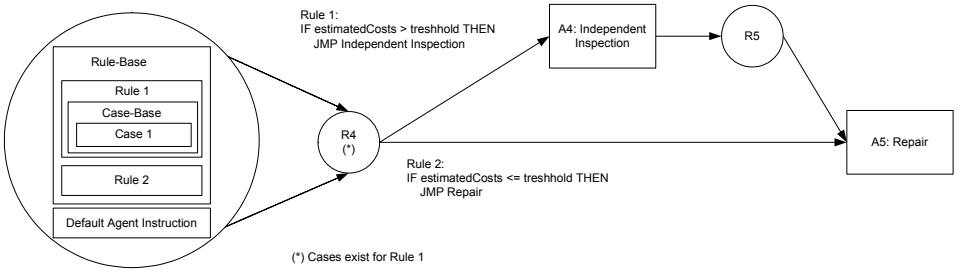

Figure 5: Structure of Routing Node RN4 after adding the case

In the example, the change has been motivated by the very high probability of failing to repair the car on time and the "not prohibitively expensive" repair cost. Finally, an action in the form of an Agent Instruction is specified to perform the change in the workflow execution (i.e. skip the independent car inspection). The analysis of the Agent's history determined that routing node RN4 was the last routing node visited and Rule 1 (estimated cost > threshold) was the firing rule. Therefore the case "Case 1" is added to the Case-Base attached to Rule 1 in routing node RN4 (see Figure 5).

4.3.2 Reusing a Case

Suppose a user holding the role responsible for "scheduling and performing the independent car inspection" gets a new insurance claim in her To-Do-List. The entry is marked, indicating that cases exist for the last routing node visited by the Agent representing this workflow instance. The user can then optionally initiate the CBR sub-system to check the available cases for possible reuse.

The retrieved case has been applied in the past in a situation where the risk to fail to repair the car on time was high and the repair cost was low. The same holds true for this insurance claim and therefore the user decides to reuse the case. The action specified in the case is then applied and the activity 'independent car inspection' is skipped.

4.3.3 Abstracting a Case

When the frequency of case usage exceeds the predefined threshold, the workflow management system sends a notification to the workflow modeler to perform the suggested case abstraction. To get an overview of the cases in the Case-Base, the workflow modeler examines them and reflects about which rules could be abstracted from the retrieved case(s). Cooperating with the domain experts and the frontline workers the workflow modeler then defines a set of rules, adds them to the Rule-Base and removes the corresponding case(s) from the Case-Base.

5 Related work

This paper is based on the idea of integrating workflow management and case-based reasoning. In related work CBR has been applied to support workflow modeling [KSL01], to the configuration of complex core processes [Wa98] and to the handling of exceptions [Lu00].

METEOR [Lu00] models organizational processes with justified ECA-rules and supports the handling of exceptions by using CBR. When no rule fires or any other exception occurs the WFMS tries to derive a capable exception handler, and, if no handler is found, CBR is used to retrieve similar cases to reuse the information about previous exceptions. Human interaction is only necessary if no acceptable solution can be automatically derived.

Our approach uses CBR not only for the handling of exceptions, but empowers the user to implicitly update the workflow model to changing requirements without demanding an exception generated by the system. CBR can not only be applied when no rule fires, but also to annotate a rule with cases in order to make deviations from the rule. The decision whether to reuse a case is not automated, but is always left to the user.

In WorkBrain [Wa98], CBR is used for the configuration of complex core processes using process components. Workflows are configured at their start (=entry time) by combining process components to reduce the number of possible process variants (i.e. reducing of the combinatorial multiplicity), and to provide the needed flexibility. For configuration purposes predefined template components as well as components from historical cases can be used. Due to the long configuration time needed this approach is most suitable for long-running complex core processes. CBR is only used at entry time, there is no support for workflow adaptation during run-time.

DWMSS (document-based workflow modeling support system) uses a CBR approach to support workflow modeling [KSL01]. Existing know-how about the business processes, stored as cases, is reused for the modeling of new workflows.

6 Conclusion

This paper proposes an agile approach to workflow modeling. The application of case-based reasoning to workflow management relaxes the strict separation between build-time and run-time and supports just-in-time updates to the workflow model. This immediate feedback resolves uncertainty, and permits to rapidly incorporate the results of the learning processes into subsequent workflow executions. The tight iterations foster an option-based approach to workflow management and allow the WFMS to respond quickly to facts rather than forecasts. The decision how to model a business process precisely can be delayed until the company's needs are more clearly understood (just-in-time modeling by using CBR). Requirements are modeled when they arise and when their modeling provides a clear business benefit. Shortening the modeling phase allows a sooner productive use of the system and enables earning business value from early-on.

7 Further Research

To demonstrate the applicability of our approach, a research prototype has been developed [We03]. Further research will include the evaluation of this prototype on several real world business processes and how to balance the trade-off between flexibility and security. Additionally, clear criteria for the decision when to stop the initial modeling should be defined and guidelines for refactoring frequently reused cases into rules should be developed to support the user and the workflow modeler. In order to free the case-base from obsolete knowledge, an aging or MRU (most recently used) factor should be implemented into the CBR sub-system and be evaluated.

References

[Aa01] van der Aalst, W. M. P.: Exterminating the Dynamic Change Bug. A Concrete Approach to Support Workflow Change. In: Information Systems Frontiers 3 (2001) 3, pp. 297-317.

[Al97] Alonso, G.; Agrawal, D.; El Abbadi, A.; Mohan, C.: Functionality and Limitations of Current Workflow Management Systems. In: IEEE Expert, Special Issue on Cooperative Information Systems 1997.

[Ag01] The Agile Alliance. Agile Manifesto (2001). Available at http://www.agilemanifesto.org, visited on January 15, 2004.

[Aa03] van der Aalst, W.M.P.; ter Hofstede, A.H.M.; Kiepuszewski, B.; Barros, A.P.: Workflow Patterns. In: Distributed and Parallel Databases 14 (2003) 3, pp. 5-51.

[AK99] Amram, M.; Kulatilaka, N.: Real Options: Managing Strategic Investment in an Uncertain World. Harvard Business School Press, Boston, Massachusetts 1999.

[Ca98] Casati, F.; Ceri, C.; Pernici, B.; Pozzi, G.: Workflow Evolution. In: Data and Knowledge Engineering 24 (1998) 3, pp. 211-238.

[DP94] Dixit, A.; Pindyck, R.: The Options Approach to Capital Investment. In: Harvard Business Review, 73 (1995) 5 & 6, pp. 105-115.

[EF02] Erdogmus, H.; Favaro, J.: Keep Your Options Open: Extreme Programming and Economics of Flexibility, In: Marchesi, M.; Succi, G.; Wells, D.; Williams, L. (eds.): Extreme Programming Perspective, Addison Wesley 2002, pp. 503-552.

[Ga02] Gadatsch, A.: Management von Geschäftsprozessen, Methoden und Werkzeuge für die IT-Praxis: Eine Einführung für Studenten und Praktiker, 2. Auflage, Vieweg Verlag, Braunschweig, Wiesbaden 2002.

[Go02] Goesmann, T.: Ein Ansatz zur Unterstützung wissensintensiver Prozesse durch Workflow-Management-Systeme. Dissertation, Technische Universität Berlin 2002.

[KD00] Klein, M.; Dellarocas, C.: A Knowledge-based Approach to Handling Exceptions in Workflow Systems. In: Computer Supported Cooperative Work 9 (2000) 3-4, pp. 399–412.

[KSL02] Kim, J.; Suh, W; Lee, H.: Document-based workflow modeling: a case-based reasoning approach. In: Expert Systems with Applications 23 (2002) 2, pp. 77-93.

[Ko93] Kolodner, J. L.: Case-Based Reasoning. Morgan Kaufmann, San Francisco, California 1993.

[Lu00] Luo, Z.; Shet, A.; Kochut, K.; Miller, J.: Exception Handling in Workflow Systems. In: Applied Intelligence 13 (2000) 2, pp. 125-147.

[Ma02] Marling, C.; Sqalli M.; Rissland, E.; Muñoz-Avila, H.; Aha, D. W: Case-Based Reasoning Integrations. In: AI Magazine 23 (2002) 1, pp. 69-85.

[My77] Myers, S. C.: Determinants of Corporate Borrowing. In: Journal of Financial Economics 5 (1977) 2, pp.147-175

[RD98] Reichert, M.; Dadam, P.: ADEPTflex – Supporting Dynamic Changes of Workflows Without Loosing Control. In: Journal of Intelligent Information Systems, Special Issue on Workflow Management 10 (1998) 2, pp. 93-129.

[RS89] Riesbeck, C.; Schank, R.: Inside Case-Based Reasoning, Lawrence Erlbaum, Hillsdale, N.J. 1989.

[PP03] Poppendieck, M.; Poppendieck, T.: Lean Software Development: An Agile Toolkit. 1st edition, Addison Wesley 2003.

[Sc98] Scheer, A-W.: Referenzmodelle für industrielle Geschäftsprozesse. 2nd edition, Springer Verlag 1998.

[SMO00] Sadiq, W.; Marjanovic, O.; Orlowska, M. E.: Managing Change and Time in Dynamic Workflow Processes. In: International Journal of Cooperative Information Systems 9 (2000) 1 & 2, pp. 93-116.

[Sh96] Shet, A.; Georgakopoulos, D.; Joosten, S.; Rusinkiewicz, M.; Scacchi, W.; Wilden, J.; Wolf, A.: Report from the NSF Workshop on Workflow and Process Automation in Information Systems. Technical Report UGA-CS-TR-96-003, University of Georgia, October 1996.

[SK97] Sheth, A.; Kochut, K.: Workflow Applications to Research Agenda: Scalable and Dynamic Work Coordination and Collaboration Systems. In: Proceedings of NATO Advanced Study Institute on Workflow Management Systems and Interoperability, Springer, Istanbul, Turkey 1997, pp. 35-60.

[Sl91] Slade, S.: Case-based reasoning: A research paradigm. In: AI Magazine 12 (1991) 1, pp. 42-55.

[Su96] Sullivan, K. J.: Software design: the options approach. In: SIGSOFT Software Architectures Workshop, San Francisco, California, 1996.

[SZ95] Stein, E. W.; Zwass, V.: Actualizing organizational memory with information systems. In: Information Systems Research 6 (1995) 2, pp. 85-117.

[WM94] Watson, I.; Marir, F.: Case-Based Reasoning: A Review. In: The Knowledge Engineering Review 9 (1994) 4, pp. 327-354.

[Wa98] Wargitsch, C.: Ein Beitrag zur Integration von Workflow- und Wissensmanagement unter besonderer Berücksichtigung komplexer Geschäftsprozesse. Dissertation, Erlangen, Nürnberg 1998.

[Wa97] Watson, I.: Applying Case-Based Reasoning Techniques for Enterprise Systems. Morgan Kaufmann, 1997.

[We03] Weber, B.: Integration of Workflow Management and Case-Based Reasoning: Supporting Business Process Management through an Adaptive Workflow Management System. Dissertation, Innsbruck 2003.