Computing and Informatics, Vol. 30, 2011, 89–111

A FLEXIBLE APPROACH TOWARDS SELF-ADAPTING PROCESS RECOMMENDATIONS

Thomas Burkhart

Institut für Wirtschaftsinformatik (IWi)
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
e-mail: burkhart@iwi.dfki.de

Christoph Dorn

Distributed Systems Group
Vienna University of Technology
Argentinierstr 8/184-1
1040 Vienna, Austria
e-mail: dorn@infosys.tuwien.ac.at

Dirk Werth, Peter Loos

Institut für Wirtschaftsinformatik (IWi)
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
e-mail: {werth, loos}@iwi.dfki.de

Revised manuscript received 22 October 2010

Abstract. A company's ability to flexibly adapt to changing business requirements is one key factor to remain competitive. The required flexibility in people driven processes is usually achieved through ad-hoc workflows which are naturally highly unstructured. Effective guidance in ad-hoc workflows therefore requires a simultaneous consideration of multiple goals: support of individual work habits, classification

of unstructured messages, exploration of crowd process knowledge, and automatic adaptation to changes. This paper presents a flexible approach towards the mapping of unstructured messages onto processes as well as patterns for self-adjusting and context-sensitive process recommendations based on the analysis of user behavior, crowd processes, and continuous application of process detection. Specifically, we classify users as *eagles* (i.e., specialists) or *flock*. The approach is evaluated in the context of the European research project Commius.

1 INTRODUCTION

Today, enterprise competitiveness is primarily determined by an organization's ability to adapt to dynamically changing environments. Keeping the pace with innovations to maintain a competitive advantage requires rapid assembly of value chains where multiple specialized companies cooperate in the production of increasingly complex products. As a direct consequence, established work practices – especially in people driven process environments – need to become flexible and adaptable.

Traditional workflow engines lack the required flexibility for reacting to ad-hoc changes. Their rigid underlying process model would need to foresee all possible variations, which becomes unfeasible even for simple processes. Support systems for flexible processes (e.g., Caramba [7]) recommend users to follow a predefined process path, but allow them to deviate from that process on demand (for an exhaustive survey on flexible business support systems see [5]). This paper focuses on two major challenges that remained mostly unaddressed: a) users in people-driven processes require a combination of personalized recommendations, while exploiting the best practices emerging from the overall user community; b) flexible processes need to evolve across time to reflect the changes in working style, business constraints, and impact of cross-organizational cooperation.

In this paper, we introduce two different types of process recommendations: pre- and post-process step recommendations. Both types are connected with a hybrid approach that combines user-centric process recommendations with crowd-based process knowledge. Specifically, we provide recommendations learned from previous processes executed by that user and couple them with process decisions taken from all users involved in that particular process type. Our main contribution is a self-adjusting user classification model that determines whether a user engages in individualized process adaptations (*eagle*) or whether the user follows a process step sequence generally agreed upon by the crowd (*flock*). Monitoring and recommendation evaluation continuously adjusts this classification. The underlying ad-hoc process engine allows any deviations from the modeled flow. These deviations feedback into the process model, ultimately enabling process evolution through self-learning of process patterns.

A motivating scenario sets the scene for our self-adjusting recommendation approach (Section 2). In Section 3, we continue with a brief discussion of the term *flexibility* as applied in the domain of adaptive business processes, followed by related work focusing on flexible process support systems. Section 4 describes the

process recommendation algorithm and feedback mechanism. Section 5 discusses our advanced recommendation aggregation and user classification technique. We evaluate our approach based on the scenario and our prototype implementation in Section 6. Finally, Section 7 gives a short conclusion and an outlook on future work.

2 MOTIVATING SCENARIO

Within this paper, a common show-case will be used to point out different aspects of the introduced approach. The scenario (Figure 1) focuses on a business process describing the handling of incoming orders and the subsequent dispatching of the ordered product. The model hereby represents all possible steps, while in certain cases, not all steps might be mandatory. For this paper the analysis of incoming events is not in the scope of this paper. Therefore we extinguish a prepend classification process of the incoming information (i.e. analysis of emails, documents, etc.). In the scenario, an incoming order triggers the process. Subsequently, an order confirmation is returned to the customer. Further, credit and inventory checks confirm the credit-worthiness of the customer and the availability of the parts that need to be assembled. In case some parts are not on stock, replenishment of these items is triggered. As soon as the product is assembled, the shipment as well as the corresponding invoice are prepared. The process ends with the dispositioning of the ordered items.

The scenario describes a primarily people-driven workflow. A worker responsible for an individual process instance reacts to some external events that are not under his/her control. Besides the triggering of the process, the assembly and packaging of the product are carried out by specific departments. The worker needs to wait for the respective events before the process can continue. At first sight, the process does not seem very ad-hoc. Individual workers, however, are free to choose in between the desired selection and sequence of process steps. The main purpose of the process description is obtaining a first, generic process that provides a rough guide for most cases. As business requirements change due to internal forces (e.g., new products, different customer focus) or external forces (e.g., special treatment of an important customer's demand) individual workers adapt the order of steps as they see fit. A worker, for example, can decide to ship the goods before completing billing and invoicing. Our mechanism monitors such decisions and continuously adapts to recommend the most suitable next steps.

3 RELATED WORK

3.1 Defining Flexibility in Business Processes

The term *flexibility* in the context of business processes comes with a multitude of interpretations. An overview over the most established interpretations of flexibility allows us to better compare our contribution to existing approaches.

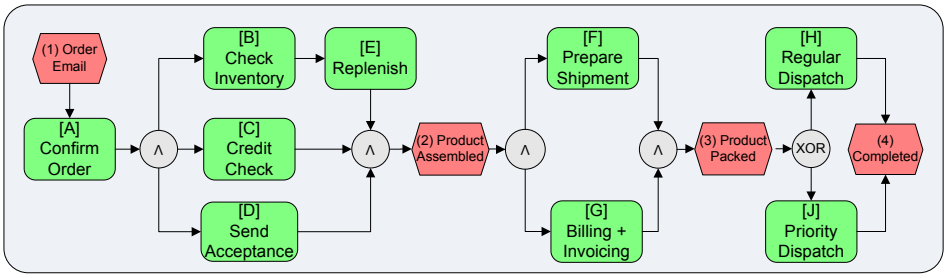

Fig. 1. Scenario: generic people-driven order process model (PM)

In the scope of this paper, we define process flexibility as the ability to adapt the process flow on demand through adding, skipping, or sequence reordering of process steps. This definition is closely related to the interpretation by Adams et al. [3] in which processes simply provide a guideline while the appropriate way of handling single tasks is chosen on an as-needed basis. In Reijers et al. [14], process models define the normal way of achieving a goal, but still offer the possibility to deviate based on available case data. Sadiq et al. [15], on the other hand, describe flexibility as the ability to deal with processes that are only partially defined at build-time. Soffer [17] distinguishes between short-term flexibility (i.e., deviations from a given model) and long-term flexibility (i.e. evolution of processes). Greiner and Rahm [10] limit the definition to exception handling capabilities in case of unforeseen events or policy changes. In contrast to the application specific perspectives, Adamides et al. [1], define strategic flexibility which describes a company's diversity of strategies and its capability to switch between them.

3.2 Flexible Process Support Systems

Research on providing recommendations in flexible workflow systems focuses on multiple aspects. The major means of providing recommendations is done by guidelines. A predefined process model assists a user in choosing how to proceed a workflow. In more detail, such a guideline can exist merely of process parts (like presented by Sadiq et al. [15]) and which thus does not require a complete process model. Alternatively, guidelines can define what has to be done in each specific process state but still not provide a complete process path [13]. Moreover, Adams et al. [2] define each process step within such a guideline as a simple placeholder task which is dynamically replaced by a context sensitive choice from an extensible catalog of suitable workflow definitions during run-time. The actual selection process is ultimately defined by so-called Ripple Down Rules [3]. In addition, recommendations can be derived based on a rough task structure [8].

In contrast to these guideline approaches that are mainly based on predefined process models and might not be instantiated at all, recommendations are based on best-practices shared by users within a company [18]. Pesic et al. and

van der Aalst [12, 11] provide recommendations based on past experiences and additionally on a specific process goal. This is achieved by comparing the current process instance with past executions (logs), while preferring those executions that satisfy the specified goal. A similar approach can be found in [16] where recommendations are generated based on similar past process executions by considering the specific optimization goals. Another approach is followed by Almeida and Casanova [4] whose recommendations are based on an ontology and semantic rules that generate possible process alternatives or suitable process steps if the execution of a workflow instance fails to proceed. Van der Feesten et al. [20] follows an approach in which, based on the information available for a case, the next step to be performed is determined using a strategy of e.g. lowest cost or shortest processing time.

While the previous approaches focus on concrete recommendations, the TIBCO Software Inc. provides detailed process information and context to the user. Thus, a user can identify which steps are required to achieve the process goal [19].

These flexible process support systems seem to be on the right track when comparing their capabilities to the stated definitions and statements concerning flexibility. According to several surveys, however, actual implications of the ad-hoc approach lack of a sufficient degree of process guidance during run-time due to their overly extensive degree of freedom (cf. [5]).

4 PROCESS RECOMMENDATION

We assume an underlying process model (PM) consisting of a set of process steps ($S \subseteq \mathcal{P}$). Each process step S specifies one or more actions a all of which the user needs to perform to complete the step. Multiple similar process steps exist when different combinations of the same actions can be sensibly combined. (E.g., instead of having one example process step S1 containing actions a1, and a2 or a3, we would define S1 requiring a1 and a2 and a second process step S2 requiring a1 and a3.)

The process model initially defines only a sensible, general purpose order of process steps which the user is free to follow or deviate from as circumstances require. The true, underlying process model emerges only through observation of and learning from the user's decisions on the process step sequences.

A recommendation r consists of a process step S, and the recommendation confidence w defined in the interval $[0, 100]$, where 100 indicates absolute certainty. Within a set of recommendations R given at a particular points within the scope of a process P, the sum of confidence values will always add up to exactly 100 ($\sum w_i = 100 \; \forall \; r_i \in R$).

Our approach combines two types of recommendations to guide a user through flexible, people-driven processes: *pre-process step recommendations* and *post-process step recommendations*. These two complementary mechanisms are illustrated in Figure 2 and work as follows:

Pre-Process Step Recommendations. Upon the occurrence of an external event, the system autonomously matches the event onto a specific process step

without further assistance and presents it to the user. In case of a mismatching, the user has the possibility to manually correct the decision (see Section 4.1). For example, in the scenario the system recommends *Prepare Shipment* rather than *Billing+Invoicing* upon *Product Assembled*. The ultimate recommendation depends on learned user preferences.

Post-Process Step Recommendations. Our mechanism includes recommendations beyond the immediate process step. Post process step recommendations assume that the user carries out the immediate underlying process step and therefore gives suitable follow-up actions. This allows the user to perceive the pre-process step recommendation in the context of subsequent necessary/optional process steps, weighted with probabilities (see Section 4.2). In the scenario, a pre process step recommendation of *Prepare Shipment* comes with a strong post process step recommendation including *Billing + Invoicing*, weakly recommending *Regular Dispatch* and *Priority Dispatch*.

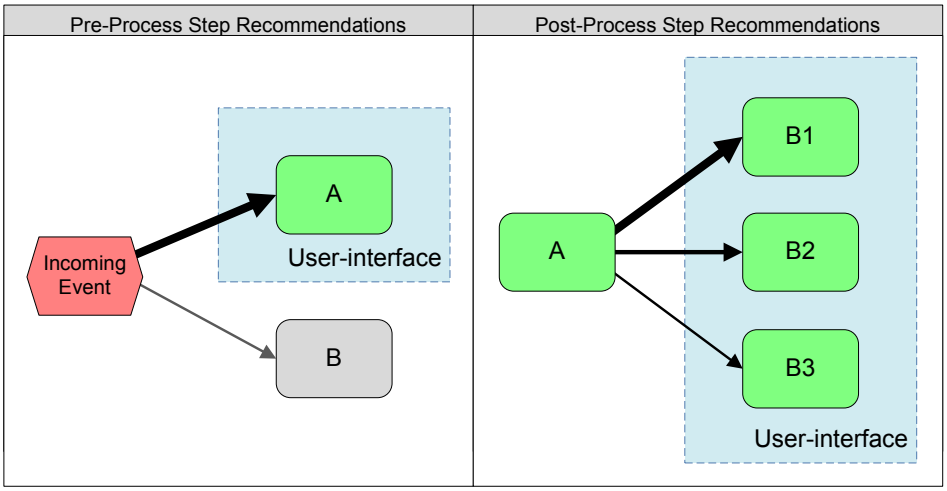

Fig. 2. Illustration of Pre- and Post-Process Step recommendations

While pre-process step recommendations occur always in the case of an external event, post-process step recommendations are always based on an actual process step. Both recommendation categories apply learning mechanisms to adapt and improve results over time through observation of user actions. While pre-process step recommendations learn only from explicit user correction, post-process step recommendations learn from the implicit user decisions which process step to carry out next. The following sections describe the two mechanisms in more detail.

4.1 Pre-Process Step Recommendations

In case of an external event, e.g. an incoming order (cf. Figure 1), the system will automatically match the event onto an executable process step without requiring any assistance of the user. These implicit recommendations are generated through the usage of process knowledge concerning the matching of an event onto a process step in a specific context. This knowledge can be generated in two ways. The matching can be done manually by a user during build-time. The second option consists in the exploitation of process knowledge, collected by the observation of user behavior in specific situations. Upon initial installation of the system, recommendations will be mainly based on build-time configuration, while during its life-cycle, the system will evolve and formulate recommendations mainly on conclusions of the prior user behavior. A more detailed description of the mechanism behind such implicit recommendations and the self-adjusting capabilities are shown in Figure 3.

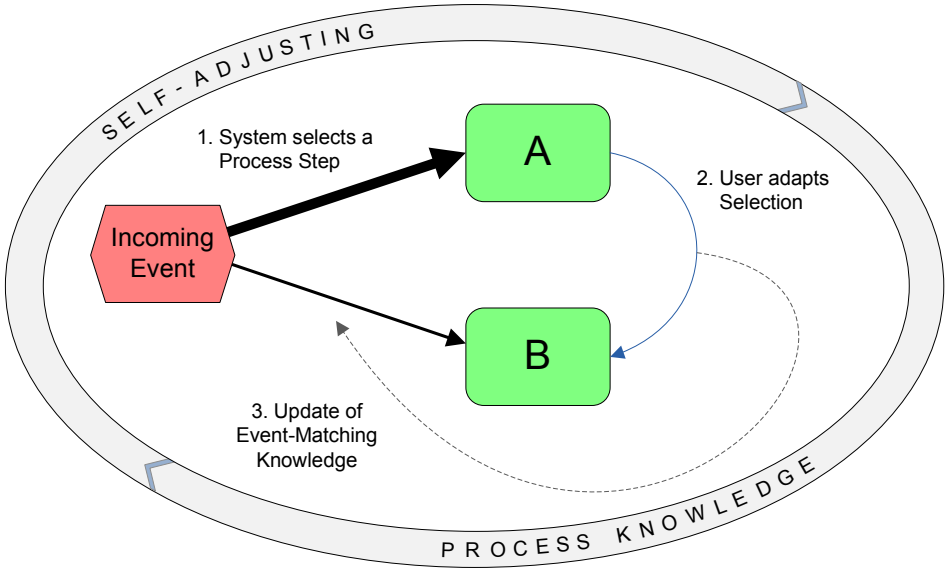

Fig. 3. Illustration of Role-Back functionality to handle pre-process recommendations

In case of the occurrence of a specific event, the system selects a corresponding process step that seems most suitable and recommends it implicitly to the user. If the user has the feeling that the presented process step is an incorrect matching or another one would fit his/her needs in a better way, s/he has the opportunity to make use of the "roll-back feature" allowing to change the process step during run-time. The "roll-back" enables the user to alter the decision of the system and manually connect an occurring event with a process step. As mentioned before, the system exhibits simple self-adapting patterns to extend and adjust the initial

process model. Therefore, the usage of the "roll-back" is tracked and processed for future recommendations (cf. Figure 3).

If the user does not change the recommended step, the system assumes that his implicit recommendation has been accepted and was correct. This acceptance is also tracked and raises the rating of the specific matching, which leads to a higher probability that this recommendation will be given again in future situations regarding the same context. In contrast, if the roll-back functionality is used, the system implies an incorrect recommendation, which consequently lowers the possibility of the same recommendation in the specific context. In case of processes which repeat on regular basis (e.g. order process, invoices, quote requests) the system will learn over a certain period of time and adapt itself to a certain user behavior.

4.2 Post-Process Step Recommendations

The recommendation mechanism applies two related data sets: the process model (PM, Figure 1) and the sequence graph (SG). Figure 4 a) displays the sequence graph for the first steps of the scenario process. The sequence graph $SG(S, E)$ comprises nodes representing the individual process step S. A directed edge $e \in E$ in SG between two nodes A and B describes a temporal sequence that process step B follows immediately after A. Whenever a user conducts process step B after process step A we increase the edge value. The SG accumulates all individual process step sequences for a particular process type.

It thus describes the likelihood (i.e., preference) of following a particular path through the process. In Figure 4a, the arc thickness indicates this preference. The sequence graph emerges from the process log sequences that are collected for each executed process instance. When we subsequently apply existing process techniques [9] on the sequence graph, we obtain the corresponding process model. It describes the dependency between process steps such as joins, splits and sequential steps.

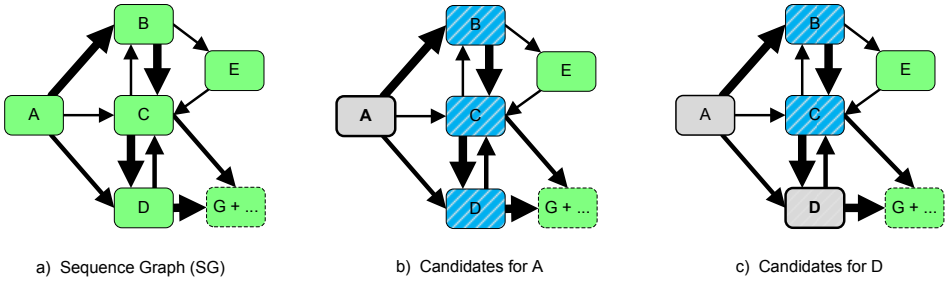

Fig. 4. Sequence Graph (SG) excerpt of process step candidates: completed process steps in dark gray, candidates in shaded blue, and inactive process steps in green

When a process is started, we derive a copy of the process model to track process progress. We utilize the process to select the relevant process steps that are

sensible to enact in the current process state. This is the task of the *Process Instance Manager* (Figure 7). For each point in time, it keeps track of process steps that have been completed, which are active (i.e., all process step preconditions are fulfilled, but the step has not been carried out), and which steps need to be (de)activated. For our recommendation purpose the *Process Instance Manager* provides a list of process step candidates that are ready to be carried out.

The sequence graph then provides the information to establish which of the possible process steps to carry out first. The algorithm in Listing 1 describes the recommendation procedure in detail. The algorithm requires the sequence graph (SG), process model (PM), and current (respectively just finished) process step (N) as input.[1]

For our example process in Figure 1, we suppose the user has completed the order confirmation process step A. The *Process Instance Manager* now identifies *Check Inventory* B, *Credit Check* C, and *Send Acceptance* D from the PM as sensible next steps (Figure 4 b). Subsequently, the SG weights these candidates according to the edge values (e.g., here $w(AB) = 70, w(AC) = 10$, and $w(AD) = 30$).

Algorithm 1 selects all active process steps (lines 1 to 5) which at the current process status include process steps B, C, and D. In lines 7 to 9, the algorithm determines the likelihood that any of the candidates is an immediate successor of the recently finished step A. For now we store merely the edge weight in the recommendation set R. For now, we assume the algorithm finishes here and just return the current content of R.

Suppose the user does not follow the top-rated recommendation B and selects step D instead (Figure 4 c) and executes the associated actions. Now, a simple recommendation based on the SG would suggest primarily to continue with the process ($G + \ldots$) and as second choice continuing to C (because arc DG yields a higher edge weight than edge DC). A pre-selection of valid edges based on the PM, however, identifies B and C as the only sensible next process steps as the process model specifies steps B, C, and D in *AND* branches and thus will only activate G once all three steps have been completed or explicitly skipped by the user.

Having only B and C in the candidate set $Cand$, lines 7 to 9 will process only the edge towards C. We should not, however, focus only on recommending subsequent process steps, as the user is free to select any process step. Lines 10 to 23 in Algorithm 1 analyze the process for skipped and out-of-order process steps. Thus, after finishing D (Figure 4 c), the algorithm also checks any preceding steps of D that are still active (i.e., only C) and scores them according to outgoing weights (i.e., $w(CD)$) – lines 10 to 14.

Finally, the algorithm also considers candidates that are not directly connected to the last completed step (i.e., B). The SG will not exhibit an edge between two nodes, if the corresponding sequence has never occurred in the logs before. It does

[1] Note that the user can request a recommendation of a process step before finishing the underlying process step already to obtain timely information on subsequently recommended steps.

not indicate, however, that the user is restricted to follow that sequence, but it will not be recommended to him/her. Here lines 15 to 23 process candidates based on their aggregated weight on their respective incoming edges (i.e., $w(AB)$). We limit the incoming edges to those that originate at already completed process steps. We, thus, prefer candidates that follow already completed steps and that are frequently traversed as it is more likely that completed steps have produced a result that is needed by the immediate successors, rather than somewhere later in the process.

Before the recommended process step in R are returned to the user, the steps' weights are normalized to add up to 100. Then the steps are sorted to have the most likely next step at first position in the set. In short, the sequence graph by itself cannot give recommendations that respect control flow constraints. The process model by itself, on the other hand, cannot provide suggestions on the order which process step to carry out first.

4.3 User vs. Crowd-based Recommendations

The generic scenario process gives rise to distinctive process adaptations as required by different environment needs. We observe the behavior of following three example users. User 1 is responsible for normal customers that order standard products which get automatically restocked once a certain threshold is undercut. Normal customers receive their goods via regular shipping (Figure 5 a). User 2 serves to premium customers that have a high order volume, pay on time and thus need not go through a credit check. Premium customers receive priority shipment (J) to deliver their ordered goods as fast as possible (Figure 5 b). User 3 handles special cases. Being a new employee, s/he tends to forget certain process steps. Specifically, s/he never returns order confirmations (D), and occasionally misses the preparation of billing information (G).

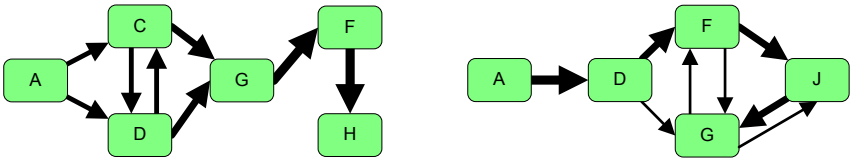

Fig. 5. Sequence graph for User 1 (handling standard orders) and User 2 (serving premium customers)

Each individual user exhibits a very personalized process that deviates considerably from the standard order process.[2] While personalized recommendation would

[2] Note that for collaborative processes (i.e., multiple interacting users) the personalized process and respective recommendations cover only the part of the process in which the user is involved in.

Algorithm 1 Crowd-based Recommendation Algorithm $\mathcal{A}(SG, PM, N)$.

```
1: for all ProcessStep S ∈ PM do                ▷ Get list of process step candidates.
2:     if state(S) == active then
3:         Cand ← S
4:     end if
5: end for                                      ▷ Initialize process step ranking scores.
6: R ← ∅
7: for all ProcessStep S ∈ Cand do              ▷ For all consecutive process steps of N.
8:     R[S] = SG.getEdge(N, S).weight
9: end for
10: if hasActivePredecessors(S) then            ▷ If a preceding process step has been
   temporarily skipped.
11:     for all ProcessStep S ∈ Cand do         ▷ Extract incoming edge weight from
   SG.
12:         R[S] += SG.getEdge(S, N).weight
13:     end for
14: end if ▷ For any other active process step that is not directly connected to N.
15: for all ProcessStep S ∈ Cand ∧ R[S] == 0 do
16:     wsum = 0
17:     for all Edge e ∈ SG.getInEdge(S) do
18:         if state(sourceNode(e)) == completed then   ▷ Count the edge weight
   only if the predecessor step has been completed.
19:             wsum += e.weight
20:         end if
21:     end for
22:     R[S] = wsum
23: end for▷ Normalized candidates probabilities in R to obtain a total sum of 100
24: norm(R)                                     ▷ Sort candidates by probability in R descending
25: sort(R)
26: return R
```

yield highly relevant process step rankings, these recommendations cannot exploit alternative activities when exceptions such as delayed shipping, or partial order content is out of stock. Moreover, pure personalized recommendations will reinforce inefficient or even incorrect sequences such as inadvertently skipping an important process step. Crowd-based recommendations mitigate this shortcoming.

Crowd-based recommendations enrich the set of relevant possible process paths through aggregation of the process experiences from multiple users. Personalized processes capture the habits of an individual user. They are, however, limited to process step sequences that a particular user has executed so far. Alternative sequences that potentially reduce overall processing time remain unavailable. Also, a personalized process cannot be applied for giving advice in exceptional situations that have not been encountered by the user before.

Figure 6 displays the aggregated process model for users 1, 2, and 3. To obtain the aggregated process model, we combine all process step sequences from every user executing the order process and generate one sequence graph. From this sequence graph, the process mining technique referenced above then generates the corresponding crowd-based process model.

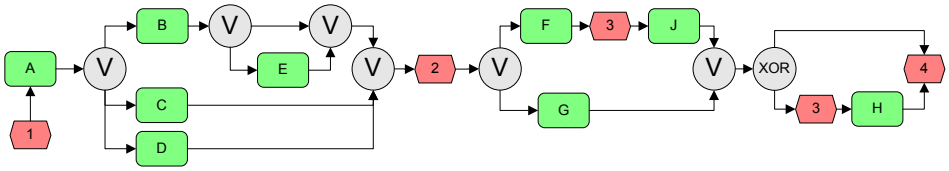

Fig. 6. Crowd-based Process Model for User 1, 2, and 3

4.4 Overall Recommendation Cycle

A complete recommendation cycle is depicted in Figure 7. The cycle outlines how process model, sequence graph, recommendations, roleback, and user actions are connected. The exact mechanisms for aggregation user-centric and crowd-centric recommendations is covered then in the subsequent Section 5.

An incoming request for recommendations triggers the recommendation mechanism (1). When the request coincides with an external event (2), the Event Monitor notifies the Process Instance Manager (3) to check which process steps should be activated (4). The Process Recommender then selects a single process step (5) but also keeps the alternatives to allow for simple role-back by the user. Steps 2 to 5 are skipped when no external event is present.

The process recommender then collects information from the process instance manager (6a) and the sequence graphs (6b) to aggregate sensible upcoming process steps. The process recommender retrieves this information from the SG or PM. The recommender subsequently provides the user with the recommended process steps (7).

The user checks the correctness of the pre-process step recommendation and forces a role back when another process step is more suitable (8). This choice is logged (9a) and post-process step recommendations are repeated (9b). The user then carries out the immediate underlying process step and other recommended steps which do not rely on external events. A process step is enacted by clicking, for example, on a link on the user's interface (10). Note that the user does not explicitly agree or disagree with a recommendation. Instead, the system monitor observes the user's actions (11) to determine the true process progress. The system monitor updates the process instance manager whenever a process step has been completed (12). The process instance manager in turn updates the sequence graph for each completed step (13). In regular intervals, the process miner mechanism takes a sequence graph and generates an updated process model (14). We apply an aging

mechanism to reduce the effect of old, potentially outdated, process sequences. For every new incoming process sequence we remove the oldest sequence. We add role-back preferences to the resulting process model to update event-to-process step mappings (15).

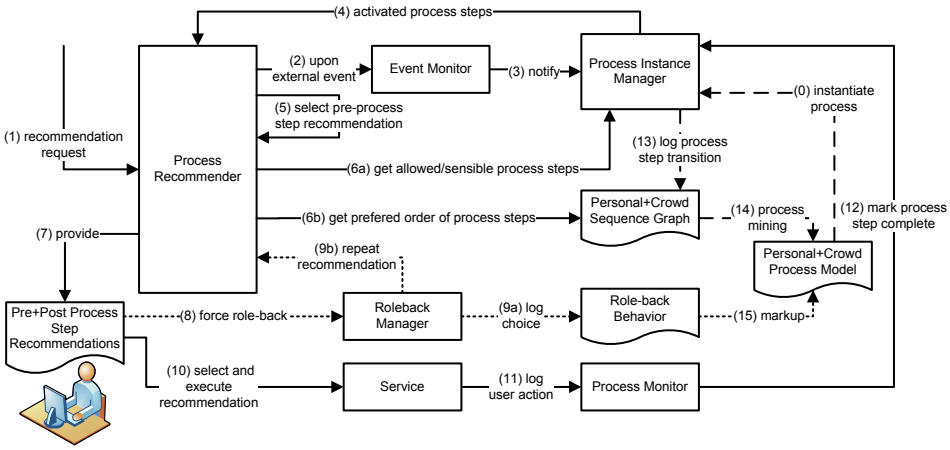

Fig. 7. Feedback cycle for personal and crowd-centric recommendations. The role-back cycle is highlighted by dotted lines, while the process refinement cycle is given in dashed lines.

The recommendation cycle provides recommendations based on personal and crowd-centric PM and SG. To obtain both types of recommendations, for each process the Process Instance Manager instantiates two instances of the PM and access to two sequence graphs. One instance of the personal process model and corresponding sequence graph delivers the recommended steps and order of steps based purely on previous user actions. The other process model instance reflects the crowd knowledge. The crowd-based process model and sequence graph deliver best practices recommendations derived from all users. Ultimately, the Process Instance Manager keeps track of the process progress in both instances. When the logged user sequence data is mined from time to time (step 14), this is done twice: once only based on user log sequences and once taking all log sequences.

5 SELF-ADJUSTING RECOMMENDATION MODEL

Users receive recommendations derived from the personal PM and SG or the crowd-centric PM and SG based on their classification as *eagle* or *flock*. This classification is captured by the parameter α. It describes the user on a scale between 0 and 1, where 1 denotes a user always adhering to his individual work style – the *eagle*. At the other extreme end of the classifier ($\alpha = 0$), a user follows generally applied work practices – *flock*. We determine α for each user and process type as a user's work

style potentially deviates for each process type. The overall recommendation merges user-centric and crowd-based recommendations according to the following formula:

$$R_{overall} = \alpha \cdot R_{user} + (1 - \alpha) \cdot R_{crowd}. \tag{1}$$

When a user tends towards *eagle* we give more importance to the recommended steps from the personal PM and SG and vice versa. To do so, we multiply the recommendation weights w_{REC} within R_{user} with α and repeat the same for R_{crowd} with $(1 - \alpha)$. Sorting the merged list provides the overall recommendation.

Suppose following simple example consisting of user- and crowd-based recommendations: R_{user} recommending S_1, S_3, S_4 and R_{crowd} recommending S_2, S_3, S_5, S_6. We obtain following overall recommendation for $\alpha = 0.5$ (Note that the weights for S_3 are aggregated.):

$$0.5 \cdot \begin{bmatrix} S_1 & 70 \\ S_3 & 15 \\ S_4 & 15 \end{bmatrix} + 0.5 \cdot \begin{bmatrix} S_2 & 80 \\ S_3 & 10 \\ S_5 & 5 \\ S_6 & 5 \end{bmatrix} \longrightarrow \begin{bmatrix} S_2 & 40 \\ S_1 & 35 \\ S_3 & 12.5 \\ S_4 & 7.5 \\ S_5 & 2.5 \\ S_6 & 2.5 \end{bmatrix} \tag{2}$$

In our example, we set $\alpha = 0.5$ to denote a user that has not been classified as *eagle* or *flock* yet. We reject a fixed configuration of the parameter α. Instead, dynamic classification adjustment reflects a user's adaptation to changing process requirements and learning effects. To this end, we observe the user's selection of recommended process steps. We increase the value of α when the user carries out a process step that originated from R_{user}. Similarly, we reduce the value of α when the user follows crowd-based recommendations. The four factors determining the amount to which α is changed are:

(i) similarity of user-centric and crowd-based recommendations,

(ii) process success,

(iii) current value of α,

(iv) explicit user feedback via role-back.

5.1 Recommendation Similarity

We calculate the similarity of user-centric and crowd-based recommendations implicitly by comparing the actual user actions with both recommendations. We assume that users deviate slightly from recommended process steps on a regular basis. Suppose that the user-centric recommendation suggest process step $S_1(a_1, a_2)$ and the crowd-based recommendation suggest process step $S_2(a_2, a_3)$. Due to unforeseen circumstances the user executed actions a_2, and a_4. Thus, we first combine the user's

actions (a_2, a_3) in an anonymous process step type and then compare that process step with the given recommendations.

We determine the similarity of two process steps by observing the overlap of common and individual actions. Specifically, we apply the weighted Jaccard similarity measurement

$$sim_{wJaccard}(S_x, S_y) = \frac{\sum_{a \in S_x \cap S_y} w_{IDF}(a)}{\sum_{a \in S_x \cup S_y} w_{IDF}(a)} \tag{3}$$

where $w_{IDF}(a)$ is the weight function describing the frequency of action a occurring in a process step s. Here the weight function is the inverse document frequency (IDF) of a. In our case, a document is a process step S where the words correspond to the involved actions a. The overall document corpus is equivalent to the set of all specified process steps $mathcal P$ that occur in the underlying process model. The weight for a particular action a_i is defined as

$$w_{IDF}(a_i) = log \frac{|S|}{s : a_i \in s} \tag{4}$$

where $|S|$ is the number of all process steps and $s : a_i \in s$ counts all process steps that contain action a_i. Actions that occur in most process steps will thus yield low weight when comparing two process steps, while rare actions will yield a high weight.

The similarity of user actions and recommendation derive the recommendation's success. Each recommended process step is additionally weighted by the recommendation's weight. For an anonymous process step A we calculate:

$$succ(R, A) = \sum_{i}^{R} sim_{wJaccard}(A, s_i) \cdot w_{REC}(s_i). \tag{5}$$

The overall effect on α moving towards *eagle* or *flock* is then simply derived through comparison of personalized and crowd-based recommendation success:

$$\delta(A) = succ(R_{user}, A) - succ(R_{crowd}, A). \tag{6}$$

5.2 Avoiding Classification Lock-In

An *eagle* remains locked-in in his classification when he repeatedly fails to successfully complete a process but continues to receive exclusive personal recommendations. In this case, we have to abandon the underlying classification. A user is considered locked-in, when his/her average process success rate falls below the average process success rate of the top 50 % *flock* users. Specifically, we sort all users (U) according to their current classification value α in ascending order and select the process success rate *psucc* of all users having α equal or below the second quartile. We set $\alpha = 0.5$ for user u if s/he fails to meet the following threshold

condition:

$$\bar{psucc}(u) > \frac{2 \cdot \sum_i \bar{psucc}(u_i)}{|U|} \qquad \forall u_i : \alpha_i \leq Q_2. \tag{7}$$

This is expected to raise the number of successful processes as the user is presented with process step alternatives s/he did not consider before. The user classification might again deviate towards *eagle* again, but this time resulting in more sensible process steps.

5.3 Accelerated Classification Divergence

When recommendations combine profile-based and crowd-based recommendation to approximately equal extent, the top recommended process steps are potentially similar or, on the other hand, completely contradicting. We apply a sigmoid function to avoid remaining too long in the middle between *eagle* and *flock* ($\alpha \sim 0.5$). The sigmoid function ensures that we can quickly move from the middle in both directions. However, we will only move if $\delta(A) \; ! = 0$ (i.e., when there is a trend towards *eagle* or *flock*); otherwise we remain with the previous α value. Based on $\delta(A)$ and current classification value α_t, we determine the new α_{t+1}:

$$\alpha_{t+1} = \begin{cases} \min[\max[(1 + e^{-10 \cdot (\alpha_t + \delta(A)) + 5})^{-1}, 0], 1] & \text{if } \delta(A) \neq 0, \\ \alpha_t & \text{if } \delta(A) = 0 \end{cases} \tag{8}$$

where the Min and Max operators limit α to the interval $[0, 1]$.

5.4 Explicit Role-Back

Feedback from role-back decisions are immediately available but only applied when external events occur. For each process, we track which process step followed upon occurrence of a particular event (this is stored in the *Roleback Decision DB*, Figure 8). Similar to the sequence graph, the database captures all experienced sequences of event x followed by a process step y from which we derive the dependency probability $p(eventx, stepy)$ by calculating $\#x \; followedBy \; y / \#x \; occurred$. Again we track these dependencies for individual users and for all users separately.

Ultimately, we change α by the difference of the selection probabilities: when personal and crowd-based probabilities differ strongly, also α is subject to a large change.

$$\delta_{roleback}(x, y) = [p_{user}(x, y) - p_{crowd}(x, y)] \cdot \gamma \tag{9}$$

In case of a role-back, α is subject to a more significant update ($\gamma = 0.5$) since the role-back indicated an incorrect recommendation. When no role-back occurs, alpha receives only a minor update ($\gamma = 0.05$) as we assume the current configuration is suitable.[3]

[3] Suitable values of γ were determined through experiments.

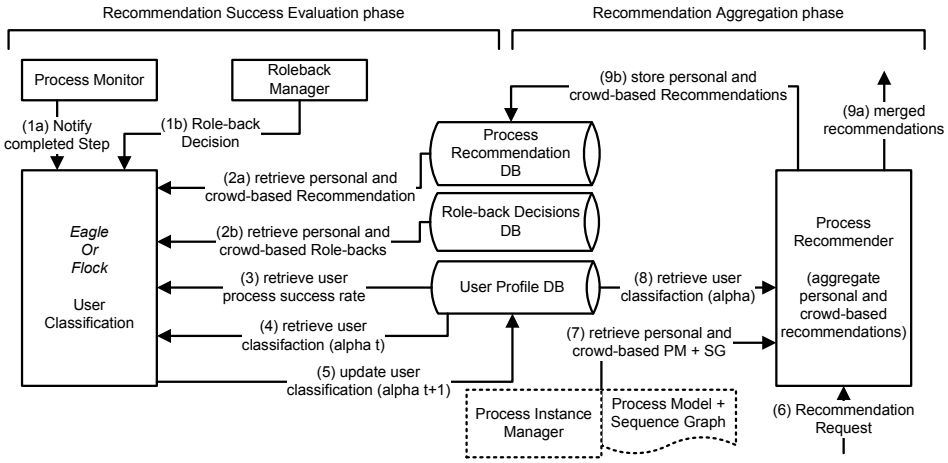

Fig. 8. Self-tuning of classification parameter α based on recommendation success and role-back

5.5 Classification Self-Tuning Cycle

The complete classification self-tuning cycle consists of the *Recommendation Success Evaluation phase* and the *Recommendation Aggregation phase* (Figure 8). For each completed process step (1a) the *User classification* component retrieves the corresponding personal and crowd-based recommendations (2a) from the *Process Recommendation DB*. Alternatively, upon a role-back (1b) probabilities on the collected role-back decisions for the corresponding external event are required to determine the basic trend towards *eagle* or *flock* (2b). Next, we apply the recommendation similarity comparison. Subsequently, we evaluate the user process success rate (3) to check for classification lock-in. The *User Profile DB* manages classification values for the various process types and the corresponding process success information. We calculate the new classification value based on the previous value (4). The previous value is neglected if the lock-in check triggers a classification reset. Finally, the new classification value is stored (5).

The recommendation aggregation phase provides more details on how the *Process Recommender* – first introduced in Figure 7 – merges personal and crowd-based recommendations. Upon an incoming recommendation request (6), the recommender retrieves personal and crowd-based PM and SG (7). For each set, the ranking algorithm in Listing 1 determines the top process step candidates. The two sets are then aggregated applying the classification parameter (8). While the user receives the merged recommendations (9a), the process recommender stores the two output rankings of the recommendation algorithm separately (9b).

6 EXPERIMENTS

6.1 Scenario Evaluation

We demonstrate the effect of user classification based on the motivating scenario, in particular based on the behavior of the three user types. Figure 9 provides the process recommendation evaluation results for 10 instances of the order process for each user (dashed lines) and the corresponding effect on the user classification α (full lines). We applied rapid aging in the experiment to visualize the convergence towards *eagle* or *flock* more clearly (i.e., new process sequences have an early and strong impact).

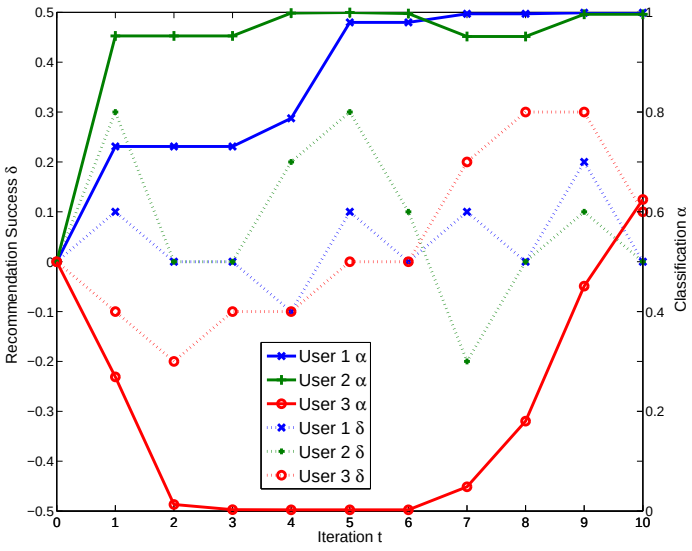

Fig. 9. Classification change α (full lines) and user recommendation success δ (dashed lines) for User 1, 2, and 3 across 10 time intervals

User 1 engages in order processes involving normal users; thus his/her individual working style does not deviate much from the initial process model, and from the emerging crowd-based process model either. Thus s/he takes up some crowd-based recommendations but remains slightly with the personalized recommendations (i.e., δ on average between 0 and 0.1). We have a delayed convergence towards *eagle*; however, deviations towards *flock* have no effect. User 2 displays also *eagle* behavior, albeit diverges more quickly as his/her process for premium customers deviated more clearly from the best practices.

User 3 exhibits a typical learning behavior. As s/he realizes to execute process step D, s/he strongly deviates from the personal recommendation, thus s/he becomes a *flock* member (interval t_1 to t_4). As his/her corrected behavior becomes more present in the personal flow model, the differences between personal and crowd-based recommendations decrease (t_5 to t_6) and his/her personal sequence preferences start to show effect (t_7 to t_{10}). Multiple, sequential recommendation evaluations towards *eagle* ($\delta > 0$) cause his/her reclassification.

As users learn and adapt their behavior, new flow control structures emerge from the crowd sequence graph. Once User 3 apprehends to always send order confirmation, the evolved crowd-based PM (Figure 10) identifies process step D as mandatory (and no longer optional).

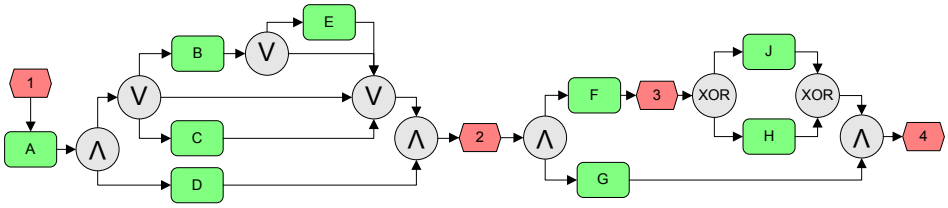

Fig. 10. Evolved Process Model for Users 1, 2, and 3

6.2 Prototype Evaluation

The recommendation approach introduced in this paper has been implemented as a proof-of-concept within a software prototype of the European research project Commius. The prototype connects to a standard email environment – intercepting and analyzing email traffic – in order to detect process steps from the communication behavior of a user.

The users apply the process configuration tool (Figure 11 left) to define a coarse-grained structure of the desired process. Within Commius, process configuration tool currently supports only simple sequential structure; however, the integrated mining algorithm refines process model later as derived from user actions. When the system recognizes this predefined process steps in the email traffic, it will automatically enrich the corresponding emails with context sensitive information as well as process recommendation concerning further steps [6]. Figure 11 (inset) displays an example email enriched with pre- as well as post-process step recommendations. The process step sequence with highest probability is provided on the right side (here four subsequent process steps taken from the scenario). While hoovering over the actual process step, a pop-up menu gives the user the opportunity to apply the roll-back feature described in Section 4.1. Clicking on one of the provided links would lead to a re-matching of the email to an other process step.

The aggregation of personal and crowd-based recommendations exhibits a different process step sequence than the originally modeled flow. User 3 has been

classified as *flock*, thus the recommendation advises him/her to prepare order confirmation. The enhanced email also demonstrates the flexibility supported by our prototype. In case the user prefers not to follow any of the given recommended steps, s/he is free to select any other step from the underlying process. The popup contains the probabilities how well the alternative process steps match the current process context.

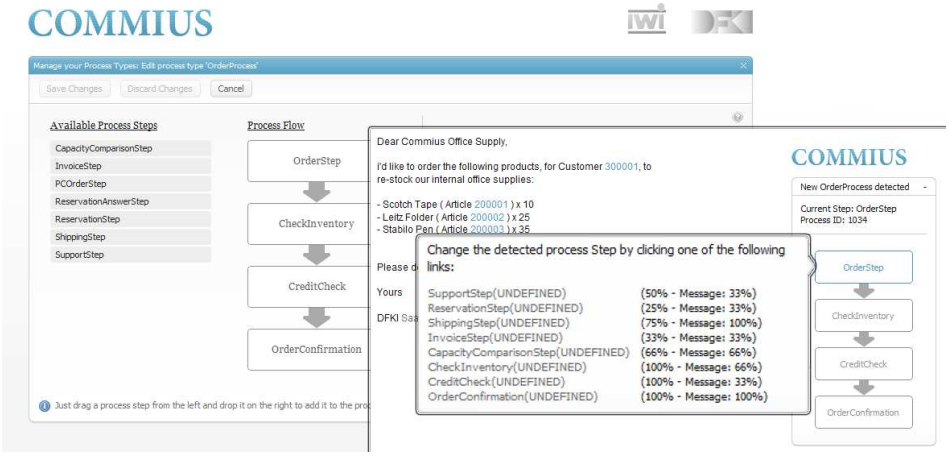

Fig. 11. Commius process modeling tool and email-based process step recommendation (inset)

6.3 Results

The evaluation results are twofold. First, we achieved the successful application of our approach in email-based process environments. Recommendation support is directly integrated in the email client. Second, we demonstrated the user classification mechanism based on three user types. Classification diverges quickly (User 1, User 2), and displays the benefit of crowd-based process model to overcome erroneous process decisions (User 3) followed by subsequent reclassification.

7 CONCLUSION AND OUTLOOK

Recommendations for people-driven ad-hoc processes exhibit maximum effectiveness when personal and crowd-based behavior is combined. Adding continuous process detection and user classification ensure valid recommendations even in case of process evolution. We introduced the concepts of *eagle* and *flock* to describe the recommendation needs of distinct user types.

Future work will focus on evaluating the recommendations in real-world environments within the scope of the Commius project. At the same time, we plan to

integrate context constraints to distinguish between process sequences that depend to a large degree on data input and/or specific environmental conditions. This will allow to give even more targeted recommendations. In addition, we intend to investigate clustering techniques for discovering conflicting recommendations in the crowd-centric process model.

Acknowledgment

This work has been partially supported by the EU STREP project Commius (FP7-213876).

REFERENCES

[1] Adamides, E. D.—Stamboulis, Y.—Pomonis, N.: Modularity and Strategic Flexibility: A Cognitive and Dynamic Perspective. In Systems Dynamics Society Conference 2005, 2005.

[2] Adams, M.—Edmond, D.—ter Hofstede, A. H. M.: The Application of Activity Theory to Dynamic Workflow Adaptation Issues. In 7th Pacific Asia Conference on Information Systems, 2003, pp. 1836–1852.

[3] Adams, M.—Hofstede, A.—Edmond, D.—van der Aalst, W.: Facilitating Flexibility and Dynamic Exception Handling in Workflows Through Worklets. In Proceedings of the CAiSE '05 Forum, FEUP, 2005, pp. 45–50.

[4] Almeida, T.—Vieira, S. C.—Casanova, M. A.: Flexible Workflow Execution Through an Ontology-Based Approach. In Workshop on Ontologies as Software Engineering Artifacts (OOPSLA), 2004.

[5] Burkhart, T.—Loos, P.: Flexible Business Processes – Evaluation of Current Approaches. In Proceedings of Multikonferenz Wirtschaftsinformatik – MKWI 2010, 2010.

[6] Burkhart, T.—Werth, D.—Loos, P.: Commius – An Email Based Interoperability Solution Tailored For SMEs. Journal Of Digital Information Management 6, 2008.

[7] Dustdar, S.: Caramba-Process-Aware Collaboration System Supporting Ad Hoc and Collaborative Processes in Virtual Teams. Distributed Parallel Databases, Vol. 15, 2004, No. 1, pp. 45–66.

[8] Eichholz, C.—Dittmar, A.—and Forbrig, P.: Using Task Modelling Concepts for Achieving Adaptive Workflows. In EHCI/DS-VIS, 2004, pp. 96–111.

[9] Gaaloul, W.—Baïna, K.—Godart, C.: Log-Based Mining Techniques Applied to Web Service Composition Reengineering. Service Oriented Computing and Applications, Vol. 2, 2008, No. 2-3, pp. 93–110.

[10] Müller, R.—Greiner, U.—Rahm, E.: $A_{gent}w_{ork}$: A Workflow System Supporting Rule-Based Workflow Adaptation. Data Knowl. Eng., Vol. 51, 2004, No. 2, pp. 223–256.

[11] Pesic, M.—Schonenberg—M. H.—Sidorova, N.—van der Aalst, W. M. P.: Constraint-Based Workflow Models: Change Made Easy. In OTM Conferences, 2007, No. 1, pp. 77–94.

[12] Pesic, M.—van der Aalst, W. M. P.: A Declarative Approach for Flexible Business Processes Management. In Business Process Management Workshops, 2006, pp. 169–180.

[13] Polyvyanyy, A.—Weske, M.: Flexible process graph: A prologue. In OTM Conferences 1, 2008, pp. 427–435.

[14] Reijers, H.—Rigter, J.—van der Aalst, W.: The Case Handling Case. International Journal of Cooperative Information Systems, Vol. 12, 2003, pp. 365–391.

[15] Sadiq, S. W.—Sadiq, W.—Orlowska, M. E.: Pockets of Flexibility in Workflow Specification. In ER, 2001), pp. 513–526.

[16] Schonenberg, H.—Weber, B.—Dongen, B.—van der Aalst, W.: Supporting Flexible Processes Through Recommendations Based on History. In BPM '08: Proceedings of the 6th International Conference on Business Process Management, Berlin, Heidelberg, 2008, Springer-Verlag, pp. 51–66.

[17] Soffer, P.: On the Notion of Flexibility in Business Processes. In Proceedings of the CAiSE '05 Workshops, 2005, pp. 35–42.

[18] Stoitsev, T.—Scheidl, S.—Spahn, M.: A Framework for Light-Weight Composition and Management of Ad-Hoc Business Processes. In TAMODIA, 2007, pp. 213–226.

[19] TIBCO, Software, and Inc.: Tibco Iprocess Conductor. 2007.

[20] van der Feesten, I. T. P.—Reijers, H. A.—van der Aalst, W. M. P.: Product Based Workflow Support: Dynamic Workflow Execution. In CAiSE, 2008, pp. 571–574.

Thomas Burkhart is a researcher at Institute for Information Systems (IWi) at the German Research Center for Artificial Intelligence (DFKI). In 2008 he received his degree in business administration at the German University in Saarbrücken. His major research fields are business interoperability as well as ERP systems. He is currently working within the European research Project Commius which aims at developing an email based interoperability solution for SMEs.

Christoph Dorn received his M. Soc. Ec. Sc. (Magister) in Business Informatics in 2004, and his Ph. D. in Computer Science (Dr. techn.) in 2009, both from the Vienna University of Technology. Currently he works as a Post-Doc researcher at the Distributed Systems Group. His research interests cover context-aware computing, people-driven ad-hoc workflows, Web service adaptation, social network analysis, and collaborative working environments.

Dirk Werth is Head of Project Group Business Integration Technologies at DFKI and holds leading positions in multiple industrial consulting projects. He started his career as scientific staff member at Saarland University, where he still lectures on Enterprise Resource Planning and Business Integration. He holds diplomas in Business Administration and in Computer Sciences as well as a Ph. D. in Economics. His Ph. D. thesis got the Special Award for Business Process Management from the Federal Association of German Political and Business Economists. His research activities comprise collaborative business processes, business integration and advanced business information systems. He wrote and edited several books and published scientific papers and articles in international journals and proceedings.

Peter Loos is Director of the Institute for Information Systems (IWi) at the German Research Institute for Artificial Intelligence (DFKI) and Head of the Chair for Business Administration and Information Systems at Saarland University. His research activities include business process management, information modelling, enterprise systems, software development as well as implementation of information systems.