Self-adjusting Recommendations for People-Driven Ad-Hoc Processes

Christoph Dorn[1], Thomas Burkhart[2], Dirk Werth[2], and Schahram Dustdar[1]

[1] Distributed Systems Group
Vienna University of Technology
1040 Vienna, Austria
lastname@infosys.tuwien.ac.at
[2] Institute for Information Systems (IWi)
German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
lastname@iwi.dfki.de

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. Effective guidance in ad-hoc workflows requires simultaneous consideration of multiple goals: support of individual work habits, exploration of crowd process knowledge, and automatic adaptation to changes. This paper presents a self-adjusting approach for providing 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 the rapid assembly of value chains where multiple specialized companies cooperate in the production of increasingly complex products. As a direct consequence, established work practises—especially in people driven process environments—need to become flexible and adaptable.

Traditional work-flow engines lack the required flexibility for reacting to ad-hoc changes. Their rigid underlying process model would need to foresee all possible variation, which becomes unfeasible even for simple processes. Support systems for flexible processes (e.g., Caramba [1]) recommend a user to follow a predefined process path, but allow them to deviate from that process on demand (For a exhaustive survey on flexible business support systems see [2]). 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 practises 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.

R. Hull, J. Mendling, and S. Tai (Eds.): BPM 2010, LNCS 6336, pp. 327–342, 2010.

In this paper, we introduce a hybrid approach that combines user-centric process recommendation 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 generally agreed upon process step sequence (*flock*). Monitoring and recommendation evaluation continuously adjusts this classification. The underlying ad-hoc process engine allows any deviations from the modeled flow. These deviations feed back 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 Supporting Flexibility in People-Driven Processes

Based on the continuum given in [3] business processes can be classified as follows:

Structured processes represent the traditional work flows with full automation capacity. Structured process-models determine a-priori the complete process flow, agents, alternative paths etc. and remain unchanged for all process instances.
Semi-structured processes—or case-oriented processes [4]—reside between ad-hoc processes and structured processes. They follow certain rules but cannot be entirely standardized.
Ad-hoc processes represent the most flexible type of processes because their actual execution path is completely defined at run-time with no given structure forcing a certain course of action.

We focus on ad-hoc processes where users are free to decide which process steps to execute. In order to provide recommendations, however, we allow users to model processes which are then refined through user monitoring. We apply the taxonomy by Regev et al. [5] to characterize our supported process flexibility:

Abstraction Level: Runtime changes affect only the process instance. However, these changes eventually cause an evolution of the process model through process learning.
Subject of Change: Process advice supports only behavioral changes, as we recommend only the selection and order of process steps. (Functional, operational, information, and organizational perspectives remain outside the scope of this paper).
Properties of Change: Our main focus lies on process learning which results in small, incremental changes to the process. Revolutionary changes through business process reengineering, however, are also supported. Changes to the process model affect only new

instances (i.e., *deferred* as opposed to *immediate*) and emerge from past user actions in an ad-hoc fashion.

2.1 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 describes the handling of incoming orders and the subsequent dispatching of the ordered items. 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 ordered items. In case the ordered goods are not on stock, the replenishment of the items is triggered. As soon as all required items are gathered, 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 work flow. At first sight, the process does not seem very ad-hoc. The individual workers, however, are free to select 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., important customers demand special treatment) the 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 always the most suitable next steps.

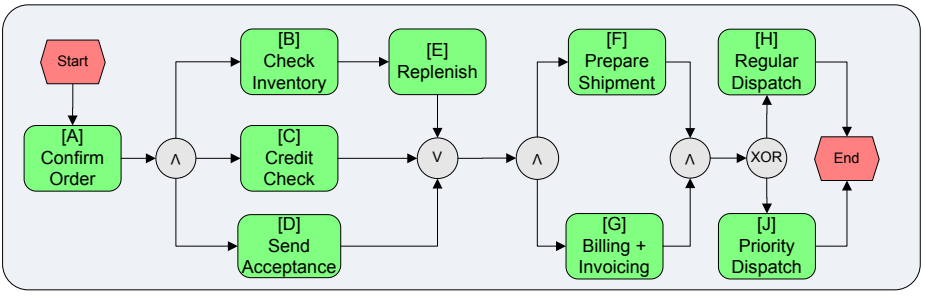

Fig. 1. Scenario: generic people-driven order process model (PM)

3 Related Work

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.

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. [6] 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. [7], 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. [8], on the other hand, describe flexibility as the ability to deal with processes that are only partially defined at build-time. Soffer [9] 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.[11], define strategic flexibility which describes a company's diversity of strategies and its capability to switch between them.

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 exists merely of process parts (like presented by Sadiq et al. [8]) 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 [12]. Moreover, Adams et al. [13] 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 [6]. In addition, recommendations can be derived based on a rough task structure [14]. 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 [15]. Pesic et al. and van der Aalst [16,17] 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 [18] where recommendations are generated based on similar past process executions by considering the specific optimization goals. Another approach is followed by Almeida and Casanova [19] 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. Vanderfeesten et al. [20] follow 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 [21].

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. [2]).

4 Process Recommendation

The recommendation mechanism applies two related data sets, the process model (PM, Figure 1) and the sequence graph (SG). Figure 2a) displays the sequence graph for the first steps of the scenario process. The sequence graph $SG(P, E)$ comprises nodes representing the individual process steps P. An 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 yields the likelihood (i.e., preference) of following a particular path through the process. In Figure 2a, the arc thickness indicates this preference. The flow control model describes the dependency between process steps. Joins, splits and sequential steps are extracted from the sequence graph by existing process techniques [22].

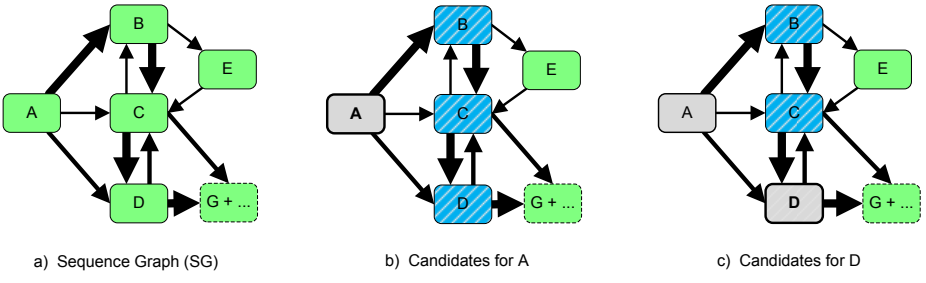

Fig. 2. Sequence Graph (SG) excerpt and process step candidates: completed process steps in dark gray, candidates in shaded blue, and inactive process steps in green. (Colors online)

When a process is started, we derive an instance of the process model. As we track the progress of the process, we utilize the process instance 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* (Fig. 5). 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. After completion of the order confirmation process step A, the *Process Instance Manager* identifies *Check Inventory B*, *Credit Check C*, and *Send Acceptance D* from the PM as valid next steps (Figure 2b). 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, the sum of confidence values will always add up to exactly 100 ($\sum w_i = 100 \; \forall \; r_i \in R$).

Subsequently, the SG weights these candidates according to the edge values (e.g., here $w(AB) = 70, w(AC) = 10$, and $w(AD) = 30$). The first phase ranks all process step candidates that are active and exhibit incoming edges from the recently finished process step (here A) as detailed in lines 1 to 12 in Algorithm 1. Let us assume the user does not follow the top-rated recommendation B and instead selects step D (Figure 2c). Now, simple recommendation on the SG would suggest primarily to continue with the process ($G+$ ...) 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. In short, the sequence graph by itself cannot give recommendations that respect control flow constraints. The flow control model by itself, on the other hand, cannot provide suggestions on the order of which process step to carry out first.

We cannot focus on recommending subsequent process steps only, as the user is free to select any process step. Lines 13 to 26 in Algorithm 1 analyze the process for skipped and out-of-order process steps. Thus, after finishing D, the SG will analyze both C and B. First, we check any preceding steps of D that are still active (i.e., only C) and score them according to outgoing weights (i.e., $w(CD)$) — lines 13 to 18.

Finally, if the SG does not exhibit any edge between the candidates and the last completed step (i.e., B), the candidates are ranked 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 after already completed steps and that are frequently traversed (lines 19 to 26).

The algorithm recommends only the next, active process steps that need completion for sake of simplicity (as opposed to process step sequences). The process model and sequence graph, however, contain the required information to provide also multi-step recommendations. The recommendation model does not support dependencies between process steps explicitly. When the users are aware of such limitations, the process logs and subsequent mined process models will eventually reenforce such dependencies implicitly.

4.1 User-Based Recommendation

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 regular customers that order standard products which get automatically restocked once a certain threshold is undercut. Standard customers receive their goods via regular shipping. Consequently, User 1's personal process model deviates from the standard order process. Figure 3 display PM (a) and SG (b) for User 1. The graphs does not display process steps B, E, and J as they are never invoked. Note that User 1 embraced the habit of always preparing the billing before triggering the shipment, having steps G and F in sequence (Figure 3a). From the sequence graph (Figure 3b) we learn, that User 1 has hardly any preference on whether to execute C or D first.

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

```
1: for all ProcessStep PS ∈ P do                          ▷ Get list of process step candidates.
2:     if state(PS) == active then
3:         R ← PS
4:     end if
5: end for                                                ▷ Initialize process step ranking scores.
6: W ← ∅
7: for all ProcessStep C ∈ R do                           ▷ For all consecutive process steps of S.
8:     W[C] += SG.getEdge(S, C).weight
9: end for
10: if hasActivePredecessors(S) then                      ▷ If a preceding process step has been temporarily
    skipped.
11:     for all ProcessStep C ∈ R do                      ▷ Extract incoming edge weight from SG.
12:         W[C] += SG.getEdge(C, S).weight
13:     end for
14: end if                          ▷ For any other active process step that is not directly connected to S.
15: for all ProcessStep C ∈ R ∧ W[C] == 0 do
16:     for all Arc a ∈ SG.getInEdge(C) do
17:         if state(sourceNode(a)) == completed then     ▷ Count the edge weight only if the
    predecessor step has been completed.
18:             wsum += a.weight
19:         end if
20:     end for
21:     W[C] += w
22: end for                                               ▷ Rank candidates by weights W
23: sort(R, W)
24: return R
```

User 2 serves to premium customers that have a high order volume, pay regularly and thus need not go through a credit check. Premium customers receive priority shipment (J) to deliver their order goods as fast as possible. Similar to User 1, User 2 doesn't check the availability of stock before confirming an order either. Consequently, steps A and D become a sequence as steps B, C, E, and H are missing (Figure 3c+d). This user also exhibits a strong tendency to first trigger shipment preparations (F) and dispatching the goods (J) before preparing invoicing (G). User 3 handles special cases. Being a new employee, he tends to forget certain process steps. Specifically, he never returns order confirmations (D), and occasionally misses the preparation of billing information (G). Consequently, the process extracted from the sequence graph joins steps F and G via an *OR* instead of an *AND* (Figure 3e+f).

Each individual user exhibits a very personalized process that deviates considerably from the standard order process.[1] While personalized recommendation would yield highly relevant process step rankings, these recommendations cannot exploit alternative activities when exceptions such as delayed shipping, or partial order content is

[1] 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.

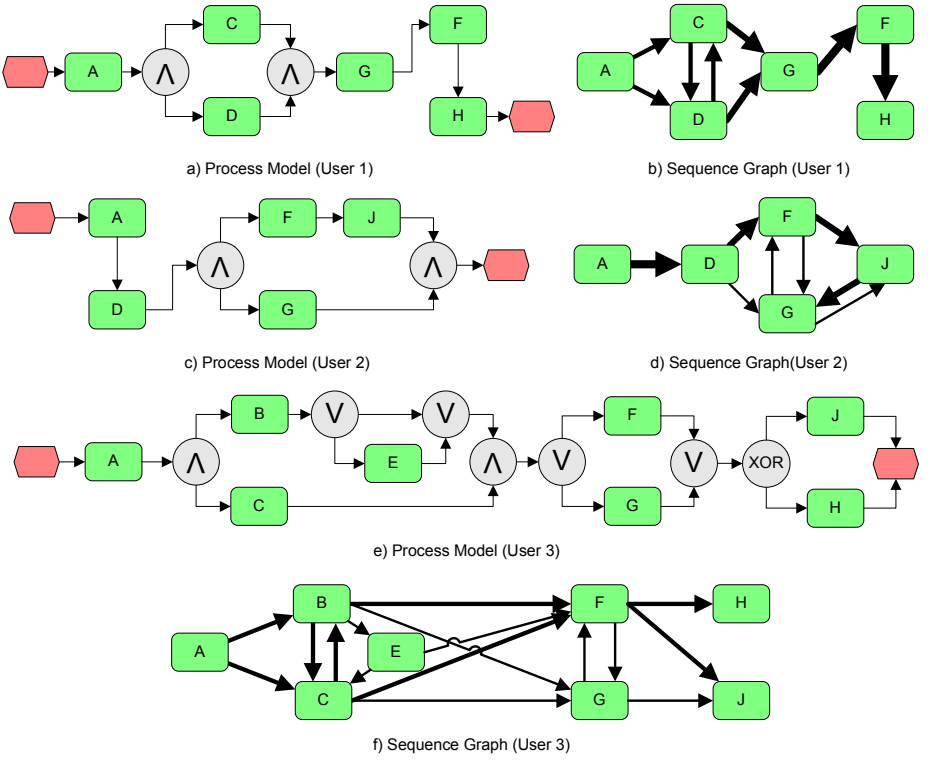

Fig. 3. Process Model and Sequence graph for User 1 (handling standard orders), User 2 (serving premium customers), and User 3 (handling special cases)

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.

4.2 Crowd-Based Recommendation

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 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 4 displays the aggregated flow control model and sequence graph for users 1, 2, and 3. The SG is a simple aggregation of all process step sequence from process instances completed by the three users. The process mining technique reference above then generates the corresponding PM. Note, due to User 3, the PM joins steps *F* and *G* via an *OR*.

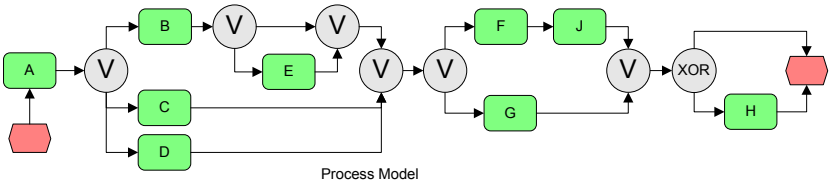

Fig. 4. Aggregated Process Model for User 1, 2, and 3

The complete recommendation cycle is depicted in Figure 5. An incoming request for recommendation triggers the recommendation mechanism (1). The recommender collects information from the process instance manager (2a) and the sequence graph (2b) to aggregate sensible upcoming process steps. The process recommender retrieves this information from both personal and crowd-based SG, respectively PM. The exact aggregation of personal and crowd-based recommendations is outlined in Section 5. The recommender subsequently provides the user the recommended process steps (3).The user selects one process step and enacts it by clicking, for example, on a link in his/her user interface (4). Note that the user doesn't explicitly agree or disagree with a recommendation. Instead, the system monitor observes the user's actions (5a), and other system events (5b) to determine the true process progress. The system monitor updates the process instance manager whenever a process step has been completed (6). The process instance manager in turn updates the personal and crowd-based sequence graph for each completed step (7). In regular intervals, the Process Miner takes a sequence graph (8) and generates an updated process model (9). This procedure is performed for each process type to generate the crowd-based PM and for each individual user and process type to derive the personal PM. 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.

5 Self-adjusting Recommendation Model

The overall recommendation combines user-centric and crowd-based recommendations according to the classifier 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:

$$R_{overall} = \alpha * R_{user} + (1 - \alpha) * R_{crowd} \quad (1)$$

Specifically, we multiply a recommendation's weight 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

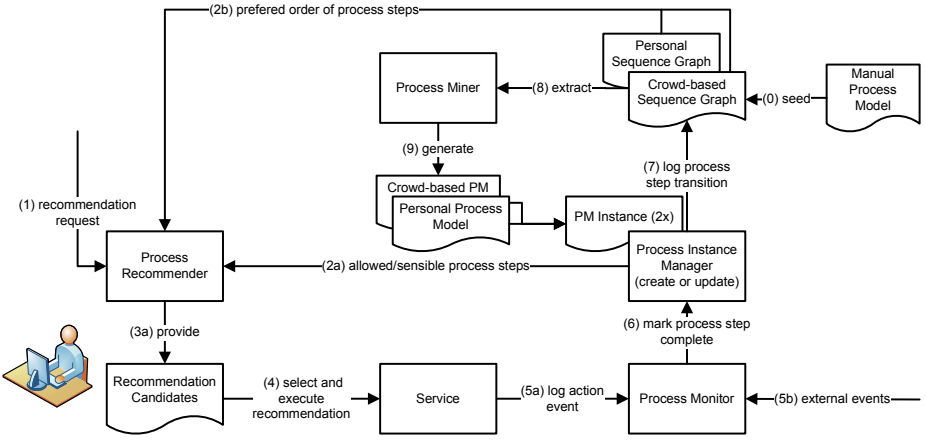

Fig. 5. Feedback cycle for personal and crowd-centric recommendations

obtain following overall recommendation for $\alpha = 0.5$ (Note that the weights for S_3 are aggregated.):

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

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 steps that originated from R_{user}. Similarly, we reduce the value of α when the user follows crowd-based recommendations.

Following factors determine the amount to which α is moved:

- Similarity of user-centric and crowd-based recommendations: We cannot clearly distinguish between distinctive user behavior when both recommendation set contain similar top-rated process steps.
- Process success: When α remains close to 1 but process success declines, we have to assume that the personalized recommendations fail as they most likely reinforce bad decisions. In this case, we need to push α towards the neutral value to introduce again crowd-based recommendations.
- Current value of α: Remaining in the middle between *eagle* and *flock* is not desirable, as we neither provide user-centric know how, nor exploit the wisdom of the crowd. Consequently, moving away from the middle is reinforced. Reaching the absolute extremes, however, is hard to allow for a quick transition between the two behavior types.

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. Subsequently, we first combine the user's actions 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.

$$sim_{wJaccard}(s_1, s_2) = \frac{\sum_{a \in s_1 \cap s_2} w_{IDF}(a)}{\sum_{a \in s_1 \cup s_2} w_{IDF}(a)} \tag{3}$$

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, having the document base comprise all process steps defined in the 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:

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

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

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

Avoiding Classification Lock-in. An *eagle* remains locked-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 average process success rate falls below the average process success rate of the top 50% *flock* users. Specifically, we sort all users 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 he fails to meet following threshold condition:

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

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

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 (see Fig. 7) 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}:

$$\alpha_{t+1} = \begin{cases} Min[Max[(1 + e^{-10*(\alpha_t+\delta(A))+5})^{-1}, 0], 1] & \text{if } \delta(A) \neq 0, \\ \alpha_t & \text{if } \delta(A) = 0. \end{cases} \quad (8)$$

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

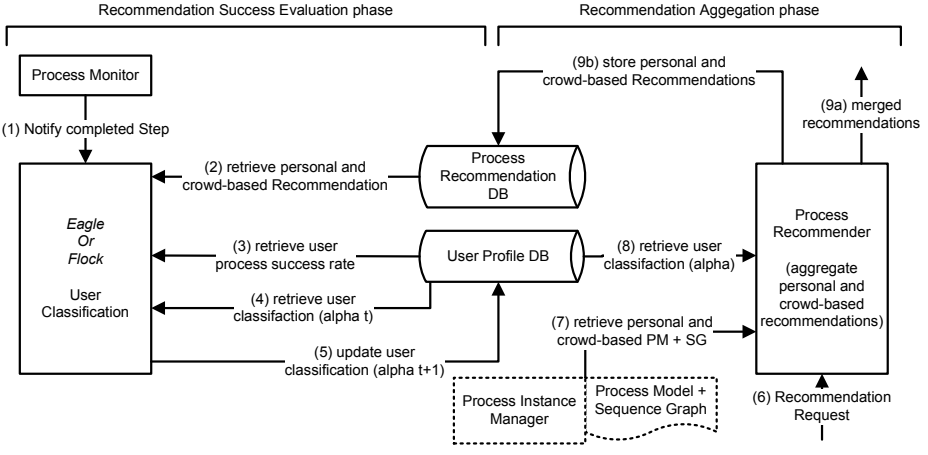

Fig. 6. Self-tuning of classification parameter α based on recommendation success

Classification Self-Tuning Cycle. The complete classification self-tuning cycle consists of the *Recommendation Success Evaluation phase* and the *Recommendation Aggregation phase* (Figure 6). For each completed process step (1) the *User classification* component retrieves the corresponding personal and crowd-based recommendations (2) from the *Process Recommendation DB*. 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 5 — merges personal and crowd-based

recommendations. Upon an incoming recommendation request (1), 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

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 8 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).

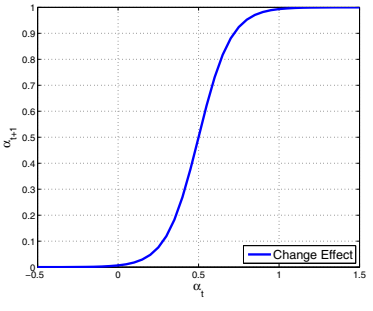

Fig. 7. Sigmoid function: the new user classification (α_{t+1}) depends on the previous classification value (α_t) and the shift towards *eagle* or *flock*

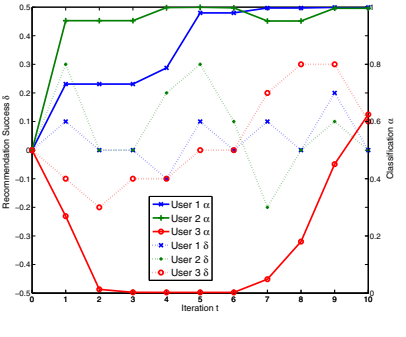

Fig. 8. Classification change α (full lines) and user recommendation success δ (dashed lines) for User 1, 2, and 3 across 10 time intervals

User 1 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. User 3 exhibits a typical learning behavior. As he realizes to execute process step *D*, he strongly deviates from the personal recommendation, thus he becomes a *flock* member (t_1 to t_4) . As his 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 personal sequence preferences start to show effect (t_7 to t_{10}). Multiple, sequential recommendation evaluations towards *eagle* ($\delta > 0$) cause his reclassification. As users learn and adapt their behavior, new flow control structures emerge from the crowd

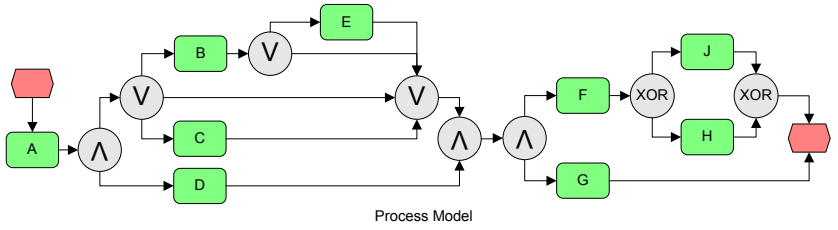

Fig. 9. Evolved Process Model and Sequence Graph for User 1, 2, and 3

sequence graph. Once User 3 apprehends to always send an order confirmation, the evolved crowd-based PM (Figure 9) identifies process step *D* as mandatory (and no longer optional).

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 10) to define a coarse-grained structure of the desired process. Within Commius, we allow only a simple sequential structure to enable process modeling also for non-experts. The refined process model is later 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 [23]. Figure 10 (inset) displays an example email enriched with process recommendations. The process step sequence with highest probability is provided on the right side (here four subsequent process steps taken from the scenario). 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 to prepare an 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.

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.

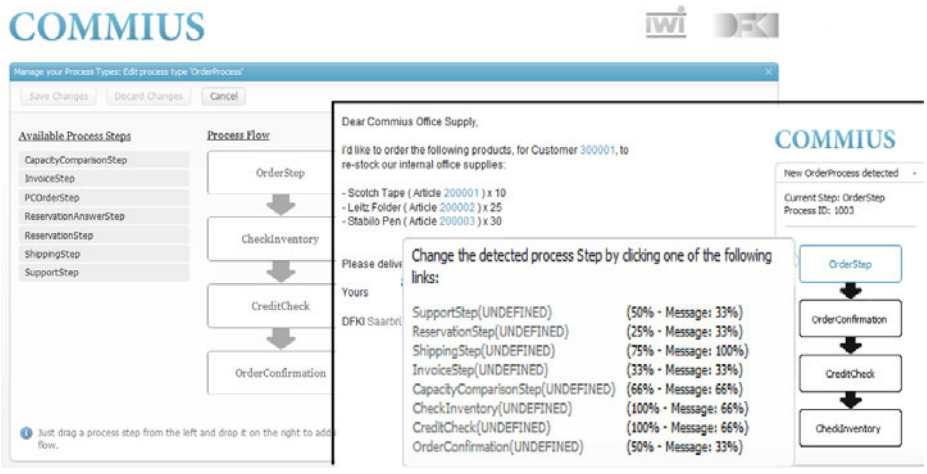

Fig. 10. Commius process modeling tool and email-based process step recommendation (inset)

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. Dustdar, S.: Caramba Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Team. Distributed Parallel Databases 15(1), 45–66 (2004)
2. Burkhart, T., Loos, P.: Flexible business processes - evaluation of current approaches. In: Proceedings of Multikonferenz Wirtschaftsinformatik - MKWI 2010 (2010)

3. Huth, C., Erdmann, I., Nastansky, L.: Groupprocess: Using process knowledge from the participative design and practical operation of ad hoc processes for the design of structured workflows. In: HICSS (2001)
4. Dellen, B., Maurer, F., Pews, G.: Knowledge based techniques to increase the flexibility of workflow management. In: Data and Knowledge Engineering.North-Holland, Amsterdam (1997)
5. Regev, G., Soffer, P., Schmidt, R.: Taxonomy of flexibility in business processes. In: BPMDS (2006)
6. Adams, M., Hofstede, A., Edmond, D., van der Aalst, W.: Facilitating flexibility and dynamic exception handling in workflows through worklets. In: CAiSE 2005, pp. 45–50 (2005)
7. Reijers, H., Rigter, J., Aalst, W.V.D.: The case handling case. International Journal of Cooperative Information Systems 12, 365–391 (2003)
8. Sadiq, S.W., Sadiq, W., Orlowska, M.E.: Pockets of flexibility in workflow specification. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) ER 2001. LNCS, vol. 2224, pp. 513–526. Springer, Heidelberg (2001)
9. Soffer, P.: On the notion of flexibility in business processes. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS. Springer, Heidelberg (2005)
10. Müller, R., Greiner, U., Rahm, E.: Agent work: a workflow system supporting rule-based workflow adaptation. Data Knowl. Eng. 51(2), 223–256 (2004)
11. Adamides, E.D., Stamboulis, Y., Pomonis, N.: Modularity and strategic flexibility: a cognitive and dynamic perspective. In: Systems Dynamics Society Conference 2005 (2005)
12. Polyvyanyy, A., Weske, M.: Flexible process graph: A prologue. In: OTM Conferences, vol. (1), pp. 427–435 (2008)
13. 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, pp. 1836–1852 (2003)
14. Eichholz, C., Dittmar, A., Forbrig, P.: Using task modelling concepts for achieving adaptive workflows. In: EHCI/DS-VIS, pp. 96–111 (2004)
15. Stoitsev, T., Scheidl, S., Spahn, M.: A framework for light-weight composition and management of ad-hoc business processes. In: Winckler, M., Johnson, H., Palanque, P. (eds.) TAMODIA 2007. LNCS, vol. 4849, pp. 213–226. Springer, Heidelberg (2007)
16. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Business Process Management Workshops, pp. 169–180 (2006)
17. Pesic, M., Schonenberg, M.H., Sidorova, N., van der Aalst, W.M.P.: Constraint-based workflow models: Change made easy. In: OTM Conferences, vol. (1), pp. 77–94 (2007)
18. Schonenberg, H., Weber, B., Dongen, B., Aalst, W.: Supporting flexible processes through recommendations based on history. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 51–66. Springer, Heidelberg (2008)
19. 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)
20. Vanderfeesten, I.T.P., Reijers, H.A., van der Aalst, W.M.P.: Product based workflow support: Dynamic workflow execution. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 571–574. Springer, Heidelberg (2008)
21. TIBCO, Software, Inc.: Tibco iprocess conductor (2007)
22. Gaaloul, W., Baïna, K., Godart, C.: Log-based mining techniques applied to web service composition reengineering. Service Oriented Computing and Applications 2(2-3), 93–110 (2008)
23. Burkhart, T., Werth, D., Loos, P.: ”Commius An Email Based Interoperability Solution Tailored For SMEs. Journal Of Digital Information Management 6 (2008)