Advanced Social Features in a Recommendation System for Process Modeling

Agnes Koschmider[1], Minseok Song[2], and Hajo A. Reijers[2]

[1] Institute of Applied Informatics and Formal Description Methods
Universität Karlsruhe (TH), Germany
koschmider@aifb.uni-karlsruhe.de
[2] School of Industrial Engineering
Eindhoven University of Technology, The Netherlands
{m.s.song,h.a.reijers}@tue.nl

Abstract. Social software is known to stimulate the exchange and sharing of information among peers. This paper describes how an existing system that supports process builders in completing a business process can be enhanced with various social features. In that way, it is easier for process modeler to become aware of new related content. They can use that content to create, update or extend process models that they are building themselves. The proposed way of achieving this is to allow users to generate and modify personalized views on the social networks they are part of. Furthermore, this paper describes mechanisms for propagating relevant changes between peers in such social networks. The presented work is particularly relevant in the context of enterprises that have already built large repositories of process models.

Keywords: Social Networks, Business Processes, Modeling Support, Personalization.

1 Introduction

Social networking sites such as `Flickr` or `Facebook` enjoys great popularity because they enable to connect and share information with each other and thus support to build online communities of people who have similar interests, or who are interested in exploring the interests of others [1].

An activity that may profit from a transfer of social networking features is *business process modeling*. Earlier, we developed a support system for people involved in this act [2]. The system takes into account a process builder's modeling intention and patterns observed in other users' preferences and uses a repository of process model parts to recommend her efficient ways to complete the model under construction. Recently, we extended this system with some social networking features [3].

Through this extension, process builders can establish who already selected and reused specific process models. Novice process builders in particular can profit from this, as the system encourages user trust and participation. To implement these features, three kinds of social networks have been implemented

W. Abramowicz (Ed.): BIS 2009, LNBIP 21, pp. 109–120, 2009.

in the system: (1) a social network from a *process model repository*, (2) a social network from a *user history* and (3) a social network from an *insertion history*. The social network from the *process model repository* provides an organizational view of business processes. The social network from *user history* shows the relationship among modelers who use the recommendation system. Finally, the social network from *insertion history* shows the relationship among modelers who decided for equal recommendations.

Over the past months, we evaluated the 'socially enhanced' recommendation system in practice, most notably through an experiment at the University of Karlsruhe that involved 30 novice process modelers. These modelers had to build a fairly large process model for which they had all the features of the recommendation system at their disposal, including information on previous usage of process model parts by people being socially close to them. We came to the conclusion that the new social features are not sufficient to propagate trust and thereby to influence novice modelers in their choice of process modeling parts. The main reason we identified is that the current system does not consider *personalized* interests and knowledge of the modelers. In particular, a user can not include her capabilities or working environment characteristics in her network, which makes it difficult for other modelers to evaluate the relevance of her modeling actions. Additionally, we realized that process builders may have an ongoing interest in notifications of changes in process models that are relevant in their social network, i.e. receive and send updates on process models beyond the exact times that they are actually busy modeling.

Against this background, this paper deals with two types of extensions of the social features of our recommendation system:

1. The *personalization of social networks* will stimulate the development of communities of process builders that share the same interests or knowledge, and activate new ways of collaborating. An example scenario would be a group of process modelers that work in different geographical units of a multinational organization, but through the recommendation system work on the modeling of similar procedures in their respective units.
2. The *propagation of changes* will allow users to become aware of potentially relevant updates to the process models they are building (of have built) themselves. Conversely, process builders can actively inform other process builders about relevant changes in process models. In this way, enhancements of process models, e.g. in response to changing legislation of market conditions, will not need to be re-invented over and over again.

To explain how the described extensions are achieved, the following section provides an overview of our metadata model, which lays the foundation for personalized social networks. In Section 3, a method for personalizing the three kinds of social networks will be presented. In Section 4 two algorithms will be presented supporting the propagation of changes within such networks. Section 5 gives an overview on related work. Finally, our paper concludes with a discussion of our approach and an outlook on future research.

2 Metadata Model

To personalize the three kinds of social networks sketched above (social network from a *process model repository*, social network from a *user history* and social network from an *insertion history*) it is necessary to build a metadata model, which describes the skills, and knowledge of users. Based on this metadata model the recommendation system can suggest additional connections to process builders on the basis of common skills. Generally, the interest or the knowledge of process builders is driven by four properties. They are *subject domain*, *capability*, *work environment*, and *individual and group behavior*. Note that users can select predefined properties for the first three properties. Individual and group behavior patterns can be derived from the user history. More specificly the properties are defined as follows:

1. *subject domain* describes the topic the user is working on or has interest (e.g., engineering, medicine, government or manufacture). Process models are usually classified based on their domain and process builders usually have their own subject domains. For example, a process builder who has some previous experience on governmental processes can more easily understand similar processes in government settings. Thus, when a process builder is working on a specific domain, she can consult with the people who have an expertise in it.
2. *capabilities* of process builders include general literacy and technical skills. Also expertise or special skills of users can be described by this property. For example, if a process builder is modeling a quality management process in a software company, people who have deep knowledge of software engineering can help her.
3. *work environment* describes the department/project the user is involved in. The work environment is the functional department or project team to which a process builder belongs. Such properties can normally be derived easily from general users' profiles. And even when a new process builder is added into a social network, those properties can be easily derived from a past track record.
4. *individual and group behavior* patterns and history describes the degree of sociality of users. Note that this property is normally not specified when a process builder becomes active for first time. It is based on the process model design history in the recommendation based support system.

Table 1 shows an example of the user history generated from the modeling history of the community of users. In the table, each row refers to a user (u) and a column corresponds to a process model (p) in the repository. Also, each cell (c_{ij}) shows the number of use of the process model (p_j) by the user (u_i). From this kind of table, we can derive *individual and group behavior patterns* [3]. For example, by applying data mining techniques (e.g. K-means clustering), we can extract several groups in which people have a similar behavior pattern or similar preferences. Arguably, the people who have a similar pattern can provide

Table 1. User History

	P_1	P_2	P_3	P_4	P_5	...	P_N
$user_1$	2	1	0	0	0	..	2
$user_2$	2	1	0	0	0	..	2
:	:	:	:	:	:	:	:
$user_M$	0	0	1	0	0	..	0

more useful information for a process builder. From the above definitions, the metadata model can be defined as follows.

Definition 2.1 (Metadata). A metadata, M, is a 9-tuple $(N, D, C, W, B, P_D, P_C, P_W, P_B)$ where

(i) N is a set of users,
(ii) D is a set of subject domain,
(iii) C is a set of capabilities,
(iv) W is a set of work environment,
(vi) B is a set of behavior patters,
(vii) P_D, P_C, P_W, and P_B is a set of profiles, where $P_D \subseteq (N \times D)$, $P_C \subseteq (N \times C)$, $P_W \subseteq (N \times W)$, $P_B \subseteq (N \times B)$.

For convenience, we define an operation on profiles: $\pi_N(p_D) = n$, $\pi_D(p_D) = d$ for some profile $p_D = (n, d) \in P_D$. We also use $\pi_C(p_C), \pi_W(p_W), \pi_B(p_B)$ in the same manner. For example, there are two users ($user_1$,$user_2$). $User_1$ has a specialism in "government" processes, has a capability of "process design", and belongs to the "IT dept". $User_2$ is working on processes in a "bank", is able to do "programming", and works at the "CS dept". Their metadata is defined as follows.

$N = \{$"$user_1$", "$user_2$"$\}$,
$D = \{$"*government*", "*bank*"$\}$,
$C = \{$"*process design*", "*programming*"$\}$,
$W = \{$"*IT dept*", "*CS dept*"$\}$,
$P_D = \{$("*user*1", "*government*"), ("*user*2", "*bank*")$\}$,
$P_C = \{$("*user*1", "*process design*"), ("*user*2", "*programming*")$\}$,
$P_W = \{$("*user*1", "*IT dept*"), ("*user*2", "*CS dept*")$\}$

3 Personalized Social Networks

The metadata model builds upon the three kinds of social networks as mentioned above. We regard these social networks as public, in the sense that any process builder using the recommendation system can access these networks. However, since the potentially huge size of such networks and the risk of information overflow it will be difficult for process builders to retrieve the proper content. Therefore, *personalized* social networks are needed. In this section, we explain how to generate such personalized social networks from the public social networks. First of all, we define a social network as follows.

Definition 3.1 (Social Network). A social network, S, is a 3-tuple (N, A, σ) where

(i) N is a set of nodes,
(ii) $A \subseteq (N \times N)$ is a set of arcs,
(iii) $\sigma : A \rightarrow \mathbb{R}$, is a weight function of an arc.

N refers to a set of users, A shows the relationship between users, and σ is a function indicating the weight of each arc.

To reduce the size of the network, a process builder can consider two kinds of information. Firstly, she can take into account people who she already knows well. In this case, she does not need any metadata of the people, but the name (or ID) of the people would be sufficient to reduce the size of the network. Secondly, she can utilize metadata of people. Based on her own interest, a user can remove some uninteresting nodes from the network and generate a personalized network. To do this, we define a filtering function.

Definition 3.2 (Filtering Function). A filtering function, $F : (S, M, N_1, P_{D1}, P_{C1}, P_{W1}, P_{B1}) \rightarrow S$ where $N_1 \subseteq N, P_{D1} \subseteq P_D, P_{C1} \subseteq P_C, P_{W1} \subseteq P_W, P_{B1} \subseteq P_B$, is defined as follows: $F(S, M, N_1, D_1, C_1, W_1, B_1) = \{(n, a, \sigma) | (n, a, \sigma) \in S \wedge (p_D, p_C, p_W, p_B) \in M \wedge \pi_N(p_D) = n \wedge \pi_N(p_C) = n \wedge \pi_N(p_W) = n \wedge \pi_N(p_B) = n \wedge n \in N_1 \wedge \pi_D(p_D) \in P_{D1} \wedge \pi_C(p_C) \in P_{C1} \wedge \pi_W(p_W) \in P_{W1} \wedge \pi_B(p_B) \in P_{B1}\}$

A public social network (S) and the original metadata (M) are given as inputs of the filtering function. In addition, the function requires filtering options. In the formula, N_1 represents a set of process builders who will be included in the personalized network. It enables a user to include people in whom she is interested in. A user can also utilize metadata. P_{D1}, P_{C1}, P_{W1}, and P_{B1} refer to a set of *subject domain*, *capability*, *work environment*, and *behavior pattern* respectively. They cover metadata which a user wants to contain in the personal network. As a result, the function returns a network in which irrelevant nodes are removed from the original network, such that it can be used as a personalized social network.

Figure 1 shows an example of such personalization. Figure 1(a) shows a public social network which contains 10 users. Table 2 shows a fragment of the metadata of the users. For example, $user_6$ has "government" as a subject domain and her capacity is "programming". She works at "IT dept." and her behavior pattern is not specified. If a user wants to personalize the public network including $user_5$, $user_6$, $user_9$ and $user_{10}$, "government" as a subject domain, and "IT dept." as a work environment, she can use F(SN,M,{$user_5$, $user_6$, $user_9$, $user_{10}$}, {"government"},{*},{"IT dept."}, {*}) and derive the personalized social network in Figure 1(b).

4 Propagation of Changes in Public and Personalized Social Networks

The public and the personalized social networks in Figure 1 are subject to contant dynamic modifications, for example because a user will select new process

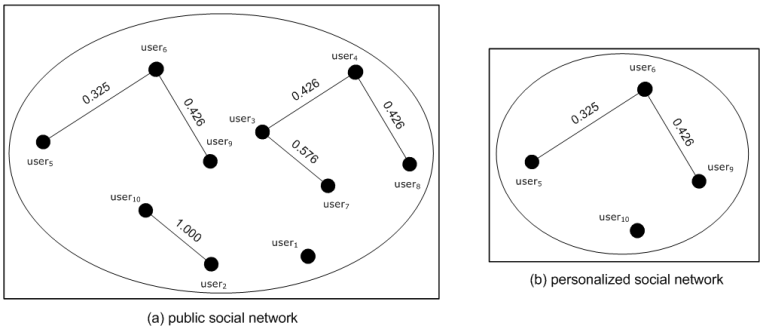

Fig. 1. Social networks

Table 2. Metadata of users

	subject domain	*capability*	*work environment*	*behavior pattern*
:	:	:	:	:
$user_5$	{government, bank}	{process design}	{IT dept.}	{}
$user_6$	{government}	{programming}	{IT dept.}	{}
$user_7$	{bank}	{accounting}	{IT dept.}	{}
:	:	:	:	:
$user_9$	{government}	{accounting}	{IT dept.}	{}
$user_{10}$	{government}	{process design}	{IT dept.}	{}

models in her job. (Note that social networks can change due to structural updates, e.g., when somebody moves from Marketing to the IT department, but these changes are rather straightforward and will not be considered in this paper.) Assume *user 10* starts selecting the same process models as *user 6*. Consequently, over time *user 10* will become connected to *user 7* and thus emulate recommendation patterns of *user 10*. In practice, we realize that process builders want to be informed if they are affected by modifications and users want to notify others if adaptation in their models are required.

We have implemented such change propagations with *pull* and *push* services. *Push* services in the recommendation system involve actively sending (or pushing) information to a specific process builder that the process builder knows to be interested in this info. This requires an active participation of users. In contrast are the *pull* services, which are more passive then the *push* service. A *pull* service involves process builders that specified that they want to receive information if a certain process model has been changed. Pull services will be explained in Section 4.1 and push services in Section 4.2.

4.1 Pull Service

A pull service can be implemented for the propagation of modifications in process models, in model parts or of process elements. In order to consider these three applications of propagation we initially differentiate the types of changes, which

may occur in process models, model parts and for process elements. Changes can be of a primitive or of high-level nature, as defined in [4]. High-level changes in our context affect process model parts and are the following:

1. *insert process model part*: a completely new process model part is inserted in a business process model, which has been created by a previous user,
2. *delete process model part*: a process model part has been completely deleted in a business process model.
3. *move process model part*: a process model part has been moved in a business process model.

Primitive changes affect process elements and process models, and pertain to the following: 1)*insert* completely new node and 2)*delete*, 3)*rename* or 4)*move* a node in a certain process model, which is already stored in a process repository.

Primitive and high-level changes affect public and personalized social networks if process builders prefer process models of other and thus emulate their recommendation patterns. The consequence of this amalgamation is that users will be connected to other process builders and thus the social network structure will change. To notify process builders of changes in process models that are relevant in their social network the recommendation system incorporates two algorithms for the calculation of primitive and high-level changes.

To calculate the number of primitive changes in a process model we use the following data structure. We define a class *Node*[1] with the following five attributes, that describe the state of a node: (1) *bInitial* of type *boolean*. If *bInitial* is *false* then a node has been newly inserted. Else no change has been performed, (2) static final *MOVED* of type *int*. This state is valid, if a node has been moved, (3) static final *RENAMED* of type *int*. This state is valid, if a node has been renamed, (4) static final *DELETED* of type int. This state is valid, if a node has been deleted.

To query the state of a node we use the variable *iState* of type *int*. The variable *iCounter* of type *int* is used to count the number of states of a node. The *list nodeList* is used to store all nodes of a business process model. With this data structure we can define an algorithm calculating the number of primitive changes as described in algorithm 1.

This algorithm differentiates between three cases. Case 1 (line 4) is valid, if all elements of a process model stored by a process builder in the repository have been reused by another user without any change. Case 2 (line 7) is valid, if the state of a process model stored in the process repository has been modified by moving, renaming or deleting process elements. In case 3 (line 10) the algorithm counts the number of newly inserted elements and the number of movements, renamings, and deletions of initial available process elements. If the user is interested in the modification of nodes of a specific process model then we prepend to line 1 an if-clause that checks a certain process model name.

[1] The business processes and parts stored in the repository have been modeled with Petri nets. Thus, in this context a node can be a place or a transition.

Algorithm 1. Algorithm to calculate the number of primitive changes

```
1. int iCounter;
2. int iState;
3. for all Node n in nodeList do
4.    if ((n.bInitial) && (n.iState & Node.MOVED == 0) && (n.iState &
      Node.RENAMED == 0)&& (n.iState & Node.DELETED == 0)) then
5.       iCounter++;
6.    end if
7.    if ((n.bInitial) && ((n.iState & Node.MOVED == Node.MOVED) || (n.iState
      & Node.RENAMED == Node.RENAMED) || (n.iState & Node.DELETED ==
      Node.DELETED))) then
8.       iCounter++;
9.    end if
10.   if ((n.bInitial == false) && ((n.iState & Node.MOVED == Node.MOVED)
      || (n.iState & Node.RENAMED == Node.RENAMED) || (n.iState &
      Node.DELETED == Node.DELETED))) then
11.      iCounter++;
12.   end if
13. end for
```

Analogous, we can calculate the number of high-level changes. For this, we define a class *ProcessPart* with attributes that describe the state of a process model part (*bInitialPart* of type *boolean*, *DELETEDPart* of type *int* and *MOVEDPart* of type *int*). For instance, if *bInitialPart* is *false* then a new process model part has been inserted in the process model. Else no change has been performed.

To query the state of a process model part one can use the variable *iStatePart* of type *int*. The *list processpartList* is used to store all process model parts of a business process model. The algorithm for process model parts can be structured in the same way as in Algorithm 1. In the algorithm for calculating the number of high-level changes, case 1 would be valid, if a whole business process model has been reused by another user without any change. Case 2 would be valid, if the state of a process model part stored in the process repository has been modified by moving or deleting a whole process model part. In Case 3 the algorithm (to calculate the number of high-level changes) counts the number of newly inserted process model parts and the number of movements or deletions of whole process model part being initiate available in the process model.

Based on these two algorithms for primitive and high-level changes we can provide process builders with an alert function. The capability of this function is to inform users when specific changes have been performed. If a process builder subscribes to the alert function then she will receive reliable information about primitive, high-level changes and newly stored process models.

Figure 2a shows the interface of the alert function of the social networks enhanced recommendation system. The process builder can decide if she is interested in an alert on high-level or primitive changes or on newly inserted process model (parts) in the repository. The user can apply the alert function for all process models, respectively parts. Process models and model parts are listed

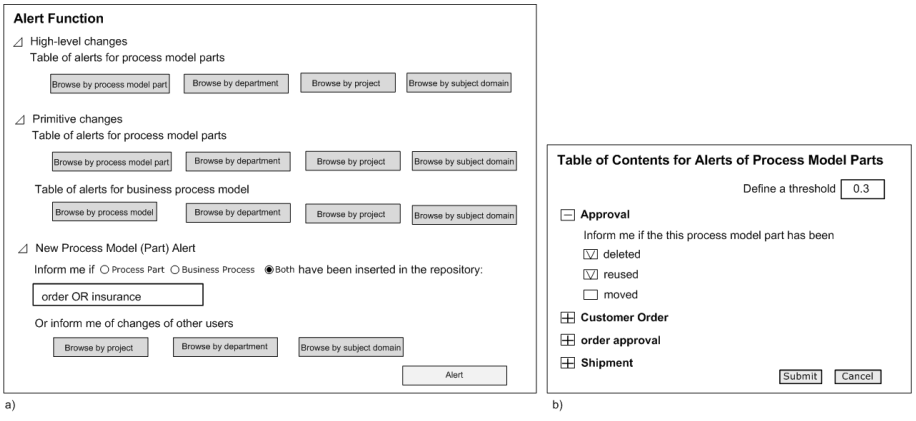

Fig. 2. Table of Contents of Pull Services

alphabetically, or in case that process builders are working in organizational boundaries they can be listed by departments, projects or subject domains.

The alert function on new process model (parts) can be activated when filling in keywords in a text field or when browsing by the project, the department or the subject domain. Then the system will automatically send a notice without a specific request from the subscriber if process builders with specific metadata have inserted a process model matching the user's keywords. In Figure 2a the user wants to be informed if process models and parts regarding *order* or *insurance* have been stored in the repository.

To determine relevant process models and parts we use the keyword respectively tag extraction method explained in [2]. Initially, we extract the words and remove common stop words, which yields the set t_{raw}. The remaining tag candidates t_{raw} are then expanded with their related synonym sets, which are determined via WordNet[2], resulting in the set t_{query}. The relevant process models and parts are then determined by querying an index, where the query term is the concatenation of all tags in the set t_{query}. Whenever new models are stored in the repository the system poses the query t_{query} and notifies the process builder if a new process model matching the process builder's criteria has been stored.

If process builders decide to be informed about primitive changes in process models then they have to specify a threshold, i.e., a value that indicates a degree of significant changes in a process model. In Figure 2b the user indicated a threshold of 0.3., that means that primitive changes should be considered if 30% of elements in the process model *Process for checking orders* have been deleted or renamed. A threshold of 0.3 tends to be coherent. If a process builder wants to be informed about high-level changes, a threshold like this will not be necessary but the rest of the interface will be similar.

[2] http://wordnet.princeton.edu/

4.2 Push Service

The push service of the recommendation system allows process builders to notify other process builders when changes have been performed or relevant content has been stored in the repository. Figure 3 shows the interface for the push service. The process builder can either push the information to her public social networks, to process builders belonging to a subject domain, a work environment or even to specific process builders (e.g., friends). We assume that the number of possible

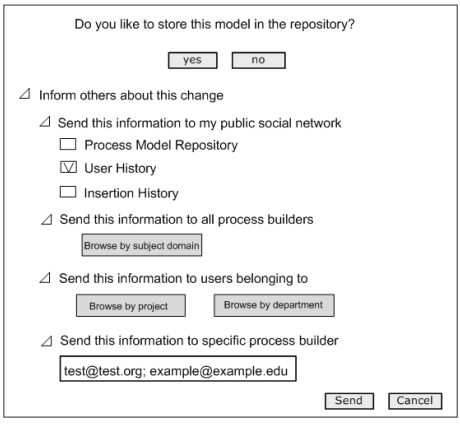

Fig. 3. Table of Contents of the Push Service

domains and work environments is usually fixed within a company and hence we can provide the process builder with an interface where she can choose from a list of subject domains and work environments to whom the information should be forwarded. Whenever the process builder stores her model in the repository she can activate the push service for this model.

5 Related Work

The initial idea of a recommendation-based process modeling support system has been described in [2]. An extension of the system is presented in [5], which considers the modeling perspective when suggesting appropriate recommendations. This paper extend the previous works in the perspective of social networks.

In the field of social network, privacy has been studied in several approaches [6,7]. [6] proposes a model of privacy for reconstructing a graph privately; the approach does focus more on security aspects and uses a completely different metamodel than the one presented in this paper. The approach in [7] could be used in our scenario in order to join two (related) personalized social networks. Such a join could be regarded as an extension of our approach, which would provide synergy effects for process builders. Regarding the searching of expertise

in social networks, the set of approaches found in the literature [8,9] follow a different metamodel and use different kinds of social networks. E.g., [9] retrieves the expertise through out semi-structured content. [8] e.g., uses organization's email data set. The work in [10] is the first approach to apply social networks in the BPM area. However, their focus is not on process modeling, but on the analysis of interpersonal relationships in an organization.

Push service and *pull service* are related to *alert service* and *propagation*. Alert services are particularly known for digital libraries [11] where users want to be notified whenever a new publication in a specific research area is published. In our context, we use the alert service to notify process builders about changes in process models and thus we use a different basis for our algorithms. Propagation in networks have been suggested for innovations [12] or effects of a change in technology [13]. These approaches have a different focus than the one presented in this paper. Approaches presented for pull and push technologies on the web [14] or for mobile devices [15] can not be applied in our context.

6 Conclusion and Reflection

The contribution of this paper is that it presents new social features of a recommendation system for process builders. Specifically, user-specific information can be added to member profiles so that modelers can build personalized views on their social networks. Furthermore, facilities are described to propagate relevant changes through such social networks. Both types of facilities are expected to increase the 'sociability' of that recommendation system, such that process builders can more easily leverage the modeling efforts by their peers.

Currently, we are implementing the described features into our recommendation system and plan to initiate a new round of practical evaluations under realistic process modeling conditions. It is our belief, cf. [16], that only through integrating both design and evaluation activities it is possible to develop our ideas towards a truly supportive recommendation system for process builders.

There is notable increase of enterprises that possess large repositories of process models which they manage and extend as valuable assets, see e.g. [17]. While much of the current research on process modeling focuses on issues with a single process model, the demand will rise for insights on how to extend, maintain, and disclose such large sets of process models. We hope that our work will contribute to the development of systems that can deal with these needs.

References

1. Wasserman, S., Faust, K., Iacobucci, D., Granovetter, M.: Social Network Analysis: Methods and Applications. Cambridge University Press, Cambridge (1994)
2. Hornung, T., Koschmider, A., Lausen, G.: Recommendation based process modeling support: Method and user experience. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 265–278. Springer, Heidelberg (2008)

3. Koschmider, A., Song, M., Reijers, H.A.: Social Software for Modeling Business Processes. In: Ardagna, D., et al. (eds.) BPM 2008 Workshops. LNBIP, vol. 17, pp. 666–677. Springer, Heidelberg (2009)
4. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features - enhancing flexibility in process-aware information systems. Data & Knowledge Engineering 66, 438–466 (2008)
5. Koschmider, A., Habryn, F., Gottschalk, F.: Real support for perspective-compliant business process design. In: Ardagna, D., et al. (eds.) BPM 2008 Workshops. LNBIP, vol. 17, pp. 32–43. Springer, Heidelberg (2009)
6. Frikken, K.B., Golle, P.: Private social network analysis: how to assemble pieces of a graph privately. In: Proceedings of the 5th ACM workshop on Privacy in electronic society, pp. 89–98. ACM, New York (2006)
7. Brickell, J., Shmatikov, V.: Privacy-preserving graph algorithms in the semi-honest model. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 236–252. Springer, Heidelberg (2005)
8. Zhang, J., Ackerman, M.S.: Searching for expertise in social networks: a simulation of potential strategies. In: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work, pp. 71–80. ACM, New York (2005)
9. Tsai, T.M., Shih, S., Stu, J., Wang, W.N., Chou, S.C.T.: Beyond web-log: Transform blog into personal expertise and social network via myfoaf support. In: Workshop on Social Web Search and Mining (2008)
10. van der Aalst, W.M.P., Reijers, H.A., Song, M.: Discovering social networks from event logs. Computer Supported Cooperative Work 14, 549–593 (2005)
11. Faensen, D., Faulstich, L., Schweppe, H., Hinze, A., Steidinger, A.: Hermes: A notification service for digital libraries. In: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries, pp. 373–380 (2001)
12. Mason, W.A., Jones, A., Goldstone, R.L.: Propagation of innovations in networked groups. Journal of Experimental Psychology 137, 233–422 (2008)
13. Burkhardt, M.E., Brass, D.J.: Changing patterns or patterns of change: The effect of a change in technology on social network structure and power. Administrative Science Quarterly 35, 104–127 (1990)
14. Kendall, J.E., Kendall, K.E.: Information delivery systems: an exploration of web pull and push technologies. Communications of the Association for Information Systems 1 (1999)
15. Qaddour, J.: Wap and push technology integrated into mobile commerce applications. In: Proceedings of the IEEE International Conference on Computer Systems and Applications, Washington, DC, USA, pp. 779–785. IEEE Computer Society, Los Alamitos (2006)
16. Hevner, A., March, S., Park, J., Ram, S.: Design Science in Information Systems Research. Management Information Systems Quarterly 28, 75–106 (2004)
17. Reijers, H., Mans, R., van der Toorn, R.: Improved Model Management with Aggregated Business Process Models. Data & Knowledge Engineering 68, 221–243 (2009)