FlowWiki: A Wiki Based Platform for Ad-hoc Collaborative Workflows☆

Jae-Yoon Jung[a], Kwanho Kim[b], Dongmin Shin[c], Jonghun Park[b,*]

[a]Kyung Hee University, Yongin, Gyunggi, 446-701, Korea

[b]Seoul National University, Seoul, 151-744, Korea

[c]Hanyang University, Ansan, Gyunggi, 425-791, Korea

Abstract

Traditional workflow management systems provide rich capabilities for designing, executing, and monitoring well-defined collaborative processes. Yet, for many occasions of collaboration, we do not often have sufficient information about who will participate, what activities people will carry out, and how the entire workflow will change. Accordingly, the problem of managing flexible workflows has been receiving increasing attention during the last decade. This paper presents a novel approach by which collaborative workflows can be configured independently as needed by participants and managed in an ad-hoc way. Following the emerging paradigm of collective intelligence, the proposed platform, named FlowWiki, provides a set of useful mechanisms to enable dynamic collaborations without requiring prescribed collaboration model. FlowWiki is an extension of conventional wiki system, and it aims for flexibly managing collaborative workflows by allowing on-demand workflow configuration and event-driven interactions.

Keywords: Wiki; Ad-hoc collaboration; Collaborative workflows; Emergent collaboration

---

☆ This work was supported mainly by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST) (No. R01-2007-000-11167-0), and partly by Engineering Research Institute at Seoul National University.

* Corresponding author. Tel.: +82-2-880-7174; fax: +82-2-889-8560.

E-mail address: jonghun@snu.ac.kr (J. Park).

Mailing address: Dept. of Industrial Engineering, Seoul National University, San 56-1, Silim-Dong, Gwanak-Gu, Seoul, 151-744, Korea.

Published in Knowledge-Based Systems, Vol. 24, No. 1, Feb 2011, pp. 154-165.

1. Introduction

Workflow management systems (WfMSs) have achieved significant commercial success particularly in supporting collaboration by automating the execution and monitoring of routine business processes. A variety of commercial WfMSs are currently available, and they typically include integrated process modeling tools, external integration functionalities, sophisticated user interfaces, and powerful analytics [1,17].

In spite of these benefits, incorporation of real world collaborative processes into WfMSs is still challenging in that collaborations are often unstructured and dynamically changing [7,11,20,23,24,25,27]. The workflow of collaboration cannot be fully defined in advance, but often changes during execution. In addition, dynamic collaboration in distributed environment is increasingly becoming common in a wide range of organizations, which also makes it difficult for a few people to comprehend and coordinate the entire collaboration.

Generally, collaborative workflows evolve constantly [5]. The processes, participants, and resources for the collaboration are subject to change, requiring the significant number of re-definitions and re-instantiations of workflows. Technically, these changes often raise considerable difficulties including conflict resolutions caused by confused modification, migrations of numerous cases to changing workflow models, and even re-deployments of ongoing workflow processes [19,21,29,30].

With the advent of Web 2.0, wiki has appeared as a simple yet powerful web-based collaborative authoring system with high flexibility for creating and editing content. The system allows anyone to create a new article or revise an existing article and also to link relevant contents to each other on the web [12]. Since its introduction in the mid 1990s, wiki has been used for business as well as education and online publication as a useful tool that promotes sharing and collaborative creation of web content.

A number of different wikis have been suggested in the past for supporting various types of collaborative tasks such as knowledge management and project planning [9, 26], and many enterprises are integrating useful web content in wiki systems with their information systems. Unfortunately, most enterprise wiki systems tend to merely connect wiki content to traditional workflow systems or simply embed workflow design modules into the wiki system, still being limited to the scope of document wiki [14].

Motivated by the potential benefits of wiki and the above remarks on the limitations of the collaborative workflows, we attempt to extend the concept of wiki to managing ad-hoc and collaborative workflow, which calls for a novel approach for the way workflows are defined and executed. The approach proposed in this paper is in line with the concept of "emergent collaboration" suggested by McAfee [16] in that it allows not only pre-structured or categorized collaborations, but it also provides tools that enable them to emerge.

The proposed platform, named FlowWiki, aims at supporting highly flexible workflows without requiring underlying, pre-specified workflow model. In FlowWiki, ad-hoc and collaborative workflows are created and specified by distributedindependent participants on the fly. As such, FlowWiki pursues dynamism and flexibility as end users are allowed to join and leave their collaborative processes dynamically and to carry out workflow design and coordination at run-time [15].

FlowWiki presents several benefits from the perspective of managing ad-hoc and collaborative workflow that requires frequent redeployment of workflows, participants, and associated resources. First, workflows can be defined locally by end users in a bottom-up way without requiring the users to be aware of global workflow specification. Any part of a workflow can be modified by relevant authorized users anytime while keeping other parts remain untouched. This gives greater flexibility in terms of workflow specification compared to the centralized approach.

Second, the proposed platform supports workflows to evolve over time. In FlowWiki, users are interacting asynchronously via events, and users can change their part of event flow instantly if they find any other effective ways they can perform their tasks better. In this manner, the cumulative changes to event flows made by users are expected to lead to intelligent workflow evolution.

Finally, FlowWiki enables collaborations to occur among a variety of participants including humans and applications across multiple organizations on the web. Since the control flows in FlowWiki are specified through the use of hyperlinks and necessary information is delivered in the form of RSS feed [28], users can easily connect their partially specified workflows with other relevant RSS feeds available on the web.

The rest of this paper is organized as follows. In Section 2, we present the related work on ad hoc and collaborative workflow. In Section 3, the proposed platform for ad-hoc collaborative workflow is introduced accompanied by an example scenario. A model that formally defines the structures and behaviors of FlowWiki is presented in Section 4. We present a FlowWiki system implementation with the example scenario in Section 5. Finally, we conclude the paper in Section 6.

2. Related work

Much research has been conducted to address the problem of incorporating flexibility into WfMSs through providing various solutions for handling dynamic modification of instantiated processes, ad-hoc workflow changes, and partially specified workflow models [2,3,8,30]. Blumenthal and Nutt [2] develop a Petri-net variant for modeling unstructured workflows that consist of exceptional and non-exceptional activities. The non-exceptional activities correspond to ad-hoc collaborations, and they consider a case in which process designers do not have knowledge about what activities are to be carried out.

In [10], ad-hoc and collaborative processes that may evolve at run-time through refinement are considered, and a messaging based model built on shared objects, named Caramba, is suggested for the purpose of coordinating process-aware collaborations among virtual teams. Although Caramba appears to be fundamentally different compared to approaches presented by many existing adaptive workflow systems, it still requires that process designers be aware of participants and their intended interactions before execution.

Sadiq et al. [24] propose the concepts of “pockets of flexibility” and “constraints specification” to support ad-hoc changes and construction of workflows for highly flexible processes. Their approach allows execution of a partially specified workflow model while full specification of the model is made at run-time. A similar prob-

lem is also considered in [6] where a flexible modeling method, named eFlow, is presented to support enactment of incompletely specified web service composition models.

Bonita model proposed by Charoy et al. [7] attempts to support execution of cooperative processes with ad-hoc changes, and it allows a process to be extended at run-time by creating new partial processes or by importing existing processes. Pesic and Aalst [20] take a declarative approach to management of dynamic process structures by requiring what workflow participants should produce, and propose a process enactment system based on temporal logic and automata simulation. Both of these approaches, however, require a centralized process execution engine.

In parallel to the theoretical development of models and methods for accommodating flexible workflows, there have been several practical approaches to integrating workflows for web applications. Noll and Scacchi [18] develop a scripting language to specify a process-oriented hypertext for a variety of organizational computing applications. Brambilla et al. [4] propose a graphical CASE tool for modeling processes of interacting web applications as well as web service compositions. Furthermore, SAP research team [10] develop a interesting prototype, named Gravity, of incorporating collaborative process modeling to Google Wave, which is a Web 2.0 tool of real-time collaboration platform of Google, although the purpose of the system is only collaborative workflow modeling, not the execution platform. As described above, several frameworks have attempted to realize collaborative workflows on the web, but all of them still take the form of imperative language, which makese it difficult to apply them to dynamically changing, ad-hoc collaborative workflows.

More recently, service oriented architecture (SOA) is often adopted to implement ad-hoc collaborative workflows. Truong and Dustdar [27] present the VOIA framework to support online interactions for ad-hoc collaborative processes in SOA-based environments. Rosenberg et al. [23] apply RESTful web services to compose collaborative workflows. They present a lightweight language, Bite, for the web-scale collaborative workflow. Just like the goal of our proposal, their framework also aims at realizing ad-hoc and person-to-person workflows. Their platforms yet remain the approaches of system integration such as SOA and web services. On the other hand, we adopt the web 2.0 platform that has prevailed among recent web users, which effectively supports users' familiarity with the platform and software developers' readiness to extend the existing wiki platform at their convenience depending on their purposes of collaborative environments.

3. Ad-hoc collaboration model

3.1. FlowWiki platform

In this section, we describe an ad-hoc collaborative workflow model on which FlowWiki platform is based. The platform is described from the viewpoints of design-time and run-time phases for the sake of clarity of discussion. FlowWiki consists of four design-time components, namely *user*, *work structure*, *event*, and *hyperlink*, and three run-time components, *event instance*, and *event causality*.

User is a human worker who participates in collaboration and carries out his/her work within the platform. In contrast to the traditional workflow systems, a user plays roles as a workflow manager and a designer as well. Users can create, delete, and modify the other three design-time components (i.e., work structures, events, and hyperlinks), and also create event instances and event causalities at run-time.

Work structure represents an abstraction of work that users carry out for collaboration. The work structure can be created and updated by a specific type of users, called *owners* and also shared with other users. To each work structure, FlowWiki assigns a unique URL on the platform in order for other users to identify and access it on the web. Herein, a work structure is of either *public* or *private* access type. The *private* type work structure does not allow an access from other users but the owner, and accordingly others are not able to share and connect to the work structure without the owner's permission.

The work structure may accompany one or more *user-defined events* that are associated with information about the status of the work. Similar to the work structure, each event also has a unique URL. We do not impose any limitation on the semantics of events. For instance, a work structure named "write report" may have events corresponding to "just started," "half done," or "finished." In addition to the user-defined events, every work structure has *system-level events* to provide information about any changes made to the work structure. Whenever a work structure or its user-defined event is updated or deleted, the corresponding system-level event is generated so that the relevant participants can be notified.

Two work structures can be connected via a *hyperlink* that acts as a directed arc from a source work structure to a sink work structure for event delivery at run-time. Besides a hyperlink between work structures, a hyperlink can also be created from a work structure to an event of other work structure for the purpose of filtering events of interest, and this type of hyperlink is called to be *restricted*. Furthermore, any user can create a hyperlink to a work structure of public type to obtain information about the status change of the work, and such a user is called a *monitor*.

In addition to the design-time components, FlowWiki has three run-time components. An *event instance* represents an occurrence of event generated by a work structure owner at a certain moment of time. As such, each event instance is identified by an event, a timestamp, and a *producer* (i.e., work structure owner), and it may carry additional information in the form of file attachment.

Upon receiving an event instance through a hyperlink, a work structure may generate a subsequent event instance based on the information contained in the received instance. This relationship between the *consumed* instance and the *produced* instance is referred to as *event causality*. In this workflow environment, exact event processing logic of a work structure is not necessarily known, thereby it may not be possible to know which event instance among delivered to the work structure has caused generation of a new event instance of the work.

Therefore, the event causality is introduced to enable users to specify the cause and effect relationships among their event instances. For instance, when event instance e_B is created in response to the received event instance e_A, user can specify this causality by indicating e_A as e_B's predecessor before creating e_B, and the event causality can be considered a sequence flow from e_A to e_B.

In summary, under the proposed FlowWiki framework, workflows are implicitly defined through specifying hyperlinks among work structures at the design-time whereas workflow enactment is made through generation and transmission of event instances via hyperlinks among work structures at run-time. Therefore, collaborative workflows are defined entirely locally and they can evolve autonomously since the work structures, events, and hyperlinks can be added, updated, or removed anytime by participants as needed.

3.2. Example scenario

In this section, an example scenario of organizing a conference is presented to illustrate how the FlowWiki platform can support an ad-hoc collaborative workflow. Specifically, the scenario consists of several partial workflows for conference organization, workshop organization, paper review, and paper preparation as shown in Fig. 1.

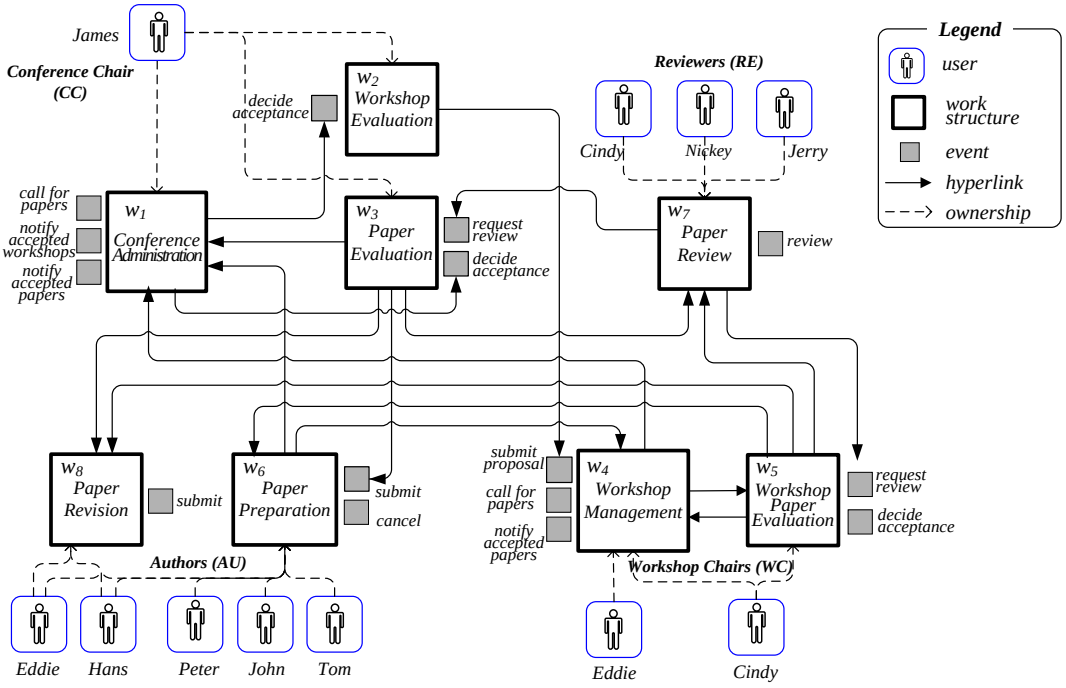


Fig. 1. An example workflow for conference organization.

We start with a single user, James, who is in charge of the conference as a conference chair (CC). First, James creates three work structures, w_1 (Conference Administration), w_2 (Workshop Evaluation), and w_3 (Paper Evaluation) in FlowWiki to organize a new conference. He then announces the conference to the public. People who are interested in the conference can now participate in FlowWiki to contribute to the conference or to obtain detailed information.

Next, Eddie and Cindy create a work structure w_4 (Workshop Management) with an event "submit pro-

posal" in order to start organizing a workshop as workshop chairs (WC). They also create a hyperlink from the w_4 to w_1 to let CC be aware of this new workshop. Note that the hyperlinks represent information needs by source work structures, and the direction of actual information flow at run-time is opposite to that of hyperlink.

With the recognition of the new workshop, CC creates a hyperlink connecting from w_2 to work w_4 to get information about the workshop related events. However, in this case, CC is only interested in the workshop proposal rather than individual workshop papers. Accordingly, CC makes a *restricted* hyperlink to event "submit proposal" of w_4. Later, when CC has finished evaluating the workshops, a list of approved workshops is delivered to w_1 as a result of event "decide acceptance" of w_2. w_4 is then notified of the decision from w_1 through the hyperlink Eddie and Cindy created before.

Subsequently, five authors (AU), Eddie, Hans, Peter, John, and Tom, create work structure w_6 (Paper Preparation) and make a hyperlink to w_1 and w_4. CC and WC create hyperlinks respectively from w_3 and w_5 to the authors' work structure (w_6) in order to receive their paper submissions. When CC and WS have received all the submitted papers, they then respectively generate "request review" event instances which are hyperlinked from work structure w_7 (Paper Review) to inform the reviewers. Afterwards, CC and WS receive the review results from the reviewers through the hyperlinks defined from w_3 to w_7 and from w_5 to w_7, respectively. Finally, the authors who are notified of acceptance create work structure w_8 (Paper Revision) for submitting revised papers.

In this way, the conference organization workflow naturally evolves over time as more and more users participate. The initial workflow was started with only principal parts which were required at that moment. Then, as more users participate, necessary events, works, and hyperlinks for the conference organization have been defined, and the workflow has grown into the one with full functionalities.

We remark that there exist several difficulties when implementing the above scenario with conventional workflow systems. First, conference organization workflow is usually somewhat involved that a single person may not know in advance the entire workflow structure and participants. To address this situation by using conventional WfMSs, the workflow definition needs to be changed whenever a new activity or control flow requirement is identified, which leads to difficulties in migrating workflow instances in many cases.

Second, nowadays most conference organization workflows are executed in a distributed web environment. To realize an effective management of conference organization, it is desirable for paper authors to be able to freely join and leave the conference workflow without prior request or invitation. Unfortunately, with the conventional WfMSs, every author needs to be registered as a workflow participant to be assigned for some roles beforehand.

Finally, conference organization workflow is quite dynamic and unpredictable since other exceptional or unanticipated requirements from users may need to be handled in a timely manner. For instance, the problems concerning hotel accommodation, joint conferences, post-conference discussions, and possible journal publication may arise in an ad-hoc way. It will be very difficult to manage such a dynamic workflow with traditional WfMSs.

4. Formal model of ad-hoc collaborative workflows

In this section, we present formal definitions of components regarding the ad-hoc collaborative workflow model employed by FlowWiki. The model is described in terms of design-time and run-time phases, and it provides a structural and behavioral specification for designing and executing ad-hoc collaborative workflows. By means of the components defined in the model, run-time aspects can be described in a generic way, which means a formal functional characterization of workflow management can be specified by unambiguous expressions. Furthermore, formally defined components can serve as an integration interface with other external workflows when a workflow of FlowWiki is integrated with others.

4.1. Design-time components

Definition 1. (FlowWiki) A *FlowWiki* $\mathcal{FW}$ is defined as 6-tuple <*U, W, E, O, ε, L*> where *U* is a set of users, *W* is a set of work structures, *E* is a set of events, $O \subseteq (U \times W)$ is a set of ownerships, $\varepsilon: E \rightarrow W$ is a function that maps an event to a work structure, and $L \subseteq W \times (W \cup E)$ is a set of hyperlinks.

In Definition 1, FlowWiki is defined in terms of three components (i.e. users, work structures, and events) and three relationships (i.e. work structure ownership, event membership, hyperlinks between work structures and events). User $u \in U$ owns zero or more work structures and the ownership is specified by relationship *O*. In contrast, an event belongs to only a single work structure, and it is specified by membership function *ε*.

Hyperlinks can be defined in two ways: one way is to define an *open* hyperlink connecting from source work structure w_i to sink work structure w_j (i.e. $L \subseteq W \times W$), which allows w_i to be notified of all events of w_j. The other way is to define a *restricted* hyperlink connecting from w_i to a certain event of w_j, which allows w_i to be notified of the specific event of w_j.

Definition 2. (Listed event of work structure) Let $w \in W$ be a work structure and $e \in E$ be an event. If $\varepsilon(e) = w$, *e* is called a *listed event of work structure w*. The set of listed events of *w* is denoted as $E(w) = \{e \in E \mid \varepsilon(e) = w\}$. The total set of events in $\mathcal{FW}$ is represented by $E = \bigcup_{w \in W} E(w)$.

Definition 3. (Links between work structures) Let $w_i, w_j \in W$ be two distinct work structures. Link between work structures is defined in terms of hyperlink l_{ij} between them. If l_{ij} is an *open hyperlink* from w_i to w_j, it is denoted by $w_i \triangleright w_j$ or $w_j \triangleleft w_i$. On the other hand, if it is a *restricted hyperlink* from w_i to event *e* of w_j, it is denoted by $w_i \triangleright^e w_j$ as depicted in Fig. 2. In addition, the sets of *outgoing* and *incoming hyperlinked work structures* of w_i are respectively denoted as $\phi^-(w_i) = \{w_j \in W \mid w_i \triangleright w_j\} \cup \{w_j \in W \mid \forall e.(w_i \triangleright^e w_j)\}$ and $\phi^+(w_i) = \{w_j \in W \mid w_j \triangleright w_i\} \cup \{w_j \in W \mid \forall e.(w_j \triangleright^e w_i)\}$.

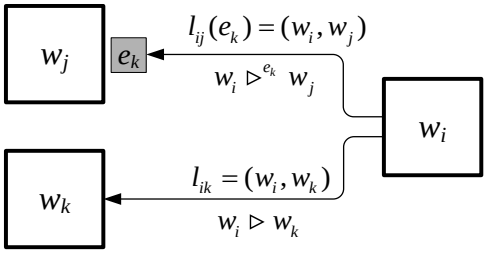


Fig. 2. Links between work structures.

From Definition 2 and Definition 3, the events that are received by a work structure or notified to other work structures through hyperlinks can be unambiguously specified. The owners of work structure *w* can receive the events to which *w* has restricted hyperlinks and also can receive the listed events of the work structure that has an open hyperlink from *w*. That is, from the viewpoint of work structure w_i to which *w* has a hyperlink, event occurrences of w_i, are notified to *w* through the hyperlink. As such, it should be noted that the directions of event flows are opposite to those of hyperlinks.

Definition 4. (Incoming and outgoing event set) *Incoming event set of work structure w* is the set of events that are delivered to *w* from other work structures through the hyperlinks connected from *w*, and *outgoing event set of work structure w* is the set of events of which occurrences are notified to other work structures through the hyperlinks connected from the other work structures to *w*. Formally, the incoming event set of work structure *w*, $E^+(w)$, is defined as $E^+(w) = \bigcup_{w_k \in \phi^-(w)} \{e | \varepsilon(e) = w_k\}$, and the outgoing event set of *w*, $E^-(w)$, is defined as $E^-(w) = \bigcup_{w_k \in \phi^+(w)} \{e | \varepsilon(e) = w_k\}$.

Since every event is restricted to belong to only one work structure, it follows that $E^+(w) \cap E^-(w) = \varnothing$. We illustrate the above definitions in the context of the conference organizing scenario in Fig. 1. Consider work structure w_4 (Workshop Management). The listed event set of w_4 is $E(w_4)$ = {"submit proposal", "call for papers", "notify accepted papers"} = $\{e_{41}, e_{42}, e_{43}\}$, where the *k*-th event of w_i is denoted as e_{ik}. w_4 has three incoming hyperlinks, $w_4 \overset{e_{41}}{\triangleleft} w_2$, $w_4 \triangleleft w_5$, and $w_4 \triangleleft w_6$, and two outgoing hyperlinks, $w_4 \triangleright w_1$ and $w_4 \triangleright w_5$, where $w_4 \overset{e_{41}}{\triangleleft} w_2$ is restricted to event e_{41}, and the others are open hyperlinks. The sets of *outgoing* and *incoming hyperlinked work structures* of w_4 are $\phi^-(w_4) = \{w_1, w_5\}$ and $\phi^+(w_4) = \{w_2, w_5, w_6\}$, respectively. Finally, it can be inferred that the users of w_4, Eddie and Cindy, can receive any event defined in $E^+(w_4) = \bigcup_{w_j \in \{w_1, w_5\}} E(w_j) =$

$\{e_{11}, e_{12}, e_{13}, e_{51}, e_{52}\}$.

Users participating in collaborative workflows may need to identify the owners of workflows as well as their events. It is also necessary for work structure owners to find those who have created hyperlinks to their work structures. In the following Definition 5, the relationships between the work structure and user are defined so that they can serve as useful abstractions to support collaborations among participants.

Definition 5. (Ownership) Let $u \in U$ be a user, and $w \in W$ be a work structure. If u is authorized to modify or delete w, u is said to be an *owner of* w, denoted by $u \in \omega(w) = \{u \in U \mid (u, w) \in O\}$, and w is said to be work structure *belonging to* u, denoted by $w \in \beta(u) = \{w \in W \mid (u, w) \in O\}$.

From Definition 4 and Definition 5, we can induce the set of the users who monitor a work structure and thus are able to receive events from the work. Given work structure w, *monitors* of w, denoted by $\mu(w)$, is expressed as $\mu(w) = \{u \in U \mid \forall i.\ (u \in \omega(w_i)) \wedge (w_i \in \phi^+(w)\} = \bigcup_{w_i \in \phi^+(w)} \omega(w_i)$. Similarly, given user u, *notifying work structure* for u, denoted by $\nu(u)$, is expressed as $\nu(u) = \{w \in W \mid \forall i.\ (w \in \phi^-(w_i)) \wedge (w_i \in \beta(u))\} = \bigcup_{w_i \in \beta(u)} \phi^-(w_i)$.

As an example, let us consider the owners and the monitors of w_4 from Fig. 1. The owners of w_4 are $\omega(w_4)$ = {Eddie, Cindy}. And, from the definition of incoming hyperlinked work structures of w_4, the monitors of w_4 can be identified such that $\mu(w_4) = \omega(w_2) \cup \omega(w_5) \cup \omega(w_6)$ = {James, Cindy, Eddie, Hans, Peter, John, Tom}. As for the work structures that notify events to Eddie who is the owner of work structures w_4 and w_6, the notifying work structures for Eddie are ν(Eddie) = $\phi^-(w_4) \cup \phi^-(w_6) = \{w_1, w_5\} \cup \{w_1, w_4\} = \{w_1, w_4, w_5\}$. Fig. 3 shows an abstracted part of the scenario in Fig. 1 depicted by definitions constructed in this paper.

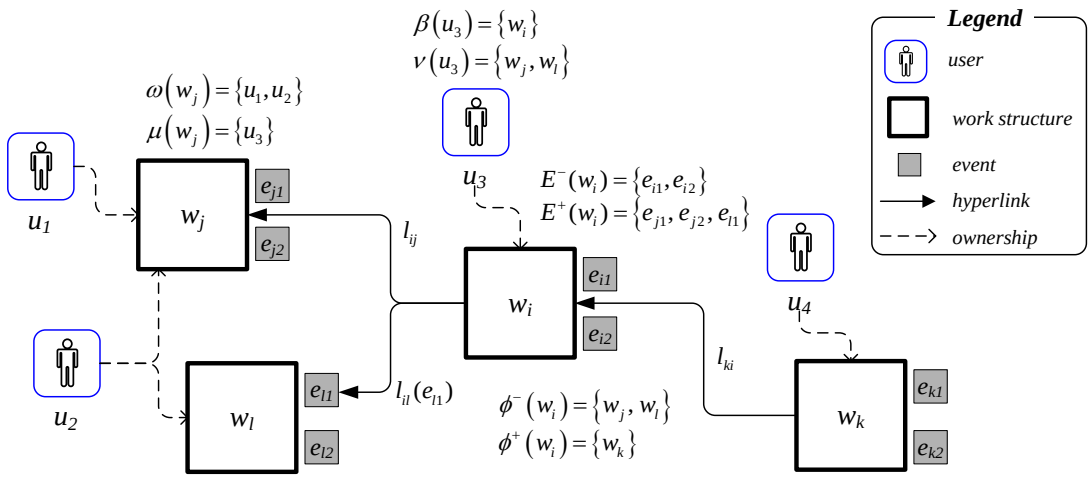


Fig. 3. Part of notational definitions for the scenario presented in Fig. 1.

4.2. Run-time components

For run-time phase, FlowWiki defines two additional components, namely event instances and event causalities to facilitate ad-hoc collaborations. We first specify the list of events a user can produce or consume in terms of the relationships between design-time components. *Producible events* of user *u* are defined by the set $E_U^-(u)=\bigcup_{w\in\beta(u)}E^-(w)$, and their occurrences are delivered to other hyperlinked work structures when *u* generates one of the listed events of the work structure she or he owns. On the other hand, *consumable events* of user *u* are defined by the set $E_U^+(u)=\bigcup_{w\in\beta(u)}E^+(w)$ since a user can only consume the events delivered to work structures under his or her ownership.

As an example, consider the user Eddie from Fig. 1. He can produce events belonging to $E_U^-(\text{Eddie}) = \bigcup_{w\in\{w_4,w_6\}}E^-(w) = \{e_{41},e_{42},e_{43},e_{61},e_{62}\}$, and he can also consume events in $E_U^+(\text{Eddie}) = \bigcup_{w\in\{w_4,w_6\}}E^+(w) = E^+(w_4)\cup E^+(w_6) = \bigcup_{w_j\in\{w_1,w_5\}}E^-(w_j)\cup\bigcup_{w_j\in\{w_3,w_5\}}E^-(w_j) = \{e_{11},e_{12},e_{13},e_{31},e_{32},e_{51},e_{52}\}$.

The event associated with a work structure is realized into event instances which contain specific information the owner has generated as a result of the task done by him or her. An event instance is described in terms of the event, user, and timestamp, and it is formally defined as follows.

Definition 6. (Event instance) Given a work structure *w* in $\mathcal{FW}$, an event instance is defined as a 3-tuple *s* = <*e*, *u*, *t*>, where *e* is an event of *w*, *u* is an owner of *w* who produces an occurrence of *e*, and t is an timestamp indicating when event instance *s* is created. Two sets of *received* and *produced* event instances by *w* are denoted by $S^+(w)=\{s=\langle e,u,t\rangle \mid e\in E^+(w)\wedge u\in\omega(w)\}$ and $S^-(w)=\{s=\langle e,u,t\rangle \mid e\in E^-(w)\wedge u\in\mu(w)\}$, respectively.

In an ad-hoc collaboration environment , a work structure may need to be carried out in relation to other work structures due to the precedence constraints, and it may require making a reference to the results of other work structures as its predecessors. In order to model this requirement in a formal manner, FlowWiki provides a means for users at run-time to specify the relationships between the events they consume for their work structures and the event instances they produce. This relationship is called event causality and is formally defined in Definition 7.

Definition 7. (Event causality) Given a work structure w, let s_i and s_j respectively be the consumed and produced event instances of w. If an owner of w has referenced s_i in producing s_j, we say that there is an *event causality* between s_i and s_j, which is denoted by $s_i \succ s_j$. The event causality set of w is defined by $C(w) = \{(s_i, s_j) \mid \forall i\ \forall j.\ s_i \succ s_j \text{ s.t. } s_i \in S^+(w) \text{ and } s_j \in S^-(w)\}$.

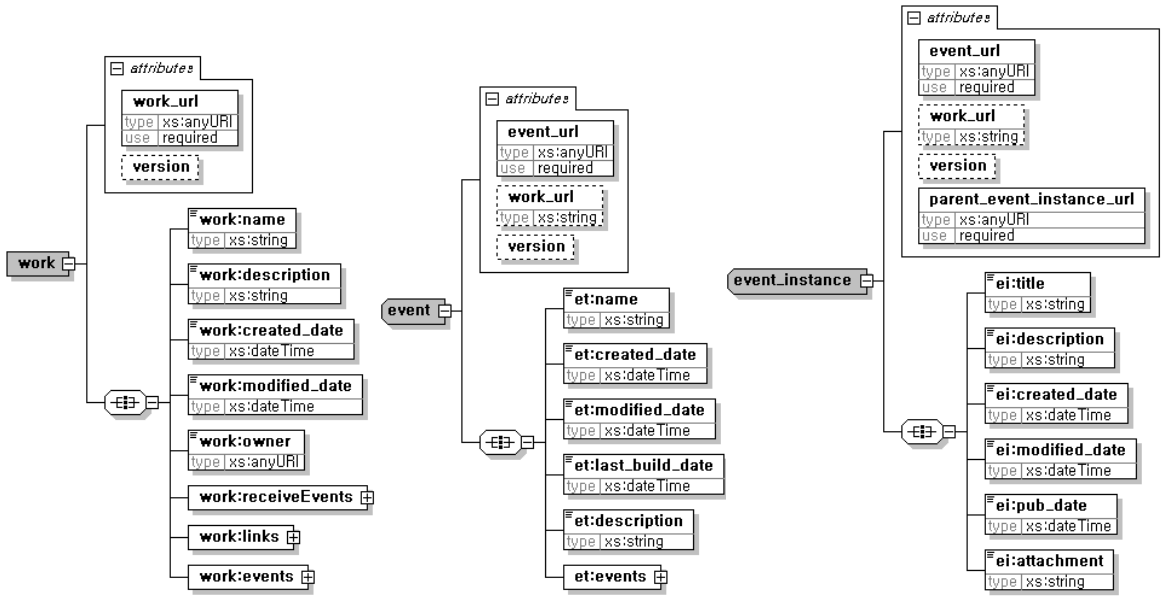
As an example, suppose that Eddie and Cindy of Fig. 1 respectively produce the instances of event "submit proposal" of w_4, which are denoted by s_E = ("submit proposal", "Eddie", t_1) and s_C = ("submit proposal", "Cindy", t_2), where t_1 and t_2 are timestamps. Later, the consumer of these two event instances, James, evaluates two workshop proposals and produces an instance of event "decide acceptance" of w_2, which is denoted by s_J = ("decide acceptance", "James", t_3). Accordingly, James can specify the event causalities as $s_E \succ s_J$ and $s_C \succ s_J$, and the event causality set of w_2 becomes $C(w_2)=\{(s_E, s_J), (s_C, s_J)\}$. The set of the consumed event instances of w_2 in this case is $S^+(w_2) = \{s_E, s_C\}$, while the set of the produced event instances is $S^-(w_2) = \{s_J\}$.

5. Platform implementation

In order to demonstrate key concepts of FlowWiki platform, a prototype has been implemented based on a conventional wiki system. While existing core components of the wiki system such as search, change control, and discussion are improved in such a way that collaborative authoring capacities are enhanced, three major changes have been implemented. First, wiki pages in the wiki system are used for implementation of work structures to support user interfaces. Second, RSS feeds in wiki pages which are originally used to syndicate the list of recent changes are revised so that users can create and modify them at their needs. Finally, the hyperlink components are enhanced to allow dynamic delivery of event instances among work structures, which is not supported in the conventional wiki system. Through the enhancement of wiki system, we have successfully transformed a collaborative content authoring system into a flexible collaborative workflow management system. Details are described in what follows.

5.1. System components

Work structure, event, and event instance are implemented in XML [13] in order to facilitate data exchanges and also to support interoperability with external systems. Fig. 10 shows the designed XML schemas for the components.



(a) Work structure (b) Event (c) Event instance

Fig. 10. XML schemas of work structure, event, and event instance.

A work structure in FlowWiki is identified by a unique name (e.g., "Paper Preparation" or "Evaluate Papers" in the example scenario of Section 2.2). Similar to wiki system, a name is used to create a hyperlink that points to a target URL of work structure or specific event. FlowWiki automatically converts a work structure name into a hyperlink for users to be able to access work structure via a simple click. In this way, a work structure may contain multiple hyperlinks to point to other work structures or events, and hyperlinked work structures can be navigated easily by just following the links provided FlowWiki.

An event belonging to a specific work structure is also identified by a unique name, and an event name followed by time descriptor represents an event instance. When an event instance is created, FlowWiki assigns a unique identifier to each event instance by concatenating the event name with timestamp that takes 0 time as January 1, 1970 00:00:00 GMT. For instance, if an event instance is created at May 8, 2007 23:28:07, the instance is identified by the specific number 1178635687 along with the corresponding event, work structure, user, and system URL such that *http://systemURL/Hans/PaperPreparation/submit/1178635687*. Each event instance contains its title, content, and other details such as the creator and URLs of related event instances to indicate event causalities between consumed and produced event instances.

Furthermore, every component in FlowWiki has a unique REST style URL [22] to support machine query and human accessibility effectively. For instance, "Paper Preparation" work structure of user Hans has an URL

like *http://systemURL/Hans/PaperPreparation*. Accordingly, users can easily access to a specific work structure when they know the user name and the work structure name. URLs of events are also defined in the same way. Since each event name is unique for a given work structure, we can extend the above URL scheme for a work structure to conveniently identify a specific event such as *http://systemURL/Hans/PaperPreparation/submit*, which means "submit" event of work structure "Paper Preparation" owned by user "Hans".

Finally, each event instance is delivered in the form of RSS to achieve flexibility and efficiency. The RSS feeds provide a client application with URLs of event instances so that the client application can assist users in receiving them in a timely manner. Since every part of the RSS format is designed to be optional, users can compose content of a work structure for better readability. Fig 11 shows an example RSS fee generated for event "submit" of work structure "Paper Preparation". The feed contain two event instances generated by two users "Hans" and "Tom".

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns="http://purl.org/rss/1.0/"
             xmlns:dc="http://purl.org/dc/elements/1.1/"
             xmlns:pw="http://di.snu.ac.kr/pw/0.9">
<channel rdf:about="http://di.snu.ac.kr/pw/0.9/pw.rdf"
             hlw:work="http://di.snu.ac.kr/pw/0.9/work">
    <title>submit</title>
    <link>http://di.snu.ac.kr/pw/PaperPreparation/submit</link>
</channel>
    …
  <item>
    <name>Paper submission</name>
    <description>This paper … </description>
    <pw:sent_by>Hans</pw:sent_by>
    <dc:date>2007-03-05 AM 10:30</dc:date>
    <pw:attachment>Hans.doc</pw:attachment>
    <link>http://di.snu.ac.kr/pw/PaperPreparation/submit/1173058207</link>
    <pw:urgency_level>3<pw:urgency_level>
  </item>
    …
  <item>
    <name>Submission of workshop paper</name>
    <description>Topic of this paper …</description>
    <pw:sent_by>Tom</pw:sent_by>
    <dc:date>2007-03-03 PM 2:00</dc:date>
    <pw:attachment>Tom.doc</pw:attachment>
    <link>http://di.snu.ac.kr/pw/PaperPreparation/submit/1172898013</link>
    <pw:urgency_level>2<pw:urgency_level>
  </item>
    …
</rdf:RDF>
```

Fig. 11. An example RSS feed of FlowWiki.

5.2. Prototype architecture

As shown in Fig. 12, the prototype system consists of client and server parts. FlowWiki server is responsible for managing system components as well as sending and receiving event instances. The server is further composed of two modules: collaboration designer and enactment engine. Collaboration designer takes care of creating and managing core design-time components of FlowWiki, and also parsing and validating them so that

they are compliant with the wiki system.

Enactment engine receives RSS feeds from internal and external event sources, and delivers event instances to work structures of appropriate users. In particular, the event engine is responsible for analyzing event instances as well as pushing newly received event instances into desktop widgets though the event push handler. The server has two databases: one for storing design-time components such as work structures, events, hyperlinks, and users, and the other for storing run-time components such as event instances and event causalities that are used to identify workflow and support run-time monitoring.

On the other hand, FlowWiki clients support two types of interfaces: web interface and desktop widget. Web interfaces are implemented to allow users to access FlowWiki through web browsers. They provide various functionalities, including editing of system components, configuration setting for search, and managing event instances. Desktop widget is used to alert users and to support them to perform work associated with newly received event instances. This mechanism can increase efficiency of workflow execution by immediately raising user's attention on the event instances received. It can also alleviate the burden of users by eliminating the need to pull their tasks periodically. Desktop widget also allows users to create new event instances based on the received instances to support easy specification of event causalities.

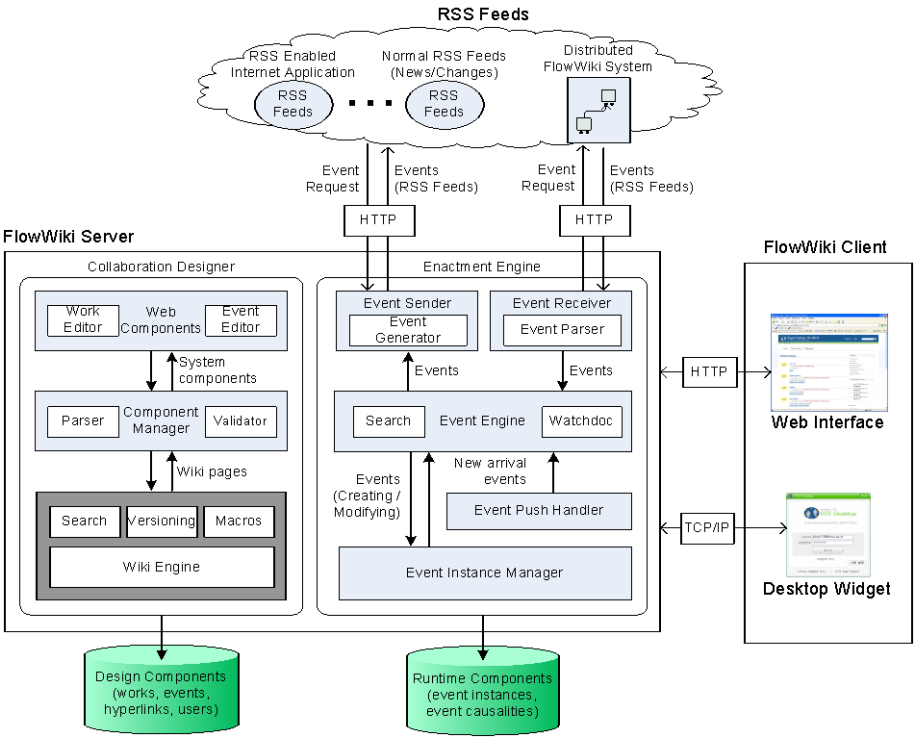
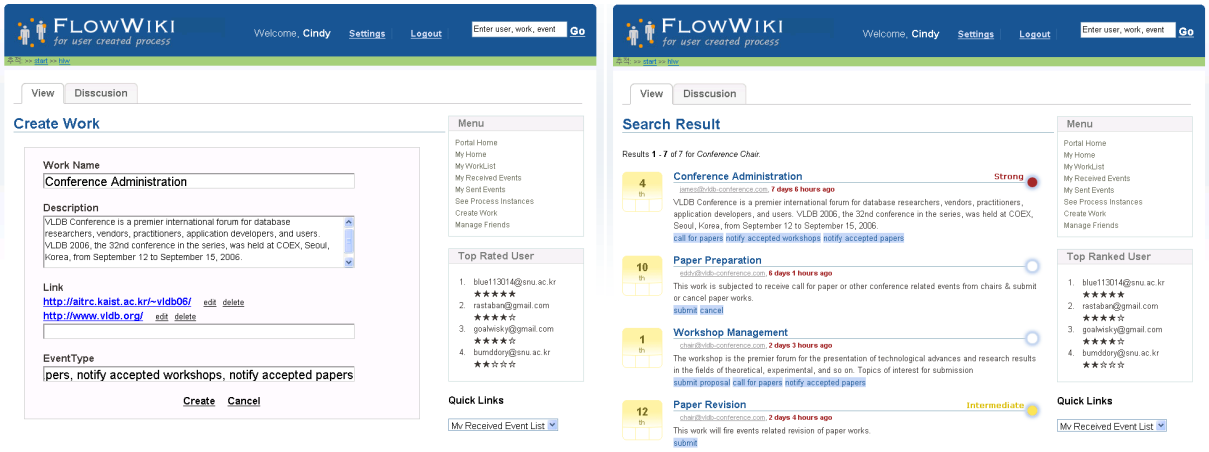


Fig. 12. System architecture of FlowWiki.

5.3. FlowWiki in action

We demonstrate how FlowWiki supports definition and execution of ad-hoc collaborative workflows through the example scenario presented in Section 2.2. Once a user logs into FlowWiki via web interface, a list of user's work structures is displayed along with appropriate actions. Through accessing the web interface, users can conveniently manage work structures, events and their instances, and hyperlinks. Fig. 13 (a) shows a screen-shot in which James (CC) in the example scenario creates a work structure "Conference Administration".

Moreover, when a user wants to find any design-time components, s/he can search them by user name, work structure name, or event name. Fig. 13 (b) shows an example search result on the James' work structures. In the presented search result, work structures are ranked according to the degree of how active work structure is. Currently, the rank is measured in terms of the number of incoming hyperlinks, the number of outgoing hyper-links, and the number of event instances received or created by work structure.



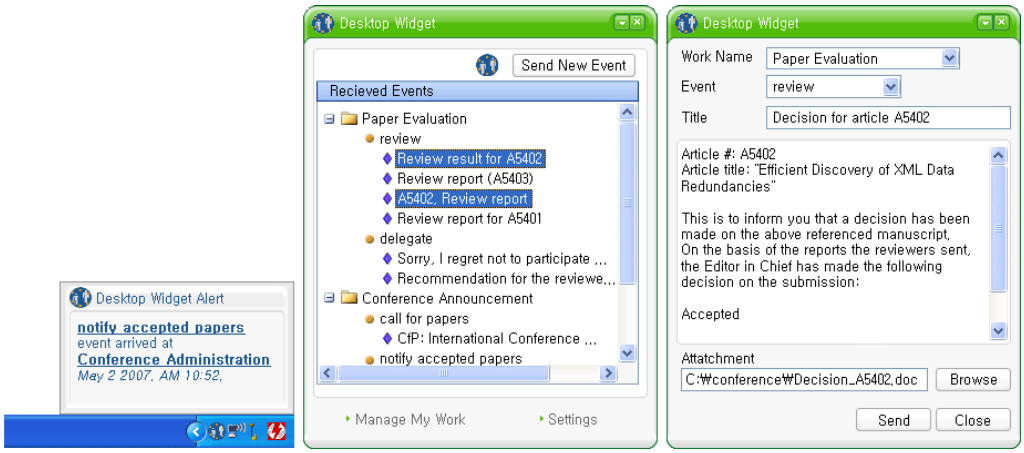
(a) Work creation form (b) Work search results

Fig. 13. Screenshots of FlowWiki web interface.

Fig. 14 shows three screenshots of desktop widgets in FlowWiki. First, Fig. 14 (a) is a pop-up alert message notifying a user of new arrival of event instances. By clicking an alert message, users can view detailed content of the received event instance. Users can also browse a list of received event instances as shown in Fig. 14 (b) where work structures, event types, and event instances are hierarchically displayed as a tree. To create a new event instance at the desk widget, users select multiple received event instances to indicate the fact that the new instance will have event causality with the selected instances. Finally, Fig. 14 (c) shows a window for creating an event instance. Users are allowed to attach a document to an event instance for additional information.

We remark that one of the salient characteristics of FlowWiki compared to conventional WfMSs is to allow early start of work even before its predecessor is not completed. In practice, there are many occasions in which execution of two tasks defined as a sequence actually can be partly overlapped without affecting the correctness of tasks. For instance, in the example presented in Section 2.2, the workshop chairs can start preparing an event instance "notify accepted papers" even before all the review results are delivered to them via "review" event

instances from several reviewers. (Note that the actual generation of "notify accepted papers" event instance will be done only after all review results are received.)



(a) Event alert message (b) Received event list (c) Event instance creation window

Fig. 14. Screenshots of desktop widget of FlowWiki.

6. Conclusions

Recent years have seen the trend of business virtualization and globalization which urgently requires dynamic and flexible collaboration not only among people but also between organizations. We proposed a new wiki based approach to effectively managing ad-hoc collaborative workflows that require high flexibility for specification as well as execution.

FlowWiki provides several benefits compared to conventional WfMSs. First, under the proposed platform, no explicit workflow specification is required. Instead, partial workflows can be dynamically created, updated, and shared by participants through defining work structures and hyperlinks as needed. Therefore, there is no need for some authorized people to be responsible for or aware of the entire workflow, which makes the workflow execution decentralized and also scalable.

Second, FlowWiki is lightweight mainly because it supports transaction and integration with existing enterprise applications in a decoupled manner through APIs and adapters. FlowWiki aims at complementing existing transactional WfMSs by supporting highly flexible workflows, and it provides RSS as a means to integrate with external systems. Furthermore, in contrast to existing wiki systems that only have RSS generators to syndicate changes of the wiki pages, work structures of FlowWiki have both RSS receivers and RSS generators, enabling them to react upon events delivered through hyperlinks. Since RSS is now widely used in most web applications, any RSS feeds from external applications can be easily integrated with FlowWiki as long as they are hyperlinked from some work structure of FlowWiki.

Third, while users in FlowWiki interact asynchronously with each other in an event-driven way through sending and receiving event instances, they can also define event causalities by specifying the relevant received event instances when a new event instance is created. In this way, event causalities can be easily generated, and they can be used to trace and monitor the collaboration.

Finally, although no single user is required to understand the entire workflow in FlowWiki, the proposed approach is expected to facilitate gradual workflow evolution since participants can flexibly customize and optimize their partial workflows over time to meet their individual needs. Future work includes application of the proposed platform to various real-world domains such as health care management, e-learning, and new product development.

References

[1] W.M.P. van der Aalst, K.M. van Hee, Workflow Management: Models, Methods and Systems (MIT Press, 2002).

[2] R. Blumenthal, G.J. Nutt, Supporting Unstructured Workflow Activities in the Bramble ICN System, Proceedings of the 1995 ACM Conference on Organizational Computing Systems (1995) 130-137.

[3] D.P. Bogia, S.M., Kaplan, Flexibility and Control for Dynamic Workflows in the WORLDS Environment, Proceedings of the 1995 ACM Conference on Organizational Computing Systems (1995) 148-159.

[4] M. Brambilla, S. Cere, P. Fraternali, I. Manolescu, Process Modeling in Web Applications, ACM Transaction on Software Engineering and Methodology 15 (4) (2006) 360-409.

[5] F. Casati, S. Ceri, B. Pernici, G. Pozzi, Workflow evolution, Data & Knowledge Engineering, 24 (3) (1998) 211-238.

[6] F. Casati, M.C. Shan, Dynamic and Adaptive Composition of e-Services, Information Systems 26 (2001) 143-163.

[7] F. Charoy, A. Guabtni, M.V. Faura, A Dynamic Workflow Management System for Coordination of Cooperative Activities, in: Proc. BPM 2006 Workshops, Lecture Notes in Computer Science, 4103 (2006) 205–216.

[8] P.W.H. Chung, L. Cheung, J. Stader, P. Jarvis, J. Moore, A. Macintosh, Knowledge-based process management – an approach to handling adaptive workflow, Knowledge-Based Systems 16 (2003) 149-160.

[9] M. Cubric, Wiki-based process framework for blended learning, in WikiSym ’07: Proceedings of the 2007 International Symposium on Wikis (2007) 11-22.

[10] A. Dreiling, Gravity - Collaborative Business Process Modelling within Google Wave, SAP Community Network Blogs (2010), http://weblogs.sdn.sap.com/pub/wlg/17826.

[11] S. Dustdar, Caramba - A Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams, Distributed and Parallel Databases 15 (2004) 45–66.

[12] A. Ebersbach, M. Glaser, R. Heigl, Wiki: Web Collaboration, Springer (2006)

[13] C.F. Goldfarb, P. Prescod, XML Handbook, 5th Ed., Prentice Hall (2004).

[14] M. Heck, Web 2.0: New technologies greet the enterprise, InfoWorld 29 (1) (2007) 20-20.

[15] F. Heylighen, Collective Intelligence and its Implementation on the Web- Algorithms to Develop a Collective Mental Map, Computational & Mathematical Organization Theory 5 (3) (1999) 253-280.

[16] A.P. McAfee, Enterprise 2.0: The Dawn of Emergent Collaboration, MIT Sloan Management Review 47 (3) (2006) 20-28.

[17] D. Miers, Best practice BPM, ACM Queue 4 (2) (2006) 40-48.

[18] J. Noll, W. Scacchi, Specifying Process-Oriented Hypertext for Organizational Computing, Journal of Network and Computer Applications 24 (2001) 39–61.

[19] G.J. Nutt, The Evolution towards Flexible Workflow Systems, Distributed System Engineering 3 (4) (1996) 276–294.

[20] M. Pesic, W.M.P. van der Aalst, A Declarative Approach for Flexible Business Processes Management, in: Proc. BPM 2006 Workshops, Lecture Notes in Computer Science 4103 (2006) 169–180.

[21] M.U. Reichert, S.B. Rinderle, U. Kreher, P. Dadam, Adaptive Process Management with ADEPT2, Proceedings of the 21st International Conference on Data Engineering (2005).

[22] L. Richardson, S. Ruby, RESTful Web Services, O'Reilly (2007).

[23] F. Rosenberg, F. Curbera, M.J. Duftler, R. Khalaf, Composing RESTful Services and Collaborative Workflows: A Lightweight Approach, IEEE Internet Computing 12 (5) (2008) 24–31.

[24] S.W. Sadiq, M.E. Orlowska, W. Sadiq, Specification and Validation of Process Constraints for Flexible Workflows, Information Systems, 30 (5) (2005) 349–378.

[25] E.Y. Shan, F. Casati, M.C. Shan, Adaptive Process Management, Workflow Handbook (2004) 103-113.

[26] R. Tolksdorf, E.P.B. Simperl, Towards Wikis as Semantic Hypermedia, Proceedings of the 2006 International Symposium on Wikis (2006) 79-88.

[27] H.-L. Truong, S. Dustdar, Online Interaction Analysis Framework for Ad-Hoc Collaborative Processes in SOA-Based Environments, Transactions on Petri Nets and Other Models of Concurrency, Lecture Notes in Computer Science 5460 (2009), 260–277.

[28] G. Vossen, S. Hagemann, Unleashing Web 2.0: From Concepts to Creativity, Morgan Kaufmann (2007)

[29] M. Wang, H. Wang, D. Xu, The Design of Intelligent Workflow Monitoring with Agent Technology, Knowledge-Based Systems 18 (2005) 257-266.

[30] J. Yan, Y. Yang, G.K. Raikundalia, SwinDeW – A p2p-Based Decentralized Workflow Management System, IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans 36 (2) (2006).