

Enhancing collaboration through notifications

Michael Sonntag¹

Asynchronous communication on web platforms is currently difficult, as it requires regular visits and checks, whether new messages have been posted, and finding out, whether these are of interest or not. This is a special problem for collaboration in E-Learning, where the focus is mainly on such asynchronous interaction modes, and class sizes are large, but communication should still take place speedily. One possibility to achieve this are notifications for users when "interesting" things happen. Methods for performing these notifications are discussed in the paper, as well as how to reduce the work for configuring them: pre-configuration as well as transfer from similar objects. The implementation of basic notifications in a learning platform is described, as well as a new transfer method through a personalized RSS feed, and a first implementation for automatically adding new notifications.

1. Motivation

Online E-Learning platforms usually contain facilities for asynchronous communication, typically in the form of discussion forums. While this is a useful feature, sometimes it can also lead to problems. E.g. in large courses discussions might be lively, with lots of new messages, so it can be difficult to identify those, which might be of interest to individual learners. Similar problems exist for coaches, but to an even larger degree, as usually the ratio students/teacher is increased compared to conventional courses. For them important are for example questions, which have not been answered after some time (e.g. by tutors or other students) or those pertaining to certain topics. Another difficulty can be group results: When one participant updated some content, the other should be informed about this new status to avoid conflicts and speed up his/her reaction to these modifications. With regard to coaches this is important e.g. for uploads of results to be inspected or corrected. Unlike [8] (notifications used in a workflow of authoring E-learning content), we focus on the actual holding of the course, where usually no structured interaction exists. There communication happens rather spontaneously and information on its reason can hardly be predicted or derived for future notifications.

¹ Institute for Information Processing and Microprocessor Technology (FIM), Johannes Kepler University Linz, Altenbergerstr. 69, A-4040 Linz, Austria. E-Mail: sonntag@fim.uni-linz.ac.at

Such difficulties can be reduced through introducing various kinds of notifications, i.e. "active" messages to inform members of the platform of certain actions within it. These notifications can be classified into three kinds:

1. **Passive notifications:** These are informational content, which is just presented unobtrusively to the user. If interested in it, he/she can follow it to receive more detailed information. Examples for this are small icons (see e.g. [4]) or awareness information to be observed peripherally ([1]). Drawbacks of this scheme are, that only a very limited amount of information can be presented to avoid cluttering the "main" information and taking up valuable screen space. In addition, no proof (or guarantee) is possible, that the user actually observed and/or noticed the information.
2. **Semi-active notifications:** These are notifications presented actively to the user while being online in the learning platform. Examples are interstitials (webpages presented in-between; when clicking on a link, a webpage is inserted before presenting the actual page) or very obtrusive information (automatically opened windows, playing sounds, etc.). Another example is the BSCW event monitor [5], which must be explicitly started and monitored. Advantageous of this form is that partly presentation to the user can be proved, and that it is closer in time (nearly synchronous) than plain passive notifications. A drawback is their interference with actions of users and their similarity to advertisement forms widely detested. They can be useful, but whether to use them or other kinds of notifications must be decided upon carefully to avoid user annoyance.
3. **Active notifications:** These are notifications, which can be at least sent and (possibly) received even when users are not logged into the learning platform. Examples are E-Mails or SMS. Although RSS is intended for public use, through a slight modification (encoding user identifications into the base URL) this can also be used for notifications. Advantages are again the proof of sending/delivery, the immediateness even when offline, and the integration into other common services. Drawbacks can be additional costs or that these services usually require "outside" help. So even though the platform itself might be working, problems can exist somewhere else (e.g. mail server offline; SMS received on the mobile phone but unnoticed; unread E-Mails; ...), preventing delivery.

2. Impact of notifications

Notifications can improve collaboration/cooperation between learners and teachers (and within those groups) in two main areas. The first is to speed up interaction through hints when interesting (depending on the person) things happen, instead of waiting until they are "manually" discovered. The second is added interaction through avoiding lost updates or orphan messages.

2.1. Collaboration between learners

Collaboration between learners improves when "empty" time is avoided, i.e. through accelerating the (asynchronous) communication. Through this shortened response time, questions are answered when they are pressing and not when already partly forgotten, solved by the student him-/herself, etc. Nevertheless, because of the high number of posts in absolute measurement, a filtering stage must be introduced: Not for every new message a notification to each participant should be generated. Therefore, interests of students must be identified and matched with the message content. Only if reasonable overlap exists, the participant should be informed. This applies not only to questions but also to messages in general, so discussions arise more easily and are performed with more interactions and faster.

Another area is shared creation of work products (e.g. seminar papers). Although learning platforms usually provide only limited support compared to groupware, this is still important. Here often not the upload/change of a document is important (but useful for speed up), but rather the opposite: Actions, which did not happen within a certain timeframe.

2.2. Cooperation between teachers

Especially in larger courses (or when several teachers jointly hold a course, like in seminars), cooperation is important but can grow difficult. One example is task sharing with regard to student questions: Who is responsible for answering certain questions? This can be difficult if there are many questions (assigning them), or if there are very few questions (no regular visits so they might go unnoticed for some time). Here retractable notifications (e.g. when it is no longer applicable as someone else has already answered the question) could be useful. However, not all notification methods allow this (e.g. E-Mail, SMS or alert windows cannot be retracted, while notifications through RSS or icons could be removed). Here notifications must be especially tied to intelligent systems as otherwise the configuration will be quite difficult. Intelligent agents can e.g. decide whom of several coaches to notify based upon their explicitly configured (or implicitly derived) interests/areas of responsibility.

2.3. Interaction between learners and teachers

Improvements are possible e.g. when learning material is updated or added: the system can check whether the learner has already seen this element/part. If not, there is no need for a notification (except perhaps that something possibly interesting was added, to be determined similar to messages), but if this part has already been viewed, it should perhaps be revisited. The same applies to tasks or assignments, which, when changed by the teacher, must be communicated rapidly to the students. Practical examples in CS courses are e.g. new versions of libraries to be used or generally additional hints/clarifications of

tasks. Interaction can also be increased similar to between students through notifications on unanswered questions. Desirable would also be a verification of student's answers of questions or even direct automatic answers as in [7] , but this is currently not possible: The system would have to understand both question and answer semantically, not only as data or its syntax (through natural language processing).

3. Automatic generation of notifications

Configuring notifications can be difficult: E.g. for each forum or folder a separate configuration is needed, although these might be rather similar (but not necessarily identical). "General" notifications in the sense of applying to multiple objects (e.g. all forums within the system or of a certain course) are not that useful, as they are either very complicated (specific objects should perhaps again be treated differently), or the rate of "false", i.e. unwanted, notifications could be quite high. One approach to ameliorate this problem is automatic generation of notifications. The following approaches can be identified:

- **Pre-configuration:** When creating objects, matching notifications can be generated automatically. Care must be taken that these depend only on the specific object and its (default or adapted) configuration, but not on any external elements like interests or preferences of the creator or its intended use. An example is automatically creating a notification for a forum for when the first answer to an own message (this is probably interesting for every user) or new messages containing keywords of interest are posted. For the latter the actual content is different for each user, but generally postings with such interesting keywords will always be important for all users. A drawback of this approach is, that only rather general notifications can be generated in this way.
- **Transferal:** Existing notifications from similar objects are transferred to newly created ones. Dependent on the information available these could also be adapted simultaneously (e.g. exchanging the teacher from the source course with the one of the target course if this data is available). This allows generating notifications that are more complicated: manually configured/adapted ones can be included. Even when "wrong" notifications are transferred (e.g. with keywords applicable only to the source course), these will usually not be triggered. In some cases difficulties can occur, e.g. when a user is in a special role for the source object (like responsible author for documents or moderator for forums), and not for the target. Such notifications will not apply to other elements, unless the role also applies to them. However, such rules are often not made explicit or detectable. If there is for example a group named "forum moderator", how can the system detect its semantics (like a notification on every single message to approve or reject it) if this is not configured explicitly? Gen-

erally, erroneous configurations will mostly remain untriggered, but can still serve as starting points for manual modifications to adapt them to their new location.

- **Derivation:** Based on intelligent subsystems rules could also be created fully automatically. One example would be that when a user's interests have been ascertained with a specific certainty, appropriate notification rules are added or modified, e.g. adding keywords to existing rules or introducing new ones. Another approach could be transferring manually created new rules to similar existing objects. In this category cooperation between users can serve not as the target but rather the source: Notifications can also be derived from other users (e.g. those with similar notifications, interests, browsing habits or groups). This is however a difficult area and depends largely on the intelligence of the base system.

An important aspect for the acceptance of automatic generation of notifications by users is information: When new notifications are generated: what are they and why were they created. This provides feedback to users helping them understand the system and allows suggesting changes or improvements. Additionally it could be used for an adaptation cycle with explicit feedback, as otherwise only modifications or removal of auto-generated notifications can serve as implicit input, which is much less reliable.

4. Implementing notifications: WeLearn

Notifications have been added to the web based learning platform WeLearn [9]. These can be configured manually, but also initial support for automatic generation exists (see below).

4.1. Kinds of notifications

Notifications can be split into two separate groups. The first group are notifications of a general nature, which are not connected to any specific object, but rather pertain to the platform as a whole. In this group currently only notifications on logins/password changes are implemented. A notification can be sent to the user when the password has not been changed for some time. This can be defined either by a number of logins or as a time span since the last change. It can be configured automatically for each user upon creation of an account. While the fact of its triggering might be of interest to the administrator or course manager, it was decided that only the user him-/herself will be notified because of privacy reasons: It is not possible to configure such notifications for other persons, although this could be implemented easily. Obviously, there is no guarantee of enforcement, as users can instead of changing their password just as easily delete this notification and not receive any notices in future, but this too is on purpose.

The second group of notifications are associated with various objects within the learning platform. Currently two types of object are supported, although enlargement is planned for the future. A common problem is monitoring uploads to folders, especially where students have to hand in exercises, or within the workspace of a smaller group creating a shared result. A notification can be sent either when an object was uploaded (for workgroups), regardless of who uploaded it or what type of object it is, or when an object was uploaded and there are now X objects of different users within this folder. The last is intended for folders where students hand in exercises. Usually they can upload them several times (the last one is actually rated), so a distinction between the total number of objects and the number of objects from different users is necessary. Coaches can use this to be notified when all (or a specific part) of students have uploaded their results. Another possible, but not yet implemented, notification could be the upload of an object by a specific user or group: this could be interesting for students, e.g. to be notified of new (or new versions) of documents added/replaced by coaches.

The second object type supported is forums, which are especially important for cooperation. While their asynchronicity should not be removed (the goal is not a structured kind of chat), interaction should be sped up. Notifications are supported for the following events: The first reply to a posting authored by the creator of the notification. This avoids e.g. the problem of continuously looking for answers to a question posted. If a forum is very interesting, a notification on every new message is then possible. As only forums can be selected and not sub elements like individual messages, this always applies to the whole forum. A more personal option is specifying several keywords of interest. When one of those appears within a new message, the user is notified. As users probably cannot exactly foresee such words, regular expressions can be used. If the extension of the learning platform through the agent system is available, an additional option can be selected. The agent system identifies keywords of interest from the user's actions, and these can be included in the keywords [6]. In this way adaptation through the agents is possible and the notification is enhanced dynamically. Another option is triggering a notification when a message is posted, which seems to contain a question. Currently only basic textual analysis takes place, but enhancement through an algorithm performing actual grammar parsing is possible. This should help teachers find postings, that should be replied to, as mere comments or answers need less attention. The last notification is based on a message being posted by a certain person or group. This can serve both learners (e.g. hint on posts by coaches) and teachers (close supervision of persons requiring special attention).

A third group of notifications is theoretically possible, which are related to several platform objects. This could be a hierarchy (all objects within/below a container) or a freely defined set. However, configuration of such notifications would be very complicated and useful applications could not be found yet.

4.2. The notification subsystem

Notifications within the system are based on events, which are triggered through various actions. These can be logged, but are sent also to the notification subsystem. Within the notification subsystem these events are added to a rules engine containing both the general rules for notifications as well as all notifications configured by users. Based on this information the rules engine fires rules if appropriate (see Figure 1). Information on fired rules is stored for later presentation on the personal page or through RSS within the rule base. After all matching rules have fired the event is removed from the system. Otherwise the large number of events would dramatically increase information within the engine. Therefore, notifications requiring persistent information store it within the engine explicitly (internal data). An example for this is information on when users last changed their passwords, so their expiration can be calculated.

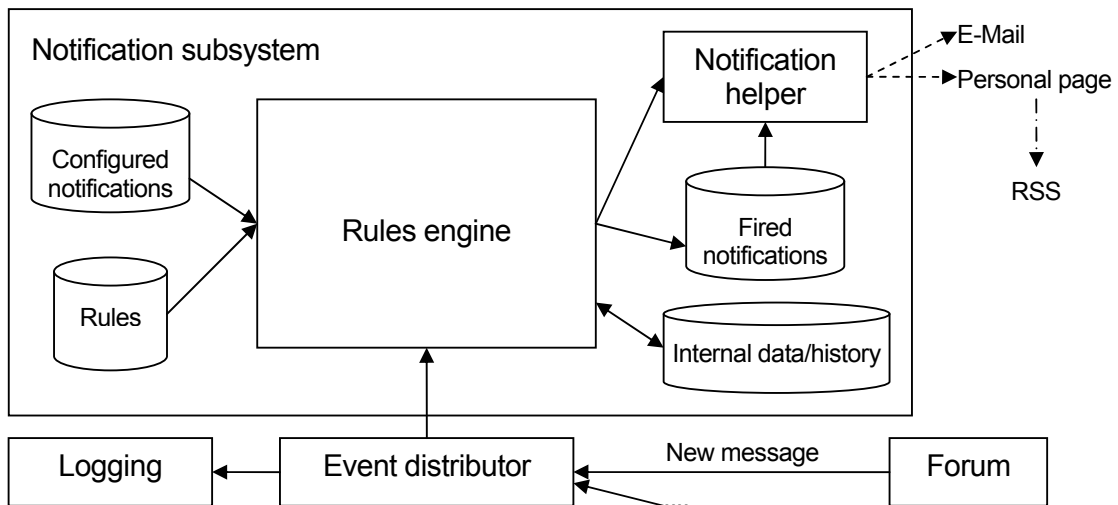


Figure 1: The notification subsystem

Each notification can be delivered using several mechanisms (see below), which can be configured separately for each notification: Some might be deemed very important (→E-Mail, SMS), while others are not that urgent and are only shown on the personal page. The actual "transfer" is done through a helper, which either assumes the role of an active (in case of E-Mail, or SMS employing public webpages) or a passive (personal page and RSS: extracts information on fired notifications from the rules engine and formats it appropriately) component.

A peculiarity of the rules engine used (DROOLS, [3]) is, that no new rules can be introduced at runtime: All rules must be configured at its creation. To circumvent this problem, notifications are not rules, but themselves (persistent) input objects. Rules possess usually two inputs: a notifications object and an event. An (abbreviated, e.g. imports omitted) listing of a sample rule is shown in Listing 1 (as rules are

configured in XML files, '&' and '<' must be escaped through character references). The example contains five conditions, before creating a holder for the fired notification and actually sending it. The first two conditions verify that the user the notification belongs to is not disabled or blocked and that the event was generated by the forum to be monitored by this notification. The third element checks the type, that it is actually a "first reply" notification. The next verifies whether it is a reply to an own posting: The owner (=creator) of the parent message (if one exists), must also be the owner of the current notification. The last condition verifies that this is the only child of the parent message, i.e. the first reply.

```

<rule name="ForumNotifications: First reply to own posting">
  <parameter identifier="notification">
    <java:class>ForumNotifications</java:class>
  </parameter>
  <parameter identifier="event">
    <java:class>ForumCreateMessageAction</java:class>
  </parameter>

  <java:condition>Notification.isUserActive(notification.getOwner())</java:condition>
  <java:condition>notification.getMonitoredObject().getMonitoredObjectOID().
    equals(event.getTargetObjectId())</java:condition>
  <java:condition>notification.getForumData().getType().equals(
    ForumNotifications.NotificationCode.FIRST_REPLY)</java:condition>
  <java:condition> DiscussionForumHelper.getParent(event.getMessageOID())!=null
    && DiscussionForumHelper.getParent(event.getMessageOID()).
    getOwner().getOid().equals(notification.getOwner())</java:condition>
  <java:condition>DiscussionForumHelper.getParent(event.getMessageOID())!=null
    && DiscussionForumHelper.getParent(event.getMessageOID()).
    getChildren().size()<=1</java:condition>

  <java:consequence>
    FiredNotificationData fnd=new FiredNotificationData(notification.getOwner(),
      notification,event);
    NotificationsObject.getInstance().fireNotification(fnd);
  </java:consequence>
</rule>

```

Listing 1: Exemplary rule: First reply to own posting

This approach of integrating the notification framework directly into the learning platform instead of moving it into the agent system, while improving performance, has also an important drawback: Timed notifications are not possible. As a servlet can be removed if unused, there is no guarantee that it will continually execute. Notifications like "Notify me if there has been no answer to a forum question after 1 week" are therefore not possible. One way to circumvent this is using other events as triggers, which can be seen at the example of changing the password. Theoretically, a user should be notified e.g. exactly three weeks after last changing the password. In the implementation this happens on the first login after three weeks, with the login being the initiating event. This makes no difference for notifications on the personal page, but if E-Mail is selected or the RSS feed monitored, these will also only be triggered upon the next login. This could be extended to arbitrary events (whenever an event comes in, the current time is checked and rules fired if overdue). Another possibility could be adding an external actor (e.g. the agent system), which either regularly "pings" the learning platform (so that an integrated scheduler will

be active and is not deactivated through the servlet container), or directly serves as a store and initiator of timed events.

4.3. Notification delivery

For actual transport, E-Mail, SMS employing public websites, RSS and in-system presentation are available. To enable E-Mail notifications, an E-Mail server must be configured for the system and the user in question must have configured an E-Mail address. Only one attempt of delivery is made: errors are logged, but delivery is not retried later. To make sure a notification is not lost therefore the internal presentation should be selected in addition to E-Mail notification.

For in-system presentation, only passive notification is possible, partly depending on the actually implemented rules: None of them is that urgent to warrant semi-active delivery. For each user there is a personal page showing dynamic information for him/her. If the agent system is present, this contains the last position (e.g. facilitating easy continuing at the place left off) within learning material or forums, a brief history of such positions, and information on keywords of interest. Notifications add to this page through a list of all fired notification instances, for which this delivery method was selected. These elements can be removed separately or all at once. Each element provides information on when and which notification fired (through the title and the description), as well as the object it pertains to. So when a new message is posted which triggers a notification for whatever reason, the information contains a link directly to this message (otherwise it might be quite difficult to find it). Reading and answering it is therefore simple and straightforward.

An active notification supported is through a personalized RSS feed. RSS is usually public and intended for general news items, as it is referenced just through an URL. In our implementation the URL is created specifically for each user, incorporating the login name and password of the user. This method allows integrating notifications into any RSS reader, so notifications are presented similarly as current news items. This kind of delivery cannot be selected separately from others; it is bound to the internal presentation on the personal page. It simply mirrors this information to external readers. For security reasons, the password is integrated into the URL only after employing a one-way hash function, so deriving the password from it is impossible. This is important, as the communication link is not encrypted and does not employ additional access control, e.g. through HTTP passwords. Therefore, everyone who can achieve knowledge of the URL can view the notifications of another person. While this is certainly an information leak, we do not think it is very dangerous. It is especially not possible to access any actual content (e.g. messages or documents), as within RSS notifications no links to the content exist (instead of

a link to a message, only its title is shown). The intended procedure is then logging into the platform and navigating to the specific message via the (identical) personal page. As the link also contains the password, it is not possible to calculate the URL for other persons, as would e.g. be the case if only the user-name were encoded. The only possibility of attack is therefore observing the link in some way.

5. Implementation of automatic new notifications

While adding new notifications is not complicated, it can get tiresome, especially for coaches assigned to several courses. Therefore, automatic addition of new notifications is integrated into the system. Now only a single class of notifications is supported: notifications pertaining to forums. The initiating event is the creation of a new forum. Automatic configuration supports pre-configuration and transferal. For pre-configuration the example presented above, first reply to own posting, is used. Transferal tries to port all existing notifications of a single or several "comparable" forum(s).

Finding a comparable forum is based on the following algorithm: If a sibling forum exists (i.e. within the same folder), this is used. The next step are descendants of the parent container, i.e. forums within folders, that are siblings of the forum to find a similar one for. If still no forum was found, the search continues recursively up to the top of the system hierarchy. It moves up one step, looks within this container and then searches all descendants in BFS order. If there is more than one forum at a level, only notifications common to all forums are transferred. For a single comparable forum, all notifications are copied exactly. A peculiarity of this algorithm is, that not necessarily the nearest forum in terms of distance of steps is chosen for comparison. I.e. the grand-grand-child of a sibling folder is preferable to a forum within the parent folder. This is based on the assumption that courses (and therefore also the object hierarchy in the system) are based on a hierarchical model. This mirrors the basic principle of the learning platform, which is similar to a file system: objects can contain other objects or "leaves" (e.g. a forum or a document). While links do exist, these are rather rare and exceptions to the general principle. Therefore a course will usually be modeled as a folder with all pertaining elements being contained within. Because of this organization, preferably moving down and only up if there is no other possibility finds forums "closer" from a didactical point of view, but possibly rather more remote in the hierarchy than others in terms of navigational steps.

To avoid user confusion about new notifications appearing seemingly at random, each automatic configuration of a new notification rule also produces a new notification itself (on the personal page only) about this fact. Through this mechanism users are notified about the new notification, receive a brief description, and can easily (through a link) modify or delete it if desired.

One limitation of automatic notification generation is the initiating event: As notifications must be configured for many users, the access rights at this moment are very important. While new notifications might be useful for the creator of the forum (e.g. the administrator or the coach), they should also be created for others, namely students. This provides some difficulty in WeLearn, as there is no explicit notion of a "course". Therefore notifications are created for all users having at least the right to read the forum. If, however, subsequently access rights are modified, this is not taken into account. It might therefore happen that users receive new notification rules for forums, which they later are unable to access. Similarly, users receiving access at another point in time will have to configure such notifications manually. This problem could be solved through integrating access right changes as events too, which later on remove automatically created notifications or add new ones. This could also be generalized so notifications, which never can fire (e.g. access was revoked), are deleted, even when created manually. But this idea also possesses some difficulties, e.g. wrong assignment of rights, which are quickly repaired, would lead to notifications disappearing or being modified (if recreated with different parameters). These would be very difficult for users to understand. Adding notifications upon access right changes is much less of a problem, as these will just not be triggered then (but still fill up the list of notifications).

Another approach to ameliorate this problem could be avoiding an implicit event as initiator for generating notifications. E.g. a coach would create a forum, configure it, and, if everything is complete, start (e.g. through the properties page) generation of notifications. This could also happen in the form of a wizard, allowing to specify additional properties, e.g. groups this forum applies to (removing the need to derive this information from the access rights), or kinds of notifications to add. But this would first require additional interaction through the teacher (which as the general idea should be obviated), and second put the teacher in the position to configure notifications for learners. While this is no inherent problem, psychologically it might not be desirable by both teachers (implicitly prescribing actions based on these notifications) and learners (patronizing by the teacher). Also, even small errors in the configurations would be attributed to the coach instead of the system otherwise. So while this is certainly a possibility, it seems worse than integrating changes in rights as triggers for modifications to notifications.

In some cases this problem is much less severe: E.g. adding notifications on password expiry can be easily configured upon creation of a new user. No difficulties arise as these apply only to this user (and not others as in case of forums) and moreover do not change over time (like access rights). If undesirable, the user can modify or delete them. However, one optimization could be of interest: If the user is disabled (e.g. at the end of a course), all notifications remain active. Therefore new notifications will accumulate,

although they cannot be read any more (apart from RSS; could be a potential problem). This is however a general problem and was solved through adding a related check to all rules.

6. Conclusions

Notifications on interesting events can be performed in several ways, but selection of the transfer method is important to avoid annoying users. While presentation within a system is easy, it suffers from the drawback of being easily overlooked and possible only, when users are online. On the other hand, sending E-Mails gets easily classified as Spam. Fine-grained and diverse possibilities to decide on the notification method are therefore important. A new method was added through a personalized RSS feed. Additionally, automatic configuration of notifications is very conservative, favoring online over offline configuration (minimum of methods of comparable forums used).

While notifications can improve cooperation, a problem factor is, that configuration is usually rather complicated, so they are created rather rarely and therefore are of little influence. Either only few categories or possibilities exist, but then notifications are not very helpful, or configuring them gets difficult and is avoided then. An important aspect is therefore easing or replacing this task, which we attempt through various forms of auto-configuration. Some notifications can be preconfigured (e.g. the first reply to own postings, which is especially helpful for communication through speeding up replies), but most must be adapted to the individual users interests and preferences. This is especially helpful for coaches in contrast to learners, as these will be involved in more areas. While personalization commonly targets the learner [2], this is therefore an aspect more important to teachers. The approach currently used is transferring notifications from similar locations, which will be extended in the future to new types of notifications.

Acknowledgment

This paper is a result of the project "Integrating Agents into Teleteaching-Webportals" sponsored by the FWF Austria (Fund for the support of scientific research; Project number P15947-N04).

References

[1] Bentley, R., Appelt, W., Busbach, U., Hinrichs, E., Kerr, D., Sikkell, K., Trevor, J., Woetzel, G.: Basic Support for Cooperative Work on the World Wide Web. In: International Journal of Human Computer Studies 46(6) 1997, 827-846

- [2] Bodendorf, F.; Schertler, M.: Improving Interactive E-Learning Frameworks by Fostering the Teacher's Role. In: Szücs, A.; Wagner, E.: The Quality Dialogue. Proc. EDEN Annual Conference 2003. Eden, Budapest 2003; S. 114-120
- [3] Drools: <http://drools.org/> (6.6.2005)
- [4] Dolog, P.; Sintek, M.: Personalization in Distributed e-Learning Environments. In: Proce. of WWW2004. ACM Press, New York, 2004. <http://www.www2004.org/proceedings/docs/2p170.pdf> (6.6.2005)
- [5] Koch, T., Appelt, W.: Beyond Web Technology - Lessons Learnt from BSCW. WETICE 1998: 176-181
- [6] Sonntag, M.: Metadata in E-Learning Applications: Automatic Extraction and Reuse. In Hofer, C.; Chroust, G. (Hrsg.): IDIMT-2004. 12th Interdisciplinary Information Management Talks. Universitätsverlag Rudolf Trauner, Linz, 2004; S. 219-231
- [7] Taylor, J. C.: Automating e-Learning: The Higher Education Revolution. In Schubert, S.; Reusch, B.; Jesse, N.: Informatik bewegt. Proc. der 32. Jahrestagung der GI 2002. GI, Bonn, 2002; S. 64-82
- [8] Telecken, T. L.; Lima, J. V.; Zeve, C. D.; Pinheiro, M. K.; Edelweis, N.. A Cooperative Environment for E-Learning Authoring. Document Numérique, França, v. 5, n. 3-4, p. 89-114, 2002. http://www-lsr.imag.fr/Les.Personnes/Manuele.Kirsch-Pinheiro/Draft_DocNum_KirschPinheiro.pdf (6.6.2005)
- [9] WeLearn - Web Environment for Learning: <http://www.fim.uni-linz.ac.at/research/WeLearn/> (6.6.2005)