Making Legacy LMS adaptable using Policy and Policy templates

Arne W. Koesling [1,2], Eelco Herder [1], Juri L. De Coi [1], Fabian Abel [1]
[1]L3S Research Center
[2]Hannover Medical School
Hanover, Germany
{koesling, herder, decoi, abel}@L3S.de

Abstract

In this paper, we discuss how users and designers of existing learning management systems (LMSs) can make use of policies to enhance adaptivity and adaptability. Many widespread LMSs currently only use limited and proprietary rule systems defining the system behaviour. Personalization of those systems is done based on those rule systems allowing only for fairly restricted adaptation rules. Policies allow for more sophisticated and flexible adaptation rules, provided by multiple stakeholders and they can be integrated into legacy systems. We present the benefits and feasibility of our ongoing approach of extending an existing LMS with policies. We will use the LMS ILIAS as a hands-on example to allow users to make use of system personalization.

1 Introduction

Our working life is accompanied by the growing need for lifelong learning. Web-based learning systems have since long been deployed in universities and enterprises to help closing the knowledge gaps of students and employees respectively. Furthermore, they are used by people in their leisure time. Therefore, lifelong learning is associated with a large diversity in interests, knowledge, and backgrounds.

Conventional learning management systems (LMSs) provide rich functionality, but lack adaptation features [Hauger and Koeck, 2007] and therefore cannot cater all individual user needs. Such individual needs could be adaptive navigation or adaptive presentation of content for students. Teachers might be interested in system adaptability features, e.g. means to let the system notify the user on certain events, trigger actions on events or easier integration of adaptation. Adaptive Educational Hypermedia Systems (AEHSs) do offer adaptation for students, but they are often prototypic systems that provide hand-tailored, application-specific user and domain models and that are used only by a small audience [Paramythis and Loidl-Reisinger, 2004]. By contrast, conventional LMSs are already used by a large number of institutes and users. Therefore, it would be very useful if one could integrate adaptation features into such legacy learning management systems.

Policy languages, together with engines that interpret the policies, can offer an easy-to-integrate solution for doing so. Depending on the language used, policies can be applied for negotiations, for access control and explanations. This provides means for making a system scrutable.

In this paper we describe our approach to extend an existing LMS – the ILIAS learning management system [ILIAS Website, 2008] – with adaptive features by means of policies. We will show how policies can be used for creating flexible adaptation rules and for lowering the burden of system administrators.

The paper is structured as follows. In the next section we introduce a real-world scenario, which reveals the current need for adaptation at the Hannover Medical School[1]. In Section 3 we introduce the concept of policies and describe our first application: utilization of policies for social navigation support. The architecture, implementation, workflow, and furthermore a discussion of our approach are presented in Section 5. We end this paper with related work, conclusions, and future work.

2 Real-World Scenario

The Hannover Medical School (MHH) makes use of a learning management system, called ILIAS [ILIAS Website, 2008], as a central repository provided for their medical students containing learning material for many courses. Since eLearning has been integrated into the MHH curriculum and the use of ILIAS is now mandatory, the activity on the learning platform has significantly increased. There are currently more than 1200 active users (activity within two months before ascertainment). Most of the material is intended to be used in a blended learning fashion, but teachers also provide a lot of additional material for self-study of the students. The structure of medical studies is organized in school years. Therefore, the top categories in ILIAS were organized according to the academic year of the students. This structure allows for rudimentary adaptivity. For example, if a student is in her second year, she can access 'study year 2' and 'study year 1' (for reference purposes), but all following study years are not accessible.

2.1 Disorientation and Information Overload

We learned from non-published usability oriented surveys that the structure described previously is insufficient. Although the learning material is organized in learning module hierarchies and below this level even by topics, many students complained about losing overview not knowing which of the material is relevant for them. The vast amount of learning material available led the students to get disorientation similar to *lost in hyperspace* [Edwards and Hardman, 1999]. To make things even worse, ILIAS currently also offers only rudimentary features for users keeping overview over their own learning history.

[1]http://www.mh-hannover.de

```
(01)  execute(adaptLearningUnitColor(LU, blue)) ←
(02)      currentRequester(RequestingUser),
(02)      RequestingUser.studyYear : StudyYear,
(03)          in([VisitNumber], rdbms : query("ILIAS_3_9_4",
(04)              "SELECT COUNT(*) " &
(05)              "FROM lo_access " &
(06)              "WHERE lm_id =' "& LU & "' " &
(07)              "AND usr_id IN (" &
(08)                  "SELECT usr_id FROM rbac_ua " &
(08)                  "INNER JOIN object_data ON (rol_id = obj_id) " &
(08)                  "WHERE type =' role' AND title =' "& StudyYear & "')"
(09)          )),
(10)      VisitNumber >= 20.
(11)  currentRequester(RequestingUser) → type : provisional.
(12)  currentRequester(RequestingUser) →
(13)      ontology :< http : //www.L3S.de/policyFramework#currentRequester > .
```

Figure 1: Example policy for color-coding a learning unit blue under certain conditions

Beside this, there are more issues that we identified by surveys and discussions with all stakeholders of the ILIAS system. For instance, medical studies are characterized by their intensive and very continuous learning process. At the MHH, summative assessments in all courses are performed at very high frequency. Most of those assessments are performed by all students of the same study year at the same day. Even though the students do not carry out the assessments directly in ILIAS, ILIAS provides practice material for these formative tests. Therefore, a very high activity can be expected on the learning platform some days before such a summative assessment, especially in the night until the morning (see [Koesling et al. (GMA), 2008]). Due to the time pressure, the students' problems in finding the desired material are even impaired. They lose time in searching appropriate material rather than working with it. For this reason, teachers seek means to stimulate or enforce students to spread their learning and training activities more evenly in time.

2.2 Identified Issues

The demands of the stakeholders we identified mainly point toward adaption and system personalization of the LMS. This is not a specific problem of ILIAS. [Hauger and Koeck, 2007] showed that widespread LMSs generally lack adaptability and adaptivity features. There is an alternative, Adaptive Educational Hypermedia Systems (AEHS), but those systems are not that widespread as LMS and do focus on a very limited knowledge domain, which is usually hand-crafted. However, there are already approaches to extend those systems with policies [Koesling et al. (AH), 2008] to overcome specific limitations.

In case of LMSs, we found that all issues identified are too complex to be implemented each separately in an economic way into the ILIAS source code. Similar to AEHSs, ILIAS – like many LMSs – has internal, proprietary rule systems. There is need for a generic solution that opens the behaviour of the LMS for a more flexible control instance than an internal and proprietary rule system. To implement the required degree of flexibility, it is also not sufficient to let only administrators adapt the behaviour. All stakeholders of the system should be given means to adapt the LMS to their needs.

We therefore analysed the ILIAS system and plan to extend this LMS with a system personalization functionality that is flexible enough to enable each of the stakeholders to adapt their learning environment by the use of so-called *policies*. There are currently several policy languages available a detailed comparison is provided by [De Coi and Olmedilla, 2008]. Following their conclusions, we choose *Protune* [Bonatti and Olmedilla, 2005], as this policy language seems currently the most mature one. The main features of Protune are described in 3.

In order to enhance ILIAS with adaptive functionality our solution furthermore has to access internal system functions of ILIAS. Our architecture will use generic components to encapsulate LMS-specific functionality, so that it can easily applied to any other web-based LMS.

3 Policies

A policy is generally understood as a statement that defines the behaviour of a system. Policies are intended to guide decisions and actions. In today's software systems, policies are primarily used for solving security and privacy concerns – such as controlling access to sensitive user data – and to model business rules, for example: *New customers of an online shop have to pay in advance, while regular costumers may be allowed to pay after delivery*. In the scope of eLearning, similar policies would be possible.

3.1 Policy Example: Color-coding

Let us assume that the administrator of an LMS wants to define a policy, that learning units that had at least 20 visits of other students of the same study year should be blue-colored to indicate that it was deemed interesting by other students. In the Protune policy language this may be written as shown in Figure 1. Similar to logic programs (cf. [Lloyd, 1987]), the predicate *execute* in line 1 holds, if each statement in the lines 2-10 hold. The variable *VisitNumber* has to be greater or equal to 20 (cf. line 10). *VisitNumber* is set by executing a SQL query, which returns the number of students, who have accessed the learning unit *LU* and are in the same *StudyYear* as the *RequestingUser*. Lines 11-13 represent meta-rules defining additional statements about the predicates used. Line 11 states the type of the predicate, in particular it defines that the predicate *currentRequester* is an action to be performed, which is uniquely identified by means of the ontology provided in line 13.

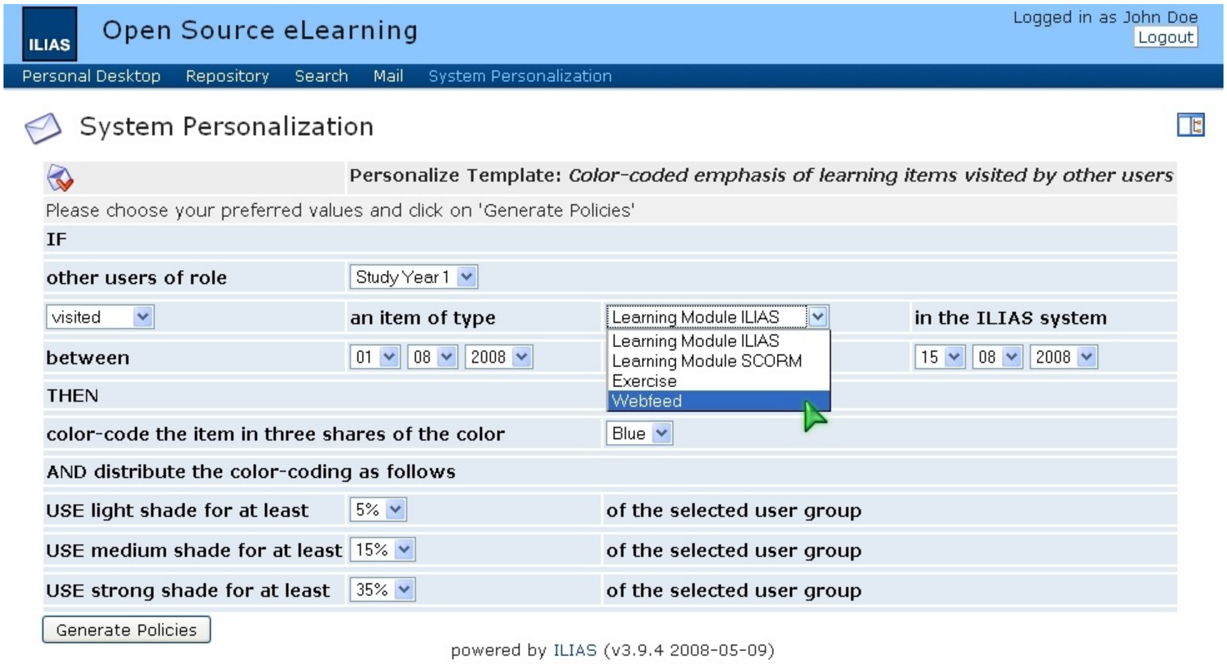

Figure 2: Personalization of User Template

Of course, there can be more sophisticated policies defined, also allowing for various degrees of coloring, depending on the amount of visits. The example in Figure 1 is intended to demonstrate that the policy can include logic statements, known from programming languages like Prolog, but also other elements, like SQL statements. It is intended to give a flavor of the Protune policy language.

The general applicability of policies in open infrastructures for lifelong learning was examined by [De Coi et al. (EC-TEL), 2007]. They gave an overview of both policy languages and policy engines, which are used to evaluate policies. The declarative nature of some policy languages enables users to define *what* the system should do, and do not require knowledge about *how* the system realizes it. Policy engines like Protune [Bonatti and Olmedilla, 2005], operate on a rule-based policy language, that has a declarative nature. In general, policy engines also provide reasoning support. In addition, Protune offers the previously mentioned explanations. This means, users have the possibility to specifically ask the system, *why* a certain answer was deduced or a decision was taken.

A remarkable feature of Protune policies is that they also allow for integrating external or environmental information into the decision making process. By performing negotiations, the user can be asked for particular preferences, credentials, etc. Furthermore, integration of policies into existing systems can be easy. The Protune policy engine is in further development to be called in a service-oriented manner.

3.2 The Need for Policy Templates

Policies can provide learners and teachers with a very flexible means to personalize the system to their needs. However, our observations show, that policies are still complex to be set up freely by the regular user. Looking at the example policy in section 3, it is unlikely that a regular user will be able to define a sophisticated policy. On the contrary, administrators can be expected to have the skills to set up policies as complex as needed and are also aware of specific actions and events that can be used in the policies within the LMS. It is therefore desirable to let administrators define templates for policies for regular users. Those templates (possibly wrapped by user-friendly interfaces) can recommend available options for the students and restrict them in their choices. This proceeding makes the creation of own policies easier, since the process is mainly a personalization. The use of policy templates does not only make the creation of policies easier for users, the restriction also allows the administrator to limit the events and actions that the user can use within her own policies to adapt the system. Policy templates can thus be compared to email filter rules or personal firewalls.

4 Use Case: Social Navigation Support

We designed an architecture to enable the flexible use of policies within a learning management system, which is described in the following sections. To demonstrate the usefulness of this architecture, we created policies and templates for a particular, simple use case. As described in Section 2, the students using ILIAS asked for a functionality to emphasize learning units that were visited by other students, leading them to relevant material. Such a social navigation support has already been explored in many systems like, e.g. Knowledge Sea II (see [Farzan and Brusilovsky, 2005]). However, we only aware of systems, that implement this functionality as fixed component. We are not aware of any system that allows the addition of such behaviour afterwards on a flexible base.

In our use case, the students will be enabled to use prebuilt policy templates (see Section 3.2), which are decoupled from the core system. Those policies enable the students to color-code learning units in three shades of an arbitrary color, according to some selectable preferences. Hence, they personalize the policies according to their needs. A student may choose to count only visits from students from specific groups or roles, e.g. students of her own study year (see Figure 1). He can choose the amount of visits needed to instruct the system to use a specific shade of a user-defined color or leave the coloring based on average visiting numbers computed within the policy. He may also choose to limit the color-coding to visits that happened in a certain period of time. Another choice is the option not to count visits, but annotations that other students left in learning units. Figure 2) demonstrates these user options in a form-based web interface. However, user input could also be collected, e.g. by more guided wizard dialogs. In [Farzan and Brusilovsky, 2005], annotations were recognized as being even more significant than visits. However, the students in the ILIAS of the MHH make rarely use of annotations. The system could thus recommend certain values to the user, making the selection process very easy and fast.

However, because the students will visit new learning units without any coloring support, the system has to deal with a *cold start problem*. The first students may browse or visit less important learning units first, resulting in wrong color-coding in the end. In the initial implementation of our system we solve this problem by enabling teachers to use policies in order to pre-indicate some relevant learning units, based on estimations of their relevance.

5 Implementation

In this section we explain the general architecture of our implementation, that consists of several elements (see also Figure 3). The policy engine as interpreter of the policies is the core element. According to [Westerinen et al., 1999]), this element is the *Policy Decision Point* (PDP). Since Protune is currently realized in Java while ILIAS is based on PHP, there are several ways to access Protune from ILIAS. We decided for requests based on web services because this results in well-defined interfaces and enables us to benefit from the advantages of service-oriented architectures, like easy replaceability. The PDP has access to a *Policy Information Base* (PIB), containing all policies defined by the stakeholders. As we found in 5.2, there is a also need for *Policy Authoring Points* (PAP) for different kind of stakeholders, presenting different web interfaces. While the administrator has access to direct editing of a policy, other stakeholders do get a specific interface for personalizing policies. In order to store the template defined by the administrator, we also need a template repository.

To extend ILIAS by a sophisticated rule system, like the Protune policy engine, the implementation needs to execute system functions of ILIAS on system level, to set system or object properties and to enforce PDP decisions: the *Policy Enforcement Point* (PEP). The PDP furthermore needs to request a variety of system properties to be included as triggering conditions for the stakeholders' policies within the policy engine. The PEP will be integrated in a *wrapper*, which is specific to the LMS used within the concrete implementation. The realizaton of the wrapper also determines all conditions and actions that can be used within the policies. In contrast to the wrapper, all other components of the architecture are generic and are applicable for arbitrary LMS.

5.1 Workflow

The workflow of this architecture has to be initiated from certain positions within the LMS. In detail, there have to be several control points within the control structure of ILIAS. If such a point is reached, a call to the policy engine is initiated. In case of our approach, the policy engine returns additional commands that have to be executed within the LMS or properties that have to be changed. The control points need to be placed before or during certain activities that are executed in the LMS. Those activities can be directly initiated by the system, e.g. generation of webpages, or initiated by the user, e.g. the start or downloading of learning units. Pointcuts known from aspect oriented programming (AOP) are a possibility to implement such control points without touching the original code of ILIAS.

There is also a need to initiate policy engine calls on events occuring in the LMS not directly initiated by the user being affected, like e.g. the login of a user or the receive of a chat message. The amount and integration locations of those points within the control structure of the LMS determine the spectrum of design freedom that is available with policies afterwards. Policies can only react on events and initiate actions that are enabled by the wrapper within the LMS. Those events and action are LMS-specific. For our use case, a call during the webpage generation is sufficient.

5.2 Discussion

The policy code of Section 3 contains an important drawback. In this case, the system would have to ask the PDP specifically whether a certain learning unit has to be colored in blue. Policies were developed in the research field of *trust management* to determine access rights on objects or services and return boolean values indicating those rights or returning a list of items, for which access is granted. Policies were not intended to extend the flexibility of an LMS. If the policies are formulated as above, the system needs either to anticipate the kind of adaptation of the user or to check for each possible adaptation, during the webpage generation. The first option would require pre-evaluation of policies. [De Coi et al (PEAS), 2007] presents an approach to pre-evaluate polices, but only regarding to deduction of access rights. Since we have to decide on actions and not access rights, pre-evaluation may not be feasible for our approach. The second option, to check for each possible adaptation, will fail mainly because of performance issues.

In evaluations [De Coi et al. (EC-TEL), 2007] it was found that one call to the policy engine using the TuProlog logic interpreter currently takes approx. 200 milliseconds. If called several times in a row, this duration is pretty long for web-based applications, enforcing economical use of those calls. Therefore, we developed the approach to insert calls to the policy engine only at the control points presented in Section 5.1 and those policy engine calls return actions and parameters to be executed. Those actions are afterwards performed within the LMS. Because we also used an old, non-optimized version of the Prolog policy engine we do also expect a high increase in performance when switching to the newest version.

6 Related Work

General learning management systems like Moodle, Sakai, or ILIAS have very simple rule systems. Those systems offer no or rudimentary adaptivity features. However, there are already attempts to enhance generic LMS like Moodle with adaptive functionality (see [Tiarnaigh, 2005]).

Policies based on the Ponder policy language are explored in [Yang et al., 2002] within collaborative eLearning

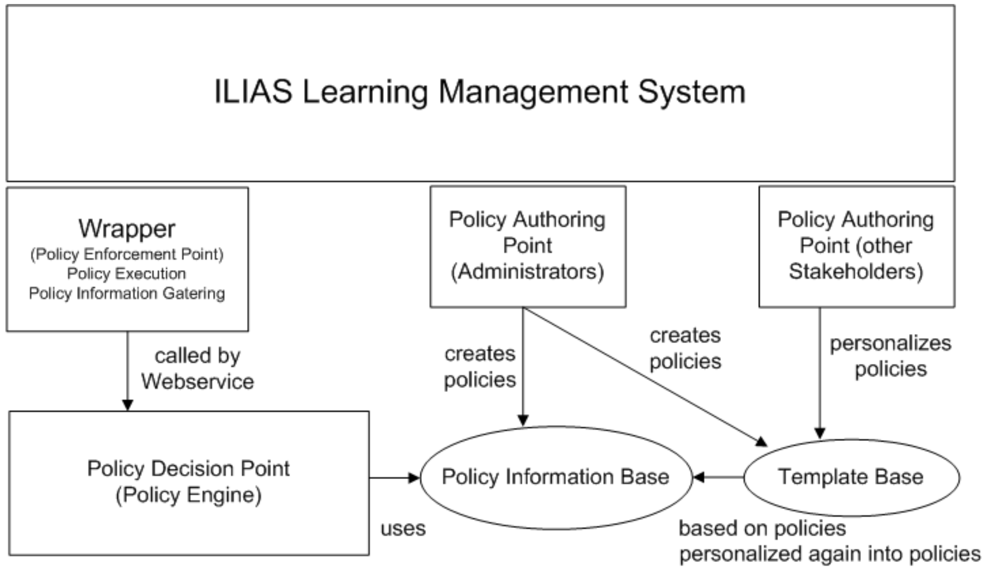

Figure 3: General architecture for ILIAS rule system extension

systems, but this work only focusses on security and privacy aspects, not on extending the adaptivity of eLearning systems. Another work that is very close to our approach is the SeLeNe project [SeLeNe Project Website, 2004] running until 2004. SeLeNe developed a so-called reactivity feature comprising a change detection mechanism based on ECA rules. However, conditions were based on RDF query languages only and actions were limited to notifications. Using a policy language like Protune allows for arbitrary conditions and actions.

The general idea of integrating sophisticated policies into eLearning environments, as we intend it in this paper, is discussed in [De Coi et al. (EC-TEL), 2007]. However, the idea of enhancing legacy learning management systems was only explored on a general level and without addressing the policy creation problem for different stakeholders. We are not aware of any other advanced research on policy-based behaviour control in technology-enhanced learning environments.

The idea of providing policy templates is not new, but there are currently no sophisticated policy template editors available, allowing for definition of policies, based on logic rules. We are furthermore not aware of any similar work enabling learners to adapt learning environments by predefined sophisticated policy templates outside the focus on security or privacy.

7 Conclusions and Future Work

In this paper we presented an approach to use policies for extending existing general-purpose learning management systems with adaptive features. By means of rules that can be developed by administrators, teachers, and learners, all stakeholders can adapt the system to their needs and requirements, which can be created from policy templates. Therefore, we presented a generic architecture and we will demonstrate the practicability of our approach by extending the ILIAS system with adaptable social navigation techniques.

We are currently researching and developing tools for many of the open issues we pointed out in this paper. For example, we are enhancing the ILIAS extension and are working on an editor that allows administrators to define policies templates for users.

An important aspect of our work is that it takes place in a 'real life' situation. This creates the opportunity to test and further refine the adaptive features and the way they can be configured and manipulated, based on usage statistics and feedback from a large pool of users. Adaptive and adaptable functionality is specifically demanded by the stakeholders and not imposed as an interesting technique that might be useful. In this paper we demonstrated, how Policies can be used for system personalization, but Policies can be means to make also more sophisticated adaptive functionality in legacy systems possible.

We are currently implementing our approach for a visual adaptation: We will further need to investigate in detail, how it fits with content-related adaptations and the exact limitations of policies within the eLearning context.

Acknowledgments

We would like to thank Daniel Olmedilla, Daniel Krause and Philipp Kärger for contributing with suggestions and remarks. The work reported in this paper is partially funded by the European Commission in the TENCompetence project (IST-2004-02787).

References

[Bonatti and Olmedilla, 2005] Piero A. Bonatti and Daniel Olmedilla. Driving and monitoring provisional trust negotiation with metapolicies. In *In 6th IEEE Policies for Distributed Systems and Networks*, IEEE Computer Society, (June 2005).

[De Coi et al (PEAS), 2007] Juri L. De Coi, Ekaterini Ioannou, Arne W. Koesling, and Daniel Olmedilla. Access Control for Sharing Semantic Data across Desktops. In *1st International Workshop on Privacy Enforcement and Accountability with Semantics*, (November 2007)

[De Coi et al. (EC-TEL), 2007] Juri L. De Coi, Philipp Kaerger, Arne W. Koesling, and Daniel Olmedilla. Exploiting Policies in an Open Infrastructure for Lifelong

Learning. In *2nd European Conference on Technology Enhanced Learning*, volume 4753 of Lecture Notes in Computer Science, pp. 26–40, Crete, Greece (2007)

[De Coi and Olmedilla, 2008] Juri L. De Coi, Daniel Olmedilla. A Review of Trust Management, Security and Privacy Policy Languages. In *3rd International Conference on Security and Cryptography*, Porto, Portugal (2008)

[Edwards and Hardman, 1999] D. M. Edwards and L. Hardman. Lost in Hyperspace: Cognitive Mapping and Navigation in a Hypertext Environment. In *Hypertext: theory into practice*, pp. 90–105, Exeter, UK (1999).

[Farzan and Brusilovsky, 2005] Rosta Farzan and Peter Brusilovsky. Social Navigation Support Through Annotation-Based Group Modeling. In *Proceedings of 10th International Conference on User Modeling*, (2005).

[Hauger and Koeck, 2007] D. Hauger and M. Koeck. State of the Art of Adaptivity in E-Learning Platforms. In *ABIS 2007 - 15th Workshop on Adaptivity and User Modeling in Interactive System*, pp. 355–360, Halle, Germany (2007).

[ILIAS Website, 2008] The ILIAS Learning Management System http://www.ilias.de (Link visited 27.06.2008)

[Koesling et al. (AH), 2008] Arne W. Koesling and Eelco Herder and Daniel Krause. Flexible Adaptivity in AEHS Using Policies In *Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, (2008).

[Koesling et al. (GMA), 2008] Arne W. Koesling, Gustav Meyer, Joern Krueckeberg, and Herbert K. Matthies. ILIAS im Rahmen des Modellstudiengangs HannibaL am Beispiel von Lerninhalten aus dem Chemischen Praktikum. *Jahrestagung der Gesellschaft fuer Medizinische Ausbildung*, (2007).

[Lloyd, 1987] John W. Lloyd. Foundations of Logic Programming, 2nd Edition. Springer, (1987)

[Paramythis and Loidl-Reisinger, 2004] A. Paramythis and S. Loidl-Reisinger. Adaptive Learning Environments and e-Learning Standards. In *Electronic Journal of e-Learning*, 2 (2), 2004, Paper 11, (2004).

[SeLeNe Project Website, 2004] The Self eLearning Networks Project http://www.dcs.bbk.ac.uk/selene (Link visited 27.06.2008)

[Tiarnaigh, 2005] M. Tiarnaigh. Adaptive Moodle. An Integration of Moodle (Modular Object Oriented Dynamic Learning Environment) with an AHS. Final Year Project, University of Dublin, (May 2005).

[Westerinen et al., 1999] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, and J. Perry, S. Waldbusser. Terminology for Policy-Based Management Request for Comments: 3198, (November 2001)

[Yang et al., 2002] Yang, Lin and Lin. Policy-based privacy and security management for collaborative e-education systems. 5th IASTED Multi-Conference Computers and Advanced Technology in Education, Cancun, Mexico, (2002).