Personalisation in Elena: How to cope with personalisation in distributed eLearning Networks

Peter Dolog and Wolfgang Nejdl

Learning Lab Lower Saxony,
University of Hannover,
Expo Plaza 1, 30539 Hannover, Germany,
{dolog, nejdl}@learninglab.de,
http://www.learninglab.de/~dolog,
http://www.kbs.uni-hannover.de/~nejdl

Abstract. One of the aims of ELENA project (www.elena-project.org) is to support personalized access to distributed learning repositories. In this talk we will present an approach to personalization we employed in ELENA. We take advantage of semantic web technologies and metadata description standards. Explicit descriptions of learning objects described in RDF bindings of LOM and DC, and learners in integrated RDF schema of PAPI and IMS LIP standards enable to employ reasoning and querying facilities of P2P Edutella infrastructure. Our approach is based on rule based matching of learning objects and learners descriptions to recommend learning services or learning objects provided by different providers, or to adapt and customize access, delivery, and consuming of the learning services and learning objects.

Keywords: Rule-based personalisation, Edutella, RDF, Semantic web

1 Introduction

Internet as an open environment provides us with the opportunity to share and reuse resources already available. Heterogeneity of users and resources in the web stresses the importance of customized (personalized) delivery of the resources. As we described in [7] a wide range of personalization techniques has been introduced based on metadata about a user or learner. The first category of techniques is based mostly on adapting user interfaces, navigation and content selection and presentation according to the user's performance in a particular domain. The performance is often evaluated in a small closed domain, e.g. an electronic course at the open university. These techniques are usually called Adaptive hypermedia techniques [5]. Another type of techniques is based on interests, preferences, likes, dislikes, and goals a user has. This information is mostly stored at some kind of modelling server [12]. These are the so-called filtering and recommendation techniques. They recommend resources according to features extracted from resource content or according to ratings of a user or learner of similar profile. In this section we show how to apply some of the Adaptive Hypermedia Techniques for personalisation in Elena network according to [6, 8].

2 Resource Metadata as Constraint on Use

We described schemas used for metadata descriptions about resources (learning material and learning services respectively) in D2.4. According to D2.4 we are using LOM for describing learning resources (Note that learning resources encapsulate here both learning objects and learning services).

Besides other use and interpretation of resource metadata there is one particularly suitable for personalisation. Resource metadata are used as some kind of constraints on use in this context. In this section we describe some of the metadata fields form this point of view. Personalized access means that resources are tailored according to some relevant aspects of the user. Which aspects of the user are important or not depends on the personalization domain. For educational scenarios it is important to take into account aspects like whether the user is student or a teacher, whether he wants to obtain a certain qualification, has specific preferences, and, of course, which is his knowledge level for the topics covered in the course. Preferences about learning materials can be easily exploited, especially if they coincide directly with the metadata and metadata values used. For users preferring PowerPoint presentations for example, we can add the literal dc:format(Resource, PowerPoint) to queries searching appropriate learning materials.

2.1 Topic Ontologies for Content Classification

We annotate each document by the topics covered in this document. Topics can be covered by sets of documents, and we will assume that a user fully knows a topic if he understands all documents annotated with this topic. However, though the standards we have just explored only provide one attribute (dc:subject) for annotating resources with topics, in reality we might want to have different kinds of annotations, to distinguish between just mentioning a topic, introducing a topic, and covering a topic. In the following we will simply assume that dc:subject is used for covered topics, but additional properties for these annotations might be useful in other contexts.

Additionally, it is obvious that self-defined keywords cannot be used in our context, as we have to use a controlled vocabulary / ontology for annotating documents and describing user knowledge. Defining a private ontology for a specific field unfortunately works only in the closed microworld of a single university. To be more general, we therefore decided to use ontologies which are already part of internationally accepted classification systems.

ACM CCS as a topic ontology for learning objects. The ACM Computer Classification system [13] has been used by the Association for Computer Machinery since several decades to classify scientific publications in the field of computer science. On the basic level, we find 11 nodes that split up in two more levels. Part of the classification hierarchy is reproduced in the following.

- A. General Literature
- B. Hardware
- C. Computer Systems Organization
- D. Software

    - D.0 GENERAL
    - D.1 PROGRAMMING TECHNIQUES
        * D.1.0 General
        * D.1.1 Applicative (Functional) Programming
        * D.1.2 Automatic Programming
        * D.1.3 Concurrent Programming
        * D.1.4 Sequential Programming
        * D.1.5 Object-oriented Programming
        * D.1.6 Logic Programming
        * D.1.7 Visual Programming
        * D.1.m Miscellaneous
    - D.2 SOFTWARE ENGINEERING
    - D.3 PROGRAMMING LANGUAGES
    - D.4 OPERATING SYSTEMS
    - D.m MISCELLANEOUS
- E. Data
- F. Theory of Computation
- G. Mathematics of Computing
- H. Information Systems
- I. Computing Methodologies
- J. Computer Applications
- K. Computing Milieux

The classification has a fourth level containing unordered keywords, thus including about 1600 entries on all four levels. For our use of the ACM CCS as a classification, we also numbered the keyword lists in the fourth level to receive unique ids like: D.1.3.1 for the keyword Parallel programming that is accessible via the taxon path: Software(D)/PROGRAMMING TECHNIQUES(D.1)/Concurrent Programming(D.1.3).

In the context of the ULI project this classification turned out to fit very well, because it covers the whole field of computer science, just as the different ULI courses cover the whole discipline. Typically a course received approximately 5 classification entries from the ACM CCS, and one entry per chapter was a typical distribution. Therefore classification with ACM CCS is excellent for the exchange of complete knowledge modules. If we look for a taxonomy that allows us to annotate different submodules and small, single learning resources, we have two other possibilities: extending the ACM CCS, or looking for another classification system. These techniques are discussed in more detail in [4].

To classify a resource, the IEEE Learning Object RDF Binding Guide (Draft Version) [10] suggests the use of dc:subject with elements of a taxonomy that must be found on the Internet. Such a taxonomy hierarchy is an instance of lom-cls:Taxonomy and must be formatted in a RDF [11] file where the topics and subtopics are separated using `lom_cls:Taxon` and `lom_cls:rootTaxon`. As discussed, we used ACM CCS, the appropriate RDF files can be found at `http://www.kbs.uni-hannover.de/Uli/ACM_CCS.rdf`. The subpart of the taxonomy transformed to a TRIPLE is as follows:

```
acmccs:'I.2.4.2'[lom_cls:taxon -> acmccs:'I.2.4.2.1'].
acmccs:'I.2.4.2'[dc:title -> nodeurl1].
nodeurl1[dc:language -> uli_lang:en].
nodeurl2[rdf:value -> 'Predicate logic'].
acmccs:'I.2.4.2.1'[rdf:type -> acmccs:'ACMClassification'].
acmccs:'I.2.4.2.1'[dc:title -> nodeurl3].
nodeurl3[dc:language -> uli_lang:en].
nodeurl3[rdf:value -> 'Skolem Functions'].
```

This subset of the ontology shows the definition of one main node of the ACM CCS (Predicate logic), refined in this example into one subtopic (Skolem Functions).

To annotate our learning resources, we link dc:subject to the entry in the ontology:

```
url:'Praedikatenlogik3.pdf'[dc:subject -> acmccs:'I.2.4.2.1'].
```

2.2 Accessibility Constraints

To specify the required level of knowledge we could introduce a new category to LOM (adaptation category for example). Another possibility is to use a relation category and the properties of PAPI or IMS. The second case makes it easier to query for appropriate resources, because we can directly map and compare what we have in the user profile and what we have in the resource description. It also means that we need to classify the learning resource according to the user profile required for accessing this learning object. LOM provides the classification category with the purpose element to do this. The purpose element has several subelements: prerequisite, educational, objective, accessibility restrictions, educational level, skill level, security level, or competency. We decided to use the accessibility restriction subelement, in order to define constraints for accessing the learning object.

```
Resource1[lom-cls:accessibilityRestrictions ->
        student:performance_1].
student:performance_1[rdf:type->papi:Performance].
student:performance_1[papi:performance_value ->
        greater_then('0.5')].
student:performance_1[papi:performance_metric -> '0-1'].
student:performance_1[papi:performance_coding -> 'number'].
student:performance_1[papi:granularity -> topic].
student:performance_1[papi:learning_experience_identifier ->
        url:'Praedikatenlogik3.pdf'].
student:performance_1[papi:learning_competency ->
        acm_ccs:'I.2.4.2.1'].
student:performance_1[papi:issued_from_identifier ->
        url:'Test_Praedikatenlogik3.pdf'].
```

Directly using these user model fields (PAPI) allows us to directly search for resources, which conform to the user profile. For example, the resource with the restricted access specified in previous example is intended for a user, whose level of knowledge about the skolem functions topic from ACM CCS is greater than 0.5.

3 Other Aspects

Additional attributes from LOM can be useful as well. For example intended user role can constrain a resource just for using it by specific role like Manager.

```
Resource1[lom-edu:intendedEndUserRole -> lom-edu:Manager].
```

Educational context can determine in which context the material can be used, e.g. school or vocational training.

```
Resource1[lom-edu:context -> lom-edu:School].
```

We can also use dcterms:audiencelevel and the lom:AgeRange for focussing on specific audiences:

```
Resource1[dcterms:audience->ID1[lom:AgeRange->7-12]].
```

And for preference:

```
Resource1[lom-edu:language->ID2[dcq:RFC1766->en]].
```

4 Describing Learner

In general we have three general possibilities how to handle adaptation and user profiles in an open environment: Overlay model based on resources, Overlay model based on ontology, Overlay or stereotype model based on ontology and resource description.

Overlay model based on resources The first possibility is to use an overlay model based on resources. This means that user knowledge is measured according to the resources which have been read / visited etc. The user model is maintained at the user site. The problem with this approach in our open environment is that we can have several resources, which belong to one topic or concept from a vocabulary.

Overlay model based on an ontology The second approach is similar to the first one, but the level of knowledge is handled according to the conceptual model or ontology. This means that we can precisely identify what resources should not be displayed because they belong to topics already understood by a user. The user model is maintained at the user site. The problem here is to map different topologies and to measure distance between topics in one or more ontologies.

Overlay or stereotype model based on ontology and resource description The third approach is based on the idea that resources and constraints for their use should be stored together as closely as possible. This means that we state directly in the resource description, for which group of students, which level of knowledge or for from which specific domain the resource is appropriate. The resource then contains descriptions like:

"I (resource xyz) am intended to be used by users, who are studying computer science, telecommunications, or physics and have knowledge about the topic I cover greater than average and who are interested in this topic".

The user model is maintained at the user site but is matched directly with the resource descriptions. An example of a learner profile employing PAPI [9] is:

```
student:student1[rdf:type -> elena:Learner].
student:student1[papi:has -> student:performance_1].
student:performance_1[rdf:type->papi:Performance].
student:performance_1[papi:performance_value -> '0.6'].
student:performance_1[papi:performance_metric -> '0-1'].
student:performance_1[papi:performance_coding -> 'number'].
student:performance_1[papi:granularity -> topic].
student:performance_1[papi:learning_experience_identifier ->
        url:'Praedikatenlogik3.pdf'].
student:performance_1[papi:learning_competency ->
        acm_ccs:'I.2.4.2.1'].
student:performance_1[papi:issued_from_identifier ->
        url:'Test_Praedikatenlogik3.pdf'].
```

The example depicts a performance record of a learner student1. He knows about Skolem Functions at the level of 0.6. This level of knowledge has been derived from an appropriate annotation for the (already read) Praedikatenlogik3.pdf resource and evaluated by the test `Test_Praedikatenlogik3.pdf`. For the topic we use the competence field from the PAPI profile. To indicate the level of knowledge, we use granularity (i.e. we measure the level of knowledge for each topic), performance coding (in numbers), performance metric (from 0 to 1) and performance value (0.6). We also use bucket to specify the time, which was required for performing the test.

5 Matching Learner and Resource Metadata

5.1 Recommendation and Filtering Based on Level of Knowledge

Reccomendation and filtering based on level of knowledge is based on matching available information about a learner with available information about constraints on resources. Matching can be described as inference rules which infer whether a document is recommended or filtered to particular user or not. Here we describe four basic scenarios which are already implemented in our ELENA network. For the scenarios we have to first infer some basic facts about the available resources. As a test bad we use an ULI provider which is one of the providers in the ELENA network.

5.2 Functionalities

The basic functionalities are:

- A document is recommended when all topics represented in all prerequisite documents are learned.
- A document is annotated as recommended when all topics represented in all prerequisite documents are learned. Otherwise it is annotated as not recommended.

The realisation of these two scenarios can be done according to two matching functionalities:

- The matching is considered as exact matching. It means that the matching rule is considered as true only when there exists information which precisely determines whether the document matches a user profile.
- The matching is considered as free matching. It means that the matching rule is considered as true when when there exists information which precisely determines whether the document matches a user profile and when there is no such information.

5.3 Realization of the Functionalities

To realize the scenarios, we need first to know what we mean by documents. Note that view @uli:ki represents our knowledgebase with facts from ULI provider resources. There are two possibilities how to handle the documents. One possibility is to use the “rdf:type” to provide exact type as an entry from ontology. This entry is in our case “elena:LectureNotes” from ontology described in the D1.1 deliverable. The rule for inferring documents from the resource metadata can look like as follows:

```
FORALL D  correct_document(D) <- D[rdf:type ->
        elena:LectureNotes]@uli:ki.
```

Another possibility is to infer that document is document when it has a dc:title. This provide us with possibility to consider more resources and to assume that author of annotation just forgot to describe a type of a resource.

```
FORALL D  document(D) <- EXISTS T D[dc:title -> T]@uli:ki.
```

To determine topics which are required for recommendation of particular resource we need to know prerequisite documents of particular resource. We use dcq:requires attribute for these purposes. The rule in TRIPLE for inferring prerequisite documents can look like as follows:

```
FORALL D, Dl prereq(D, Dl) <- D[dcq:requires->Dl]@uli:ki.
```

We need to determine topics represented or covered in particular resource. We use dc:subject for this purpose. Rule for inferring topics of a resource can look like as follows:

```
FORALL D, T topic(D, T) <- D[dc:subject->T]@uli:ki.
```

We need to determine who is a user / learner in our context. We use a type “Learner” for this purpose. A rule in triple can look like as follows:

```
FORALL U  user(U) <- U[rdf:type->elena:Learner]@uli:learner.
```

User performance is needed to match a resource to a user and to infer whether it is recommended or not. We need two rules for this purpose. One for determining what is a performance and a second for inferring values of the performance. The rules for inferring a user performance can look like as follows:

```
FORALL P performance(P) <-
        P[rdf:type -> papi:Performance]@uli:learner.
FORALL P, U user_performance(U, P) <-
        U[papi:has -> P]@uli:learner.
```

We can now realize rules for recommendation with previous facts. As we mentioned there are two recommendation scenarios and two realisation scenarios.

The first recommendation scenario can be realised in TRIPLE as follows:

```
FORALL U, D recommended_restricted(U, D) <-
      user(U) AND correct_document(D) AND
      FORALL Dl (prereq(D, Dl) ->
                  (FORALL T (topic(Dl, T) -> (EXISTS P
                    (U[papi:has->P]@uli:learner
                    AND performance(P) AND
                  P[papi:learning_competency -> T]@uli:learner
                  )))))).
```

It says that a document is recommended (the recommendation assume just exact match restricted recomendation) when for all prerequisites document all topics have at least one performance record in a learner profile. The restriction on documents which are returned as a result of the query is encoded by the `correct_document(D)` rule.

A rule for recommendation which considers also documents which are not classified by type LectureNote is the same but instead of the rule `correct_document(D)` we use the rule `document(D)`. The rule can look like as follows:

```
FORALL U, D recommended(U, D) <- user(U) AND document(D) AND
  FORALL Dl (prereq(D, Dl) ->
              (FORALL T (topic(Dl, T) -> (EXISTS P
                (U[papi:has->P]@uli:learner AND
                    performance(P) AND
                  P[papi:learning_competency -> T]@uli:learner
                ))))).
```

To extend the first scenarion to second scenarion we need to infer additional facts. One fact is that a document is recommended and another fact is that a document is not recommended. In our example all documents are first considered as not recommended. We introduced additional predicate for these purposes (document_state). The rules can look like as follows:

```
FORALL U, D document_state(U, D, not_recommended) <-
        document(D) AND user(U).
```

The state of a document is inferred as recommended to a user when documents fulfil the rule recommended. The rules can look like as follows:

```
FORALL U, D document_state(U, D, recommended) <-
        recommended(U, D).
```

We need a rule for restricted recommendation as well to realise the second realisation scenario. The rule can look like as follows:

```
FORALL U, D document_state_restricted(U, D, recommended) <-
        recommended_restricted(U, D).
```

With this information we can infer a new annotation for a document and to assign a triple whether a document is recommended or not. A document will be annotated as recommended when his state is recommended.

```
FORALL U, D, S recommendation(U, D, recommended) <-
        document_state(U, D, recommended).
```

A document will be annotated as not recommended when its state is not recommended and it is not in the recommended documents.

```
FORALL U, D, S recommendation(U, D, not_recommended) <-
        document_state(U, D, not_recommended)
        AND NOT document_state(U, D, recommended).
```

The same is valid for restricted recommendation but using the predicate with postfix restricted, which realize restricted recommendation.

```
FORALL U, D, S recommendation_restricted(U, D, recommended) <-
        document_state_restricted(U, D, recommended).
FORALL U, D, S
        recommendation_restricted(U, D, not_recommended) <-
        document_state(U, D, not_recommended) AND NOT
        document_state_restricted(U, D, recommended).
```

6 An Implementation Prototype

We have been implementing a prototype of personalized search service which implements described rule-based personalization approach for query reformulation and recommendation of learning resources.

Figure 1 depicts a user interface for formulating a query for a particular concept or competence a user would like to acquire. User can type the concept or concepts into three provided fields or can select the concepts from an ontology provided.

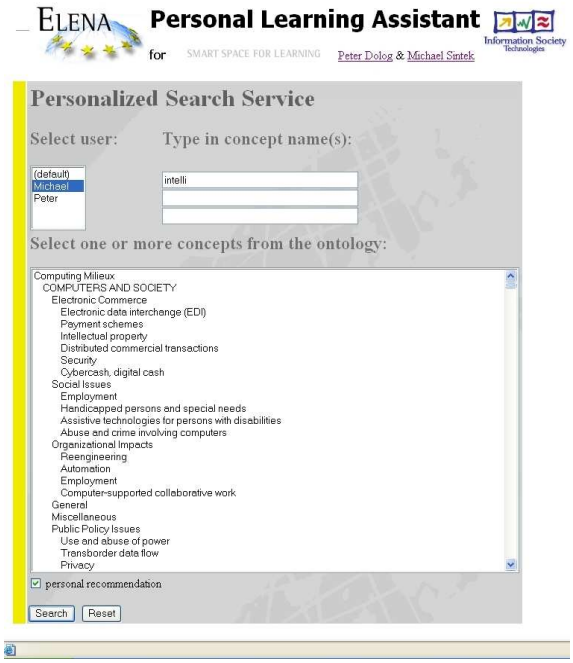

Fig. 1. A prototype for search user interface.

Personal learning assistant then creates an Edutella QEL query. The query is extended with restrictions by query rewriting service according to preferences in the learner profile before submitting to a P2P network. After receiving results, personal learning assistant contacts particular recommendation/personalization service to customize the results with a recommendation information. As it was discussed, a resource is recommended if all its prerequisite concepts are under-

stood. It is not recommended when no prerequisite concepts are understood. If some prerequisite concepts are understood, document is partially recommended.

Figure 2 depicts a prototype of a user interface for personalized search results. As you can see, we use traffic light metaphor to annotate resources with recommendation information.Green ball stands for recommended resources, red ball stands for not recommended resources and yellow ball stands for partially recommended resources.

The personal recommendation (based on learner personal profile) is depicted in the firs column (PReco). There is a second column (Reco), which provides learner with group based recommendation. Such group-based recommendation is calculated according to recommendations of learners from the same group who are using the same personal learning assistant.

Fig. 2. A prototype for personalised search results user interface.

7 Related Work

A very interesting paper in this context is the one by Bailey et al. [1], which aims to describe adaptive hypermedia techniques in an open hypermedia environment. [1] relates basic fundamental open hypermedia model concepts with adaptive hypermedia techniques. These basic concepts of open hypermedia models are data objects, context objects, behavior objects, concepts, levels of detail, links, and tours.

Data objects represent information items, context objects are associated to data objects and state in which context items are visible. Behavior objects are associated to data objects and include actions, which are performed, whenever some event on the data object occurs. Links, concepts, levels of detail, and tours represent different association over objects. These concepts are used within the contextual link server, which groups and references the resources in the web by means of these concepts.

In our work we have built upon yet a more open environment, where data objects are resources, and managed somewhere in the P2P network. These resources are annotated by RDF metadata representing the different kinds of attributes described [1], but as general metadata instead of as specific kinds of objects. Resources also can have associated accessibility restrictions, which are visibility constraints — contexts — from a user point of view. Because we provide a more expressive language for specifying contexts — Datalog based queries and constraints —

we can have more complex rules for specifying accessibility in general, not only visibility constraints. Behavior like update of user profiles can also be associated within the RDF annotation of the resource and as Datalog programs. RDF annotations provide several possibilities for specifying relationships and association, as defined by the RDFS schema, and topic ontologies are defined as RDF data again in the form of topic ontologies.

If we compare our work with standard models for adaptive hypermedia systems such as the one used in AHA! [3] for example, we observe that they define several model like conceptual, navigational, adaptation, teacher and user models. These models either correspond to ontologies / taxonomies in our case, to different schemas describing teacher and user profile, and to schemas describing the navigational structure of a course. Adaptation functionalities are expressed as Datalog queries in our model, while the adaptation model in AHA uses a rule based language encoded into XML. At the level of concept or information items AHA! provides functionalities to describe requirements [2] for the resource, which state what is required from a user to visit that information. In our case we describe these requirements by Datalog based accessibility constraints.

8 Conclusions and Further Work

We described a principle of an Elena project rule-base personalization in this paper. We discussed several recommendation functionalities and how they can be implemented. We also discussed a learner and learning resource metadata and how they can contribute to personalization problem.

We will further investigate extensions of personalization functionalities towards personalized learning services and improvements of personalization services based on learner profiles.

References

1. Christopher Bailey, Wendy Hall, David Millard, and Mark Weal. Towards open adaptive hypermedia. In *Proccedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002)*, Malaga, Spain, May 2002.
2. P. De Bra, A. Aerts, D. Smits, and N. Stash. AHA! version 2.0. In *Proceedings of the AACE ELearn'2002 conference*, pages 240–246, October 2002.
3. Paul De Bra, Geert-Jan Houben, and Hongjing Wu. AHAM: A dexter-based reference model for adaptive hypermedia. In K. Tochtermann, J. Westbomke, U.K. Wiil, and J. Leggett, editors, *Proc. of ACM Conference on Hypertext and Hypermedia*, pages 147–156, Darmstadt, Germany, February 1999.
4. J. Brase and W. Nejdl. Ontologies in elearning. Technical report, university of hannover, November 2002. to be published in "Handbook on Ontologies", Springer-Verlag 2003.
5. Peter Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87–100, 2001.
6. Peter Dolog, Rita Gavriloaie, Wolfgang Nejdl, and Jan Brase. Integrating adaptive hypermedia techniques and open rdf-based environments. In *Proc. of 12th International World Wide Web Conference*, Budapest, Hungary, May 2003.
7. Peter Dolog and Wolfgang Nejdl. Challenges and benefits of the semantic web for user modelling. In *In Proc. of AH2003 — Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems, User Modelling Conference 2003*, Pittsburgh, PA, June 2003.
8. Nicola Henze and Wolfgang Nejdl. Logically characterizing adaptive educational hypermedia systems. In *International Workshop on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2003)*, Budapest, Hungary, 2003.

9. IEEE. IEEE P1484.2/D7, 2000-11-28. draft standard for learning technology. public and private information (papi) for learners (papi learner). Available at: http://ltsc.ieee.org/wg2/. Accessed on October 25, 2002.
10. IEEE-LTSC. Ieee lom working draft 6.1. Available at: http://ltsc.ieee.org/wg12/index.html. Accessed on October 25, 2002.
11. O. Lassila and R.R. Swick. W3c resource description framework (rdf) model and syntax specification. Available at: http://www.w3.org/TR/REC-rdfsyntax/. Accessed on October 25, 2002.
12. M. Nilsson. Ims metadata rdf binding guide. http://kmr.nada.kth.se/el/ims/metadata.html, May 2001.
13. Assosiation of Computing machinery. The acm computer classification system. http://www.acm.org/class/1998/, 2002.