A Software Product Line Approach for Handling Privacy Constraints in Web Personalization[1]

Yang Wang and Alfred Kobsa

Donald Bren School of Information and Computer Sciences
University of California, Irvine, U.S.A.
{yangwang, kobsa}@ics.uci.edu

Abstract. Web personalization has demonstrated to be advantageous for both online customers and vendors. However, its benefits are severely counteracted by privacy concerns. Personalized systems need to take these into account, as well as privacy laws and industry self-regulations that may be in effect. When these constraints are present, they not only affect the personal data that can be collected, but also the methods that can be used to process the data. The present research aims at maximizing the personalization benefits, while at the same time satisfying the currently prevailing privacy constraints. Since such privacy constraints can change over time, we seek a systematic and flexible mechanism that can cater to this dynamics. We looked at several existing approaches and found that they fail to present a practical and efficient solution. Inspired by the ability of software product lines to support software variability, we propose a user modeling architecture based thereon that supports architectural level configuration management to dynamically select personalization methods that satisfy current privacy constraints. A pilot experiment is being carried out with the support of an existing user modeling server and a software architecture based development environment.

1 Introduction

Personalization technologies have been successfully introduced on the World Wide Web where they are mostly used for customer relationship management [1]. A number of studies show that personalization has provided benefits for both online customers and vendors [2, 3].

However, personalization benefits are offset by privacy concerns [4-7]. Since personalized websites collect personal data, they are also subject to privacy laws and regulations if the respective individuals are in principle identifiable. A review of nearly 30 international privacy laws [8] shows that if privacy laws apply to a personalized website, they often not only affect the data that are collected by the website and the way in which data is transferred (e.g., to which party), but also the methods that may be used for processing them (and consequently the components that embed such methods). For instance, the German Teleservices Data Protection Act [9] mandates personal data to be erased immediately after each session except for very

[1] This research has been supported through NSF grant IIS 0308277. We would like to thank André van der Hoek and Eric Dashofy for their helpful comments.

limited purposes. This provision could affect the use of machine learning methods where the learning takes place over several sessions.

From a personalization point of view, we ask the research question: how can personalized web-systems maximize the personalization benefits, while at the same time being compliant with the privacy constraints that are currently in effect (such as privacy laws, industry and company regulations, and the privacy preferences of the current user)? The remainder of this article is organized in the following way: we will discuss several existing approaches in Section 2, our proposed software product line approach in Section 3, our pilot experiment in Section 4, our example in Section 5, and finally present conclusions in Section 6.

2 Existing Approaches

2.1 Anonymous Personalization

Basically, this approach allows users to remain anonymous with regard to the personalized system and the whole network infrastructure, whilst enabling the system to still recognize the same user in different sessions so that it can cater to her individually [10]. At first sight, this seems to be the panacea because in most cases privacy laws do not apply any more when the interaction is anonymous. However, anonymity is currently difficult and/or tedious to preserve when payments, physical goods and non-electronic services are being exchanged, it harbors the risk of misuse, and it hinders vendors from cross-channel marketing (e.g. sending a products catalog to a web customer by mail). Moreover, users may still have additional privacy preferences (e.g., they do not want profiling even when it is only done pseudonymously), to which the personalized system needs to adjust.

2.2 Largest Permissible Dominator

Ideally, this approach means that only those personalization methods that meet all privacy laws and regulations are used. The Disney website for instance meets both the U.S. Children’s Online Privacy Protection Act (COPPA) as well as the European Union Directive [11]. This solution is likely to run into problems if more than a very few jurisdictions are involved, since the largest permissible denominator may then become very small.

2.3 Different Country/Region Versions

In this approach, personalized systems have different country versions, with personalization methods only that are admissible in the respective country. If countries have similar privacy laws, separate versions can be built for these countries combined, using the above-described largest permissible denominator approach. For example, IBM’s German-language pages meet the privacy laws of Germany, Austria and Switzerland [12], while IBM’s U.S. site meets the legal constraints in U.S. This approach is also likely to be infeasible as soon as the number of countries/regions, and hence the number of different versions of the personalized system, increases.

3 Our approach

User modeling systems are widely used for supporting user-adaptive applications. In industry, most such systems use a client-server architecture. A User Modeling Server (UMS) stores and represents user characteristics and behavior, integrates external user-related information, applies user modeling methods to derive additional assumptions about the user, and allows multiple external user/client adaptive applications to retrieve user information from the server in parallel [13]. Since privacy constraints directly affect UMSs, we suggest addressing them in the UMS design.

3.1 A Dynamic Privacy-Enabling User Modeling Architecture

For many personalization goals, more than one method can often be used that differ in their data and privacy requirements and their anticipated accuracy and reliability. For example, a personalized website could use incremental machine learning (that discards all raw data after the end of a session) to provide personalization to web visitors from Germany[2], while it can use possibly better one-time machine learning with the data stored across several sessions to provide personalization to web visitors from the U.S. who are not subject to this constraint. We propose a software architecture that encapsulates different personalization methods in individual components and, at any point during runtime, ascertains that only those components can be operational that are in compliance with the currently prevailing privacy constraints. Moreover, the architecture can also dynamically select the component with the optimal anticipated personalization effects among those that are currently permissible. To implement this design, we choose the Software Product Line (SPL) approach from software architecture research.

3.2 Software Product Line Architecture (PLA)

Software Product Lines have been successfully introduced in industrial software development for improving productivity, software quality and time-to-market [14]. A product line architecture represents the architectural structure for a set of related products by defining *core elements* that are present in all product architectures, and *variation points* where differences might occur among specific product architectures.

[2] This is not yet a complete solution though since the German Teleservices Data Protection Act also mandates that profiling requires the use of pseudonymous or the consent of the user.

Each variation point is guarded with a Boolean expression. Given a set of desired properties or bindings (expressed in name-value pairs), a particular product architecture can be selected out of a product line architecture by resolving the Boolean guards of each variation point.

Treating software as a product line is a new approach to support software variability from design-time to invocation-time to run-time [15]. We conceive our user modeling server as a product line architecture, where each personalization component (embedding a specific personalization method) forms a variation point in the architecture. Components that are in compliance with the currently prevailing privacy constraints will be flagged, and only those will be able to operate. If required, the architecture can additionally select the component with the highest personalization benefits based on a designer-specified preference order.

4 Pilot Experiment

We are conducting a feasibility study based on an existing user modeling server and an architecture-based software development environment. Figure 1 shows our privacy-enabling user modeling architecture implemented as a product line architecture.

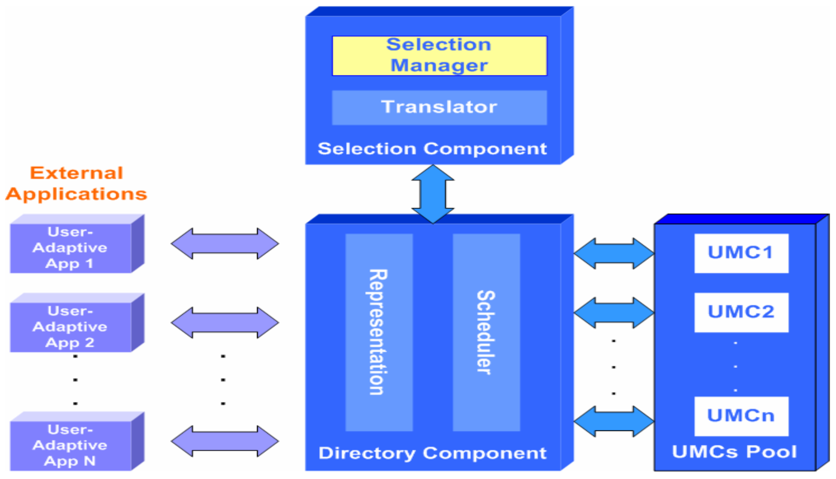

Fig. 1. A Dynamic Privacy-Enabling User Modeling Architecture

4.1 ArchStudio 3.0 – An architecture-base development environment

Our user modeling architecture is implemented in Archstudio 3.0 [16], which supports architecture-level configuration management (such as versioning, diff and merge operations) and deployment of product line architectures. We express the UMS in xADL 2.0 [17], the underlying XML-based architectural description language for ArchStudio 3.0.

4.2 External User-Adaptive Applications

On the left side of Figure 1, we see several external user-adaptive applications that query user information from the UMS in order to provide personalization services, and supply new information about the user. They communicate with the UMS using standard LDAP operations such as add, search, bind and unbind.

4.3 Directory Component

We used an existing LDAP-based[3] common user data repository [18] as our test bed for managing user models. The repository is represented as *Directory Component* in Figure 1 and is composed of two sub-systems: *Representation* and *Scheduler*. The Representation sub-system is in charge of managing directory content (i.e., mainly user-related information). The Scheduler sub-system is responsible for the communication between the Directory Component and various *User Modeling Components* (UMC).

4.4 User Modeling Components (UMC)

On the right side of Figure 1, we see a set of user modeling components. Each of these components embodies a user modeling method (e.g., collaborative filtering, domain-based inferences). A UMC can subscribe to certain types of internal events of Directory Component by maintaining event subscriptions in the Service Model hosted by the Representation sub-system. After the launch of the UMS, the Scheduler loads event subscriptions from the Service Model. Subsequently, the Scheduler periodically checks the Service Model for new entries and, if necessary, updates its internal subscription tables accordingly. Henceforth, the Scheduler acts as an event broker that supervises LDAP events within the Directory Component and communicates them together with associated data to UMCs.

In our product line architecture, UMCs are treated as variant components guarded by Boolean expressions which express privacy constraints[4] pertaining to the personalization methods incorporated in these components.

4.5 Selection Component

The Selection Component which is shown in the top part of Figure 1 subscribes to the

[3] Our approach can use any form of user data repository, e.g., a database, as long as personalization methods can plug into the repository.

[4] While ultimately these constraints should be expressed in privacy constraint specification languages (such as APPEL [19] or EPAL [20]) or semantic web technologies [21], we currently use a set of variables only.

LDAP events of the Directory Component. The Selection Component comprises two sub-components: *Selection Manager* and *Translator*. The main use of the Selector Manager is to carry out the dynamic user modeling component selection described in Section 4.6. The Selection Manager is implemented in a component-based and message-based architectural style called C2 [22]. The Translator maps the LDAP events to C2 messages. Figure 2 shows the internal architecture of the Selection Manager. The xArchADT stores the architectural description of the PLA. The Selector performs the selection of personalization methods, and the Manager orchestrates the whole selection and instantiation process.

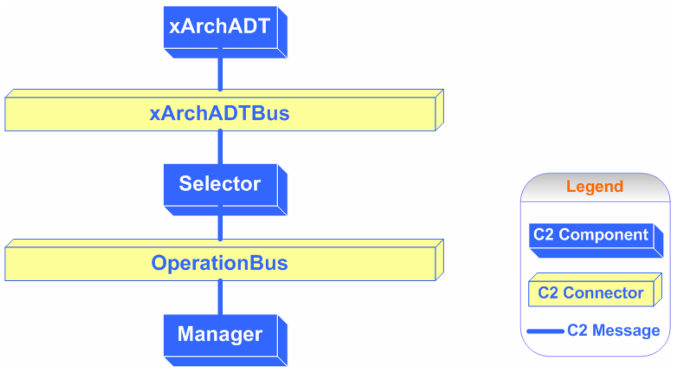

Fig. 2. Internal Structure of the Selection Manager

4.6 Dynamic Selection Mechanism

The Manager monitors the start and end of user sessions via external applications' LDAP bind and unbind operations onto the UMS. When the Manager detects the start of a user session, it initiates a *Privacy Context Detection* process that will collect all the active privacy constraints and then generate corresponding variable bindings for the privacy constraints of all UMCs. A similar process will be carried out whenever during a user session the Manager learns about new or changed privacy requirements (which for all practical purposes will stem from user preferences since privacy laws and regulations are unlikely to change during a user session).

The bindings are fed into the Selector that will carry out a *PLA selection process.* First, the Boolean guards of all UMCs are evaluated based on their variable bindings, to determine whether or not these UMCs may be included in the user modeling architecture for the current user session. A binary Privacy Constraint Satisfaction (PCS) vector is constructed whose n^{th} element represents whether or not the n^{th} UMC may be used. The Selector checks whether a run-time system instance with such a PCS already exists. If so, the Manager will assign the user session to the existing run-time system instance that has the same PCS. If not, the Selector will perform *PLA Pruning* that automatically removes any disallowed components from the architecture, and then the Manager instantiates a new run-time system instance for the user session.

The following pseudo-code illustrates the above dynamic selection mechanism:

```
The Selection Manager monitors LDAP bind and unbind events of user sessions:
   On bind:
      Privacy Context Detection:
           Collect the current privacy constraints (e.g., privacy laws and regulations);
           Generate variable bindings for Boolean guard expressions of UMCs;
      PLA selection, based on variable bindings:
           Evaluate Boolean guards for UMCs;
           Construct a new Privacy Constraints Satisfaction (PCS) vector V where element
           V[i] signifies whether UMC_i may be used (1) or not (0) for the user session;
      IF there already exists an identical PCS THEN
           Assign the user session to the existing run-time system instance;
           numOfSessions ++; //add 1 to the number of sessions handled by this instance
      ELSE
           PLA Pruning:
               Prune out UMCs whose Boolean guards are resolved to FALSE, meaning
               they are in conflict with the privacy constraints in effect;
           Instantiate a new run-time system instance for the user session;
           numOfSessions =1; // assign the user session to the new run-time instance

   On unbind:
      numOfSessions - -; // decrease the number of sessions handled by this instance by 1
      IF numOfSessions == 0 THEN
           Kill the corresponding run-time system instance;

   If new/changed user privacy preferences are detected:
      Similar process as on bind, but simplifications are possible;
```

5 Example

UniversalFriends.com is a website run by UniversalFriends LLC in the USA, a signatory of the U.S. Network Advertisers Initiative. The goal of this website is to bridge physical distances between people and to foster universal friendship via information technology. It provides personalized services to help customers make friends worldwide. Upon registration, each user will be asked to choose a pseudonymous user ID along with a password and provide some information about themselves (e.g., their hobbies). Users will be given some space on the UniversalFriends web server to create their own homepages. Based on a user’s characteristics, the system will recommend a personalized list of likely friends, and will automatically send invitations for pairwise virtual meetings.

The UniversalFriends web server relies on our privacy-enabling user modeling server to provide inferred information about users to recommend potential friends. More specifically, inferences about a user are calculated by different user modeling components from the User Modeling Components Pool as shown in Table 1.

User Modeling Component	**Data used**	**Methods used**
UMC_1	• Demographic data (age, gender, profession, education level and so forth)	Clustering techniques using demographic data (e.g., recommend people in the same profession cluster).
UMC_2	• User-supplied data (e.g., a user indicates her levels of interests in different topics)	Rule-based reasoning (e.g., if a user indicates a high interest in a specific topic, we infer that she would like to meet people with similar ratings for the topic).
UMC_3	• User-supplied data	Fuzzy reasoning with uncertainty (e.g., if a user indicates a high interest in a specific topic, we are 95% confident to infer that she would like to meet people with similar ratings for the topic).
UMC_4	• Demographic data • User-supplied data	Rule-based reasoning
UMC_5	• Demographic data • User-supplied data	Fuzzy reasoning with uncertainty
UMC_6	• User-supplied data • The UniversalFriends pages the user visited in the current session	Incremental machine learning
UMC_7	• User-supplied data • The UniversalFriends pages the user visited across sessions	One-time machine learning
UMC_8	• Demographic data • User-supplied data • The UniversalFriends pages the user visited across sessions	One-time machine learning, Fuzzy reasoning with uncertainty

Table 1. Different User Modeling Components in the User Modeling Components Pool

Let us assume that we have three users: Alice, Cheng and Bob. Table 2 describes their information:

Name	**Nationality / Current Location**	**Privacy preference**
Alice	Germany	None
Cheng	China	Dislike being tracked
Bob	The United States	None

Table 2. Information about our hypothetic users

Figure 3 illustrates the process of how our privacy-enabling user modeling architecture caters to each individual user.

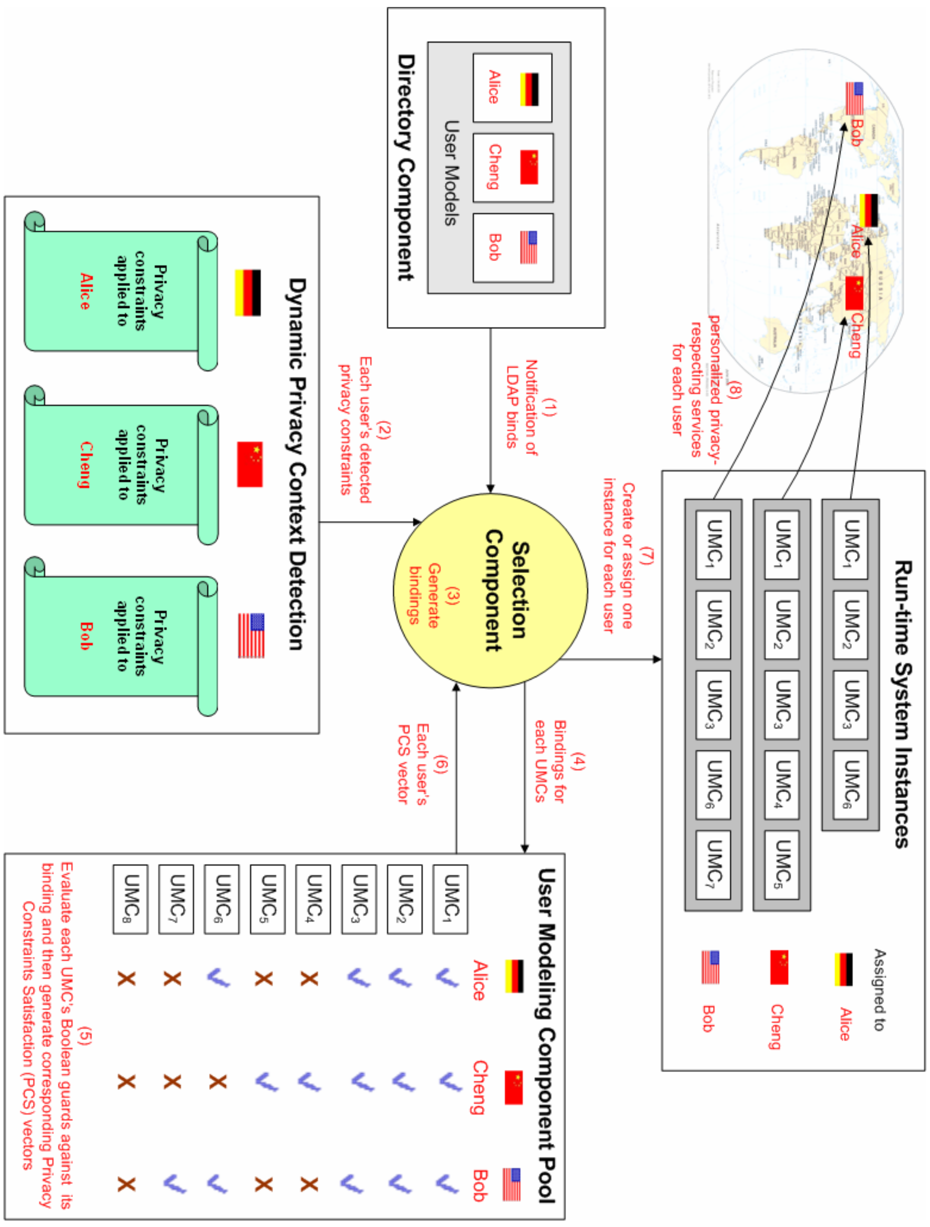
Directory Component
User Models
Alice
Cheng
Bob
(1) Notification of LDAP binds
Dynamic Privacy Context Detection
Privacy constraints applied to Alice
Privacy constraints applied to Cheng
Privacy constraints applied to Bob
(2) Each user's detected privacy constraints
Selection Component
(3) Generate bindings
(4) Bindings for each UMCs
User Modeling Component Pool
Alice
Cheng
Bob
UMC_1
UMC_2
UMC_3
UMC_4
UMC_5
UMC_6
UMC_7
UMC_8
(5) Evaluate each UMC's Boolean guards against its binding and then generate corresponding Privacy Constraints Satisfaction (PCS) vectors
(6) Each user's PCS vector
(7) Create or assign one instance for each user
Run-time System Instances
UMC_1 UMC_2 UMC_3 UMC_6
UMC_1 UMC_2 UMC_3 UMC_4 UMC_5
UMC_1 UMC_2 UMC_3 UMC_6 UMC_7
Assigned to
Alice
Cheng
Bob
(8) personalized privacy-respecting services for each user
Bob
Alice
Cheng

Fig. 3 Privacy-enabling User Modeling Process

As is shown in Figure 3, the user modeling process is controlled by the Selection Component. Assume that the three users have requested the website for recommendations of potential friends. The web server will bind to the Directory Component (an LDAP server in our implementation) and then send relevant user information to the individual user models. The Selection Component will check users' privacy constraints via a privacy context detection, which in turn will generate the bindings of each user modeling component for every user (i.e., whether or not it can be used according to the user's privacy constraints), and filter out the UMCs that are not allowed to operate.

The privacy constraints that apply to each of the three individual users and their implications to the UMCs are discussed below:

For Alice, Germany's Tele-Services Data Protection Law applies:

- UMC_4, $UMC_{5,}$ UMC_8 are illegal because the law prohibits combining user profiles retrievable under pseudonyms with data relating to the bearer of the pseudonym.
- UMC_7, UMC_8 are illegal because the law mandates personal data to be erased immediately after each session except for very limited purposes.

Therefore, UMC_4, $UMC_{5,}$ $UMC_{7,}$ UMC_8 cannot be used for Alice.

Despite no privacy law that can apply to Cheng was found, she has her own personal privacy preference as "dislike being tracked". $UMC_{6,}$ $UMC_{7,}$ UMC_8 cannot be used because the system cannot keep track of the pages she visits on UniversalFriends.com.

For Bob from the United States, UMC_4, UMC_5 and UMC_8 cannot be used according to the NAI self-regulation [23] if he does not give consent on merging non-personally identifiable use data with personally identifiable demographic data.[5]

To provide privacy-enhanced personalized services to users, the Selector Component will produce a PCS vector and instantiate a new run-time system instance for each user, or use an existing instance if its PCS is the same as that of another user.

6 Conclusions

Our approach facilitates the construction of personalized websites operate in a privacy-aware manner (both with respect to legal and user requirements). Our software product line approach allows personalized websites to address the combinatorial complexity of privacy constraints in a systematic and flexible manner, which builds on state-of-the-art industry practice for managing software variants at runtime. We aim at exploring the feasibility of this approach using an existing user modeling server and empirically established privacy constraints.

[5] We can choose a single method from the set of permissible methods by using a partial selection preference order derived through domain analysis at design time.

References

1. Kobsa, A., J. Koenemann and W. Pohl, *Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships.* The Knowledge Engineering Review, 2001. **16**(2): p. 111-155.
2. Hof, R., Green, H., and Himmelstein, L., *Now it's YOUR WEB*, in *Business Week Oct. 5*. 1998. p. 68-75
3. Personalization Consortium, *Personalization & Privacy Survey.* 2000, Personalization Consortium: Edgewater Place, MA. http://www.personalization.org/SurveyResults.pdf
4. FOR, *The Privacy Best Practice.* 1999, Forrester Research: Cambridge, MA
5. IBM, *IBM Multi-National Consumer Privacy Survey.* 1999, IBM. http://www.ibm.com/services/files/privacy_survey_oct991.pdf
6. DePallo, M., *AARP National Survey on Consumer Preparedness and E-Commerce: A Survey of Computer Users Age 45 and Older.* 2000, AARP: Washington, D.C. http://research.aarp.org/consume/ecommerce.pdf
7. Teltzrow, M. and A. Kobsa, *Impacts of User Privacy Preferences on Personalized Systems: a Comparative Study,* in *Designing Personalized User Experiences for eCommerce,* C.-M. Karat, J. Blom, and J. Karat, Editors. 2004, Kluwer Academic Publishers: Dordrecht, Netherlands.
8. Chen, Z. and A. Kobsa, *A Collection and Systematization of International Privacy Laws, with Special Consideration of Internationally Operating Personalized Websites.* 2002. http://www.ics.uci.edu/~kobsa/privacy
9. DE-TS, *German Teleservices Data Protection Act.* 1997. http://www.datenschutz-berlin.de/recht/de/rv/tk_med/iukdg_en.htm#a2
10. Kobsa, A. and J. Schreck, *Privacy through Pseudonymity in User-Adaptive Systems.* ACM Transactions on Internet Technology, 2003. 3(**2**): p. 149-183.
11. *Personal Communication, Chief Privacy Officer, Disney Corporation. 20*02
12. *Personal Communication, Chief Privacy Officer, IBM Zurich. 20*03.
13. Kobsa, A., *Generic User Modeling Systems.* User Modeling and User-Adapted Interaction, 2001. 11(**1**-2): p. 49-63.
14. Bosch, J., *Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach. 20*00, New York: Addison-Wesley
15. van der Hoek, A., *Design-Time Product Line Architectures for Any-Time Variability.* Science of Computer Programming, special issue on Software Variability Management, 2004. 53(**30**): p. 285-304.
16. ArchStudio, *ArchStudio 3.0. 2*005. http://www.isr.uci.edu/projects/archstudio/
17. Dashofy, E.M., A.v.d. Hoek, and R.N. Taylor. *A Highly-Extensible, XML-Based Architecture Description Language. in Proccedings of The Working IEEE/IFIP Conference on Software Architecture. 2*001. Amsterdam, Netherlands.: IEEE Computer Society.103-112
18. Fink, J., *User Modeling Servers: Requirements, Design, and Evaluation. 20*04, IOS Press, Netherlands (Infix)
19. Cranor, L., M. Langheinrich, and M. Marchiori, *A P3P Preference Exchange Language 1.0 (APPEL1.0): W3C Working Draft 15 April 2002. 2*002. http://www.w3.org/TR/P3P-preferences

20. Schunter, M. and C. Powers, *The Enterprise Privacy Authorization Language (EPAL 1.1): Reader's Guide to the Documentation. 2*003: IBM Research Laboratory. http://www.zurich.ibm.com/security/enterprise-privacy/epal/
21. Gandon., F.L. and N.M. Sadeh, *Semantic Web Technologies to Reconcile Privacy and Context Awareness.* Journal of Web Semantics, 2004. 1(3**)**: p. 241-260. doi:10.1016/j.websem.2003.07.008
22. Taylor, R.N., et al, *A Component- and Message-Based Architectural Style for GUI Software.* IEEE Transactions on Software Engineering, 1996(22(6), June, 1996): p. P.390-406.
23. NAI, *The Network Advertising Initiative (NAI) Self-Regulatory Principles.* 2001, Network Advertising Initiative