# Privacy Policy Negotiation Architecture for Pervasive Computing Environments

Nidal Qwasmi, Khalil El-Khatib, Ramiro Liscano, Julie Thorpe
University of Ontario Institute of Technology
Oshawa, Ontario, Canada

nidal.qwasmi@uoit.ca, khalil.el-khatib@uoit.ca, ramiro.liscano@uoit.ca, julie.thorpe@uoit.ca

*Abstract*— **In a highly connected and collaborated world, privacy is always a pressing issue. New technologies, such as cloud computing and service-oriented architecture, made system collaboration possible, which, in return, made privacy and security a challenge. This research paper focuses on designing a comprehensive technical architecture to negotiate and preserve privacy policy in pervasive computing environments. The architecture addresses the problem of user's privacy by developing dynamically controlled mechanisms that would allow the inhabitants of the environment to control what information is collected about them, how it is being processed, and under what circumstances it can be shared. The architecture allows users to have more control and negotiate their privacy policy as well as resolve any conflict that may occur during normal operations of the system.**

***Keywords- APPEL; privacy; architecture; conflict; P3P; resolution; exchange; negotiation.***

## I. INTRODUCTION

Establishing and maintaining individual privacy policies for Internet users remains unpopular mainly due to the technology limitations such as the complexity of Platform for Privacy Preferences (P3P), Preference Exchange Language (APPEL) and the "take it or leave it" approach for the web which make any privacy solution deployment a challenge. However, the Platform for Privacy Preferences (P3P) and preference language called APPEL [1] are considered the most recognizable efforts to enable users to have control over their privacy while online. P3P became a W3C recommendation in April 2002 [2] while APPEL became a working draft on April 20th, 2000 [2]. P3P and APPEL interact to enable a machine to programmatically check and apply user's privacy preferences and policy. Unfortunately, P3P and APPEL do not interact very well due to fundamental design choices that result in serious problems when using APPEL to define user's preferences [3]. Deficiencies with APPEL are mainly because users can only define what is unacceptable, not what is acceptable. In addition, APPEL rules are hard to express which makes it inefficient and tedious [3].

Negotiation mechanism was a working item for W3C in their early drafts of the P3P specification, but that initiative had been abandoned in support of easy and quick implementation. Negotiation mechanisms are still not part of the latest version of the P3P 1.1 specification [2]; however, it is planned for future versions of P3P [4]. Policy conflicts may cause serious privacy breaches and vulnerabilities such as blocking legitimate operations or permitting unwanted operations, hence privacy negotiation becomes a very important requirement for any privacy architecture in practice. Many real life examples demonstrate the importance and usefulness of negotiation features to any privacy architecture, for example:

- In medical environments [5], where a patient negotiates his own privacy policy with his nursing service provider.
- In distance education [6], where, for example, a trainee negotiates the access limit to his education record with an education center.
- In online retailing [7], where a customer negotiates how his address information would be handled with an on-line bookstore.

Our contribution in this research is presenting a new design of a privacy architecture for pervasive environments that supports negotiation, utilizing available technologies (P3P and APPEL). Adapting P3P and APPEL for a pervasive computing environment requires fundamental changes to P3P XML policy structure as well as its reference language APPEL. Some of these changes are necessary to accommodate the special needs of the environment; other changes are dictated by the basic system requirement of enabling conflict negotiation and resolution. Our suggested changes to P3P and APPEL are discussed in Sections III and IV. The rest of the paper is organized as follows. Section II presents related work. In Section III, we describe privacy policy for pervasive computing environment. In Section IV, we discuss reference language for pervasive computing environment. In Section V, we present our proposed architecture for privacy policy negotiation. Section VI presents our research conclusion and future work.

 ISBN: 978-1-61208-287-5 

## II. RELATED WORK

There are many previous research papers related to the topic of privacy protection in pervasive computing environments. Here, we describe these papers and how they differ from our present paper.

Shankar et al. [8] presented the necessity of having a policy-based management architecture, which is capable of handling conflict between the policies within the system. The proposed architecture is based on the concept of Event-Condition-PreCondition-Action-Post Condition (ECPAP). To accomplish this task, the authors proposed providing an axiomatic specification of rule actions to define the status of the impacted system before and after the execution of a policy. The authors in this research acknowledged the issue of policy conflict and proposed an approach to detect the conflict; however, they did not propose any viable way to dynamically resolve this conflict.

Langheinrich [9] provided comprehensive information about ubiquitous computing environments, which covers background analysis, privacy mechanisms and principles, privacy awareness systems, and related work. The authors also proposed a conceptual privacy system architected for ubiquitous computing environments, which is based on the following three main concepts:

1. An upfront announcement of a system's privacy policies.
2. An automatic configuration of available services based on the user's privacy policy.
3. An enforcement of privacy policy usage on the stored and collected information.

Langheinrich's work inspired us in designing our proposed system as it provides a set of privacy principles. However, this research did not address the issue of conflict between privacy policies and preferences.

Babbitt et al. [10] present the necessity of having a privacy management system (PMS) in ubiquitous computing environments where inhabitants can have control over their privacy. To accomplish this task, the authors proposed to create a conceptual model for a privacy management system, which can be used in the implementation process of any smart home environments. This research highlights many privacy issues within the ubiquitous environment.

Lahlou et al. [11] explained the gravity of infringing upon an individual's privacy. They also touched on the reasons behind taking privacy issues lightly by technical professionals. The authors also present the European Union's approach to enhance individual's privacy within ubiquitous computing systems. This research presents real life cases of privacy violation as well as the importance of having a privacy negotiation mechanism, which can help avoiding any unwanted consequences of privacy violation.

El-Khatib [7] has discussed a conceptual privacy negotiation protocol (PnP) protocol for web services. The author explained the benefits of having a privacy negotiation protocol, and presented an architecture for a negotiation protocol that can be used with web services. To implement the suggested protocol, the author envisions using P3P and APPEL with modifications. Our present paper differs from [7] in the targeted environment, as we envision our proposed architecture to be implemented in client-server rather than B2B (Business-To-Business) environments.

Kapadia et al. [12] presented a model for protecting user's privacy by giving users control over the dissemination of their digital footprint such as location or other private habits. Their model is based on building virtual walls, which simulate the actual live physical walls and follow the same concept of real life privacy protection mechanisms. The proposed virtual walls model is based on three levels of transparency, which are: transparent, translucent, and opaque. Users in this model can choose any level of transparency, which translates into a level of their own privacy protection. Authors in this research designed pre-determined levels of privacy and assumed that it will be applicable to all users. We believe that user privacy preferences are varied and it is too complex to be summarized in a limited number of choices.

Schmandt et al. [13] listed many real life examples and issues to support the importance of protecting an individual's privacy. Some of these examples are surveillance, grocery shopping in an intelligent environment, exchange of personal information over the Internet. The authors anticipated a controversial issue in the case of users having an interface to control their own privacy that is "what is the level of granularity at which users wish to control what is revealed about them. And how those choices are expressed" [13].

Babbitt et al. [14] proposed a privacy system model based on an authorization model, the administration model, the domain model, and the performance model. The authorization model is responsible for controlling how users, services, devices, and policies act and interact. The administration model is responsible for monitoring the changes that users need to apply on the authorization model. The domain model is responsible for controlling the context of where the user is trying to access the system, as different domains would need different privacy policies. The performance model is in charge of overseeing the efficiency and scalability of the privacy systems. This model does not deal with policy conflict or negotiation.

## III. PRIVACY POLICY FOR PERVASIVE COMPUTING ENVIRONMENT

In order to adapt P3P specifications into a pervasive environment, some modifications are necessary on the privacy policy file structure (XML); these modifications are mainly to accommodate the idea of communicating simultaneously to multiple devices with different policies while providing the capability of automated or semi-automated negotiation between a system and its clients.

Fig. 1 shows the existing P3P policy file structure, while Fig. 2 shows our modified version of the P3P policy file structure. The difference is that we have added an

 ISBN: 978-1-61208-287-5 

additional segment for negotiation and changed the existing "data segment". The proposed changes are explained here for each segment:

### A. Negotiation-group Segment:

This new segment is mandatory. The purpose of this new segment is to allow for automated or semi-automated negotiation between a system and the system's clients. In the normal state of the system, this segment should be free of any conflict elements but if conflict exists, then either side (system or client) may populate the resolution part of the segment, depending on which side had initiated the conflict. In this case, one of the two sides (system or client) either accepts the resolution or rejects it. When any of the two sides (system or client) rejects the resolution(s) then the receiver side may propose another resolution for the conflict. The negotiation process would continue until mutual agreement has successfully concluded or connection ended (time-out).

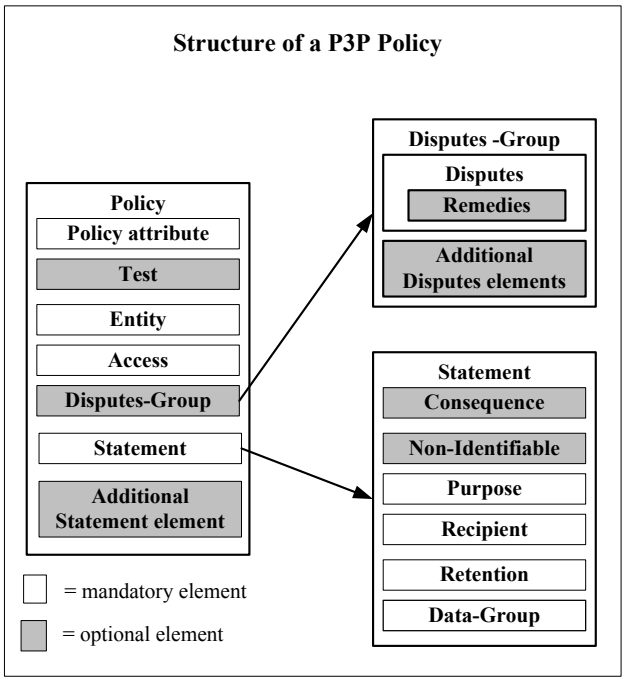


Figure 1. P3P XML policy structure [15]

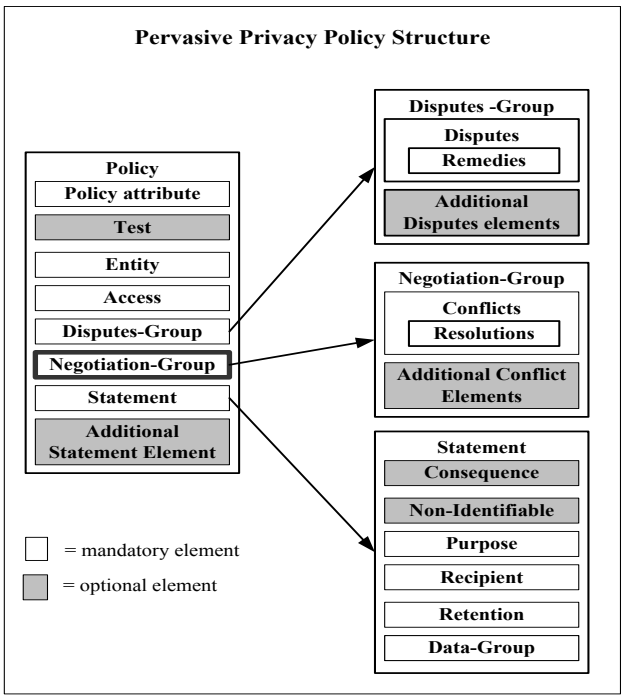


Figure 2. Privacy policy structure for pervasive computing

For our proposed model to work, we need a simple rule-set language which can be used to set and control agent's rules, conflict and resolution. The Extensible Stylesheet Language family (XSL) [18] would be an ideal tool to serve our model so the system can automatically negotiate privacy policies for the client's device(s). We discussed available language options in more detail in Section IV. This segment will also show the importance of resolving the policy's conflicts as well as providing a means to inform the client of any privacy policy violation and available paths to resolution. At the same time, this change would serve the purpose of increasing the level of client privacy by informing the user right from the beginning about any privacy policy conflict or violation and engaging the system's automated negotiation process in order to resolve the conflict according to the client's rules without much involvement from the client or disruption to the device's normal activities. If applicable, the negotiation segment would also contain data about the conflict resolution process which has not been successfully resolved by the negotiation process. Example of such data could be about other alternatives to resolve a conflict, which still exists after the end of system automated negotiation.

### B. Data-group Segment:

This group is expanded into three sub-segments: the first is for mandatory device information, the second is for personal information, and finally the third is for optional additional data that needs to be collected.

 ISBN: 978-1-61208-287-5

## IV. REFERENCE LANGUAGE FOR PERVASIVE COMPUTING ENVIRONMENT

APPEL is privacy reference execution language which is currently in use for privacy over the Internet; however it has been retired [16]. Another alternative is to use the XPref [3] and the Enterprise Privacy Authorization Language (EPAL) [17]. EPAL is a language for writing enterprise privacy policy and exchanging it in a standard format between applications or enterprises [17]. The main purpose of EPAL is to control data usage in the enterprise system according to fine-grained positive and negative authorization rights [17]. EPAL is very useful for enterprise IT systems but provide little or no benefit for client side interface where policy negotiation should take place, therefore EPAL with its features and capabilities is not suitable for client side interface.

XPref [3] is a script language based on XPath [19] and borrows some of its syntax and semantics from APPEL. The main goal of designing XPref is to overcome the deficiencies in APPEL, which are explained in [3]:

- Allows specifying unacceptable rules, but not acceptable rules.
- Rules are constrained with limited number of logical expressions that could be provided in a single rule.

Despite the good features and great flexibility that comes with the XPref language, it is still not capable of handling the negotiation part of the system. Therefore, our search extended to other alternatives that may satisfy our system requirements. The first step in selecting any programming language is to define the requirements of the system we intend to use it for. Accordingly, our system requirements are: XML based language, mature with international standard, easy to use, simple syntax, can accept XML file as an input and generate an XML file as an output, and lightweight so it can be installed on devices with low computation power such as PDA and cell phones.

To implement the system, our search concluded by selecting The Extensible Style-sheet Language Family (XSL) [18], more specifically XPath [19] and XSLT [18]. The reasons behind our selections are:

1. XSL is mainly designed for XML based applications.
2. It is a mature language as it is adopted by W3C.
3. It has a simple and easy syntax.
4. It can transform any XML file into another XML file (same or different structure), HTML, or XHTML
5. XSL client can be a standalone or can be add-on to a web browser. Most of the Internet browsers are supporting XLS. Firefox from version 1.0.2, Mozilla, Netscape from version 8 (uses the Mozilla engine); Opera from version 9 and Internet Explorer from version 6 all have supported XML and XSLT (and CSS). If a web browser add-on is being used then XSLT and XML files can be in separated files and implemented directly or "javascript-ECMAScript-262" [20] can be used to process the document.

The Extensible style-sheet language family consists of the following three main components.

- XSLT: language for transforming XML files [18].
- XPath: an expression language used by XSLT to navigate through an XML document [19].
- XSL-FO: an XML vocabulary for specifying formatting semantics [18].

The main objective of using XSL is to analyze the XML file, which carries the privacy policy, apply system's or client's privacy preferences, and to resolve any privacy policy conflict. If a conflict occurred then a new XML file should be generated with its conflict segment been populated. Both side (client and system) should be able to receive, process and generate an XML file.

As shown in Fig. 3, XML standards will be used to create privacy policies (system and client) while XSLT will be used to express privacy preferences and resolve any conflict that may have occurred. As we mentioned previously, XSL can transform any XML file into another XML file, which may be (or may not) the same structure or it can transform it into HTML/XHTML file. Also, XSL uses XPath to access any part of the XML file and it uses XSL-FO to format/control the presentation layer. These features become very valuable when a user interface is needed such as when a user needs to be prompted for a decision.

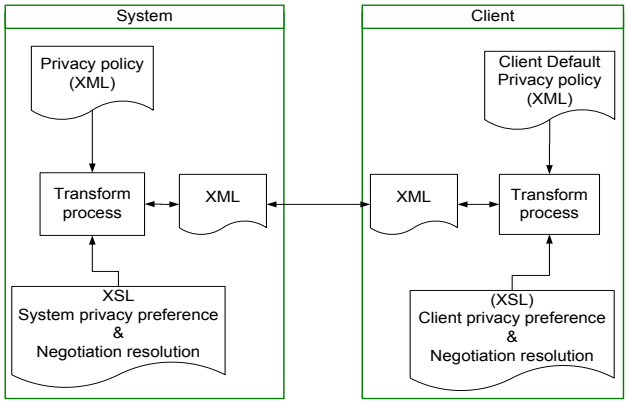


Figure 3. Privacy policy processes

## V. PRIVACY POLICY NEGOTIATION

Steven L. McShane defines negotiations as a "process occuring whenever two or more conflicting parties attempt to resolve their divergent goals by redefining the terms of their interdependence" [21]. Although this definition was in the context of human behavior, it is also applicable to privacy policy negotiation. If we take a closer look at the definition, then we find that it has three keywords that can summarize the whole process of negotiation; they are conflicting, redefining, and resolve. Indeed, it required a conflict to occur between two or more parties about something in order to start a negotiation process, which is

 ISBN: 978-1-61208-287-5 

using redefining techniques of terms and conditions to achieve the ultimate outcome of the process that is resolving the conflict. Based on this definition, a basic pervasive policy negotiation process might proceed as follows:

1. The server system tries to automatically access the client's privacy policy. The client can store the privacy policy in a pre-determined location (public access) on the device where the system can access it automatically. In the absence of the client's privacy preferences policy, the system will send its own privacy policy file (XML file format).
2. If the client's device has a publicly accessible privacy policy preference file, then the system tries to apply the client's privacy policy preferences. In the case of any conflict, the system populates the conflict and resolution segments (if applicable) in an XML file and sends it back to the client. In the absence of privacy preferences file from the client device, the server system will send its own privacy policy file. Upon receipt, the client's device using the reference language (XSL) will try to apply the client's privacy rules, fill the conflict and resolution segments (if applicable), and then send the XML file back to the server.
3. Normal operation status required that no conflict should exist.
4. If conflict existed, then either side would automatically evaluate the resolution(s). In this case either side has three options:

a) Accept the resolution and change the privacy policy accordingly.

b) Add a new resolution. Here it implies the other side had rejected the suggested resolution and would like the client to suggest another one.

c) Reject the resolution. In this case it means either side doesn't agree to the service agreement and would like to walk away, which also require prompting the user about the conflict and asking for consent to leave the system service area.

5. After a pre-determined number of automatic negotiation cycles, the client will be prompted for a decision.

Fig. 4 shows the message sequence for exchanging the privacy policy between the client and the server system. Our assumption in this scenario is that the client doesn't allow public access to the privacy policy file.

Fig. 5 shows a graphical representation of our previously described process of privacy negotiation. The left side of the graph shows the activities taking place in the system side of the process (service provider). In the right side of the graph, it shows the activities taking place in the client side of the process (service consumer). Our graph clearly shows that links between the two sides are the transactions of exchanging privacy policy file (XML file). The process of applying the privacy preference and negotiation is completely dependent on the Extensible style-sheet language family; mainly XSLT and XPath.

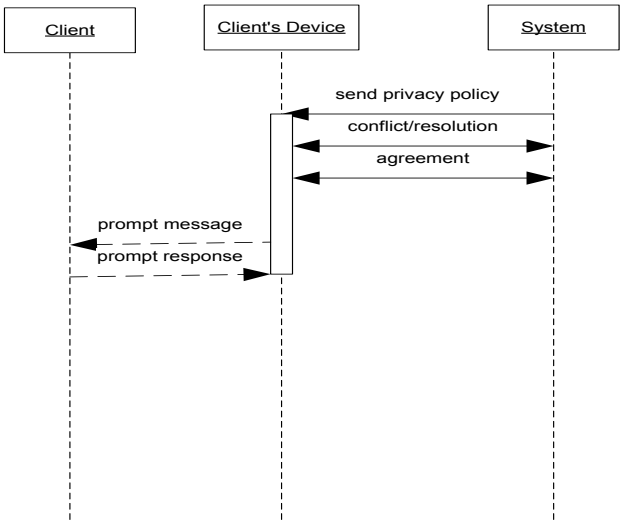


Figure 4. System sequence for accessing privacy policy

## VI. CONCLUSION AND FUTURE WORK

Our new architecture adapts P3P and APPEL for pervasive computing environments, which requires some fundamental changes to the P3P policy file structure as well as its reference execution language APPEL. Some of these changes are necessary to accommodate the environment special information needs. Other changes are dictated by the basic system requirements of enabling conflict negotiation and resolution.

Our proposed architecture requires changes to the P3P structure as well as its script reference language. Fig. 2 shows our modified version of the P3P policy file structure. As we have seen from previous sections of this research, APPEL is not even sufficient for static privacy policy needs. Our other alternatives we looked at into were XPref [3] and EPAL [17].

Our research concluded by selecting The Extensible Style-sheet Language Family (XSL) [18] more specifically XPath [19] and XSLT [18]. The reasons behind our selections are, first it is an XML based implementation, second it is a mature language, third its simplicity, fourth it can transform any XML file into another XML file (same or different structure), HTML, or XHTML, and fifth is XSL agent is flexible to implement.

With all the efforts that have been put into this research but we think that it is still coming short to cover some important areas of the privacy system for pervasive computing environment; especially the following two topics:

- Interface Agent: This research topic focuses on communicating between users and XLS. It covers the issues of converting user's privacy preferences to XLS rules. Although XLS syntax is easy to understand by technical professional but it is still very inconvenient for non-technical person to compose his preference file. Our visionary solution for this requirement is to be a Java

 ISBN: 978-1-61208-287-5 

standalone application utilizing standard web browser as presentation engine.

- Communication Agent: As we previously described in this research, the privacy system depends on exchanging privacy policies, which are an XML files. The need for communication agent is to facilitate, organize, and control the process of exchanging the privacy policy files between negotiated parties.

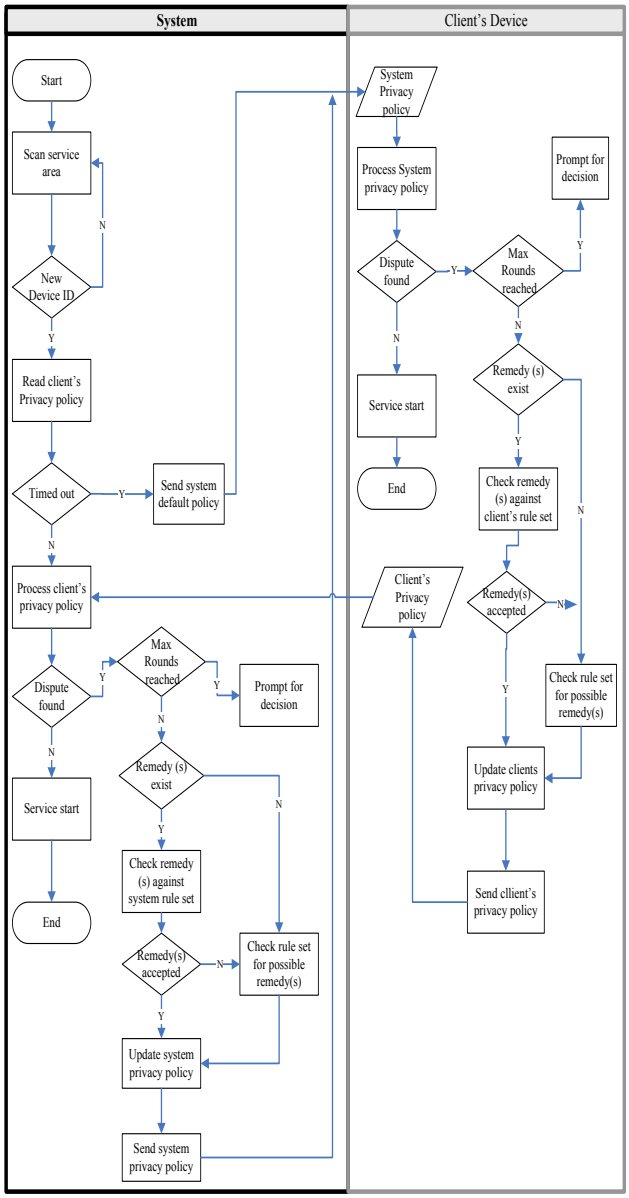


Figure 5. System negotiation flowcharts

## REFERENCES


[1] A P3P Preference Exchange Language 1.0 (APPEL1.0) http://www.w3.org/TR/P3P-preferences/ [retrieved: 04, 2013]

[2] The Platform for Privacy Preferences 1.0 (P3P1.0) Specification http://www.w3.org/TR/P3P/ [retrieved: 04, 2013]

[3] Agrawal, R., Kiernan, J., Srikant, R., Xu, Y., "XPref: a preference language for P3P", science direct-Computer Networks 48 (2005) 809–827, 16 February 2005

[4] W3C, P3P – Future Versions of P3P (Presentation, slide 10), http://www.w3.org/Consortium/Offices/Presentations/P3P/Overview.html#(10) [retrieved: 04, 2013]

[5] Yee, G., Korba, L., "Feature Interactions in Policy-Driven Privacy Management", Proceedings, Seventh International Workshop on Feature Interactions in Telecommunications and Software Systems, Ottawa, Canada, 2003

[6] Yee, G., Korba, L., "the Negotiation of Privacy Policies in Distance Education", Proceedings of the 4th International IRMA Conference. Philadelphia, Pennsylvania, USA. May 18-21, 2003.

[7] El-Khatib, K, "a Privacy Negotiation Protocol for Web Services", Workshop on Collaboration Agents: Autonomous Agents for Collaborative Environments Halifax, Nova Scotia, Canada. October 13, 2003

[8] Shankar, C., Campbell, R., "Policy-based Management Framework for Pervasive Systems using Axiomatized Rule-Actions", Fourth IEEE International Symposium on Network Computing and Applications, July 27-July 29 2005

[9] Langheinrich, M., "Personal Privacy in Ubiquitous Computing - Tools and System Support." Zurich, Switzerland: ETH Zurich, 2005.

[10] Babbitt, R., Information privacy management in smart home environments: Modeling, verification, and implementation. Proc. of 30th Annual Int. Computer Software and Application Conf. COMPSAC 2006, vol. 2, 2006, pp. 344-346.

[11] Lahlou, S., Langheinrich, M., and Ro¨cker, C., "Privacy and Trust Issues with Invisible Computers," Comm. ACM, vol. 48, no. 3, pp. 59-60, 2005.

[12] Kapadia, A., Henderson, T., Fielding, J., Kotz, D., "VirtualWalls: Protecting Digital Privacy in Pervasive Environments", In Proc. of the 5th Int'l Conf. on Pervasive Computing, 2007

[13] Schmandt, C., Ackerman, M., "Personal and Ubiquitous Computing Issue on privacy and security", Journal Personal and Ubiquitous Computing, Volume 8 Issue 6, November 2004

[14] Babbitt, R.; Wong, J.; Chang, C.; , "Towards the Modeling of Personal Privacy in Ubiquitous Computing Environments," Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International , vol.2, no., pp.695-699, 24-27 July 2007

[15] Cranor, L. F., "Privacy Policy, Law, and Technology", http://lorrie.cranor.org/courses/sp04/p3p-slides.pdf , p 19, [retrieved: 04, 2013]

[16] W3C http://www.w3.org/TR/#tr_P3P [retrieved: 04, 2013]

[17] Enterprise Privacy Authorization Language (EPAL 1.2), http://www.w3.org/Submission/2003/SUBM-EPAL-20031110/ [retrieved: 04, 2013]

[18] The Extensible Stylesheet Language Family (XSL) http://www.w3.org/Style/XSL/ [retrieved: 04, 2013]

[19] XPath http://www.w3.org/TR/xpath20/ [retrieved: 04, 2013]

[20] ECMAScript, http://www.ecmascript.org/docs.php [retrieved: 04, 2013]

[21] McShane, S., "Canadian Organization Behaviour", fifth edition, McGraw-Hill, 2004


 ISBN: 978-1-61208-287-5 