DOI : 10.3745/JIPS.2009.5.1.019

Privacy-Aware Adaptable Web Services Using Petri Nets

You-Jin Song* and Jae-Geol Yim**

Abstract: Many researchers have developed frameworks that are capable of handling context information and can be adapted and used by any Web service. However, no research involving the systematic analysis of existing frameworks has yet been conducted. This paper examines the Context Framework, an example of existing frameworks, using a Petri net, and analyzes its advantages and disadvantages. Then, a Petri net model – with its disadvantages removed - is introduced, and a new framework is presented on the basis of that model. The proposed PAWS (Privacy Aware Web Services) framework has a expandability for context management and communicates flexible context information for every session. The proposed framework can solve overhead problems of context in SOAP messages. It also protects user privacy according to user preferences.

Keywords: *Privacy, Web Service, Petri Net, Context Framework*

1. Introduction

The newly emergent ubiquitous computing environment has introduced a new paradigm of computing in which machines are able to recognize human conditions and situations and actively predict user needs, enabling them to provide the appropriate services to users instead of passively performing user requests. Therefore, the keyword in ubiquitous computing is context awareness. This context awareness cannot be realized unless all machines are connected and interchangeable. As a platform independent software application, Web services provide interchangeability in the ubiquitous environments. Therefore, it is expected that Web services will be chosen as the standard software technology for the ubiquitous environments of the future.

In order to implement a ubiquitous Web service environment, a component for processing context-awareness function, which is a major feature of ubiquitous environments, is required. However, the current standard Web service does not deal with the handling of context. As a consequence, a Web service can handle only context considered at the stage of designing the Web service. This is undesirable in a ubiquitous environment, as various contexts exist. Therefore, a framework which can conveniently add context information is required.

Markus Keidi et al. designed a Context Framework for Web services, so that a Web service can be adapted to context-aware environments [2][4]. The Context Framework performs pre/post processes on context blocks in an SOAP message. Therefore, contexts which were not considered at the design stage of the Web service can also be delivered.

However, the Context Framework does not mention the reasons for the establishment of context information [2][4]. In this case, it is not possible to determine what elements are required for a Web service. In addition, as the number of context elements in the SOAP message increases, overhead increases.

This paper proposes a new design for a framework in which the disadvantages of the existing frameworks are removed. The proposed framework is flexible in terms of context because it establishes context information when Web services are operating. In addition, the proposed framework prevents the exposure of private information from the very beginning, by reflecting user preferences in the establishment of context. Using these methods, private information can be protected, based on privacy levels.

Manuscript received September 10, 2008; accepted January 7, 2009.
This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST)(NO. R01-2008-000-20062-0(2008)).

Corresponding Author: You-Jin Song
* Dept. of Information Management, Dongguk University, Gyeongju, Korea (song@dongguk.ac.kr)
** Dept. of Computer and Multimedia, Dongguk University, Gyeongju, Korea (yim@dongguk.ac.kr)

2. Related Works

2.1 Context Framework [2]

Markus Keidi et al. designed a Context Framework which can handle context information and be adapted and used by any Web service. Context is processed as shown in Fig. 1 in the environment of a Context Framework.

① Applying the pre-process to SOAP request messages
② Delivering the context to Web services through an invocation manager
③ Receiving a reply from the Web service
④ Including the context header block in the SOAP response messages through the post process

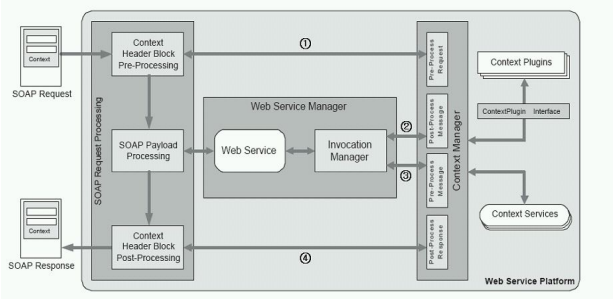


Fig. 1. Context-Based Web service Framework

However, the context framework has the following problems.

SOAP overhead problem: As the development of technology continues to progress, the information needed by a session will become ever more complicated and diverse. As a result, the number of elements (such as location, role, activity, identity, time, and so on) constituting the context information will increase. Putting all of the elements at the SOAP header will result in an occurrence of overhead, the reason for this being that a session may not need all of the elements. Therefore, it should be possible to specify the particular set of context blocks required for a session and differentiate it from the set of context blocks required for a different session. Our method establishes the required information once per session and does not send unnecessary context blocks, thereby resolving the SOAP overhead problem.

Determination of context. The existing papers deal only with the methods of context delivery and not with the criteria for determining the context blocks to be sent. A context contains the following elements: location, role, activity, identity, time, etc. These elements can be defined in tModel [2]. However it is not possible to determine which context has been delivered for the current session. Our method solves this problem by determining the context blocks by referring to the elements demanded by the requester and the owner's preference.

User's privacy protection. The user's privacy protection is not dealt with. Context delivery increases the threat of attacks on a user's private information. If a user has agreed to disclose his/her information, then this does not pose a problem. Without such agreement, however, disclosure of the user's information - including the user's current location, identity, and role - will be problematic. A context type can be defined in a tModel but the user's preference cannot be defined. In such a case, an intrusion on the private information occurs and may result in a serious problem.

The methods proposed in this paper take the owner's preference into account when determining context information. Therefore, the context elements related with the user's private information will not be delivered and the privacy will be protected.

2.2 Petri net model of Context Framework

2.2.1. Concept of the Petri Net.

The Petri net, introduced in 1962, has been successfully used in system performance testing, communication protocol testing, distributed software systems, and so on. Building a Petri net model is relatively easy, and various mathematical methods of analyzing a Petri net are available. This is the main reason for the Petri net's success. The definition of the Petri net is presented in Definition 1. The definition of the transition firing rule is as follows:

(1) A transition t is said to be enabled if each input place p of t is marked with at least w(p,t) tokens, where w(p,t) is the weight of the arc from p to t.

(2) An enabled transition may or may not fire (depending on whether the event actually takes place).

(3) The firing of an enabled transition t removes w(p,t) tokens from each input place p of t, and adds w(t,p) tokens to each output place p of t, where w(t,p) is the weight of the arc from t to p.

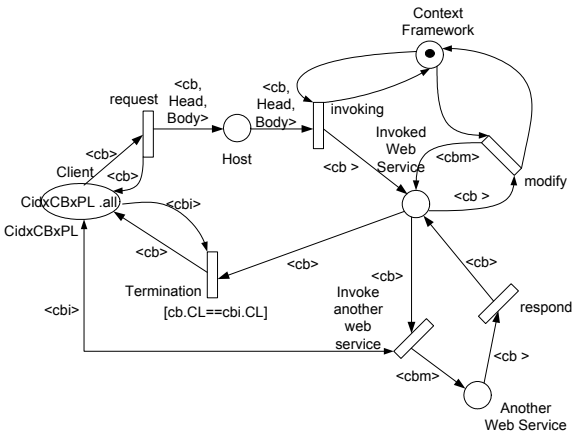
Definition 1. Petri Net

The Petri net is a 5-tuple, N = (P, T, F, W, M0) where:
P={p1, p2, ... pn) is a finite set of places,
T={t1, t2, ..., tm} is a finite set of transitions,
$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs,
W: $F \rightarrow \{1, 2, 3, ...\}$ is a weight function,
M0: $P \rightarrow \{0, 1, 2, 3, ...\}$ is the initial marking.
$P \cap T = \varnothing$ and $P \cup T \neq \varnothing$.

2.2.2. Petri Net Model Construction.

The Petri net model of the life-cycle of context in the Context Framework is presented in Fig. 2. The Place Client is associated with data type CidxCBxPL, which is the Cartesian product of Cid, CB and PL. Cid is defined as a set of integers representing the client identification number. CB represents a context block consisting of a set of context elements. PL represents the privacy level of the associated context block and, with level 4, is top secret. The '.all' at the end of CidxCBxPL.all is written in place of the Client, as a function returning all tuples included in the Cartesian product of CidxCBxPL. The 'cb' in the label <cb> is a variable which can be mapped into a tuple in CidxCBxPL. When a client requests a service, it inserts the context block <cb> into its SOAP message. The context information associated with the <cb> may be top secret. The message is sent to 'Invoked Web Service' and is then moved to 'Another Web Service' upon initiation of the 'Invoke another web service' transition. This result is disastrous because top secret information is exposed to another Web service; it is also disastrous to the 'other web service' because its context is also exposed to the caller Web service.

It is important to note that a context block consists of many elements and is inserted in the SOAP as a whole. This can result in SOAP overhead.



Cid: {1..100}; // a client´s identity number
PL: {0, 1, 2, 3, 4};
Location = product real*real; // coordinate
Action : string;
Time: integer;
Role: string;
EleName: {null, Location, Action, Time, Role};
ELE = product of EleName * PL;
CB = product of ELE*ELE*ELE*ELE*ELE;
CidxCBxPL = product of Cid * CB * PL;

Var cb, cbi, cbm : CidxCBxPL ; // cb stands for context block, cbi is another cb,
// cbm is a modified cb.

Fig. 2. Petri Net Model of Context Framework

As with Context Framework, context is delivered by putting it in the header of the SOAP as a context block. A context-type element, one of the context elements, has privacy level and context value. The privacy level and context value elements represent sensitivity and private data, respectively. The structure of the context block is described in Fig. 3. The framework proposed in this paper prevents the exposure of sensitive information by using the privacy level element.

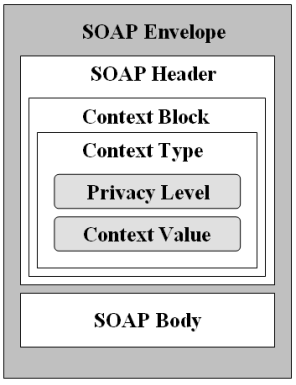
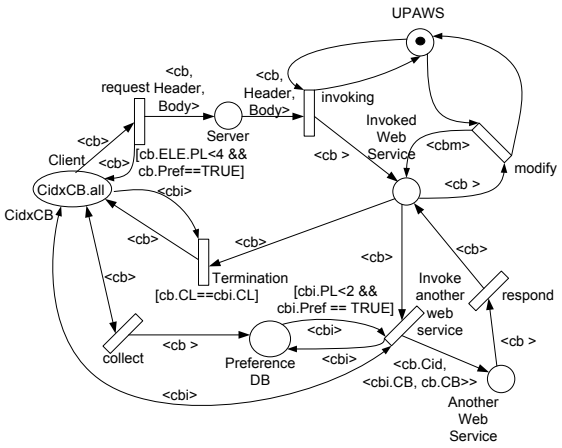


Fig. 3. CB (Context Block) Structure

3. Design of a New Framework

3.1 Modification of the Petri Net Model Constructed

Firing a ‘request’ with a token corresponding to secret context information results in disastrous results, as shown in Fig. 4.



Cid: {1..100}; // a client´s identity number
PL : {0, 1, 2, 3, 4};
Pref = product of Cid*Constraints; // A constraint is a simple proposition
Location = product real*real; // coordinate
Action : string;
Time: integer;
Role: string;
EleName = {null, Location, Action, Time, Role};
ELE = product of EleName *PL;
CB = product of ELE*ELE*ELE*ELE*ELE*Pref;
CidxCB = product of Cid * CB;

Var cb, cbi, cbm : CidxCB ; // cb stands for context block, cbi is another cb,
// cbm is a modified cb.

Fig. 4. Petri Net Model After Removing the Shortcomings of the Context Framework

This phenomenon can be avoided if a guard is attached to the transition ‘request’, so that secret information cannot move out. It has also been observed that if elements of the context block are examined one by one, and the elements related to the current session are transmitted, only then can SOAP overhead be avoided. For this purpose, user preferences are attached to a context block by using privacy level.

A transition which collects context blocks and saves preferences in a database can also be used. When another Web service is invoked, only a portion of its context, which the user’s preferences allow to be exposed, can be exposed to other Web services.

The privacy level element in the context block is presented in Table 1.

Table 1. Meaning of privacy levels

Privacy level	Meaning
4	Do not allow under any circumstances
3	Allow only to the current session, but saving is not allowed
2	Saving is allowed but use by the others is not allowed
1	Use by the others for permitted services is allowed
0	No restriction on the use of the information

Information can be delivered only if its privacy level is less than 4 and if a Web service treats the personal information as presented in Table 1 according to the level values. Personal information can be protected based on the user preference specified in the privacy level element.

3.2 PAWS Framework

3.2.1. Architecture.

The architecture of the PAWS framework is shown in Fig. 5.

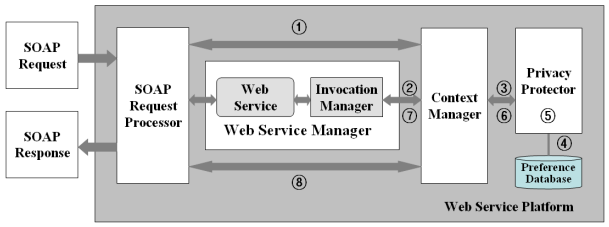


Fig. 5. The architecture of the PAWS framework

The safe delivery of a message and the context between the client and the Web service is basically achieved by the context manager and the privacy protector. A requesting client can transmit a message including context information to a web service via the SOAP request processor. The SOAP request processor can also receive context information from the context manager. The context manager must obtain permission from the privacy protector in order to include an item in a context. The privacy protector acquires the user's preferences, which are then stored in a preference database, and the user can modify his/her preferences at any time. A web service communicating with a client can invoke another web service, and the invoked web service can also use the user context.

3.2.2. Establishment of Context.

Context establishment consists of a sequence of procedures for establishing a context block. The steps for context establishment are represented in Fig. 5.

① Applying the pre-process to SOAP request messages
② Delivering the context to Web services through an invocation manager
③ Context Manager asks Privacy Protector whether the element can be opened or not
④ Privacy Protector obtains user preferences from the preference database
⑤ Privacy Protector determines the context block structure reflecting the user's preferences based on privacy level
⑥ Privacy Protector sends the context information included in the context block to Context Manager
⑦ Receiving a reply from the Web service
⑧ The context header block is included in the SOAP response messages via the post process

4. Comparative Analysis with Context Framework

Consideration of Preference. Context establishment is determined based on the user's preferences. Context delivery should be recognized by the user, and it should be possible to stop the delivery of context information if the user does not want to expose this information. The proposed method establishes a context based on the user's preferences. Therefore, private information is securely protected.

Privacy Protection. An application and a sensor in a ubiquitous environment take the current context into account when operating. If the user does not mind disclosing his/her personal information, then this will not be problematic; otherwise, revealing private user information - such as current location, identification, role, and so on - can result in privacy violations. The proposed method takes the user's preferences into account when determining whether a context block should be opened. Therefore, such a problem will not occur.

Determination of Context Per Session. The PAWS Framework determines a set of context items per session for Web service execution. Context Establishment establishes a set of context items per session. Therefore, those context items which are not needed in the current session are not delivered in the first place. Hence, our method is efficient.

SOAP Overhead. The Context Framework delivery of irrelevant context information is unavoidable. The proposed method minimizes the context to be transmitted during the process of Context Establishment. The transmission of irrelevant context causes such problems as intrusions on privacy and degradation of efficiency. In considering the negative effects of unnecessary information, the framework can optimize the number of elements used in a message. Hence, the overhead problem is solved.

A summary of the comparative analysis of the frameworks is presented in Table 2.

Table 2. A summary of comparative analysis of frameworks

Framework / Factor	Context Framework	PAWS
Expandability of Context	○	○
Determination of Context Per Session	×	○
SOAP Overhead	×	○
Consideration of Preference	×	○
Privacy Protection	×	○

○ : supported
× : Not supported

5. Conclusion

A Petri net model for a Context Framework has been proposed and its advantages and disadvantages have been compared. The model has been modified and its disadvantages removed. A new framework, the PAWS framework, has been designed on the basis of the modified model. The proposed PAWS framework has a expandability for context management and communicates flexible context information for every session. The proposed framework can solve overhead problems of context in SOAP messages and can also protect the user's privacy according to the user's preferences. The PAWS framework does not have the disadvantages of the Context Framework, and features all the advantages of the Context Framework. In the future, we plan to study experimental verification.

References

[1] Murata, T., "Petri nets: Properties, Analysis and Applications", Proceedings of the IEEE, Vol. 77, No. 4, 1989, pp. 541-580.

[2] Markus Keidi, Alfons Kemper, "Towards Context-Aware Adaptable Web Services", Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, ACM, pp.55 - 65, 2004.

[3] Patricia Dockhorn Costa, "Towards a Services Platform for Context-Aware Applications", International Workshop on Ubiquitous Computing (IWUC), INSTICC Press , 2004.

[4] Markus Keidi, Alfons Kemper, "A Framework for Context-Aware Adaptable Web Services", Proceedings of the Ninth International Conference, LNCS, Vol. 2992, pp. 826 - 829, 2004.

[5] W3C, "Simple Object Access Protocol (SOAP) 1.1", http://www.w3.org/TR/soap/

[6] Stephen Crane, "Privacy Preserving Trust Agents", HP Trusted Systems Lab, 2004.

[7] Gerhard Austaller, et al, "Using Web Services to Build Context-Aware Applications in Ubiquitous Computing", Web Engineering 4th International Conference, LNCS, Volume. 3140, pp.483-487, 2004.

[8] Richard Hull, et al, "Enabling Context-Aware and Privacy-Conscious User Data Sharing", IEEE International Conference on Mobile Data Management, pp.187- 198, 2004.

[9] Amr Ali Eldin, "An ICT Design Approach to Support Privacy Protection for Contextual Information", Proceedings of the 10th doctoral Consortium workshop in the CAiSE`03, 15th International Conference on Advanced Information Systems Engineering, 2003.

[10] Al-Muhtadi, J. et al., "Routing through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments", in Intl. Conf. Of Distributed Computing Systems (ICDCS 2002, pp. 65-74, 2002.

[11] R. Agrawal and E. L. Wimmers, "A Framework for Expressing and Combining Preferences", In the Proc. of the Intl. Conf. on Very Large Databases (VLDB), pp. 297–306, 2002.

[12] A. K. Dey, D. Salber, and G. D. Abowd, "A Context-based Infrastructure for Smart Environments", In the Proc. of the Intl. Workshop on Managing Interactions in Smart Environments (MANSE), pp. 114–128, 1999.

[13] M. Ebling, G. Hunt, and H. Lei, "Issues for Context Services for Pervasive Computing", In the Proc. of the Advanced Workshop on Middleware for Mobile Computing, 2001.

[14] W. Kießling, Foundations of Preferences in Database Systems, In the Proc. of the Intl. Conf. on Very Large Databases (VLDB), pages 311–322, 2002.

[15] W. Kießling and B. Hafenrichter, "Optimizing Preference Queries for Personalized Web Services", In the Proc. of the IASTED Intl. Conf. on Communications, Internet and Information Technology, pages 461–466, 2002.

[16] Universal Description, Discovery and Integration (UDDI) Technical White Paper. http://www.uddi.org, 2000.

[17] S. Rich´e and G. Brebner, "Storing and Accessing User Context", In Proc. of the Intl. Conf. on Mobile Data Management (MDM), LNCS, volume 2574, pages 1–12, 2003.

You-Jin Song

He received a BS degree in Engineering from Hankook Aviation Univ. in 1982. In 1987 he completed an MS degree at Kyungbuk National University, then joined the Electronics and Telecommunications Research Institute (ETRI). He also completed a Ph.D program at the Tokyo Institute of Technology in 1995. He has been a Professor of Dongguk Univ. since 1996. His research interests include cryptography and information security, privacy protection, and application security in ubiquitous computing.

Jae-geol Yim

He received his M.S. and Ph.D. degrees in Computer Science from the University of Illinois, Chicago, in 1987 and 1990, respectively. He is a Professor in the Department of Computer Science at Dongguk University in Kyungjoo, Korea.

His current research interests include Petri net theory and its applications, location-based services, AI systems, and multimedia systems.