Context-Aware Privacy Design Pattern Selection

Siani Pearson, Yun Shen

HP Laboratories
HPL-2010-74

Keyword(s):

Privacy, Design Patterns, Context Awareness

Abstract:

User-related contextual factors affect the degree of privacy protection that is necessary for a given context. Such factors include: sensitivity of data, location of data, sector, contractual restrictions, cultural expectations, user trust (in organisations, etc.), trustworthiness of partners, security deployed in the infrastructure, etc. The relationship between these factors and privacy control measures that should be deployed can be complex. In this paper we propose a decision based support system that assesses context and deduces a list of recommendations and controls. One or more design patterns will be suggested, that can be used in conjunction to satisfy contextual requirements. This is a broad solution that can be used for privacy, security and other types of requirement.

External Posting Date: July 6, 2010 [Fulltext] Approved for External Publication
Internal Posting Date: July 6, 2010 [Fulltext]
Published and presented at TrustBus 2010, Spain, May 16, 2010. The original publication is available at www.springerlink.com

Context-Aware Privacy Design Pattern Selection

Siani Pearson and Yun Shen

HP Labs Bristol
Long Down Avenue
Stoke Gifford
Bristol BS34 8QZ UK
{Siani.Pearson, Yun.Shen@hp.com}

Abstract. User-related contextual factors affect the degree of privacy protection that is necessary for a given context. Such factors include: sensitivity of data, location of data, sector, contractual restrictions, cultural expectations, user trust (in organisations, etc.), trustworthiness of partners, security deployed in the infrastructure, etc. The relationship between these factors and privacy control measures that should be deployed can be complex. In this paper we propose a decision based support system that assesses context and deduces a list of recommendations and controls. One or more design patterns will be suggested, that can be used in conjunction to satisfy contextual requirements. This is a broad solution that can be used for privacy, security and other types of requirement.

1 Introduction

There is increasing awareness that privacy should be integrated into design rather than being bolted on afterwards, and for the need to take privacy into account [**?**]. New regulations, consumer concerns and high profile cases of personal or sensitive data exposure are forcing companies to design more privacy-aware systems. Contextual and environmental factors should be taken account of in product and service design, but this can be very complex. Sometimes the time and expertise to do this is not readily available even with the presence of system administrators: this is especially the case for dynamic environments. User-related contextual factors affect the degree of privacy protection that is necessary for a given context. Such factors include: sensitivity of data, location of data, sector, contractual restrictions, cultural expectations, user trust (in organisations, etc.), trustworthiness of partners, security deployed in the infrastructure, etc. The relationship between these factors and privacy control measures that should be deployed is too complex to be modelled in a tabular form. By breaking the complex modelling issue down to relatively simple rules and combining these using the proposed reasoning engine, we are able to model the complex relationships mentioned above.

The core problems we address in this paper are:

- How to aid product and service design, whilst taking into account the context and environment in which the product or service is to be deployed.

– How to help non-expert developers/architects locate design patterns [**?**] that are particularly relevant to their problem space.

In essence, our solution to these problems is a system that gathers context relating to the design required and inputs this to a rule-based system, to trigger decisions about which control measures it could be appropriate to use within that context. The tool helps to determine appropriate design patterns that could be used to address privacy, security and other requirements. The solution is targeted at non-expert developers and architects. It may be useful in management products for servers, storage, networking, etc., in cloud environments and in other domains.

The rest of the paper is organised as follows. We describe our solution in Section 2. A detailed example is further discussed in Section 3 showing how the system can generate a list of candidate privacy design patterns with regard to specific contextual factors. We consider related work in Section 4 and finally conclusions are given in Section 5.

2 Framework Overview

Our approach is to use a specialised tool in order to aid a designer to make decisions. It is a type of decision-based support system that interacts with designers in order to gather appropriate context, and that assesses this context and outputs a list of recommendations and controls that it would be appropriate for the designer to use within this context. One or more design patterns [**?**] will be suggested, that can be used in conjunction to satisfy contextual requirements. (Further information about design patterns and how they are extended within our solution will be given in Sections 3 and 4). The solution is rule-based and functions as an expert system. A domain expert (or experts) will create the rules and patterns, based upon industry standard techniques and patterns for specific domains. There can be a feedback process by which an architect can choose a lower ranked pattern and this goes to improve the selection process. In the rest of this section, we consider in more detail the component parts of this system and their interactions, both internal and external.

An overview of the system is shown in Figure **??**. An expert administrator can tailor the rules if required, and the user of the system is the designer that wishes to obtain advice about which controls (in the form of design patterns) they should consider using for their situation. The user interacts with the system via a questionnaire, which asks the user questions about their current goals, context and preferences. This questionnaire could be static, but ideally would be generated via an expert system such that the questions will vary according to the previous answers of the user. When the questionnaire is completed, the system outputs a ranked list of design pattern candidates. For example, it might suggest usage of Design Pattern 1 for the current context, with confidence being 1.0, and also Design Pattern 2, with confidence that this is appropriate for the current context being 0.8.

A central aspect to this approach is the mechanism that selects design patterns in a context-aware manner, such that design pattern candidates are selected with regard to the context (e.g. customer requests, regulations, policies) and pre-defined rules/knowledge base. This mechanism includes:

1. a rules repository that has two major kinds of rules: rules that apply to a context handler and rules that apply to a design pattern selection processor
2. a context handler that quantifies the context factors by applying contextual rules
3. a design patterns repository that contains fine-grained design patterns with additional parameters relating to selection criteria
4. a pattern selector which (either through a graphical user interface (GUI) to manually or a programmed process to automatically) selects a list of design pattern candidates by finding a reasonable matching between the parameters expressed in the design patterns and the selection criteria generated by the context handler and the pattern filter rules.

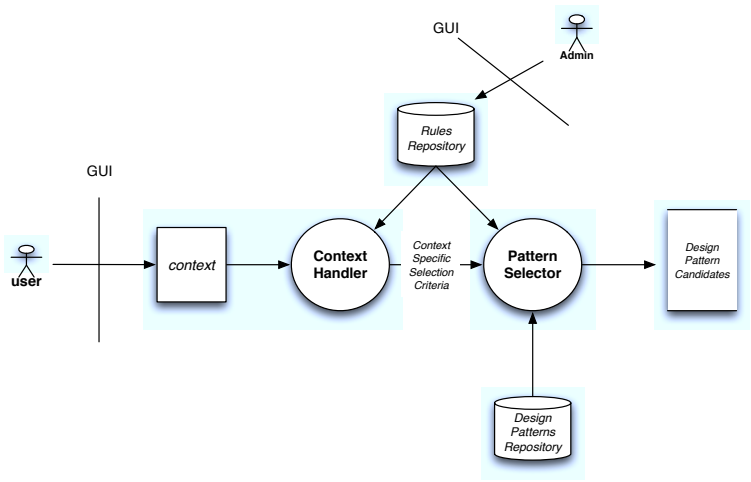

Fig. 1. Context Aware Design Pattern Selection

The main features of the system are as follows:

– **Design Stage Context** is collected from the user (in this case, the designer) through the GUI. The user can specify contexts using pre-defined keywords such as sensitive information, limited contractual restrictions, etc.
– **Context Processing Rules (CPR)** are defined by system administrator to handle contextual information specified by user. CPR will process applicable contextual information and output a set of selection criteria based on the context processing rules combination configuration

- **Context Processing Rules Combination (CPRC)** are defined to combine the results generated by the CPR process
- **Selection Criteria** are generated by the CPRC and used by the design pattern selection rules
- **Design Pattern Selection Rules (DPSR)** process selection criteria generated by the CPRC to determine a list of design pattern candidates that are close to the contextual information input by the user
- **Design Pattern Selection Rules Combination (DPSRC)** defines the algorithm to output a ranked list with regard to the DPSR

The generic process works as follows. A set of fine-grained design patterns are carefully formulated. Each design pattern has a criteria field in which a list of parameters relating to different context setup and selection criteria are defined. Two sets of rules are also constructed: one is context-related and another is pattern selection-related. The context handler responds to specific context setup at the design stage. Context-related rules are applied to quantify contextual information, e.g. 'sensitivity is high' is converted to 'sensitivity = 1.0'. After this process, contextual information collected through the GUI from the user is transformed into a list of quantified selection criteria. The pattern selector, either as a GUI or automatic process, applies a set of selection rules to match the previously generated selection criteria against the design patterns stored in the repository. The pattern selector outputs a list of design pattern candidates with a double value identifying how suitable the design pattern is to a specific context. We take an approach similar to Thesaurus [**?**]; this enables a better interaction between the system and the users.

3 Example of Our Approach

In this section we give a worked example where we base the representation used upon the design pattern format used within [**?**], and show what the corresponding rules look like in our system.

3.1 Design Stage Context

The supported privacy context is defined by the system administrator, i.e. designer of our solution (and referred to within the GUIs and rules, as explained above). It can include aspects such as:

- Sensitivity of data: {No Personally Identifiable Information (PII), PII, sensitive}
- Location of (stored) data
- Potential locations of transferred data
- Sector
- Number of users of system
- Whether an anonymous data set could be usable
- Contractual restrictions

- Cultural expectations
- User's role in the organisation
- Security deployed in the infrastructure
- Intent of system designer

Note that these are not the same as the context within the design pattern, nor the intent of the pattern (see Section 3.8, where we refer to this privacy context as a new category called 'applicable context' within the design patterns).

This Design Stage Context is collected from the user through the GUI. The user can specify contexts using pre-defined keywords such as sensitive information, limited contractual restrictions, etc. A natural language processing module can also be applied in this stage. Examples include the following context, where some phrases may be varied across pre-set options (e.g. {a large number, a small number, one, no}; {personal information, sensitive information, not personal information}):

1. Protected storage of data should be enabled over a large number of distributed servers in different countries.
2. Data is not sensitive personal information.
3. Assurances are required that the data cannot be re-assembled within a jurisdiction that is either not permitted to process the data set, or by a single malicious entity within the storage chain.
4. There are limited contractual restrictions between data storage locations.

3.2 Context Handler (CH)

The Context Handler (CH) processes and quantifies context information input by the user with regard to mutual agreement between the CH and Context Processing Rules. For example, the CH will process the context stated above as:

1. data.Location is cross-border
2. data.Sensitivity is non-sensitive
3. purposeOfProcessingData is restricted
4. contractualRestrictions is limited

3.3 Context Processing Rules (CPR)

Context Processing Rules (CPR) are stored in the Rules Repository and retrieved by the CH. They are defined by the system administrator to handle contextual information specified by the user. CPR will process applicable contextual information and output a set of selection criteria based on the context processing rules combination configuration. For example:

1. CPR.r1: If (data.Location is undetermined or data.Location is cross-border) then transborderDataFlow = true else transborderDataFlow = false;

2. CPR.r2: If (data.Sensitivity = sensitive) then sensitivity of information = 1.0 else sensitivity of information = 0.0;
3. CPR.r3: If(securityLevel < idealLevel & transborderDataFlowRestrictions = true) then sensitivity of information = 1.0
4. CPR.r4: If(purposeOfProcessingData is restricted) then limited usage = 0.4 else limitedUsage =0.0;
5. CPR.r5 If(contractualRestrictions is limited) then limitedLiability = 0.7;

3.4 Context Processing Rules Combination (CPRC)

Context Processing Rules Combination (CPRC) is defined to combine the results generated by CPR process. An example is listed below. It will generate four selection criteria: one criterion from CPR.r1, one criterion from maximum value between CPR.r2 and CPR.r3, one criterion from CPR.r4 and one criterion from CPR.r5.

1. CPR.r1 append max(CPR.r2, CPR,r3) append CPR.r4 append CPR.r5

3.5 Selection Criteria (Generated by CPRC)

Selection Criteria are generated by CPRC within the CH and are passed to the Pattern Selector. They are used by the Pattern Selector in conjunction with design pattern selection rules. For example:

1. Sensitivity of information = 1.0 (result from max(CPR.r2, CPR.r3))
2. transborderDataFlow = true; (result from CPR.r1)
3. limited usage = 0.4 (result from CPR.r4)
4. limited liability = 0.8 (result from CPR.r5)

3.6 Design Pattern Selection Rules (DPSR)

Design Pattern Selection Rules (DPSR) are used by the Pattern Selector to process selection criteria generated by the CPRC to determine a list of design pattern candidates that are close to the contextual information input by the user. For example:

1. DPSR.r1. If (Sensitivity of information = 1.0) then DP1 = 1.0, DP2 = 0.6;
2. DPSR.r2 If (transborderDataFlow = true & limited usage > 0.3) then DP2 = 0.8, DP1 = 0.3;
3. DPSR.r3 If (limitedLiability > 0.5) then DP2 = 0.8, DP1 = 0.6

3.7 Design Pattern Selection Rules Combination (DPSRC)

Design Pattern Selection Rules Combination (DPSRC) defines the algorithm to output a ranked list with regard to the DPSR. For example:

1. Max(DPSR.result.all.DP1) append Max(DPSR.result.all.DP2)

Result
The system outputs a ranked list of design pattern candidates.

1. DP1, confidence is 1.0
2. DP2, confidence is 0.8

3.8 Example Design Patterns

Our solution is independent of any particular format of design pattern: an additional *applicable context* field just needs to be added into the pattern format used that is reasoned about within the pattern selection process. We show below some privacy-related design patterns that we have defined to illustrate the type of patterns that might be deployed in the knowledge base. Note that these two design patterns have the same set of Selection Rules which generate different results with regard to specific contextual factors.

1. Design Pattern 1 (DP1) Obligation Management

Applicable Context:

- Sensitivity of data
- Location of (stored) data
- Potential locations of transferred data
- Cultural expectations
- Number of users of system
- Would an anonymous data set be usable?
- Contractual restrictions
- Security deployed in the infrastructure
- Conformance to existing agreements between parties or compatibility with legacy systems

Selection Rule Repository: DPSR
Selection Rules: DPSR1.r1, DPSR1.r2, DPSR1.r3
Name: Obligation **Classification:** Data and policy management
Intent: to allow obligations relating to data processing to be transferred and managed when the data is shared
Motivation: A scenario where this would be useful is when a service provider (SP) subcontracts services, but wishes to ensure that the data is deleted after a certain time and that the SP will be notified if there is further subcontracting
Context: You are designing a service solution. You want to make sure that multiple parties are aware of and act in accordance with your policies as personal and sensitive data is passed along the chain of parties storing, using and sharing that data.
Problem: Data could be treated by receivers in ways that the data subject or initiator would not like, and/or the data subject may be contacted in ways that they would not like e.g. being contacted by a call centre when they had expressed that they did not wish to be contacted. Furthermore, the original service provider

may be legally liable if this happens (e.g. according to APEC accountability-related legislation). In addition, data could be received by receivers in ways that they would not agree with or is not conforming to the initial agreement between parties, e.g. data subject or initiator changes the data transfer protocols or predefined communication channels.
Solution: all the service providers use an obligation management system. Obligation management can handle information lifecycle management, driven by individual preferences and organisational policies. A scalable obligation management system could be deployed, driven by obligation policies and individuals preferences that would manipulate data over time, including data minimisation, deletion and management of notifications to individuals.
Consequences:

Benefits - privacy preferences and policies can be conveyed along the chain and acted on in an operational manner.

Liabilities - extra workload in that users or organisations need to set obligations.
Known uses: Pretschner et al [**?**] provide a framework for evaluating whether a supplier is meeting customer data protection obligations in distributed systems. IBM proposed Enterprise Privacy Authorization Language (EPAL) [**?**] to to govern data handling practices in IT systems according to fine-grained positive and negative authorisation rights. Casassa Mont [**?**] discussed various important aspects and technical approaches to deal with privacy obligations.
Related patterns: sticky policies (obligations can be stuck to data), identity management (e.g. user-centric obligations managed by identity management system)

2. Design Pattern 2 (DP2) Sticky Policies

Applicable Context:

- Sensitivity of data
- Location of (stored) data
- Potential locations of transferred data
- Number of users of system
- Would an anonymous data set be usable?
- Contractual restrictions
- Security deployed in the infrastructure

Selection Rule Repository: DPSR
Selection Rules: DPSR1.r1, DPSR1.r2, DPSR1.r3
Name: Sticky policies **Classification:** Policy enforcement
Intent: to bind policies to the data it refers to
Motivation: A scenario where this would be useful is to ensure that policies relating to data are propagated and enforced along all chains through which the data is stored, processed and shared
Context: You want to make sure that multiple parties are aware of and act in accordance with your policies as personal and sensitive data is passed along the

chain of parties storing, using and sharing that data.
Problem: Data could be treated by receivers in ways that the data subject or initiator would not like. The policy could be ignored, or separated from the data it should refer to.
Solution: Enforceable sticky electronic privacy policies: personal information is associated with machine-readable policies, which are preferences or conditions about how that information should be treated (for example, that it is only to be used for particular purposes, by certain people or that the user must be contacted before it is used) in such a way that this cannot be compromised. When information is processed, this is done in such a way as to adhere to these constraints. These policies can be associated with data with various degrees of binding and enforcement. Trusted computing and cryptography can be used to stick policies to data and ensure that that receivers act according to associated policies and constraints, by interacting with trusted third parties or Trust Authorities.
Consequences:

Benefits - Policies can be propagated throughout the cloud, strong enforcement of these policies, strong binding of data to policies, traceability. Multiple copies of data are OK, as each has the policy attached.

Liabilities - Scalability and practicality: if data is bonded with the policy, this makes data heavier and potentially not compatible to current information systems. It may be difficult to update the policy once the data is sent to the cloud, as there can be multiple copies of data and it might not be known where these are. Once the data is decrypted and in clear, the enforcement mechanism becomes weak, i.e. it is hard to enforce that the data cannot be shared further in clear, but must instead be passed on in the sticky policy form; therefore, audit must be used to check that this does not happen.
Known uses: Policy specification, modelling and verification tools include EPAL, OASIS XACML , W3C P3P and Ponder. Notably, a technical solution for sticky policies and tracing services can leverage Identifier-Based Encryption (IBE) and trusted technologies; this solution requires enforcement for third party tracing and auditing parties. An alternative solution that relies on a Merkle hash tree has been proposed by Pöhls [**?**]. A Platform for Enterprise Privacy Practices (E-P3P) [**?**] separates the enterprise-specific deployment policy from the privacy policy and facilitates the privacy-enabled management and exchange of customer data.
Related patterns: obligations (obligations can be stuck to data), identity management (e.g. polices bound to data managed in identity management system), audit, Digital Rights Management (DRM).

4 Related Work

Privacy design techniques are not a new concept: various companies, notably Microsoft [**?**], have produced detailed privacy design guidelines. Cannon has described processes and methodologies about how to integrate privacy considerations and engineering into the development process [**?**]. Privacy design guide-

lines in specific areas are given in [?,?]. In November 2007 the UK Information Commissioners Office (ICO) [?] (an organisation responsible for regulating and enforcing access to and use of personal information), launched a Privacy Impact Assessment (PIA) [?] process (incorporating privacy by design) to help organisations assess the impact of their operations on personal privacy. This process assesses the privacy requirements of new and existing systems; it is primarily intended for use in public sector risk management, but is increasingly seen to be of value to private sector businesses that process personal data. Similar methodologies exist and can have legal status in Australia, Canada and the USA [?]. This methodology aims to combat the slow take-up to design in privacy protections from first principles at the enterprise level, see [?] for further discussion, [?] for further background, and [?] for a useful classification system for online privacy.

In addition to this body of privacy design guidelines, practical techniques can be specified using design patterns [?]. These can be defined in various different forms, ranging from fairly informal to formal, but all having substructure. Work is currently in place to define these: for example, use-cases that drive cloud computing are familiar ones and so design patterns to fit these have started to be produced [?]. Some previous work has been carried out in the privacy design pattern area, but not for cloud computing: [?] describes four design patterns that can aide the decision making process for the designers of privacy protecting systems. These design patterns are applicable to the design of anonymity systems for various types of online communication, online data sharing, location monitoring, voting and electronic cash management and do not address use within an enterprise. In our system, we extend the usage of design patterns to cover privacy architectural options and controls that can be deployed - particularly within an organisation, with the option of providing detail right down to example code level. Furthermore, we build upon this approach to allow automated determination of a set of recommendations for designers. With the existing guidelines, these are distributed and used in an off-line way, and it can be difficult for developers to find appropriate advice.

In expert systems, problem expertise is encoded in the data structures rather than the programs and the inference rules are authored by a domain expert. Techniques for building expert systems are well known [?]. A key advantage of this approach is that it is easier for the expert to understand or modify statements relating to their expertise. Our system can also be viewed as a decision support system. Again, there is a large body of preceding research [?]. Many different DSS generator products are available, including [?,?,?,?].

Halkidis et al. [?] perform risk analysis of software systems based on the security patterns that they contain. The first step is to determine to what extent specific security patterns shield from known attacks. This information is fed to a mathematical model based on the fuzzy-set theory and fuzzy fault trees in order to compute the risk for each category of attacks. However, this approach does not handle context information and there is no rule engine provided. There has been related work carried out in the Serenity project (see especially [?,?]): a

general framework was proposed to develop secure applications based on security patterns. They used an extension of TROPOS called SI* modelling framework for modelling and analysis of security requirements. The context of security patterns was discussed, and executable components can be selected upon client request by matching the context of pre-defined patterns. Delessy et al. [**?**] also discussed how to build upon two different approaches to secure SOA applications: model-driven development and the use of security patterns. Laboto et al. [**?**] proposed to use patterns to support the development of privacy policies. However, unlike our approach, a rule engine was not proposed to automatically select appropriate patterns at the design stage, and the focus was on security.

5 Conclusions

We have presented a novel approach for automatically selecting design patterns based on context. Our approach enables contextual and environmental factors to be taken account of in product and service design, by providing suitable options for the given context to designers. This procedure is independent of the chosen format of the design patterns. One or more design patterns will be suggested, that can be used in conjunction to satisfy contextual requirements. This is a broad solution that can be used for privacy, security and other types of requirement. We are currently extending this approach within the EnCoRe project [**?**] in order to generate privacy controls (with a focus on consent and revocation mechanisms) that are appropriate for different contexts, such as: to what level of granularity of data should the policy be attached? any economically feasible mechanism to enforce the policy? whether compatible to legacy systems? whether the obligations will be an extension of access control policies, or separate policies that are dealt with in a separate manner? etc.

References

1. Information Commissioners Office: The Privacy Dividend; the business case for investing in proactive privacy protection. (2010)
2. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., Angel, S.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press (1977)
3. Miller, G. A.: WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.
4. Pretschner, A., Schtz, F., Schaefer, C., and Walter, T.: Policy Evolution in Distributed Usage Control. Electron. Notes Theor. Comput. Sci. 244 (2009)
5. IBM: The Enterprise Privacy Authorization Language (EPAL), EPAL specification, v1.2, `http://www.zurich.ibm.com/security/enterprise-privacy/epal/` (2004)
6. Casassa Mont, M.: Dealing with Privacy Obligations, Important Aspects and Technical Approaches, TrustBus (2004)
7. Phls, H.G.: Verifiable and Revocable Expression of Consent to Processing of Aggregated Personal Data. ICICS, 279-293 (2008)

8. Ashley, P., Hada, S., Karjoth, G., Schunter, M.: E-P3P privacy policies and privacy authorization. WPES '02. 103-109 (2002)
9. Microsoft Corporation: Privacy Guidelines for Developing Software Products and Services, Version 2.1a, http://www.microsoft.com/Downloads/details.aspx?FamilyID=c48cf80f-6e87-48f5-83ec-a18d1ad2fc1f\&displaylang=en (2007)
10. Cannon, J.C.: Privacy: What Developers and IT Professionals Should Know. Addison Wesley (2004)
11. Patrick, A., Kenny, S.: From Privacy Legislation to Interface Design: Implementing Information Privacy in Human-Computer Interactions. R. Dingledine (ed.), PET 2003, LNCS 2760, pp. 107-124, Springer-Verlag Berlin Heidelberg (2003)
12. Belloti, V., Sellen, A.: Design for Privacy in Ubiquitous Computing Environments. Proc. 3rd conference on European Conference on Computer-Supported Cooperative Work, pp. 77-92 (1993)
13. Information Commissioners Office: PIA handbook. http://www.ico.gov.uk/ (2007)
14. Office of the Privacy Commissioner of Canada: Fact sheet: Privacy impact assessments. http://www.privcom.gc.ca/. (2007)
15. Information Commissioners Office: Privacy by Design. Report, www.ico.gov.uk (2008)
16. Jutla, D. N., Bodorik, P.: Sociotechnical architecture for online privacy. IEEE Security and Privacy, 3(2), pp. 29-39. IEEE (2005)
17. Spiekermann, S., Cranor, L. F.: Engineering privacy. IEEE Transactions on Software Engineering, pp. 142. IEEE (2008)
18. Arista: Cloud Networking: Design Patterns for Cloud Centric Application Environments. (2009) http://www.aristanetworks.com/en/CloudCentricDesignPatterns.pdf
19. Hafiz, M.: A collection of privacy design patterns. Proc. 2006 Conference on Pattern Languages of Programs, ACM, NY, pp. 1-13 (2006)
20. Russel, S., Norvig, P.: Artificial Intelligence A Modern Approach, 2nd edition, Prentice Hall, Englewood Cliffs (2003)
21. Wikipedia, http://en.wikipedia.org/wiki/Decision_support
22. Dicodess: Open Source Model-Driven DSS Generator, http://dicodess.sourceforge.net
23. XpertRule: Knowledge Builder, http://www.xpertrule.com/pages/info_kb.htm
24. Lumenaut: Decision Tree Package, http://www.lumenaut.com/decisiontree.htm
25. OC1 Oblique Classifier 1, http://www.cbcb.umd.edu/~salzberg/announce-oc1.html
26. Halkidis, S.T., Tsantalis, N., Chatzigeorgiou, A., and G. Stephanides: Architectural Risk Analysis of Software Systems Based on Security Patterns, IEEE TDSC, Vol. 5, No. 3, 2008.
27. Kokolakis, S., Rizomiliotis, P., Benameur, A., Kumar Sinha, S.,: Security and Dependability Solutions for Web Services and Workflows : A Patterns Approach, Security and dependability for Ambient Intelligence, Springer, May 2009
28. Benameur, A., Fenet, S., Saidane, A., Khumar Sinha, S.: A Pattern-Based General Security Framework: An eBusiness Case Study, HPCC, Seoul, Korea (2009)
29. Delessy, N. A., d Fernandez, E. B.: A Pattern-Driven Security Process for SOA Applications, ARES: 416-421 (2008)
30. Lobato, L.L., d Fernandez, E. B., Sergio Donizetti Zorzo: Patterns to Support the Development of Privacy Policies. ARES: 744-74 (2009)
31. EnCoRe - Ensuring Consent and Revocation. http://www.encore-project.info/