An Information Model and Architecture for Context-Aware Management Domains

Ricardo Neisse*, Patrícia Dockhorn Costa†, Maarten Wegdam *‡, and Marten van Sinderen*
*CTIT, University of Twente, The Netherlands
{r.neisse, m.wegdam, m.j.vansinderen}@utwente.nl
† Federal University of Espírito Santo, Vitória, Brazil
pdcosta@inf.ufes.br
‡ Telematica Instituut, P.O.Box 589, 7500 AN, Enschede, The Netherlands
maarten.wegdam@telin.nl

Abstract

Context-aware service platforms use context-aware policy management solutions to manage user's privacy preferences, to manage trust relationships, and to control access to the platform resources. However, existing context-aware policy management solutions focus on at most one policy management area (e.g. trust management, or privacy, or access control) and are difficult to integrate due to their unrelated policy/context information models and semantics. This leads to an integration problem, and to a policy management nightmare, because context-aware policies of different management areas have to be managed using different tools. In this paper, we address this problem using a new context-aware policy management abstraction called Context-Aware Management Domains (CAMDs). CAMDs allow the grouping of entities, for which a common set of policies apply, based on the entities' context. In comparison to existing solutions CAMDs provide a more generic context-aware policy management abstraction. CAMDs are suitable for any policy management area, and allow context-aware obligation policies, which are not supported by existing policy management solutions.

1 Introduction

Context-aware services adapt themselves to the current user's situation. An example of this is a context-aware health service, which chooses the most suitable caregivers to help a patient who is having an epileptic seizure based on the patient's location, and the caregivers' location and availability. In order to ease the deployment of context-aware services, service platforms have been designed to support context information acquisition, reasoning, and distribution [1].

Typical context-aware service platforms have thousands or even millions of entities (users, service providers, context providers, etc.) and different types of policies need to be managed. Policies are required, for instance, to enforce user's privacy, to manage trust relationships among the entities, and to control access to the platform (e.g. context information). Policies in a context-aware service platform are influenced by context changes, for example, a patient using a context-aware health service does not want to reveal his location and health information to caregivers when he is not having a seizure.

In order to specify policies for context-aware service platforms, policy management tools can be used. However, these policy tools provide either static policy management capabilities without considering context information (e.g. management domains defined by Ponder2 ([2] [3] [4]) , or, if there is some form of context-aware policy management, it is limited to one specific policy management area (e.g., X-RBAC [5], [6], and [7] for access control and COMITY [8] for trust management).

In this paper, we combine the concept of management domains from the Ponder2 policy language [3] with a context modeling approach that uses Event-Condition-Action (ECA) rules [9][10].

From this combination, we propose a new concept for context-aware policy management called Context-Aware Management Domains (CAMDs) [11]. CAMDs are management abstractions that provide dynamic grouping of entities, based on common context situations, for which a common set of policies apply. The dynamic grouping allows us to deal with a large quantity of entities in the dynamic manner that is required for context-aware services. We use ECA rules to detect changes in the context situations, which can trigger a change in the membership of a CAMD. As a result, CAMDs allows context-aware management of different types of policies. We have already introduced the concept of CAMDs in a short paper [11]. Besides detailing the CAMD information model, this paper extends the short paper by proposing a distributed architecture for CAMDs, describing a proof of concept prototype implementation, and presenting a case study that applies our prototype in a context-aware health scenario to demonstrate the feasibility.

This paper is organized as follows. Section 2 introduces our context-aware service platform and the policy management model on which our research is based. Section 3 presents our context model and context handling platform. Section 4 explains our new concept called Context-Aware Management Domains and presents our information model and architecture for CAMDs. Section 5 explains our case study for the context-aware health service scenario and shows our proof of concept prototype implementation. Section 6 compares our work with related work on context-aware management tools and Section 7 ends this paper with conclusions and future work.

2 Policy Management Model

Figure 1 presents our target context-aware service platform and illustrates the main roles we distinguish regarding the platform management and operation layers.

Within the operation layer, a user receives an identity token (2) after authenticating with an identity provider (1). The identity token is used to access a service provider (3), which verifies the user's identity (4) and retrieves context information to adapt the service (5 and 6). The context information can be, for instance, the current activity or location of the user; however, it can also include context about other entities (context owners) that are relevant for the service used (e.g. service provider's context). For details about the operation layer, see [1].

Within the management layer, an administrator accesses the policy provider in order to manage operation policies. Policies are rules that define a choice in the behavior of the system and can be of different types such as obligation and authorization [2]. Policies of different types can also focus on different management areas such as access control, privacy, and trust management.

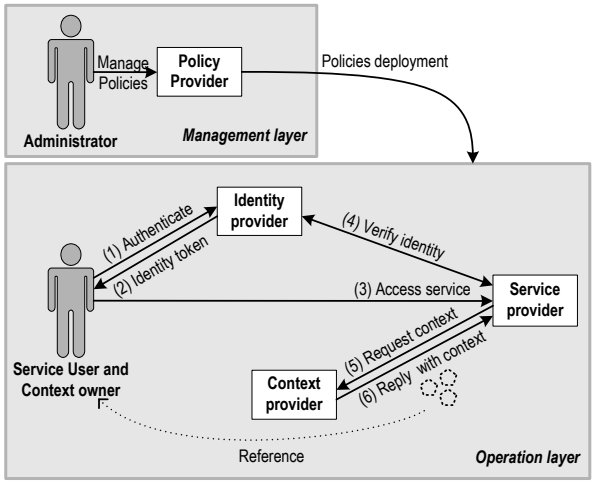

Figure 1. Policy Management in a Context-Aware Service Platform

In order to specify management policies there are many choices of policy management models available in the literature, for instance, Ponder2 [4], IETF Policy Core Information Model (PCIM) [12], and Rei [13]. In this paper, we choose the Ponder2 policy management model because it is extensible, it allows the grouping of managed objects and policies to simplify the management, and there is an implementation available. Additionally, Ponder2 can be used at different levels of entity granularity, from small embedded devices to complex services and Virtual Organizations, a desirable characteristic for our application scenario.

In the Ponder2 information model (Figure 2), a Policy is a managed object that can be triggered by events and related to other managed objects namely policy subjects and targets. Policy subjects and targets can be specified individually, as one specific managed object, or by means of management domains, which are static hierarchical sets of managed objects. The concept of management domains is similar to a directory structure, and helps in the management process because policies can be defined for a set of managed objects

instead of individual managed objects. Management domains reduce the management complexity in large systems because it is impracticable to specify and apply policies individually for each entity on a large scale. Management domains are static sets, and the inclusion and removal of entities from a management domain has to be done manually.

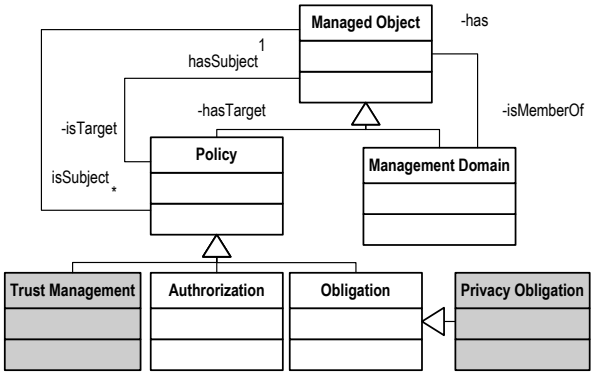

Figure 2. Extension of Ponder2 Policy Information Model

Regarding policy types, Ponder2 supports by default the definition of authorization and obligation policies. We have extended this to allow also the definition of privacy obligations and trust management policies.

Privacy obligation policies describe privacy actions that subjects must perform on targets under certain conditions. A privacy obligation policy could state, for instance, that a caregiver (subject) should delete (action) the patient location (target) 15 minutes after he gets it (condition), or that the ”identity provider” (subject) should not anonymize (action) the ”patient identity” (target) for nearby caregivers when the patient is in an imminent seizure situation (condition).

Trust policies are used to manage the bootstrapping and increase or decrease of trust values regarding different trust behaviors and aspects [14]. One trust management policy could state that every time a caregiver accepts to help a patient, the trust in the availability for the ”helping patients” aspect of this specific caregiver should be increased by one unit. For details about our trust model for context-aware service platforms please see [15].

3 Context Information Model

We adopt in this paper the context model developed in the AWARENESS research project [1], which also includes the work presented in this paper. In this context model, context is any information about an entity that is of interest to a context-aware service [10]. This context modeling approach provides us with proper conceptual foundations that can be extended and specialized with our own specific concerns. These foundations include the concepts of entity, context, and situation, as depicted in Figure 3.

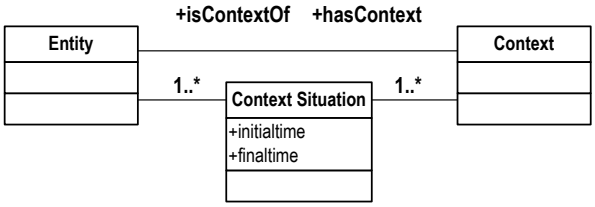

Figure 3. Context model

The concept of situation is particularly useful, since it allows us to model context graphically (using UML) at a higher level of abstraction. For example, it is possible to model a situation in which a patient stays close to a caregiver for a certain period, or a situation in which a patient has had a certain number of seizures in the past day. Situations in our context model are defined using UML class diagrams, which are enriched with OCL constraints.

Other context modeling approaches [16][17] offer limited support for situation modeling comparing to the approach we choose. Often these other approaches do not consider a suitable notion of time, and, therefore, temporal aspects, such as duration and precedence of situations, cannot be explicitly defined. In addition, there is no support for graphical representation of situations.

The context model presented in Figure 3 is only a summarized version of our context model that defines the concepts of context, entity, and context situation. A context situation is a composite class that captures situations of interest for the context-aware service. A situation has duration, which is defined by the moment the situation begins to hold (initial time), and the time the situation ceases to hold (final time). The situation realization mechanism presented in [10] implements a rule-based approach that allows the attentive detection of situations. Situation events are generated when the situation begins to hold (enter true situation event), and when the situation seizes to hold (enter false situation event).

Our context model uses Event-Condition-Action (ECA) rules defined in terms of occurrences of events,

and context and situation values. For example, with an ECA rule it is possible to specify that upon a patient's epileptic seizure (situation event), if the patient is performing a possibility hazardous activity and there is a caregiver nearby (context values), the system should send a warning message to both the patient, and the caregiver.

Figure 4 presents the architecture of our context handling platform [10], which instantiates our context information model. In this architecture, context is captured by sensors in the environment and is accessed through Context Source components. The Context Manager component uses the context information to detect Situations and to generate events when situations begin and end. The situation events are captured by a Controller component that implements Event-Condition-Action (ECA) rules and are responsible for triggering, for example, application adaptation actions. In the current implementation Context Sources, Context Managers, and Controller components are accessible through distributed objects in Java RMI and the Context Managers and Controllers are implemented as rules in a JESS rule engine [18].

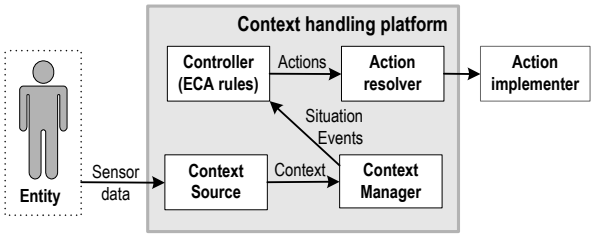

Figure 4. Context handling platform

4 Context-Aware Management Domains

The policy management model presented in Section 2 specifies policy subjects and targets individually or collectively, by means of management domains. However, even using management domains, the system administrators still have to define manually the set of entities that are part of the management domain. In this paper, we go one step further by defining management domains based on the context situations and situations events as presented in Section 3. We call this novel concept of management domains Context-Aware Management Domains (CAMDs) [11].

In order to illustrate the CAMD concept we present in Figure 5 an example where the domain "available caregivers" is mapped to every caregiver for which the current status is "available". In other words, a caregiver becomes part of this domain if its status changes to "available" and leaves the domain if its status returns to, for example, "unavailable". For this domain, the system administrator can associate policies of different types, for example, access control to patient health data, privacy obligations, or trust management policies.

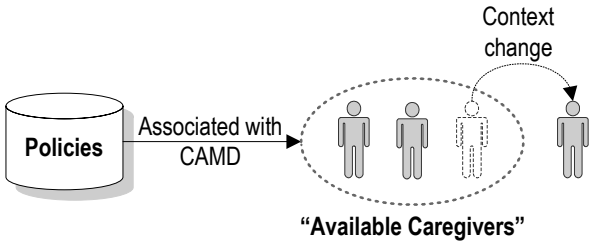

Figure 5. Context-Aware Domain Membership Changes

Figure 6 presents our information model for CAMDs which combines and extends the policy management model presented in Section 2 with the context information model presented in Section 3. In this combined information model, we define a CAMD as a sub class of the Management Domain class from the Ponder2 information model. CAMDs are associated with context situations and therefore indirect associated with the entities part of this situation. In order to allow the entities of our context model to be part of management domains entities are specified as sub-classes of the managed object class. The association of policies with CAMDs is done in the same way of the Ponder2 information model, by means of targets and subjects. Also, similarly to the Ponder2 information model, our policies can be triggered by events, which in our model can be the event of an entity joining or leaving a domain or a context situation event specific from our context handling platform.

Figure 7 presents our architecture for CAMDs. In our architecture, the system administrator manages the domains specification using ECA rules and associates policies with these domains using a graphical user interface provided by the domain manager component. The domain manager then deploys the domain specification in the context handling platform using ECA rules and the associated policies in the policy manager. The context handling platform monitors changes in the context situations and manages the

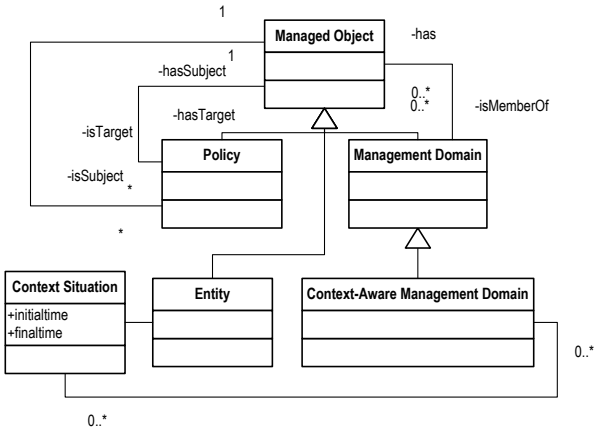

Figure 6. CAMDs Information Model

domain membership based on the events generate by these changes and the corresponding ECA rules. Simultaneously, when the policy manager evaluates a policy, he does not have to evaluate any context attributes, only the domain membership has to be checked in order to verify if a policy subject or target is part of a specific domain.

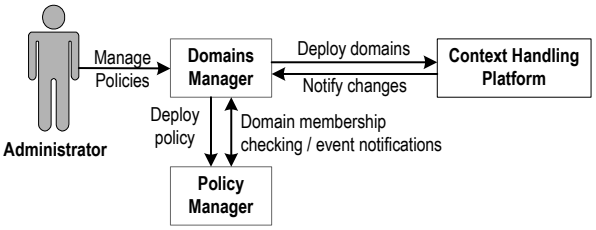

Figure 7. . Context-Aware Management Domains Architecture

In comparison to existing context-aware policy management solutions, CAMDs provide a simpler context-aware policy evaluation mechanism because policies refer only to CAMDs and do not contain anymore references to context attributes.

5 Case Study: Context-Aware Health Service

We apply the concept of Context-Aware Management Domains (CAMDs) in the management of authorization, privacy obligation, and trust management policies in the context-aware health service scenario. The context-aware health service monitors epileptic patients in order to detect or predict epileptic seizures. Upon an epileptic seizure alarm, a number of actions can be taken, such as warning the patient of an upcoming seizure, and sending notifications to caregivers that are currently nearby and available. The objective of the context-aware health service is to improve the life quality of such patients.

Figure 8 presents our CAMD graphical user interface (GUI). The GUI displays the available domains in a tree structure in the left side and the details of each domain in the right side using tabs. For each domain the tabs in the right side show the associated policies, the ECA rules defined to manage the domain membership, and a visualization of the entities that are currently member of that domain. In our current prototype the system administrator is responsible for the definition of the corresponding ECA rules to manage the domain membership, and also for defining the appropriate CAMD structure considering the service scenario and the context situations of interest. The definition of these ECA rules requires knowledge about the ECA rule language. We plan to develop as future work a graphical mechanism to assist the system administrator in this task. We expect that the context-aware application developers, which have more knowledge about the application domain and respective context model instantiated, will be responsible to provide templates of ECA rules to manage context situations.

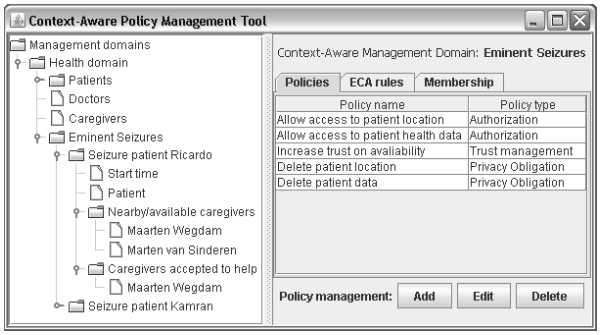

Figure 8. Prototype screenshot

Our prototype manages authorization, trust, and privacy policies for a seizure situation. First, when the seizure situation is detected for a patient, a CAMD is created and authorization policies are deployed to allow the nearby caregivers access to the patient's location. Authorization policies are also deployed to allow only the caregivers that accepted to help the patient access to the patient's health data, and trust management policies

to increase the patient's trust in these caregivers. When the context handling platform detects that the seizure situation has ended, privacy obligation policies are deployed to request the respective caregivers that have been authorized access before to delete the patient location and health data, in order to protect the patient's privacy. When the seizure situation ends the system also deletes the respective CAMD and includes this action and details about the seizure and policy deployment in a log file.

For illustration purposes, one of the ECA rules defined in our prototype to manage the membership of the CAMD "Eminent Seizure" from Figure 8 is presented bellow (Listing 1). This ECA rule creates CAMD sub-domains under the "Eminent Seizures" domain, upon the occurrence of an epileptic seizure alarm, following a pre-defined structure. The system administrator is free to define the most suitable CAMD structure to match the application scenario, which in our case study is composed of the patient having the seizure, the nearby caregivers, and the caregivers who accepted to help the patient. We also define in our prototype other rules to manage the CAMD membership, for example, to trigger the exclusion of a CAMD when the seizure situation ends, which are not present here for the sake of simplicity.

```
Scope (EpilepticPatient.*; patient) {
  Upon EpilepticAlarm (patient)
  Do CreateEminentSeizureDomain(
    patient,
    Select (
      CareGiver.*; caregivers; (
        isCareGiverOf (caregivers, patient)
        and
        SituationWithinRange (patient, caregivers)
        and
        SituationCareGiverAvailable(caregivers)
      )
    )
  )
}
```

Listing 1. ECA rule to manage Eminent Seizure CAMD

The algorithms used in our prototype to predict seizures and detect changes on the context of the entities are provided by the context handling platform and are out of the scope of this paper. These algorithms are part of output of the AWARENESS research project [1].

6 Related Work

We can separate the related work into work on policy management (including management domains) on one hand, and point solutions that use context in specific policy management areas on the other. We start by comparing our work to the first, and then discuss the most prominent representatives of the second category of related work.

The work done in the Ponder/Ponder2 toolkits [2][3][4] provides a model for policies and management domains. In this model, entities are statically associated with domains which are then associated with different types of policies. Our realization of CAMDs builds upon their model, but makes the association between entities and domain dynamic based on context. In addition, we apply their generic policy management toolkit in the area of context-aware services.

Joshi et al. [5] have extended the Role Based Access Control (RBAC) standard to support the definition of parameterized access control roles. Their proposal is called X-RBAC and provides dynamic management of access control roles based on time and location constraints. Their focus is specific in access control policies for XML document sources at different levels (conceptual, schema, XML instance, and element). Compared to our work their work is more restricted to the type of policies (only access control) and type of context (only location and time). For example, they do not support obligation policies for privacy nor context related events.

Corradi et al. [6] [8] use context information to adapt trust relationships for pervasive environments in the so called COMITY security model. In their work they associate trust degrees with context conditions which are further associated with authorization and refrain policies allowing dynamic management. Our concept of a context-aware domain can be seen as a generalization of this work because we allow context-aware management of different types of policies and do not limit ourselves to trust management.

7 Conclusions and Future Work

This paper presents a new concept called Context-Aware Management Domain (CAMD) that allows flexible and dynamic policy management for context aware service platforms. In comparison to existing context-aware policy management solutions CAMDs are more generic because they are not limited to a specific policy management area such as access control, trust management, and privacy enforcement. Furthermore, CAMDs provide an abstraction for the system administrator that supports the definition of obligation policies based on context situation events,

which is not currently supported by any context-aware policy management solution we have found in our related work studies. We have also verified the feasibility of our work through our prototype, where the different types of policies are managed in a context-aware health service scenario.

As future work, we plan to analyze policy conflicts for CAMDs and study the deploying of CAMDs in multi-administrative domains environment considering the trustworthiness and quality aspects of the context information used for context situation detection. We also want to research mechanisms to increase the reliability in case context situation events are lost or are not detected due to failures in the context handling platform, for example, due to problems in the sensors.

A preliminary performance evaluation of our ECA rule engine indicates that the response time scales for a large number of users. We want to carry out a specific performance evaluations of our CAMD mechanism considering also network bandwidth and memory consumption when a large number of domains and policies have to be evaluated.

Acknowledgment

This work is part of the Freeband AWARENESS project. Freeband is sponsored by the Dutch government under contract BSIK 03025. Ricardo Neisse is supported by a CNPq Scholarship, Brazil.

References

[1] M. J. van Sinderen, A. T. van Halteren, M. W. H. B. Meeuwissen, and E. H. Eertink, "Supporting context-aware mobile applications: an infrastructure approach," *Communications Magazine, IEEE*, vol. 44, no. 9, pp. 96–104, Sept. 2006.

[2] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The ponder policy specification language," in *POLICY '01: Proceedings of the International Workshop on Policies for Distributed Systems and Networks*. London, UK: Springer-Verlag, 2001, pp. 18–38.

[3] N. Damianou, N. Dulay, E. Lupu, M. Sloman, and T. Tonouchi, "Tools for domain-based policy management of distributed systems," *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pp. 203–217, 2002.

[4] G. Russello, C. Dong, and N. Dulay, "Authorisation and conflict resolution for hierarchical domains," in *POLICY '07: Proceedings of the Eighth IEEE International Workshop on Policies for Distributed Systems and Networks*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 201–210.

[5] J. B. D. Joshi, R. Bhatti, E. Bertino, and A. Ghafoor, "Access-control language for multidomain environments," *IEEE Internet Computing*, vol. 8, no. 6, pp. 40–50, 2004.

[6] A. Corradi, R. Montanari, and D. Tibaldi, "Context-based access control for ubiquitous service provisioning," in *COMPSAC '04: Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 444–451.

[7] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahamad, and G. D. Abowd, "Securing context-aware applications using environment roles," in *SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies*. New York, NY, USA: ACM, 2001, pp. 10–20.

[8] A. Corradi, R. Montanari, and D. Tibaldi, "Context-driven adaptation of trust relationships in pervasive collaborative environments," in *SAINT-W '05: Proceedings of the 2005 Symposium on Applications and the Internet Workshops*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 178–181.

[9] P. D. Costa, G. Guizzardi, J. P. A. Almeida, L. F. Pires, and M. van Sinderen, "Situations in conceptual modeling of context," in *EDOCW '06: Proceedings of the 10th IEEE on International Enterprise Distributed Object Computing Conference Workshops (VORTE 2006)*. Washington, DC, USA: IEEE Computer Society, 2006, p. 6.

[10] P. D. Costa, J. P. A. Almeida, L. F. Pires, and M. J. van Sinderen, "Situation specification and realization in rule-based context-aware applications," in *Seventh IFIP WG6.1 International Conference on Distributed Applications and Interoperable Systems, DAIS 2007, Paphos, Cyprus*, ser. Lecture Notes in

Computer Science, vol. 4531. Springer, 2007, pp. 32–47.

[11] R. Neisse, P. D. Costa, M. Wegdam, and M. van Sinderen, "Context-aware management domains," *First International Workshop on Combining Context with Trust, Security and Privacy*, July 2006.

[12] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "Policy Core Information Model – Version 1 Specification," RFC 3060 (Proposed Standard), Feb. 2001, updated by RFC 3460. [Online]. Available: http://www.ietf.org/rfc/rfc3060.txt

[13] L. Kagal, T. Finin, and A. Joshi, "A Policy Language for A Pervasive Computing Environment," in *IEEE 4th International Workshop on Policies for Distributed Systems and Networks*, June 2003.

[14] A. Abdul-Rahman and S. Hailes, "Supporting trust in virtual communities," in *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6*. Washington, DC, USA: IEEE Computer Society, 2000, p. 6007.

[15] R. Neisse, M. Wegdam, M. van Sinderen, and G. Lenzini, "Trust management model and architecture for context-aware service platforms," in *In: Proceedings of the 2nd International Symposium on Information Security (IS07), November 26-27, Vilamoura, Portugal*, 2007, pp. 1803–1820.

[16] K. Henricksen and J. Indulska, "A software engineering framework for context-aware pervasive computing," *percom*, vol. 00, p. 77, 2004.

[17] H. Chen, "An Intelligent Broker Architecture for Pervasive Context-Aware Systems," Ph.D. dissertation, University of Maryland, Baltimore County, December 2004.

[18] F. Cabitza, M. Sarini, and B. D. Seno, "Jess rule engine," 2008. [Online]. Available: http://herzberg.ca.sandia.gov/jess/