Context inference of users' social relationships and distributed policy management

Alisa Devlic[1,2], Roland Reichle[3], Michal Wagner[3], Manuele Kirsch Pinheiro[4,5], Yves Vanrompay[4], Yolande Berbers[4], Massimo Valla[6]

[1]Appear Networks, Kista, Sweden
[2]Royal Institute of Technology (KTH), Department of Communication Systems, Kista, Sweden
[3]University of Kassel, Kassel, Germany
[4]Katholieke Universiteit Leuven, Department of Computer Science, Leuven, Belgium
[5]Centre de Recherche en Informatique, Université Paris 1–Panthéon Sorbonne, Paris, France
[6]Telecom Italia Lab, Torino, Italy

alisa.devlic@appearnetworks.com, {reichle, wagner}@vs.uni-kassel.de, {yves.vanrompay, yolande.berbers}@cs.kuleuven.be, manuele.kirsch-pinheiro@univ-paris1.fr, massimo.valla@telecomitalia.it

***Abstract*— Inference of high-level context is becoming crucial in development of context-aware applications. An example is social context inference – i.e., deriving social relations based upon the user's daily communication with other people. The efficiency of this mechanism mainly depends on the method(s) used to draw inferences based on existing evidence and sample information, such as a training data. Our approach uses rule-based data mining, Bayesian network inference, and user feedback to compute the probabilities of another user being in the specific social relationship with a user whose daily communication is logged by a mobile phone. In addition, a privacy mechanism is required to ensure the user's personal integrity and privacy when sharing this user's sensitive context data. Therefore, the derived social relations are used to define a user's policies for context access control, which grant the restricted context information scope depending on the user's current context. Finally, we propose a distributed architecture capable of managing this context information based upon these context access policies.**

Keywords- Context inference of user social relations, context access control policies, context scope, user privacy.

I. INTRODUCTION

Deriving context information without explicit user input is a key requirement for context-aware applications development. Additionally, some information can only be inferred by analyzing the user's activities over time. An example is social context inference - i.e., deriving social relations from a user's daily communication with other people. The user's communication with others is captured on a mobile device by logging the data about sent and received SMS & MMS messages, call logs, and e-mails to a file (see Figure 1). This file is uploaded once a day to a computer for analysis of the user's communication patterns in context – in order to infer the user's social relations with the people he/she interacts with. This inferencing is based on rule-based data mining, Bayesian network inference, and user feedback to compute the probabilities of another user being in a specific social relationship with a given user. The inferred social relationships are stored in the form of a Friend-Of-A-Friend (FOAF) ontology extended with social relationship terms (i.e., family, friend, colleague, or unknown).

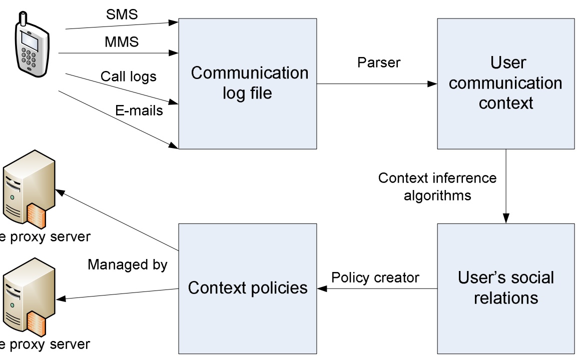

Figure 1. User social relations inference and policy management approach

Because users are sensitive about sharing their context with other users without explicitly giving their permission, a non-intrusive means of indirectly gaining the user's permission to share some context information is needed. To do this, we propose to use the user's social relations as a means to create user specific policies for granting access to their context information. This enables a user to specify different levels of access to his context information based on the relation he/she has with the other user that requests it (e.g. whether this other user is a family member, a friend, a colleague, or unknown). The decision about whether to grant access to context to the requesting entity, and at what granularity, is made based on the social relationship that this entity has with the context owner **and** the owner's current situation. As a result the user only has to explicitly determine the access to be given to a class of users, rather than to each specific user. However, these policy rules might depend upon the situation the user might be in. Therefore, in our policy design we also introduce context conditions to allow a user to define different rules (i.e., allowing distribution of a particular context to a specific scope or to deny distribution) based on his/her current situation. These context policies are utilized by multiple proxy servers which process queries for context information, thus improving scalability and avoiding single point of failure.

II. Motivating Scenario

The research presented in this paper belongs to a larger initiative, the MUSIC Project [1], aiming to develop a platform and tools for rapid development of context-aware self-adapting applications. Among the scenarios considered, the Instant Social (IS) scenario [2] proposes a mobile content sharing platform that enables users to browse, search for, and share multimedia content scattered over devices in different contexts.

In IS scenario, Paul is visiting a rock festival and wants to share photos with other users at the festival. The IS application running on Paul's mobile device communicates with other instances of itself running on nearby devices, sharing not only media content[1], but also context information about these users (their preferences, location, etc.) and their mobile devices (available memory, CPU utilization, and network bandwidth). Context information is used by the IS platform for adaptation purposes [2]. For example, these instances may utilize local services or services running on other instances in order to save memory or battery power. The proposed implementation is based on the MUSIC adaptation middleware which bases this choice on the result of a utility function that ensures sufficient replication of data for stable operation when members join and leave dynamically, and balances the load over the member devices according to their resource availability [2].

Similar to content sharing, context distribution also raises privacy issues. For most of people, social interactions with different communities during a normal day typically exhibit a pattern. These communities can be broadly categorized as family, friends, and colleagues. We have assumed that all individuals in a given class will be treated in the same manner (with regard to context access). Context information is often considered sensitive information whose distribution should be controlled and limited. Therefore, during the rock festival, Paul might share his location only with his friends & family. While to his colleagues he might simply indicate that he is "Out of the Office". In contrast, during working hours, a person's family & friends do not need to know that he is currently in a meeting; they might only be able to learn that Paul is "At work".

This example clearly shows the need for setting and following privacy constraints for both content sharing and context distribution. It also demonstrates the benefits of using the user's social relationships to reduce the burden upon the user – while preserving the desired user control. Moreover, it indicates the need for context dependent privacy restrictions, because user preferences may differ for different situations. In this paper, we focus on how to allow users to control the distribution of their context information based on their current context and their social relationships to the requesting user. These social relations are inferred based on the user's daily communication patterns.

[1] Note that this is the user's own media content and not that of a performance taking place at the festival.

III. Architectural View

The MUSIC context middleware is responsible for collecting, organizing, managing, and sharing context information as acquired by sensors with the proper context clients. Once collected, context information is cached in context storage for the period determined by the nature of information and application needs. In MUSIC, reasoners are specialized context sensors which process existing context data in order to compute higher-level context data. Sensors have plug-in components with a mechanism to activate or deactivate the sensing mechanism (previously described in [3]).

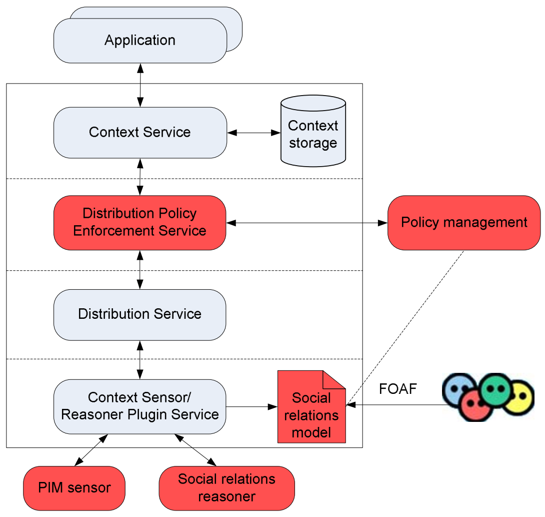

Figure 2. MUSIC Context middleware architecture

To implement our approach, we extend the existing MUSIC context management system (depicted in Figure 2) with a *Personal Information Management (PIM) sensor* which captures features from the user's daily communication patterns; a *Social relations reasoner* (applies a probabilistic rule-based reasoning to the user communication context (provided by the PIM sensor) in order to infer the user's social relationships with other users he/she was in contact with) which are stored in the user's *Social relations model*; the *Distribution Policy Enforcement Service* creates distribution policy requests and sends them to the *Policy management* module for evaluation. This *Policy management* module scopes the dissemination of context information based on the user's social relationship with the query initiator and the user's current situation. The *Policy management* module accesses the user's *Social relations model*. This model contains the user's contacts divided in social groups in the FOAF format. The different parts of this model can be distributed over multiple proxy servers to facilitate processing requests for context information. After obtaining a decision from the *Policy Management* module, the *Distribution Policy Enforcement Service* enforces this decision by forwarding the context information in the allowed scope to the *Distribution Service* or canceling the context distribution and notifying the *Context Service* about the rejection of the request. The *Policy management* module is handled by distributed proxy servers as it is computationally demanding. Sensors and reasoners do not need to run locally on the device where the application is launched, but can be accessed remotely via the *Distribution Service* (which is not described in this paper).

IV. Inference Mechanism of User's Social Relations

In order to infer the users' social relationships from logged communication activities via their mobile phone and email we follow an approach similar to that proposed in [4]. However, we modified this approach, as several prerequisites differ:

- Unlike [4] and [5], only a very limited amount of log-data is utilized (with regards to both the number of involved users and the recording time). However, we consider this to be more realistic for our application domain and as the data is collected by each participant in the communication – there is less concern about privacy. This leads to better scaling with the number of users, since each user builds their own model.
- Only log-data about communication activities via mobile phone and email are available, no information about proximity was incorporated.
- Our goal is to utilize only simple inference rules. These rules are intended to be easily understandable by human developers & users and to be updated by a learning mechanism incorporating user feedback.

Of these differences, the limited amount of log-data raises some new challenges, as special care has to be taken to avoid over constraining of the inference rules due to the limited training data. Therefore, a primary goal was to develop a robust approach that is applicable for a limited amount of log-data and does not rely on extensive (historical) logging of communication activities. The rules, once derived by a developer are further adjusted or modified by incorporating appropriate user feedback.

A. *Collection of log data*

The PIM sensor which acquires data about the users' communication activities (i.e., incoming/outgoing phone calls and SMS/MMS messages) runs as a background C# application on Windows Mobile phones. Each time the phone is synchronized using ActiveSync, incoming e-mails are also recorded. For each communication activity, a log entry with a timestamp, contact information, and duration is made. For email, the sender & receiver email addresses, and the subject are logged. If the user has specified a category for a contact in his address book, then the category of any contact involved in the conversation is also recorded in the log. Logs are stored in the phone's memory and later uploaded to a server for post processing and relationship inference.

B. *Inference approach*

The proposed inference approach for deriving user social relations from mobile phone and email log-data consists of five steps (see Figure 3):

1. The data collected by the PIM sensor are parsed and converted into standard comma separated value (CSV) format, in order to make it available to other tools.
2. A set of features is derived that might be used to distinguish between the different classes or categories of the corresponding communication partners. Examples of potentially useful features are 'contact data available' and 'number of activities' (this might lead to inference rule such as 'if there is no contact data available and there is only limited communication, then the corresponding user is regarded as a stranger).
3. The usefulness of the selected features is evaluated on the log-data and the features that best contribute to discrimination into the categories of communication partners are retained.
4. These features are utilized for rule-based data-mining approach, e.g. PART [6], in order to derive inference rules based on the log-data. The derived rules are manually inspected by the developer. Rules that hint of overfitting of data are discarded or modified.
5. These rules are used to create a classifier and a simple Bayesian network in order to estimate the confidence of the classifier in its decision. If the classifier is not able to satisfactorily classify the log-data, then go back to step 2.

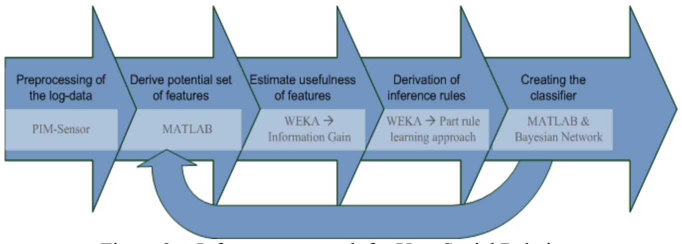
Preprocessing of the log-data
PIM-Sensor
Derive potential set of features
MATLAB
Estimate usefulness of features
WEKA → Information Gain
Derivation of inference rules
WEKA → Part rule learning approach
Creating the classifier
MATLAB & Bayesian Network

Figure 3. Inference approach for User Social Relations

In the following subsections each of the steps following the preprocessing step are described in more detail.

1) *Derive potential set of features (Step 2)*

The general objective in our proof-of-concept application was to distinguish between four different categories of communication partners: (1) stranger, (2) colleague, (3) friend, and (4) family member. For this purpose, we have derived 16 candidate features, shown in first column of Table I (each with a value stratified as being low/medium/high, except for Contact data available which has only a yes/no value).

TABLE I. Derived and Selected Useful Set of Features

Derived potential set of features	**Selected useful features**	**Abbreviations for selected features**
(1) Contact data available	Yes	[ContactData]
(2) Number of communication activities	Yes	[nInteraction]
(3) Number of phone calls	Yes	[nPhoneCalls]
(4) Number of phone calls at work time	No	
(5) Number of phone calls in free time	Yes	[nPhoneCallsFreetime]
(6) Number of phone calls at week end	Yes	[nPhoneCallsWeekend]
(7) Average duration of phone calls	No	
(8) Number of SMS/MMS	Yes	[nSMS]
(9) Number of SMS/MMS at work time	No	
(10) Number of SMS/MMS in free time	No	
(11) Number of SMS/MMS at week end	No	
(12) Number of Emails	No	
(13) Number of Emails at work time	No	
(14) Number of Emails in free time	No	
(15) Number of Emails at week end	No	
(16) Number of Business Emails	Yes	[nBusinessMails]
(subject containing e.g. 'meeting', 'deadline', 'review')	No	
Additional features		
(17) Number of SMS/MMS on Friday or Saturday night	Yes	[nSMSFrSaNight]
(18) Number of phone calls on Friday or Saturday night	Yes	[nPhoneCallsFrSaNight]

For extracting the features described above, we imported the CSV-files into MATLAB [7] and applied appropriate scripts. The statistical features are normalized with regard to the average value of all interactions of a user and discretized to the values 'low' (< 0.75*average), 'medium' (>= 0.75*average, <= 1.5*average), and 'high' (> 1.5*average).

As this set of features proved to be insufficient to discriminate between the four categories of communication partners (in particular between 'family member' and 'friend') it was enhanced by including two additional features:

(17) Number of SMS/MMS on Friday or Saturday night
(18) Number of phone calls on Friday or Saturday night

Without these two rules the classifier is not able to perform significantly better in distinguishing between 'family member' and 'friend' than a random classifier.

2) Estimate usefulness of features (Step 3)

In order to estimate the ability of the features to contribute to the discrimination of the four different categories of communication partners, we exported the log-data along with the two added features as a standard CSV file. This file was imported into WEKA (Waikato Environment for Knowledge Analysis) [8]. WEKA is a data-mining tool that allows applying a number of different feature selection algorithms on the log-data enriched with the new features. We have decided to calculate the Information Gain [9] achieved by a feature in order to estimate its usefulness. Information Gain is a well known measure to select appropriate features and is commonly used in Decision Tree Learning [9], which is quite similar to our rule-based approach. As a result, the set of features shown in the second column of Table I was selected. According to the Information Gain measure, all of the other features did not really contribute to the discrimination of the four categories of communication partners and therefore were discarded.

3) Derivation of inference rules (Step 4)

While the previous steps are quite similar to existing approaches, such as presented in [4], in the next step we go in a different direction than others, in order to cope with the additional challenges and objectives as mentioned in Section IV. To derive the inference rules we again utilized the WEKA tool and its facilities to create classifiers from training data. As our objective is to provide easy to understand inference rules, we applied the PART rule learning approach [6] to create a rule based classifier. However, the derived rules are **not** directly used. First they are manually inspected by the developer, to see if they hint at overfitting or if they correspond to the general understanding that a developer has about the characteristics of communication with a partner of a certain category.

For example, rules which incorporate a large number of different features and allows only a single value for most of the features, hints at overfitting. The same applies for rules that do not really reflect the common understanding of communication characteristics of a category. In this respect, we have to answer the question why we have chosen this approach, instead of utilizing other classification approaches and automatically tuning parameters or rules to avoid overfitting. Usually, in order to detect overfitting the training data are split up in a training set and a test set. The training set is used to create the classifier, whereas the test set is used to check if the classifier adequately generalizes the remaining data. This approach can be used to automatically tune parameters and rules. We have done the same, experimenting with different approaches. However, our experiences have shown that it is quite difficult to find an appropriate setup with a very limited amount of log-data and that the automatically tuned classifiers had a poor performance. We have also split the training data into one training set and two test sets, where one test set was used to detect overfitting and the other to check if the tuned classifier adequately generalizes the second test set as well. However, this was not always the case. Deriving simple and easily understandable rules and asking the user who is able to incorporate general knowledge about communication characteristics has revealed to be more reasonable.

For our proof-of-concept application, we derived the following rules:

(1) if (nInteraction == low and ContactData == no) or (ContactData == no and nPhoneCalls == 0 and nSMS == 0) then category = 'stranger'
(2) if not('stranger') and (nBusinessMails > low or nPhoneCallsFreetime <= medium and nPhoneCallsWeekend <= medium) then category = 'colleague'
(3) if not('stranger') and not('colleague') and nPhoneCallsFrSaNight >= medium and nSMSFrSaNight >= medium then category = 'friend'
(4) if not('stranger') and not('colleague') and not('friend') then category = 'family'

Note that the rule (4) is a result of the PART rule learning approach and the order of filters: the first filter identifies strangers and then out of these colleagues can be extracted. Assuming that only friends and family members are left, the most useful rule filters friends out of the remaining group. Finally, the group that remains are family members.

4) Creating the classifier (Step 5)

The corresponding classifier is directly created from the inference rules shown above. However, as these rules may not directly reflect the communication characteristics in every case, it is crucial to be able to estimate the confidence of the classifier in its classification result. For this purpose, we derive a simple Bayesian network from the set of rules. The conditional probability tables (CPTs) of the nodes are estimated by simply counting the occurrences in the training data that matched the user's self reported social relations whose context data was logged. From this simple Bayesian network the probabilities for a certain category can be calculated. The entropy derived from these probabilities gives a hint on the confidence the classifier has in its classification result.

C. Incorporating user feedback

Incorporating user feedback is still work in progress and also a key part of future work. Therefore, in this subsection only general ideas are presented.

If the rules reflect the training data, then usually the classification result corresponds to the category with the highest probability. However, if the classification is characterized as high entropy, then the classification can be regarded as unsure and the user has to be asked for feedback. This user feedback can be used to create new instances of labelled training data and to adjust the Bayesian Network. For this purpose, the user chooses a weight[2] to specify the influence of the new training data on the current CPTs. Obviously the updated CPTs affect the entropy values of the classifications. If the update of the CPTs results in greater uncertainty in the classification results, then the classification rules have to be adjusted. To determine the updated rules, PART rule-learning is applied again. If the resulting rules are similar to those already used in the classifier, but with changed comparison values, then the rules can be automatically adjusted. However, if the CPTs are adjusted to a large extent, then the rules may become obsolete and the structure of the rules (leaving out features, incorporate new features) will change. In this case, the rule is presented to the user in an easily understandable expression, e.g. in English language, and he/she is asked if this rule is characteristic for their communication activities with the corresponding category of communication partner.

We are aware that estimating the usefulness of rules is not always a trivial task for the user and that asking the user for such feedback is not desirable. However, learning the rules completely autonomously we regard as not feasible at the present, because of the very limited amount of log-data to be expected. Therefore, currently we consider it unavoidable to ask the user for feedback as mentioned above. However, this has to be investigated more thoroughly in the future.

D. *Evaluation results*

For our proof-of-concept application we ran the PIM sensor for three different users with a recording time of one week. This resulted in 331 communication activities with 41 different communication partners in total: 122 interactions with 15 partners for $user_1$, 121 interactions with 13 partners for $user_2$, and 88 interactions with 13 partners for $user_3$. These numbers highlight again the very limited amount of available log-data and therefore the difficulty of the problem.

Despite this limited data, using the simple inference rules presented above, 33 out of the 41 different communication partners were classified correctly with regard to the users' self reported social relations: 13 out of 15 for $user_1$, 11 out of 13 for $user_2$, and 9 out of 13 for $user_3$. This corresponds to classification successes of about 87%, 85%, and 69%, respectively. Of these, 4 out of the 8 incorrectly classified communication partners are friends that are classified as colleagues. The remaining 4 incorrectly classified partners were more or less spread equally among colleagues, friends, and family members. Unfortunately, a classification success of about 69% for $user_3$ is not very satisfactory. The reason for these incorrect classifications is that the $user_3$ had the smallest number of total interactions. Because of the lack of data about these contacts, there is little evidence to base a decision on. Hence it would be better to add a fifth category – not yet classified. However, more important than the pure classification success is the fact that all the incorrect classifications were judged to be unsure, therefore, the user would have been asked for feedback.

In conclusion, although the classification success of roughly 80% for all three users, the result is still promising, as it was achieved by applying very simple rules that reflect the common understanding of the communication characteristics of these categories of communication partners. As the user's device performs the logging, the classification accuracy for those people with whom the user regularly has contact should quickly be correct, while only new contacts will be inaccurate.

We are aware that our assumption of mutual exclusive categories of social relationships is not valid in general (e.g., a colleague can also be a friend). In fact, such relationships are seen in our classification results. We plan to extend our approach to detect characteristics of the different categories without having a discriminating classification problem in mind. However, first it has to be investigated if such a limited amount of log-data is sufficient to derive such characteristics.

V. USER SOCIAL RELATIONS MODEL

The user's social relations model stores relations between users inferred by the context inference mechanism. Several formats are emerging to store relations in social networks [24][25]. We chose to store inferred relations using the FOAF format, an RDF-based language already used by several social web services, but extended to categorize the *type of relationship*. A previous proposal to extend FOAF with relationship types is described in [26], where the property foaf:knows, the only option offered by the FOAF ontology, is extended by sub-properties such as: colleagueOf, friendOf, etc. that can be used in our case to distinguish which type of relationship is inferred between two users.

We have added relationship tags to the FOAF Person element, as showed in Figure 4, where the Person "cristina" is defined to be a colleague of Person "massimo", while Person "francesco" is defined to be a friend.

```
<foaf:Person rdf:nodeID="massimo">
    <foaf:name>Massimo Valla</foaf:name>
    <foaf:firstName>Massimo</foaf:firstName>
    <foaf:surname>Valla</foaf:surname>

    <rel:colleagueOf>
      <foaf:Person rdf:nodeID="cristina">
        <foaf:name>Cristina Fra</foaf:name>
      </foaf:Person>
    </rel:colleagueOf >

    <rel:friendOf rdf:nodeID="francesco"/>
</foaf:Person>
```

Figure 4. User social relations model using FOAF extended to support relationship types

The use of FOAF format to represent inferred relationships is also useful if external applications or web services need to access this information in a standard way. Several proposals are emerging to offer social network and relationship information to external services, the most promising one being OpenSocial

[2] The value is in the interval [0.0, 1.0]. A value of 1.0 indicates that the communication partner with unsure classification is very representative of the communication behaviour of the user. A value of 0.1 indicates that the communication partner is representative only to a limited extend, and 0.0 means not representative at all.

API [27]. Our *Social relations reasoner* implements the OpenSocial API in order to be able to export this information externally. Finally, the inferred relations could be integrated with information obtained from external social networks, for example by periodically merging FOAF data exported by such networks with the data inferred by our inference mechanisms.

VI. DISTRIBUTED CONTEXT POLICY MANAGEMENT

Based on the social relations inferred in Section IV, a user can define policies controlling the sharing of his sensitive context information with other users, based on their relations. The goal of these context policies is to grant different degrees of access (i.e., a particular context scope) for the user's different social relation groups. This context scope can be conditioned upon the user's context (i.e. time, place, or activity). The social relations are identified by the FOAF ontology mentioned in Section V. The membership in these groups is automatically inferred, as explained in Section IV.

Each context distribution policy contains a target and one or more policy rules (such as shown in Figure 5). The policy rule can be seen as a set of quadruplets "*relationship, resource, action, context condition*" in which *relationship* is the requesting entity's attribute corresponding to his/her social relationship with the user to which the policy refers to, *resource* corresponds to the context scope associated with this relationship, *action* is to grant or deny read or write access to the actor for the corresponding resource, and *context condition* specifies the user's context in which this action is taken.

```
Target:
        Resource: context information=location
Policy rule:
        Relationship: FOAF relation = friendOf
        Resource: context scope = street address
        Action: read allowed
        Context condition: activity = clubbing
```

Figure 5. Context policy example

Figure 5 depicts a policy for accessing location information of a given user in a certain context. Using this policy, a user indicates that his friends may access his street address when querying about his location if his/her activity is set to clubbing.

A. *Design of context policies based on user's social relations*

Figure 6 illustrates the same policy presented in Figure 5, but expressed using the XACML Policy Language. In this standard language, policies are defined for a given target, which can be a resource, an actor (called a "subject"), an action, or an environment involving these elements, and they are composed by a set of rules. For each rule, the policy writer defines an effect if the rule is evaluated as "true" (permit or deny), an optional target, and an optional condition. When evaluating a policy, each rule is evaluated using a rule-combining algorithm indicated in the policy definition.

In our example, the policy target is the resource (this policy limits access to location information); however this is not shown in Figure 6 due to limited space. This policy contains only one rule, whose action is to read a particular resource value and which defines, as a condition that the subject has to be in FOAF relation "friendOf" with the context owner.

```
<Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:SimpleRule1"
Effect="Permit">
  <Target>
     ...
         <ResourceMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
             <ResourceAttributeDesignator
AttributeId="urn:ist-music:names:context:model:concept:value"
DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">urn:ist-
music:names:context:model:concept:value:address</AttributeValue>
         </ResourceMatch>
      ...
 </Target>
 <Condition>
   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
         <SubjectAttributeDesignator AttributeId="relation"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
         <AttributeValue
DataType=http://www.w3.org/2001/XMLSchema#string>friendOf</Attribute
Value>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
         <ResourceAttributeDesignator AttributeId="activity"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">clubbing</Attribu
teValue>
      </Apply>
   </Apply>
 </Condition>
```

Figure 6. Policy rule in XACML

It is worth noting in Figure 6 that the granted resource value when requesting a resource is described through a string ("*urn:ist-music:names:context:model:concept:value:address*"), indicating a street address. This string refers to a particular context value represented in the MUSIC Context Model [11]. By referring to concepts represented in the MUSIC ontology, we intend to allow the definition of context distribution policies over all context concepts recognized by a MUSIC application, such as the Instant Social sharing platform [2].

B. *Context-aware policy management architecture*

The context-aware policy management architecture is shown in Figure 7. It was inspired by [10], but extended to support a distributed and decentralized policy management that controls access to context information based on the social group the context requestor belongs to and the current context of this requested entity. The distributed approach currently utilizes the user's personal proxies.

The architecture involves three main entities: *context client*, *context source* (which is the requested user's context provider), and a *proxy server*. The *context client* sends a query for context information to the *context source*. We will assume for this discussion that the query initiator is the user's friend. The context source's Policy Enforcement Point (PEP) retrieves from the Social relations model the query initiator's relationship with the user, creates a context scope request in XACML format, and sends it to the remote Policy Decision Point (PDP) at the

Home proxy server for evaluation. The PDP sends a request to the Policy Information Point (PIP) for policies related to this decision request, which retrieves these policies from the Policy Information Base (PIB). In order to evaluate which policies are applicable for the received request, the PDP queries the Context Information Base (CIB) via the Attribute Information Point (AIP) for the missing (context) attributes. After evaluating the Target element of the retrieved policies and identifying an applicable policy, the PDP evaluates the (context) condition of this policy. Note that the CIB subscribes at the *context source* node to receive updates on context values which are needed for this purpose. When a match is found, the PDP makes a decision and sends it back in the response to the PEP on the *context source* node. After obtaining the authorization decision from the *proxy*, the PEP enforces this decision. If the decision permits the access to the user's location information, the *context source* retrieves its value from its own CIB, formats it in the granted scope, and sends this scoped value in the response to the *context client*.

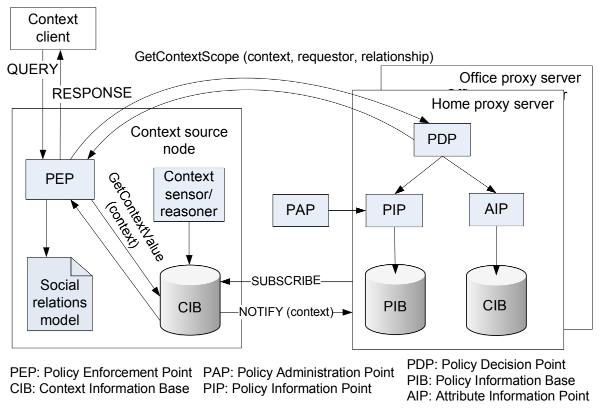
Context client
GetContextScope (context, requestor, relationship)
QUERY
RESPONSE
Office proxy server
Home proxy server
Context source node
PDP
PEP
GetContextValue (context)
Context sensor/ reasoner
PAP
PIP
AIP
SUBSCRIBE
Social relations model
CIB
NOTIFY (context)
PIB
CIB
PEP: Policy Enforcement Point
CIB: Context Information Base
PAP: Policy Administration Point
PIP: Policy Information Point
PDP: Policy Decision Point
PIB: Policy Information Base
AIP: Attribute Information Point

Figure 7. Context-aware policy management architecture

The Policy Administration Point (PAP) is the entity that creates policies and sends them to the PIP at the *proxy server*. After it has received a policy, the proxy's PIP should eliminate the rules which conditions refer to the groups for which this proxy does not manage policies, and store such a modified policy into the PIB. Therefore, in our example, a home proxy would process only rules related to family and friends.

Currently we accommodate context policies that follow the XACML standard. However, since XACML is designed for static attributes and the context information we want to introduce in the rule's condition is volatile, we plan to modify the design of these policies to allow rules to be context dependent - by creating a context-switch and inserting the rules into context conditions. This idea would follow a similar approach to [12], where we extended CPL (Call Processing Language) scripts with context parameters; however, instead of call logic actions there would be policy rules (see Figure 8).

Such a context policy design would enable the proxy to evaluate only the set of rules that are relevant to the user's current situation. This along with dividing the policy management load on several proxy servers responsible for a particular group (or a set of groups) should potentially improve the time to respond to a context query.

```
<context-switch owner="alisa">
   <context activity="clubbing">
      <!-- rule1: allow friends to see your current address -->
      <!--rule2: allow family to see your current city scope -- >
   </context>
     <context day="workday" and timeOfDay="[9a.m.,5p.m.]">
        <!--rule3: allow colleagues to see your current address -->
     </context>
   <otherwise>
      <!--reject-->
   </otherwise>
</context-switch>
```

Figure 8. An example of a CPL policy with the context-switch element

VII. RELATED WORK

Context reasoning derives higher level context information from context sensors. Probabilistic reasoning approaches like Bayesian Networks or hidden Markov models seem to be well suited for this purpose [13]. User social relationships can be inferred from context and used for sharing context information based on policies that utilize social relations. To the best of our knowledge, existing work on inference of user social relations has not yet been part of designing context management and sharing systems. However, inference of user social relations has been a research topic for a long time in the area of social network analysis [14]. Although there are many approaches for mining social relationships, such as [15], [16], and [17], these approaches differ in the number of users, the number of involved social groups, and the nature of the utilized data which they handle. However, only a few researchers have addressed the challenge of analyzing user social relations from interaction data, i.e. based on email and phone logs.

Eagle *et al.* [4] and [5] focused on predicting user social relations. They used interaction data including logs of phone calls and text messages, along with location and proximity information gathered from the cell information of the mobile phone and its Bluetooth interface. In both studies extensive log data were collected for the activities of around 100 users. For the inference task, they applied well known multivariate analysis methods, learned Gaussian mixture models, and well established methods of feature selection. However, we faced additional challenges because we did not have location or proximity data and we tried to infer user social relations from a very limited amount of log data. It is well known in data mining that it is very difficult to infer information from a quite limited amount of data [18]. While properties or behavioral characteristics may be stable over a large group of individuals, the characteristics of individuals may differ quite a lot.

Policies are used for granting (or denying) users access to given resources such as security or network management. Kamienski *et al.* [21] apply policies to the management of ambient networks. Policies are expressed using an extended version of the XACML [20]. Such policies are used to manage network composition in a general architecture for PBMAN (Policy-Based Management of Ambient Network), which is an extension of the IETF PBM framework [22]. Nupur [10] used policies based on the XACML for access control of context information within the ambient network using a centralized Policy Management System (PMS) architecture, which may lead to scalability problems. Although XACML is presented as a standard for access control policies, Toninelli *et al.* [19] and Verlaenen *et al.* [23] propose other languages for policy description. They each argue that policy definition and conflict resolution in pervasive or service-oriented environments demand reasoning capabilities that can be obtained by combining OWL and rule-based languages.

VIII. CONCLUSION

We have presented a rule-based inference approach for deriving user social relations with his/her communication partners based on the log-data collected by their mobile phone. This approach uses a PART rule-based data mining to derive relevant inference rules, which in turn are used to create a classifier and a simple Bayesian network to provide confidence values. User feedback is incorporated to adjust the Conditional Probability Tables of the Bayesian Network and tune the inference rules in order to obtain better classification results. The proposed approach was evaluated on data obtained from three users monitored for a week, resulting in classification success rates of 87% for user$_1$, 85% for user$_2$, and 69% for user$_3$, despite simple rules and very limited log-data.

The derived social relations with the contacts are stored in the FOAF ontology extended with social relation terms. We proposed these relations to be used as attributes when creating context-aware policies to check if the requesting entity belongs to a particular social group. Thus, there is no need for explicitly stating in the policy all the actors that the policy refers to. Context policies also specify the user's context in which a policy action is executed.

In future, we plan to evaluate our social inference approach with at least 12 users to prove its success - and apply it to publicly available data for mining in addition to our data. We will extend this approach to support classification of the user's communication partners to more than one social relationship category. Finally, we will extend these policies with a "context-switch" to enable rules to be context-dependent and perform a performance analysis of this context policy management.

ACKNOWLEDGMENT

The authors of this paper would like to Prof. Gerald Q. Maguire Jr. for fruitful comments to this research work.

REFERENCES

[1] EU IST MUSIC project, Self-Adapting Applications for Mobile Users in Ubiquitous Computing Environments, http://www.ist-music.eu, 2008.

[2] L. Fraga, S. Hallsteinsen, and U. Scholz, "Instant Social – Implementing a Distributed Mobile Multi-user Application with Adaptation Middleware", In Proceedings of the First International DisCoTec Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services (CAMPUS), Oslo, Norvay, June 2008.

[3] N. Paspallis et al., "A Pluggable and Reconfigurable Architecture for a Context-aware Enabling Middleware System", In Proc. 10th International Symposium on Distributed Objects, Middleware, and Applications (DOA'08), Monterrey, Mexico, Springer-Verlag, November 2008.

[4] N. Eagle, S. A. Pentland, and D. Lazer. "Mobile Phone Data for Inferring Social Network Structure". In Social Computing, Behavioral Modeling, and Prediction, pp. 79-88, January 2008.

[5] N. Eagle, A. Pentland. 2006. "Reality Mining: Sensing Complex Social Systems", Personal and Ubiquitous Computing, vol 10(4), pp. 255-268.

[6] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization", In Proc. 15th International Conf. on Machine Learning, pp. 144–151, Madison, Wisconsin, USA, July 1998.

[7] MATLAB, The language of technical computing, http://www.mathworks.com (last visited on January 2009.)

[8] University of Waikato, WEKA - Waikato Environment for Knowledge Analysis, Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/ (last visited on January 2009.)

[9] S. Russell and P. Norvig, "Artificial Intelligence: A modern approach (second edition)". Prentice Hall International, January 2003.

[10] Nupur Bahtja, "Policy Management in Context-Aware Networks", Master of Science Thesis, Royal Institute of Technology (KTH), Stockholm, Sweden, April 2007.

[11] R. Reichle et al., "A Comprehensive Context Modeling Framework for Pervasive Computing Systems", In Proceedings of the 8th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS), Oslo, Norway, Springer Verlag, June 2008.

[12] A. Devlic, "Extending CPL with context ontology", In Mobile Human Computer Interaction (Mobile HCI 2006) Conference Workshop on Innovative Mobile Applications of Context (IMAC), Espoo/Helsinki, Finland, September 2006.

[13] W. Dargie, "The Role of Probabilistic Schemes in Multisensor Context-Awareness", In Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07), pp.27-32, March 2007.

[14] S. Wasserman and K. Faust, "Social Network Analysis, Methods and Applications". Cambridge, UK: Cambridge University Press. 1994.

[15] G. Kossinets and D. J. Watts. "Empirical Analysis of an Evolving Social Network", Science 311, pp. 88-90. 2006.

[16] H. Ebel, L-I. Mielsch, and S. Bornholdt. "Scale-free topology of e-mail networks", Phys Rev E 66: 35103. 2002.

[17] W. Aiello, F. Chung, and L. Lu, "A random graph model for massive graphs", Annual ACM Symposium on Theory of Computing, Proceedings of the thirty-second annual ACM symposium on Theory of computing, pp. 171–180, 2000.

[18] J. M. Kleinberg, "Challenges in mining social network data: processes, privacy, and paradoxes". In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07), pp. 4-5. 2007.

[19] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila. Proteus, "A Semantic Context-Aware Adaptive Policy Model", Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY '07). IEEE Computer Society, pp. 129-140, 2007.

[20] T. Moses (ed.), "OASIS eXtensible Access Control Markup Language (XACML) Version 2.0". OASIS Standard, 1 Feb 2005.

[21] C. Kamienski, J. Fidalgo, R. Dantas, D. Sadok, and B. Ohlman, "XACML-Based Composition Policies for Ambient Networks", Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY '07). IEEE Computer Society, pp. 77-86, 2007.

[22] R. Yavatkar, D. Pendarakis, and R. Guerin, "A Framework for Policy-based Admission Control", IETF RFC 2753, January 2000.

[23] K. Verlaenen, B. De Win, and W. Joosen, "Policy Analysis Using a Hybrid Semantic Reasoning Engine", Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY '07). IEEE Computer Society, pp. 193-200, 2007.

[24] The Friend Of a Friend (FOAF) project, http://www.foaf-project.org/, last visited on January 2009.

[25] XHTML Friends Network (XFN), http://gmpg.org/xfn/, last visited on January 2009.

[26] I. Davis and E. Vitiello Jr, "RELATIONSHIP: A vocabulary for describing relationships between people", http://vocab.org/relationship/, last visited on January 2009.

[27] Google OpenSocial API, http://code.google.com/apis/opensocial/, last visited on January 2009.