SCIMS: A Social Context Information Management System for Socially-Aware Applications*

Muhammad Ashad Kabir, Jun Han, Jian Yu, and Alan Colman

Faculty of Information and Communication Technologies,
Swinburne University of Technology,
Melbourne, Australia
{akabir,jhan,jianyu,acolman}@swin.edu.au

Abstract. Social Context Information has been used with encouraging results in developing socially-aware applications in different domains. However, users' social information is distributed over the web and managed by many different proprietary applications, which is a challenge for application developers as they must collect information from different sources and wade through a lot of irrelevant information to obtain the social context information of interest. Combining the social information from the diverse sources and incorporating richer semantics could greatly assist the developers and enrich the applications.

In this paper, we introduce *SCIMS*, a social context information management system. It includes the ability to acquire raw social data from multiple sources; an ontology based model for classifying, inferring and storing social context information, in particular, social relationships and status; an ontology based policy model and language for owners to control access to their information; a query interface for accessing and utilizing social context information. We evaluate the performance and scalability of SCIMS using real data from Facebook, LinkedIn, Twitter and Google calendar, and demonstrate its applicability through a socially-aware phone call application.

Keywords: Social context, Online Social Networks, Social relationship, Ontology, Access control, Information management.

1 Introduction

Recently we have witnessed an increasing number of efforts aimed at providing socially-aware applications from such domains as pervasive computing, semantic web and information retrieval. These applications exploit users' social context information such as relationship, interaction history, and so on, to provide "smart" services and capabilities [1]. For example, SmartObject considers the user's relationship information to turn on the audio player when friends are present [2].

* This research was supported in part by the Commonwealth of Australia, through the Cooperative Research Centre for Advanced Automotive Technology (AutoCRC)

Review quality is quantified based on the interaction and social relationships of reviewers in [5].

On the other hand, the popularity of Online Social Networks (OSNs) such as Facebook, LinkedIn, Twitter and Google+ (being the prime examples) produce an unprecedented amount of Social Context Information (SCI) as people specify their relationships, update their status and share contents. This phenomenon offers a unique opportunity for this information to be leveraged in creating more intriguing socially-aware applications. As it is, however, the users social context information is distributed all over the web, emerging from and fragmented across many different proprietary applications. Combining this information from such diverse sources could provide a more accurate representation of the users' social world with semantically richer information, and enable a whole new set of socially-aware applications.

Most of the existing works collect and manage social information within the context of an application (as in the above examples), which has two major problems. First, it is a challenge for application developers as they must collect information from different sources and wade through a lot of irrelevant information to obtain the social context information of interest to the targeted functionality. Second, it makes it extremely hard for information owners to control how their information should be exposed to different users and applications. While early context-aware applications relied on ad hoc architectures and representations, it has already been recognized that separating the process of acquiring contextual information from actual applications is key to facilitating application development and maintenance [7]. Nevertheless, managing users' social context information for supporting the development of socially-aware applications is still a challenging task for several reasons:

- **Consistent representation of SCI:** There are different types of social context information. One type is the "object-centric" relationship, identified between people who have shown common interest (e.g., like/tag in Facebook) or participated in common activities or become members of similar groups. This type of relationship has been used in applications to infer preferences [8] and incentives of resource sharing [3]. Another type is the "people-centric" relationship, which is formal and declarative. For example, a person identifies other persons as father, supervisor, school friend, etc. This type of relationship can be used in a socially-aware phone call application as identified in [9]. The different types of social relationships need to be represented consistently to facilitate application use.
- **Inferring social relationships with SCI semantics:** An application may need social context information that is not directly available from the sources but can be derived from basic information. For instance, users may want to filter phone call based on relationship categories such as "family" and "best-friends" that are not provided directly but can be inferred from other available relationships. Thus, there is the need to define and obtain derived relationships (at different abstraction levels) based on the basic relationships (e.g., mother and school-friend) and their semantics (e.g., mother being in family) and attributes (e.g., strength and trust).

– **Preserving owner privacy by fine-grained access control over SCI:** The user's social context information is inherently sensitive and can be further used to infer sensitive information. The scenarios of emerging socially-aware applications require users to share their information for greater benefits but may also compromise their privacy. For example, allowing caller to know the status of callee before calling might reduce interruptions [9], but may also raise serious concerns regarding the privacy and access control over users' status and other data. Thus, users should be able to retain control over who has access to their personal information under which conditions. In addition, a user may want to *fine-tune* the *granularity* of the answer provided to a given query, depending on the context of that query such as *who* is asking, *what* is asked for, user's *current status*, etc. For instance, an employee may be happy that her boss knows her current status is "AtDesk". However, she may not be happy to let the boss know her status is "chatting to friends via instant messaging". Thus users should be able to control access to their information at different *levels of granularity*.

A number of existing works have attempted to gather and manage users' social data. However, they address the above issues in a very limited manner. For instance, PocketSocial [10], Prometheus [11] and MobiSoC [12] each gather a particular type of relationship information but without considering its semantics, and as a consequence are not able to answer the relationship at different levels of abstraction. While Yarta [13] does consider semantics and uses ontology in their relationship representation, it does not consider the granularity in information access in their policy model.

In this paper we introduce *SCIMS*, a social context information management system for collecting, integrating, classifying, inferring and storing social context information from diverse sources. SCIMS allows efficient access to this information while respecting user privacy, in order to facilitate the development of socially-aware applications. This research has three major contributions. First, we propose an ontology based model for representing and storing both *people-* and *object-centric* social relationships, and users' status information. Second, a rich set of social context information has been derived based on information acquired from disparate sources. Third, we propose a way to preserving owners' privacy by allowing the owners to fine-tune the granularity of information access and to specify access control policies. In addition, we have developed a number of adapters to fetch information from various sources and implemented a set of query APIs for applications to access the context information. We have used the semantic web technologies to implement the overall system and also performed evaluations of SCIMS's performance and scalability. A socially-aware phone call application has been developed on top of SCIMS to demonstrate its applicability.

The paper is organized as follows. Section 2 reviews related research. An overview of the proposed SCIMS is given in section 3. Section 4 introduces our approach to modeling and inferring social context information. An access control model to protect context information is described in section 5. Section 6 presents a prototype implementation. Section 7 reports our experimental evaluation with a case study, while section 8 concludes the paper and highlights future work.

2 Related Work

In the field of general context-aware system, there has been significant amount of research effort for modeling and managing context of a physical nature such as location, time, activity, and so on. Comparatively, there have been only limited works concerning contexts of a social nature [2]. Some research efforts have attempted to adopt and extend the Friend of a Friend (FOAF) [4] ontology for representing social relationships [6][14]. They extend the *foaf:knows* object property, the only option offered by FOAF ontology, with sub-properties such as *colleagueOf*, and *friendOf*. However, representing relationships using such object property suffers from generality for two main reasons: (i) it does not allow the specification of different attributes such as strength and trust associated with a relationship; (ii) as a consequence more abstract and rich context information cannot be derived.

OSNs like Facebook, LinkedIn and so on, offer their native APIs [15][16] for accessing their simple, unprocessed social data. These APIs do not provide access to derived relationships like "best friend" or "Colleague of a Colleague". Instead, an application must explicitly crawl through the graph to obtain them.

Policy-based access control has been the subject of extensive research over the past decade. Recent research efforts have tried to integrate semantic technologies both in access control model and policy specifications, thus enabling automated reasoning and policy enforcement over expressive access control specifications. In particular, the R*OWL*BAC [17] and ReBAC [18] models have provided us with useful insight for our socially-aware access control framework. Most policy models for social networking applications [19], and even those designed for pervasive computing applications [20] consider either *role* or *relationship* in their access control but not both. While the work [21] is very close to ours in protecting user social context information, it does not consider *role* in their access control model and therefore is not able to offer the advantages of role based access control. Moreover, they do not consider the obfuscation to support the access control at different granularity level which is an important aspect for access control [28].

Socially-aware applications are often designed from scratch by embedding all management functionalities into the application logic, providing an application-specific data representation models, and acquiring data from one or a few specific external sources (e.g., [2], [23]). In all of these cases, social knowledge has been mined in the context of a single application. Some efforts have already recognized the need to externalize the social context management functionalities and have taken steps towards systematically managing users' social context information.

Prometheus [11] collects user's social data from different OSNs represents it as multi-edged graphs, where vertices correspond to users and edges correspond to interactions between users. The interactions are described with a label (e.g., "football", "music") and a weight specifies the intensity of an interaction, and essentially represents an "object-centric" relationship. Prometheus implements a set of inference functions to answer queries like social strength, proximity, and so on, while enforcing user-defined access control policies. Like Prometheus, PocketSocial [10] also collects social data from different sources. But unlike Prometheus,

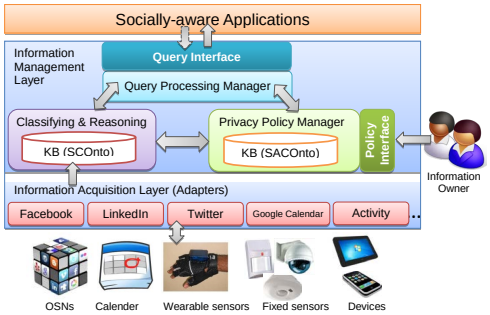


Fig. 1: Social context information management system architecture

it represents social data in JSON objects and supports only REST based APIs like Facebook, and does not provide any inference functions. Neither Prometheus nor PocketSocial represent both the *object-* and *people-centric* relationships with their semantics, and as a consequence they are not able to infer richer information or fine-tune the granularity of information access.

On the other hand, Yarta [13] adopts the FOAF ontology in their relationship representation, which has the drawbacks as discussed above. However, it only considers *people-centric* relationships and does not capture the *object-centric* ones. Even though it does consider social context in its policy model, its access control policy does not take into account granularity in information access. Moreover, its policy model is RDF-based which lacks in generality of OWL DL-based models.

Our work significantly differs from the previously noted approaches in that it not only collects users' social relationship information from multiple sources and stores it in ontologies, but also considers the owners' status information and their semantics, allowing information representation and derivation at different levels of abstraction and consequently facilitating access control and query processing.

3 Overview of the SCI Management System

We introduce an SCI management system, SCIMS, to address the challenges in developing socially-aware applications, as discussed in the introduction. Our proposed architecture for the SCIMS (see Figure 1) comprises two layers: (i) Information Acquisition and (ii) Information Management, and provides a platform for accumulating, storing and managing social context information.

The *information acquisition* layer is responsible for accumulating social context information from various sources such as OSNs, calendar entries, physical sensors, and so on. A common interface is provided so that different adapters can be built based on that interface to collect data from various sources.

For the *information management* layer, we propose an ontology based context model to store the processed social data being collected. The ontology based model provides the facilities of inferring and deducing more complex context information based on the processed data. We also propose a socially aware access

control mechanism to allow information owners to protect their information by specifying privacy policy. Our policy model reflects the human way of thinking by considering social relationship, social role and status information when defining privacy preferences. Moreover, to allow different applications to access the social context information, we introduce a query interface so that application developers can build applications without the need to deal with the details of information representation schema and management. In the sections below, we introduce the detailed capabilities of the components in this SCIMS architecture.

4 Modeling and Inferring Social Context Information

We adopt an ontology based approach to modeling and representing social context information. Ontology based approaches have been evaluated as most promising for context modeling in pervasive computing. It has two main advantages: (i) it creates a common knowledge and understanding of context information, and (ii) it allows context reasoning [24]. We use OWL 2 DL [30], a sub-language of OWL 2, to model social context information. To date, OWL 2 DL has been the most practical choice for most ontological applications as it supports maximum expressiveness while retaining computational completeness and decidability [25].

Despite its intuitive anchoring in everyday life, social relationships or context information may be very challenging to represent in a formal model. However, at a certain level of abstraction, social relationships can be defined as a possible form of "relational ties", as most of the existing works have considered (e.g., [13]). We also take this view to model social relationships. However, we model relationships as *first-class entities* rather than representing them as a generic *link* between people. This way of modeling allows us to benefit from Description Logic (DL) [26] in *classifying* and *reasoning* about relationships at different *levels of abstraction* based on the properties of these relationships.

In order to provide a suitable trade-off between formal modeling and application specific concepts, we define a set of basic concepts as building blocks for the creation of socially-aware applications. Towards this goal, we propose both upper and domain-specific (lower) ontologies. The *upper-level* ontology captures the basic concepts, abstracted from the analysis of: (i) real-world use case scenarios, like the socially-aware phone call [6]; (ii) different sources of context information, e.g., Facebook, LinkedIn, etc. Through a standard specification, this upper ontology can be shared, reused, and adapted to others systems. This can be customized to represent terms in different forms for different users. The upper ontology is also extensible to allow for the incorporation of new concepts and the specialization of concepts and constraints for a particular application, which we refer as *domain-specific* ontologies.

4.1 Core Upper Ontology

Our upper ontology model, called Social Context Ontology (*SCOnto*), defines four first-class entities, namely, *Person*, *SocialRole*, *Relationship*, and *CurrentSta-*

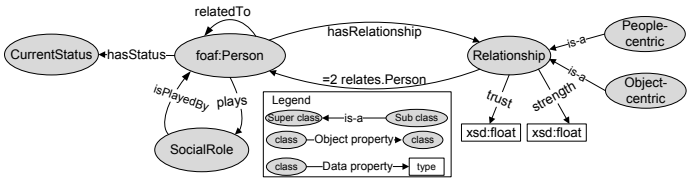


Fig. 2: Social context upper ontology – core classes and properties

tus. These entities can be organized into a hierarchy where the root of the hierarchy is the term *SCOnto*. To define *Person*, we adopt and extend the FOAF ontology. In the FOAF ontology, a person's status is represented as literal. To support reasoning over a person's status at different granularity levels we change the type of "status" from data property to object property (labeled "hasStatus") which links the Person and CurrentStatus class. The FOAF model also includes a set of attributes that provide information about a person, such as *name*, *email*, *phone-number*, *gender* and so on. In real-world, a person may have different types of relationship as she plays various roles. To identify a person at such a fine grained level, we introduce the concept of *SocialRole* which *isPlayedBy* the *Person*. The core concept of our model is the (social) *Relationship* which is a subclass of *SCOnto* and *relates* exactly two persons that can be defined in OWL 2 using the DL [25] syntax as follows:

$$Relationship \sqsubseteq SCOnto \sqcap = 2\, relates.Person$$

where those persons should be different individuals (i.e., the *relatedTo* object property type should be set to *Irreflexive*).

Figure 2 shows the *SCOnto* model that can be read as follows. A *Person* may have *Relationship*(s); each *Relationship* can be classified as *object-* and *people-centric*, *relates* exactly two Persons; thus through a *Relationship* a *Person* is *relatedTo* another *Person* and *plays* a particular *SocialRole*; also a *Person* may have a *CurrentStatus*. The set of concepts included in our ontology is obviously non-exhaustive. However, we believe that this ontology can be profitably used to model many application scenarios.

4.2 Domain-specific Ontologies

Based on the core ontology we define some domain-specific ontologies (Figure 3). In particular, we propose a fine-grained relationship model for *Family*, *EducationBasedFriend*, *Work*, and *CommonInterest* relationships. Basically, we rely on users' Facebook information to deduce *family* relationships. Facebook allows people to maintain a family list, where a user can annotate a person in his family list as *Brother*, *Father*, *Uncle* and so on, which ultimately indicates the role played by the person being annotated from the user's point of view rather than the specific relationship between the user and that person. For instance, if a user annotates a person in his/her family list as "Father", the relationship between

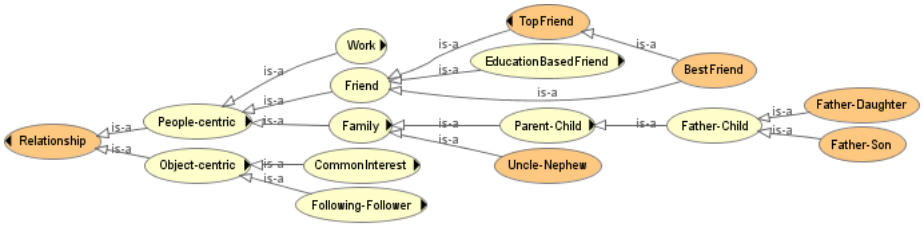


Fig. 3: Domain specific ontologies – an excerpt of different types of relationship

them could be a "Father-Son" or "Father-Daughter" relationship, which can be inferred by utilizing the user's gender information. In our ontology, we use DL for defining such derived relationships. Some examples are shown below:

$$FatherSon \sqsubseteq FatherChild \sqcap \exists relates((Person \sqcap \exists plays \cdot Father) \sqcap (Me \sqcap gender \cdot male))$$
$$UncleNephew \sqsubseteq Family \sqcap \exists relates((Person \sqcap \exists plays \cdot Uncle) \sqcap (Me \sqcap gender \cdot male))$$

Here, *Me* is a subclass of *Person* that specifies the user from whose perspective the relationship is computed. However, in some cases, we can deduce the relationship directly from the role name. For example, if a person is annotated as "husband", we can directly deduce the relationship as "husband-wife"; the same applies for *cousin* and *partner* relationships.

Facebook is also a potential source of collecting *friend* information. A person is an education-based-friend if he/she studied with the user, so we categorize *EducationBasedFriend* as *PrimarySchoolFriend*, *HighSchoolFriend*, *CollegeFriend*, and *GraduateSchoolFriend*. Such basic friendship information can be used to further deduce and classify friendship relationships such as *TopFriend*, *BestFriend* and so on. For instance, a user can define that a person is a top friend if he/she is a *HighSchoolFriend*, *CollegeFriend* and *GraduateSchoolFriend*, as follows:

$$TopFriend \equiv HighSchoolFriend \sqcap CollegeFriend \sqcap GraduateSchoolFriend$$

Also, a person can be defined as a best friend if he/she is a top friend and has relationship *strength* greater than a certain value (e.g., 0.8)

$$BestFriend \equiv TopFriend \sqcap \geq 0.8\, strength$$

This relationship *strength* information can be computed based on the user's interaction activities in OSNs [26].

We classify the *work* relationship for an educational organization as *Student-Teacher*, *Student-Supervisor*, and *Colleague* relationships. From LinkedIn, we can get current and past positions of a user and all the persons connected with that user. Based on the institution *name* or *id* and the *positions* held by them, we can deduce their work relationships. For instance, if two persons hold any of the staff roles in an organization, we can deduce their relationship as being *Colleague*. Based on the specific types of staff roles, we can further classify the relationship as: *AcademicStaff-AdminStaff*, *AcademicStaff-AcademicStaff*, *AdminStaff-AdminStaff*.

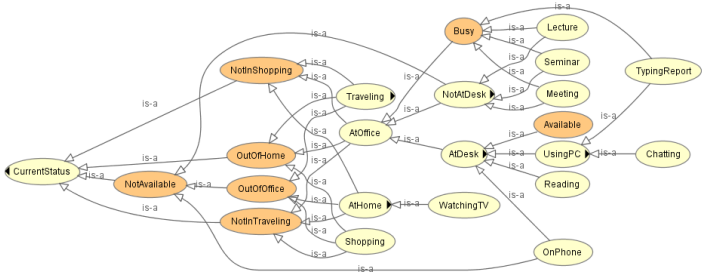


Fig. 4: An excerpt of current status ontology

Table 1: Granularity levels of classes in the *CurrentStatus* ontology

Granularity Level	Classes
1	NotAvailable, Available
2	Busy
3	OutOfHome, OutOfOffice, NotInTraveling, NotInShopping
4	Shopping, Traveling, AtOffice, AtHome
5	NotAtDesk, AtDesk, WatchingTV
6	Lecture, Meeting, Seminar, UsingPC, OnPhone
7	TypingReport, Chatting

From Twitter we can obtain *Following-Follower* relationship, where *Following* includes the person(s) whom the user is interested and following, and *Follower* includes person(s) who are following the user. *LivingAddress* based relationship can be acquired from Facebook and further categorized at *Country* and *City* levels. *CommonInterest* includes the relationships with person(s) who like the same *Digital-Content* (e.g., those people use "like" to express in Facebook), are members of the same *Group* (e.g., soccer, cricket, and so on), are interested in the same *Event* (e.g., conference), or have *CommonResearchInterest* which can be collected from LinkedIn.

Figure 4 shows part of the *CurrentStatus* ontology by considering some activities (both social activities (more than one person is involved) and individual activities) in particular domains such as home, office, shopping and travel. Again, the set of information defined in this ontology is obviously non-exhaustive and can be further extended based on the need of the application or domain of interest. We assume that we can collect a user's raw status information from the user's diary, Facebook, LinkedIn, or Twitter status update. The user's raw status information will be one of the leaf nodes in the ontology. We can deduce the status of being *Available* or *NotAvailable* for telephone call, or *Busy* based on the raw information, as follows:

$$Busy \equiv Lecture \sqcup Meeting \sqcup Seminar \sqcup TypingReport$$
$$Available \equiv AtDesk \sqcap \neg OnPhone$$
$$NotAvailable \equiv OnPhone \sqcup NotAtDesk \sqcup OutOfOffice$$

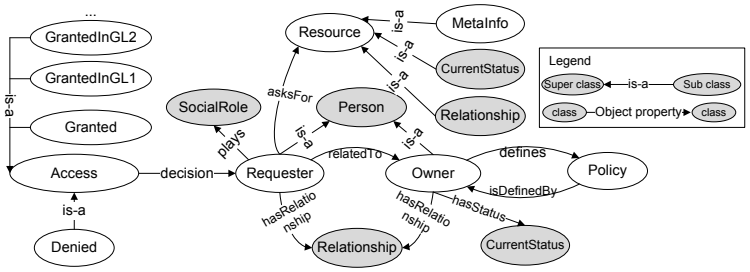


Fig. 5: Access control policy model

We have set the *granularity level* for each node/class in the ontology as an increasing number from root to leaf nodes, as depicted in Table 1. Thus for a given *status*, we can answer the *current status* of a user at different levels of granularity. For instance, for a status "Meeting" with granularity level 4, we can say that the user is "AtOffice". Similarly, we also specify the granularity level for relationship ontologies.

5 Preserving Privacy

5.1 A Socially-Aware Access Control Policy Model

For controlling access to SCI, we have proposed an access control policy model to protect owners' social context information based on the context information itself. Thus our policy model, called Socially-aware Access Control Ontology (*SACOnto*) (see Figure 5), reuses the concepts from core *SCOnto* model (shown in shaded ellipse) to define under which conditions (i.e., the social context) a given resource is accessible. The accessible resource could be any social context information (e.g., relationship, status) including the meta-information related to any relationship of a user such as the *numbers* of friends and colleagues.

Studies have revealed that users want to define policies that apply to all people with certain relationships or roles rather than explicitly name each person [20, 21]. Thus in access conditions, we consider both requester's role and relationships with information owner to reflect the way users tend to group similar sets of people when deciding to share resources to other users. Therefore, our model is able to represent existing policy models, such as popular role-based and recently emerged relationship-based [16, 17], with enhanced expressive capabilities.

In addition, we consider the owner's current status information which is another important aspect of defining policies [20]. Furthermore, our model can be easily extended to incorporate users' physical context information such as *location*, *time*, and so on, in making access decision. For the sake of brevity, however, we do not consider it here. Users typically specify their policies using combinations of the main dimensions driving access control decisions. For example,

– Who is requesting access and what is her relationship with the owner?

- What role is the requester playing?
- The current status of the owner when the request comes?
- What type of resource is being requested and what are its characteristics?

Our ontological model enables a user (owner) to specify logical relations between the above fundamental elements. In the owners' conceptual model, such who/what/when dimensions are typically interrelated. Our model (Figure 5) captures these dimensions which can be read as follows: A *Policy isDefinedBy* an *Owner* which specifies *Access decision* (*Denied* or *Granted* or *GrantedInGL1* (granted in granularity level 1) and so on) for *Requester*(s) who plays a *SocialRole*, *relatedTo Owner* and *hasRelationship* with owner named *Relationship*, *asksFor* a *Resource*, when the *Owner hasStatus CurrentStatus*. For instance, consider this policy, "Any of my colleagues can access my current status when I am in meeting" (policy #1). In such case, the owner's access decision is based on the relationship with the requester (who), the resource being accessed (what), and the status of the owner at request time (when). While traditional approaches generally consider these dimensions orthogonal, our model reflects the owner's way of thinking by supporting the cross dimensional definition of policies.

Another key feature of our model is the ability to specify access permissions at different levels of granularity, i.e., disclosing information at a certain level of abstraction by hiding the specific fact. For instance, "Upon being asked the current status by supervisor, a student may want to state her status as being 'AtDesk' (granularity 5) while she is actually chatting with friends" (policy#2).

5.2 Policy Specification and Enforcement

We use DL to specify access control policies, as one of its main advantages is that a DL reasoner can be used for automatic inconsistency detection both in policy specification and enforcement. A graphical user interface can be provided to the owner for specifying her privacy policies which can be easily transformed to the DL format. The template of our policy rule in the DL is defined as follows:

$$\begin{aligned} &\mathit{Denied}\ /\ \mathit{Granted}/\mathit{GrantedInGL}\mathbf{?n} \sqsubseteq \mathit{Access} \sqcap \exists \mathit{decision}(\mathit{Requester} \sqcap \\ &\quad \exists\ \mathit{hasRelationship}.\mathbf{?Relationship} \sqcap \exists \mathit{asksFor}.\mathbf{?Resource} \sqcap \\ &\quad \exists\ \mathit{plays}.\mathbf{?SocialRole} \sqcap \exists \mathit{relatedTo}.(\mathit{Owner} \sqcap \exists \mathit{hasStatus}.\mathbf{?CurrentStatus})) \end{aligned}$$

where bold words with a preceding '?' mark are variables that will be filled based on the owner's privacy preference statements. For example, we can represent the above policy #2 as follows:

$$\begin{aligned} \mathit{GrantedInGL}\mathbf{5} \sqsubseteq\ & \mathit{Access} \sqcap \exists \mathit{decision}(\mathit{Requester} \sqcap \exists \mathit{asksFor}.\mathbf{CurrentStatus} \sqcap \\ & \exists\ \mathit{hasRelationship}.\mathbf{Student\text{-}Supervisor} \sqcap \\ & \exists\ \mathit{relatedTo}.(\mathit{Owner} \sqcap \exists \mathit{hasStatus}.\mathbf{Chatting})) \end{aligned}$$

Once the policy is transformed to the DL format, it is asserted to the owner's access control ontology. For instance, policy #2 will be asserted to the *SACOnto* as an equivalent class of *GrantedInGL5* class. When an access request or query

comes which basically corresponds to invoking a function from the access APIs (see next section), some facts about the context of the query are temporarily asserted to the ontology – namely the requester *name* and *resource* being requested. In addition, an individual of *Access* class with a link to the requester using *decision* property is asserted. After that the reasoner is fired which classifies that individual of access class to one of its subclasses, i.e., Denied, GrantedInGL1, and so on, based on the defined policies. Query processing manager considers this result for answering the query.

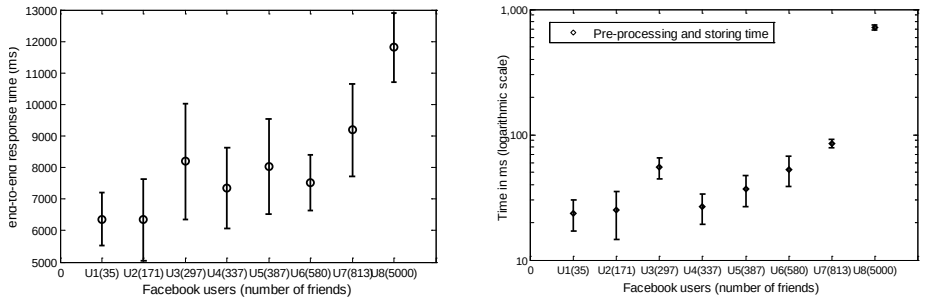
6 Prototype Implementation

We have implemented a prototype system for SCI management, SCIMS, in Java 2 Platform Standard Edition (J2SE). For managing the knowledge base (*SCOnto* and *SACOnto*), we have used OWL API 3. We wrote *adapters* for Facebook, LinkedIn, Twitter, and Google calendar using their native APIs for fetching users' social data, and follow the OAuth 2.0 protocol for authorization. For inferring and policy execution, we have used reasoners compliant with OWL 2 DL. In particular, for evaluation purposes, we have incorporated five different DL reasoners: Pellet 2.3.0, HermiT 1.3.5, TrOWL 0.8.1, Fact++ 1.5.2 and RacePro 2.0.

To aid developing socially-aware applications, we have implemented a set of APIs exposed as Web services. *Policy APIs* allow owners to add, update and delete their privacy preferences – `addPRule`, `deletePRule`, `getPRule`, *updatePRule*, and so on. *Management APIs* provide functionality (i) to manipulate ontologies, allowing inserting, editing and deleting both concepts and individuals, and (ii) to configure and execute information acquisition operations – `insertConcept`, `insertRelationship`, `updateCurrentStatus`, *startInfoAcqFromFB*, and so on. *Query APIs* for accessing both context and meta-context information – `getCurrentStatus`, `whatIsMyRelWithB`, `getRelNameByGLevel`, `isAhasRelRwithB`, `getNumberOfGraduateSchoolFriend`, *getColleagueOfAColleague*, and so on. Since most of the interface names are self explanatory, we do not provide description of those APIs. Also due to page limit, we cannot give the complete lists of the all types of APIs.

We adopt a DL based query language, named SPARQL-DL [28], for query processing and have used the derivo 1.0 SPARQL-DL query engine with above mentioned DL reasoners. Recently, SPARQL-DL was introduced as a rich query language for OWL 2, which is a distinct subset of SPARQL (a RDF based language), tailored to ontology specific queries. Therefore, different aspects of context such as granularity can be answered easily and in simpler fashion. For example, `getStatusByGLevel(A,gLevel)` can be realized in SPARQL-DL as follows:

```
PREFIX sc:<http://www.ict.swin.edu.au/Ontology/SCOnto#>
SELECT ?status {
PropertyValue(A, sc:hasStatus, ?statIns)
    DirectType(?statusName, ?statIns)
    Annotation(strictSubClassOf(?statusName, ?status),
    sc:granularityLevel, gLevel)}
```



(a) Fetching information from Facebook **(b)** Inserting information into KB

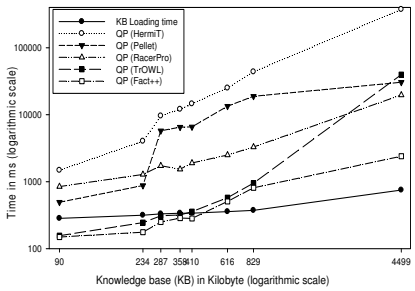
Fig. 6: Time of acquiring social context information from Facebook

7 Experimental Evaluation and Case Study

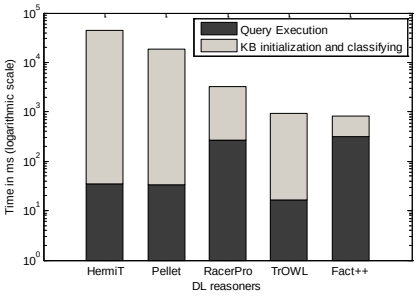
The management system prototype is deployed and evaluated on a machine with Core 2 Duo E8400 3 GHz processor, 3 GB RAM, WinXP professional edition SP3, and Java 1.6. Our evaluation had three goals: (1) assess the systems scalability in acquiring social information using different sizes of real social data of multiple users; (2) measure the systems performance in executing a set of queries over a large data set using different existing reasoners; (3) validate the system by developing a case study social application on mobile phones.

Scalability Evaluation for Information Acquisition. To assess the scalability of the system regarding *information acquisition*, we evaluate the time cost of acquiring information using two metrics: (i) end-to-end response time – time required to fetch information from different sources – Facebook, LinkedIn, Twitter, and Google calendar – to quantify the network overhead; (ii) Preprocessing and storing time – time required to extract the data of interest from fetched objects, transform it to the suitable format, and insert it into the knowledge base (ontologies), to quantify the performance of inserting information into SCIMS.

In these experiments, we choose eight different Facebook users who have increasing numbers of friends, including a user with 5000 friends which is the highest number Facebook allows. These users gave us permission to collect their social data. For each user, we run the experiment 50 times. Figure 6 shows the result where error bars depict standard deviation. For the highest number of friends (i.e., 5000), the average end-to-end response time was 12 sec (see Figure 6a) and the average time for preprocessing and inserting information into knowledge base (KB) was around 700 ms (see Figure 6b), which is an acceptable result, since this information acquisition is usually performed off-line and might not be required to update very frequently. Similarly, we have done experiments using LinkedIn, Google calendar and Twitter users; however, due to page limit those results are not included here.



(a) Loading and QP time over various size of KB using five different reasoners



(b) Details of QP time for a particular Knowledge-Base sized 829 kilobytes

Fig. 7: System performance on query processing (QP)

Performance Evaluation for Query Processing (QP). For measuring the system performance on *query processing*, we use three metrics: (i) time required to load KB, (ii) time for classifying KB, and (iii) time to answer a set of queries – `isAFamilyOfB`, `getAllRelationships` and `getStatusByGLevel`. The KB that has been built in previous experiments using users' social context information acquired from Facebook, LinkedIn, Twitter and Google calendar, is used for performance measurement. Figure 7a shows the result (average of 50 runs) of loading time which basically depends on the size of the knowledge base and not associated with any reasoners. Figure 7a also shows the result of query processing time (the sum of KB classification and query execution time) over various size of KB using five different reasoners. The result of QP shows that for a small sized KB, TrOWL performs very well. But as the size increases the computation cost increases dramatically. However, Fact++ outperforms consistently from small to large sized KBs. Figure 7b shows further details of query processing performance (separating KB classification and query execution) of different reasoners for a particular sized KB (i.e., 829 kilobytes – contains 953 relationship instances).

Case Study – A Socially-Aware Phone Call Application. To demonstrate the real-world applicability of our approach, we have developed a proof-of-concept application, called socially-aware phone call[1]. This application leverages the social context information in SCIMS to decide whether to ring, vibrate, reject, or reject and send status when a call comes. Thus for each incoming call, the application invokes `getMyCurrentStatus` and `whatIsMyRelwithB` functions to get the current status and the relationship information with the caller, respectively, and acts based on the preference defined by the callee. On the other hand, before calling, using the same application one can also obtain the status of the intended callee by invoking `getCurrentStatusOfB` function to judge the suitable time of making the call.

[1] `http://www.ict.swin.edu.au/personal/akabir/sphone/spcall.htm`

The application is written in Java for mobile devices running Android OS and was tested on LG Optimus One which communicates with SCIMS over a 3G network. We have tested meeting and seminar scenarios during a month where a user defines her *filtering preferences* using the application as "(1) If my status is *meeting* or *seminar* and a call comes from *friend*, action is reject; (2) For *family*, action is reject and forward my status at granularity level 2 (Busy); (3) for *colleague* action is reject and forward status at granularity level 6 (i.e., meeting or seminar); and (4) for *supervisor* action is vibrate". To forward the status information of the callee (for case 2 and 3), application invokes the `getStatusByGLevel` function with particular granularity level as an attribute value (as specified in the filtering conditions). We also use policy #1 as user's *privacy preference* regarding access to her status information by caller (to judge the suitable time of making the call) that SCIMS collects from user's Google calendar. We observe that the application acts well in its real-time constraint, i.e., the application is able to acquire information from SCIMS and make decision before the call is forwarded to the voice-mail of the callee.

8 Conclusions and Future Work

In this paper, we have presented a novel social context information management system, SCIMS, to aid the development of complex socially-aware applications. Our system utilizes existing OSNs to accumulate user's social context information; provides rich semantic support for representing and inferring SCI, particularly social relationships and status; offers flexible policies for controlling access to SCI based on owners' preferences; provides a generic query interface for the use of the SCI. In particular, our semantic based approach to modeling and managing SCI provides the advantages of reusability and extensibility. The SCIMS implements a set of adapters to retrieve social data from different OSNs, stores information in an ontology-based knowledge base, and provides a number of interfaces for accessing and managing this information. We have demonstrated the efficacy and usability of the proposed system by conducting a performance and scalability evaluation and by developing a socially-aware phone call application running on Android phones.

We are currently extending the SCI management system along a number of directions, including support of continuous query over SCI and system realization in different platform settings (P2P, Cloud based and Mobile).

References

1. Kabir, M. A., Han, J., Colman, A.: Modeling and Coordinating Social Interactions in Pervasive Environments. In: Proc. of the ICECCS, pp. 243–252 (2011)
2. Biamino, G.: Modeling social contexts for pervasive computing environments. In: PerCom CoMoRea Workshop, pp. 415–420. (2011)
3. Li, J.,Dabek, F.: F2F: Reliable storage in open networks. In: P2P Workshop, (2006)
4. Friend of a Friend (FOAF), `http://xmlns.com/foaf/spec/`

5. Lu, Y., Tsaparas, P., Ntoulas, A.,Polanyi, L.: Exploiting social context for review quality prediction. In: Proc. of the 19th Intl. Conf. on WWW. pp. 691–700. (2010)
6. RELATIONSHIP. http://vocab.org/relationship
7. Dey, A.K., Abowd, G.D.,Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum.-Comput. Interact., **16**(2), (2001)
8. Mislove, A., Gummadi, K.P.,Druschel, P.: Exploiting social networks for internet search. In: Proc. 5th Workshop on Hot Topics in Networks. pp. 79–84. (2006)
9. Toninelli, A., Khushraj, D., Lassila, O.,Montanari, R.: Towards socially aware mobile phones. In: 1st Social Data on the Web Wrokshop. Germany: CEUR. (2008)
10. Bo, X., Gronowski, K., Radia, N., Svensson, M.,Ton, A.: PocketSocial: Your distributed social context now in your pocket. In: PerCom Workshops, (2011)
11. Kourtellis, N., et al.: Prometheus: User-Controlled P2P Social Data Management for Socially-Aware Applications. In: Middleware, pp. 212–231. (2010)
12. Gupta, A., Kalra, A., Boston, D.,Borcea, C.: MobiSoC: a middleware for mobile social computing applications. Mobile Netw. and Appl., **14**(1), 35–52. (2009)
13. Toninelli, A., Pathak, A.,Issarny, V.: Yarta: A Middleware for Managing Mobile Social Ecosystems. In: Advances in Grid and Pervasive Comp, pp. 209–220. (2011)
14. Devlic, A., et al.: Context inference of users' social relationships and distributed policy management, In: PerCom Workshops, pp. 1–8. (2009)
15. Graph API – Facebook developers, http://developers.facebook.com/
16. LinkedIn APIs, https://developer.linkedin.com/apis
17. Finin, T., et al. :ROWLBAC: representing role based access control in OWL. In: Proc. of the 13th SACMAT, pp. 73–82. (2008)
18. Fong, P.W.L., Siahaan, I.: Relationship-based access control policies and their policy languages. In Proc. of the 16th SACMAT, pp. 51–60. ACM (2011)
19. Carminati, B., Ferrari, E., Heatherly, R., Kantarcioglu, M.,Thuraisingham, B.:A semantic web based framework for social network access control, In: Proc. of the 14th SACMAT, pp. 177–186. (2009)
20. Jagtap, P., Joshi, A., Finin, T.,Zavala, L.: Preserving Privacy in Context-Aware Systems. In: Proc. of 5th Intl. Conf. on Semantic Computing, pp. 149–153. (2011)
21. Toninelli, A., Montanari, R., Lassila, O.,Khushraj, D.: What's on Users' Minds? Toward a Usable Smart Phone Security Model. Pervasive Computing, **8**(2), (2009)
22. Khalil, A.,Connelly, K.: Context-aware telephony: privacy preferences and sharing patterns. In: Proc. of the 20th CSCW Conf., pp. 469–478. (2006)
23. Beach, A., et al.:Fusing mobile, sensor, and social data to fully enable context-aware computing. In: Proc. of the Workshop on Mob. Comput. Sys & Appl. (2010)
24. Bettini, C., et al.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput., **6**(2), 161-180. (2010)
25. Riboni, D.,Bettini, C.: OWL 2 modeling and reasoning with complex human activities. Pervasive Mob. Comput., **7**(3), 379-395. (2011)
26. Baader, F., The description logic handbook: theory, implementation, and applications. Cambridge Univ Press. (2003)
27. Xiang, R., Neville, J.,Rogati, M.:Modeling relationship strength in online social networks. In: Proc. of the Intl. Conf. on World Wide Web, pp. 981-990. (2010)
28. Wishart, R., Henricksen, K.,Indulska, J.: Context Privacy and Obfuscation Supported by Dynamic Context Source Discovery and Processing in a Context Management System. In: Porc. of the UIC, pp. 929-940. (2007)
29. Sirin, E.,Parsia, B.: Sparql-dl: Sparql query for owl-dl. In: OWLED (2007)
30. OWL 2 Web Ontology Language, http://www.w3.org/TR/owl2-overview/