Chapter 19
SocioPlatform: A Platform for Social Context-Aware Applications

Muhammad Ashad Kabir, Alan Colman and Jun Han

Abstract With an explosive growth in the popularity of social media and increasing prevalence and features of advanced mobile devices, interest has grown significantly in applications that are aware of users' social context and are able to assist them in their daily activities. A key requirement of developing social context-aware applications is the platform support to reduce the complexity of engineering such applications. In this chapter, the authors present such a platform, namely *SocioPlatform*, to aid the development of social context-aware applications by acquiring, reasoning, storing and provisioning different types of social context information, and managing their runtime interactions and adaptation. The platform hides the complexity of managing social context, and thus assists the development of social context-aware applications. The authors demonstrate the feasibility and applicability of the platform by developing two different types of such applications.

19.1 Introduction

Context-Aware Computing is a paradigm that aims to make pervasive applications more intelligent and accessible, and is increasingly gaining attention in the research community. The notion of context is widely appreciated today, and usually refers to information about systems, entities, and their environments. Software applications that adapt their behaviour with the changes of context information (e.g., location, temperature, and time) are called *context-aware* applications. A context-aware application uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task (Dey 2001).

M. A. Kabir (✉) · A. Colman · J. Han
School of Software and Electrical Engineering,
Swinburne University of Technology, John Street, Hawthorn,
PO Box 218, Melbourne, Australia
e-mail: akabir@swin.edu.au

A. Colman
e-mail: acolman@swin.edu.au

J. Han
e-mail: jhan@swin.edu.au

P. Brézillon, A. J. Gonzalez (eds.), *Context in Computing,*
DOI 10.1007/978-1-4939-1887-4_19

Humans, however, are social beings. Hence, the notion of social context-awareness (in short *social awareness*) extends the vision of context-aware computing. An application is socially-aware if it uses social context information (e.g., social relationships, social roles, social interactions and situations) to adapt its behaviour (Ferscha 2012).

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 (Dey et al. 2001; Henricksen and Indulska 2006). Therefore, a number of software architectures, frameworks and platforms have been proposed for developing and managing context aware applications (see Raychoudhury et al. 2013 for a survey). Existing software architectures and platforms for context-aware applications, however, mostly address contexts of a physical nature such as location, time, and activity, and so on.

Taking account of social context poses additional challenges for application developers as they must define or collect social context information from various sources, mediate/coordinate social interactions across parties and manage them in a consistent manner. There has been comparatively only limited work investigating contexts of a social nature such as social roles, interaction- and connection-oriented social relationships and social situations (Kabir 2014b). Even though recently some works have attempted to manage social context (e.g., Kourtellis et al. 2010), they are limited in representing different aspects of social context. Furthermore, there is a lack of support for managing the acquisition, changes and provision of various types of social context.

In this chapter, we present a platform, called *SocioPlatform*, to aid the development of socially-aware applications. The essence of our approach is to hide the complexity of acquiring, classifying, inferring, storing and managing social context by providing a supporting platform, and thus assist the development of socially-aware applications. The platform provides a number of functionalities. It acquires social context information from various sources; classifies, integrates and stores such information into a knowledge base. It supports the specification of reasoning rules and the derivation of a richer set of social context information. The platform also enforces users' privacy preferences in accessing their social context information and allows users to specify their privacy preferences in a consistent manner. The platform provides efficient access of social context information by implementing a query interface so that application developers can use the interface to access users' social context information. The platform also provides runtime environment for mediating social interactions based on interaction-oriented social relationships and supports their runtime adaptation.

The chapter is organized as follows. Section 19.2 describes social context and key requirements of developing socially-aware applications. After giving an overview of our SocioPlatform in Sect. 19.3, Sects. 19.4 and 19.5 present the two key components of the SocioPlatform architecture followed by the presentation of a prototype implementation in Sect. 19.6. Section 19.7 reviews related work and Sect. 19.8 concludes the chapter.

19.2 Social Context and Socially-Aware Applications

19.2.1 Social Context

In context-aware computing area, early works on context-awareness referred to context as primarily the location of people and objects (Schilit and Theimer 1994). In recent works, context has been extended to include a broader collection of factors, such as physical and social aspects of an entity (Dourish 2004).

Schmidt et al. (1999) present a model of context with two distinct categories: human factors and physical environment. *Human factors* consist of three categories: information about the user (*e.g.,* profile, emotional state), the user's social environment (*e.g.,* presence of other people, group dynamics), and the user's tasks (*e.g.,* current activity, goals). *Physical environment* also consists of three categories: location (*e.g.,* absolute and relative position), infrastructure (*e.g.,* computational resources), and physical conditions (*e.g.,* noise, light). This model gives a classification according to specific contextual factors, but does not provide a formal definition. Dey (2001) presents a survey of alternative view of context, which are largely imprecise and indirect, typically defining context by synonym or example. Finally, he offers the following definition of context, which is perhaps now the most widely accepted: "*Context is any information that can be used to characterise the situation of an entity. An entity is person, place or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves*". Henricksen (2003) relates context to tasks, rather than to interactions between users and applications, as in the definition of Dey. She separates the concepts of context, context modelling and context information. Henricksen argues that context represents a nebulous concept, is difficult to define and bound, where as context models and context information are well defined and understood, and are primary interest in constructing context-aware systems.

Several studies have attempted to define and represent social context from different perspectives. Han et al. (2008) define social context as the user's social surroundings, that is to say, the social relationships of the user. Eugster et al. (2009) rely on a more restricted definition of social context. They consider distributed objects as peers and the social context of a peer represents its awareness of the existence of other peers. Zheng et al. (2007) identify social context as one of the essential elements of the context space for online social interaction. They consider social context as social, cultural, psychological, and emotional influences on online social interactions. Wang et al. (2010) analyse the role of the social group in a ubiquitous computing environment as a source of contextual information. They define social context as: *"Information relevant to the characterisation of a situation that influences the interactions of one user with one or more other users"*. Biamino (2011) views social context as social aggregations or social groups, and defines social context using 3-tuple expression (<number of nodes, number of connections between them, nature of relations between the nodes>) that characterises a social network. Endler et al.

(2011) introduce the term "situated social context" to enable location-based spontaneous interaction among people and define the term as: "*Situated Social Context of an individual is the set of people that share common spatio-temporal relationship with the individual, which turn them into potential peers for information sharing or interacting in a specific situation*". Schuster et al. (2012) combines the concept of social context with pervasive context and introduce the term "pervasive social context" which they define as: "*Pervasive Social Context of an individual is the set of information that arises our of direct or indirect interaction with people carrying sensor-equipped pervasive devices connected to the same Social Network Service"*.

As can be seen from the above discussion, the term "social context" can have many meanings or definitions, but most of the above works view social context as possible forms of *relationships* and *interactions* among people. Taking this insight, the following interpretations are adopted in this chapter:

- *Social Context* characterises social milieu[1] of an individual with respect to another individual or a group of individuals.
- A *Social Context Model* represents a subset of the social milieu, which we define in terms of social roles, relationships, interactions and situations, of an individual with respect to another individual or a group. The social context model is employed by a given socially-aware application, is usually explicitly specified by the application developer but may evolve over time.
- *Social Context Information* is a set of data, gathered from various sources (*e.g.,* social media) or explicitly specified by human, that conforms to a social context model. It provides a snapshot that approximates the state, at a given time, of the subset of the social context encompassed by the model.

19.2.2 Socially-Aware Applications

Based on the factors that dominate the applications' behaviour, we categorize socially-aware applications as data-centric and interaction-centric applications.

In *data-centric* socially-aware applications, data on social context information such as social roles, social situations and connection-oriented social relationships, are the basis of the applications' behaviour. The *connection-oriented* relationships represent users' relational *ties* which can be further categorised as object-centric and people-centric relationships (Kourtellis et al. 2010). An *object-centric* relationship is identified between people who have shown common interests or participated in common activities or become members of similar groups. This type of relationship has been used in applications to infer preferences (Gummadi et al. 2006) and incentives of resource sharing (Li and Dabek 2006). The *people-centric* relationship is a formal and declarative definition of a direct relationship between people. For example, a

[1] Refers to the social setting or environment in which people live or something happens (Bauer and Gaskell 1999)

person identifies other persons as father, supervisor, school friend, etc. This type of relationship can be used in an application to turn on the audio player when friends are present (Biamino 2011), or an application to quantify review quality (Lu et al. 2010), or a socially-aware phone call application (Kabir et al. 2014a).

A smart socially-aware *phone-call application* uses the social relationship information available from online social networks to determine the type of the relationship between the caller and the callee and consequently decide whether to vibrate (for calls coming from family), or keep silent but automatically send a message to the caller (for calls from close friends) or without sending a message (for other calls), when the user is in a situation such as meeting. On the other hand, using the application a caller can obtain the situation of an intended callee to check whether it is a suitable time to call.

To develop such a data-centric socially-aware application needs to fulfill the following key requirements:

- *First*, an application should *acquire* its user's social context information. An application may need social context information that is not directly available from external sources but can be derived from the available basic information. For instance, users may want to filter phone calls based on situation categories such as "busy" that may not be acquired from sources but can be *inferred* from collected data by specifying rules (*e.g.,* meeting or seminar being in busy). Similarly, users may want to filter phone calls based on relationship categories such as "family" and "best-friends" that are not provided directly but can be inferred from the semantics of the relationship categories. Thus, it is required to collect, classify, infer and manage different social context information.
- *Second*, an application may need to allow its user to share social context information with other users. For example, allowing a caller to know the status of the callee before calling. In this regard, the application should also allow its user to specify her *privacy* preferences to retain control over who has access to her situation information under which conditions.

In *interaction-centric* socially-aware applications, interaction-oriented social relationships among collaborative actors dominate the applications' behaviour. The interaction-oriented relationships represent agreements and constraints regarding collaborative interactions among users, which are used in developing interaction-centric socially-aware applications. Such applications can assist users in their daily activities and ultimately enrich their social interactions and well-being (Lukowicz et al. 2012).

A socially-aware *telematics application* (Kabir et al. 2014a), for example, can make travel safer and more convenient by allowing drivers to form a cooperative convoy, collaborate and interact with each other based on their interaction relationships. In a cooperative convoy, a vehicle interacts with other vehicles, service providers and infrastructure systems. Through these interactions a vehicle's driver can share information (acquired from the service providers and infrastructure systems) with other vehicles' drivers in performing their tasks. Some interaction examples include—vehicles should notify each other of their positions every 10s, if a vehicle experiences

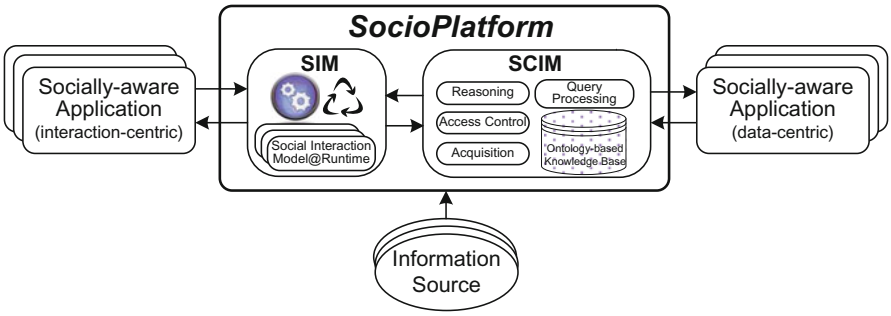

Fig. 19.1 SocioPlatform architecture overview

mechanical problems (*e.g.,* flat tyre, engine issue) it needs to notify the other vehicles as well as the road side assistance, etc.

To develop such an interaction-centric socially-aware application needs to fulfill two major requirements:

- *First*, the application should support interactions complying with the agreed interaction relationships (*i.e.,* constraints and obligations). Thus, a *runtime environment* is required to facilitate interactions.
- *Second*, the application needs to support *runtime adaptation*, as the interaction relationships evolve over time and thus need to adapt with the changes in requirements and environments. For instance, a third vehicle could join when the convoy is on the way; or the break-down of a vehicle might result in its leaving the convoy before reaching the destination.

In the next section we present SocioPlatform that addresses the above mentioned requirements of developing data-centric and interaction-centric socially-aware applications.

19.3 SocioPlatform Overview

The SocioPlatform (see Fig. 19.1) consists of social context information management and social interaction management. Collectively these two parts provide supports to building socially-aware applications with two different focuses: data-centric and interaction-centric.

Social context information management (SCIM) provides a number of functionalities to fulfill the requirements (as discussed above) of developing data-centric socially-aware applications.

Acquiring and Storing Information The advent of social media such as online social networks, blogs, and instant messaging, have radically changed the way people interact with each other and share information about their lives and works. Such

use of social media platforms produces an unprecedented amount of social context information as people specify their relationships, update their status, share interests and contents. Thus, it is now possible to acquire users' social context information from various sources such as Facebook, LinkedIn, Twitter and Google Calendar (Rosi et al. 2011, Lovett et al. 2010). SCIM acquires and integrates social context information from such diverse sources, and stores it in a knowledge base.

Deriving Information SCIM allows application developers to define and obtain derived relationships (at different abstraction levels) based on the basic relationships (*e.g.,* father-daughter and close-friend) and their semantics (*e.g.,* father-daughter being in family) and attributes (*e.g.,* strength and trust). For example, a *best-friend* could be specified as a specially *close* and *trusted* friend, *i.e.,* , $BestFriend \equiv CloseFriend \sqcap (trust > \{0.0\})$. A person's relationship with another person can be derived from their social roles and gender, for example, a '*Father-Daughter*' relationship between two persons can be derived from a person's corresponding social role, *e.g.,* '*Father*' and the other person's gender information, here '*female*'.

Inferring Situations To further enhance the services provided by the data-centric socially-aware applications, it might be required to infer *situations* based on users' social interaction events. These events can be identified from users' interaction activities in various social interaction applications (*e.g.,* Facebook, email and socially-aware telematics application). Therefore, SCIM supports *inferring* situations by observing and analysing current and past interaction events, and utilising ontological knowledge about such events.

Access Control The user's social context information is inherently sensitive. The scenarios of emerging socially-aware applications require users to share their information for greater benefits but this may also compromise their privacy. For example, allowing a caller to know the status of the callee before calling might reduce interruptions, but may also raise serious concerns regarding the privacy and access control over users' situation and other data (Khalil and Connelly 2006). 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, and the user's current situation. Thus, SCIM provides efficient *access* to this social context information while respecting information owners' *privacy*.

Social interaction management (SIM) provides the *runtime environment* and *adaptation management* of social interactions for the interaction-centric socially-aware applications.

Runtime Environment In interaction-centric socially-aware applications, interactions among collaborative users are based on predefined agreements and constraints that characterise the interaction-oriented relationships among users. We model such interaction-relationships among users from domain- and player-perspectives. The interested reader is referred to (Kabir et al. 2011, 2012, 2014a) for a more detailed description of this modelling approach. The *domain-centric social interaction*

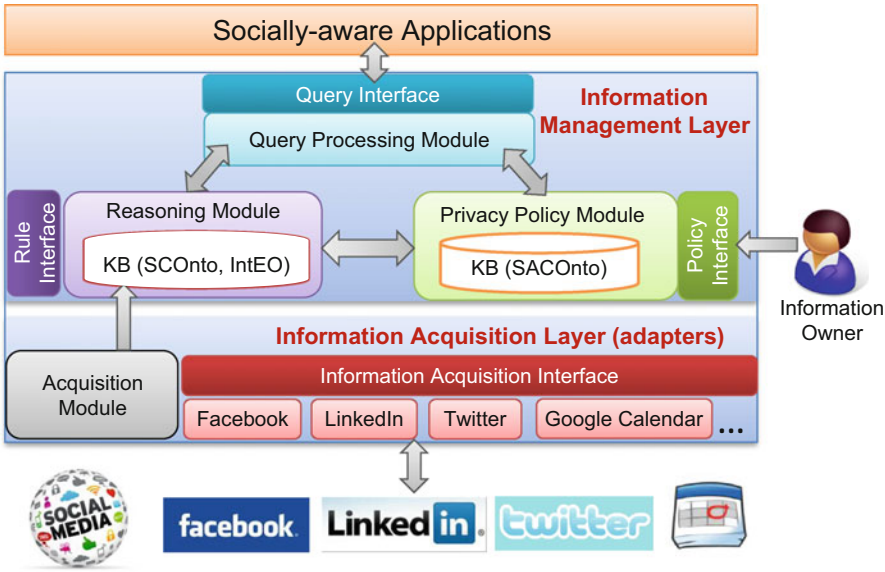

Fig. 19.2 Architecture of the social context information management (SCIM)

model (DSIM) captures a *collaborative* view of the interaction relationships among the users/actors, whereas the *player-centric social interaction model (PSIM)* captures an individual's *coordinated* view of all its interactions (across different domains), and thus supports their coordination. SIM provides the runtime environment for the application to instantiate these domain- and player-centric social interaction models. These instantiated runtime models mediate and coordinate social interactions among collaborative users according to their agreements.

Runtime Adaptation The (runtime) domain- and player-centric social interaction models typically evolve and many aspects of these models such as topology, interaction constraints and non-functional quality properties need to be modified frequently in response to changes in user requirements and environments. Thus, it is necessary to support adaptation in such runtime social interaction models. SIM implements an adaptation protocol (Kabir et al. 2014a) that ensures safe and consistent changes of the runtime models.

19.4 Social Context Information Management Architecture

The SCIM architecture comprises two layers: (i) information acquisition layer and (ii) information management layer, as shown in Fig 19.2.

The *information acquisition* layer is responsible for acquiring social context information from various sources such as Google Calendar, Facebook, LinkedIn, Twitter

and other social media. A common information acquisition interface is provided so that application developers can build different adapters based on that interface to collect data from various sources. The acquisition module is responsible for managing and operating adapters to fetch raw data from the different sources, make the data consistent (*i.e.,* remove irrelevant data and integrate data of interest) and store it into the knowledge base. The acquisition module keeps the list of the available adapters and their implemented APIs. After a certain time interval, which is configurable, the acquisition module executes the fetching and processing steps to update the knowledge base with users' recent social context information. The frequency for such information update, however, is not the same for all types of social context information. For instance, a user's situation information may need to be updated more frequently compared to his/her family relationship. To fulfil this requirement, we allow users to schedule the execution of different acquisition functions. Moreover, it is also possible to trigger the execution of these functions manually (*i.e.,* on demand).

The *information management* layer is responsible to store users' social context information as acquired and to preserve their privacy when this information is accessed. This layer consists of three main modules:

Reasoning Module It classifies users' social data collected by the information acquisition layer, and stores them into the social context ontology (SCOnto) knowledge base. SCOnto (Kabir et al. 2014b) defines general concepts such as social role, social relationship, social interaction and social situation, and extends these concepts to incorporate domain-specific concepts for domain such as Facebook, LinkedIn and Twitter. Interaction event ontology (IntEO) (Kabir 2013a) incorporates SCOnto to capture the properties about users' interaction activities in social media. Reasoning module provides a reasoning functionality to infer abstract social context information that is of interest to applications by exploiting these SCOnto and IntEO knowledge bases. The rule interface allows application developers to develop a mobile- or desktop-based graphical user interface application which can be used by the users (e.g., domain experts) to add, delete, retrieve and update reasoning rules.

Privacy Policy Module It provides a policy interface to allow users to specify and manage their privacy preferences. Users' privacy policies are stored in their socially-aware access control ontology (SACOnto) knowledge base (Kabir et al. 2014b). Like the rule interface, the policy interface allows application developers to develop a graphical user interface application which can be used by the users to add, delete, retrieve and modify their polices. The policy module automatically checks inconsistency in policy specifications and only allows users to add consistent privacy policies. It also automatically enforces the specified privacy policies while users' social context information is accessed.

Query Processing Module It allows different applications to access users' social context information and provides a query interface so that application developers can build applications without the need to deal with the details of information representation schema and management.

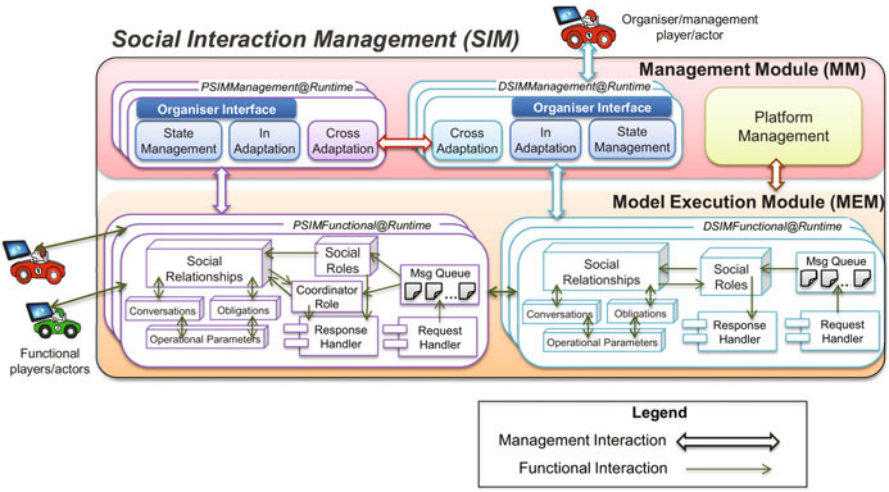

Fig. 19.3 Architecture of the *social interaction management (SIM)*

19.5 Social Interaction Management Architecture

The *Social Interaction Management (SIM)* architecture comprises two main modules: the Model Execution Module (MEM) and the Management Module (MM), as shown in Fig 19.3.

The *Model Execution Module* supports the instantiation of both the domain-centric social interaction models (DSIMs) and player-centric social interaction models (PSIMs). As described in the Sect. 19.3, these interaction models capture interaction-oriented relationships among collaborative actors in an interaction-centric socially-aware application. Therefore, at a given moment, multiple social interaction model instances may exist in parallel.

The MEM maintains a representation of all the functional elements of the domain-centric and player-centric social interaction models, called DSIMFunctional@runtime (in short, *DSIMfun*) and PSIMFuctional@runtime (in short, *PSIMfun*), respectively. The DSIMfun represents Social Roles, Social Relationships, Interactions, Conversations, Obligations and Operational Parameters. In addition, it contains a request handler, a response handler and a message queue. The DSIMfun is able to (1) handle requests received from players (*i.e.,* applications); (2) allocate requests into a message queue; (3) forward messages to corresponding social roles; (4) evaluate conditions (*i.e.,* conversation and obligation) specified in the relationships; (5) send request to relevant social roles and then to players.

The PSIMfun contains all the components of the DSIMfun. In addition, it contains a special type of social role, called the coordinator role. The PSIMfun bounds to one or more social roles in the DSIMfun(s). In the PSIMfun, all the incoming messages are first forwarded to the coordinator role. After evaluating the conditions specified in the relationships, the request message is forwarded to the coordinator player. The

message is processed further based on the decision of the coordinator player, *e.g.,* generating a reply message and sending it to the player from which the message has come. Both the PSIMfun and DSIMfun maintain the *state* of their corresponding entities such as social roles, social relationships and the social interaction model (as a whole).

The *Management Module* supports the runtime adaptation of the instantiated social interaction model. Thus, the management components, called DSIMManagement@Runtime (in short, *DSIMman*) and PSIMManagement@Runtime (in short, *PSIMman*), are instantiated for each of the DSIMfun and PSIMfun. Both of these management components implement basic management operations provided by the organiser interface. We classify these operations as structure, parameter and state related operations. The *InAdaptation* sub-component of the management component implements the structure and parameter related management operations, while the *State Management* sub-component implements state related operations. In addition to these sub-components (*i.e., InAdaptation* and *State Management*), the management component contains the *Cross Adaptation* sub-component which supports the adaptation across social interaction models. The MM also supports the platform-level management, *i.e.,* to create, retrieve, delete, deploy and undeploy social interaction models dynamically. These APIs allows application developers to build a graphical user interface application for a user to perform administration level management.

19.6 Prototype Implementation

We have implemented a SocioPlatform prototype (see Sect. 19.6.1) and demonstrated its applicability and feasibility by developing both data-centric and interaction-centric socially-aware applications (see Sect. 19.6.2). We have also quantified the adaptation overhead and efficacy of the platform by conducting a series of experiments. The experimental results, as reported in (Kabir et al. 2014b), show that the platform is robust and efficient.

19.6.1 SocioPlatform Prototype Implementation

The SocioPlatform prototype is implemented in Java. As part of SCIM, we have written adapters for Facebook, LinkedIn, Twitter, and Google calendar using restfb 1.6.7[2], linkedin-j 1.0.415[3], twitter4j 2.2.4[4], and gdata-calendar 2.0 [5], respectively, to fetch users' social data. We have implemented *SCOnto*, *IntEO* and

[2] http://restfb.com/
[3] http://code.google.com/p/linkedin-j/
[4] http://twitter4j.org/en/index.html
[5] http://code.google.com/p/gdata-java-client/

SACOnto knowledge bases using OWL API 3[6] to store and manage users' social context information and their privacy policies. We adopt a description logic (DL) based query language, namely SPARQL-DL (Sirin and Parsia 2007), and have used the *derivo 1.0*[7] SPARQL-DL query engine with the TrOWL[8] DL reasoner for reasoning about social context information, executing users' privacy policies and processing applications' query.

We have implemented SIM by adopting and extending the ROAD4WS (Kapuruge et al. 2011) which is an extension to the Apache Axis2[9] web service engine for deploying adaptive service compositions. SIM exploits JAXB 2.0[10] for creating DSIMs and PSIMs runtime from their XML descriptors. JAXB helps the generation of classes and interfaces of runtime models automatically using an XML schema. It exposes each *social role* as a *service*, the associated *interactions* of the role as operations of that service. The constraints specified in the *interaction-oriented social relationship* are evaluated as *event-condition-action* rules and implemented using Drools engine[11]. The *runtime adaptations* are supported by the Java reflection mechanism and the Drools engine. To cope with the changes in environments and requirements, at runtime, Javassist[12] allows generation of *new* classes and *modification* of existing classes, which helps to add new social roles/relationships and change existing roles/relationships, respectively. Drools engine allows the SIM to inject new rules and delete existing rules from the working memory which facilitates the addition and deletion of constraints (conversations, obligations and parameters) in the relationships.

19.6.2 Developing Socially-Aware Applications

To demonstrate the real-world applicability and feasibility of our approach, we have developed a data-centric socially-aware applications for Android mobile devices, named socially-aware phone call application (SPCall) (Fig. 19.4a shows a screen-shot), and an interaction-centric socially-aware application for Android mobile devices, named socially-aware telematics application (SocioTelematics) (Fig. 19.4b shows a screen-shot), using our SocioPlatform.

The *SPCall* application aims to reduce phone call interruptions and considers both the caller and callee perspectives. The application allows the caller to know the situations of the intended callee to check whether it is suitable time to call. Accessing the callee's situations is also subject to the callee's privacy policies. In this regard,

[6] http://owlapi.sourceforge.net/
[7] http://www.derivo.de/en/resources/sparql-dl-api/
[8] http://trowl.eu/
[9] http://axis.apache.org/
[10] http://jcp.org/en/jsr/detail?id=22
[11] http://www.jboss.org/drools/
[12] http://www.jboss.org/javassist

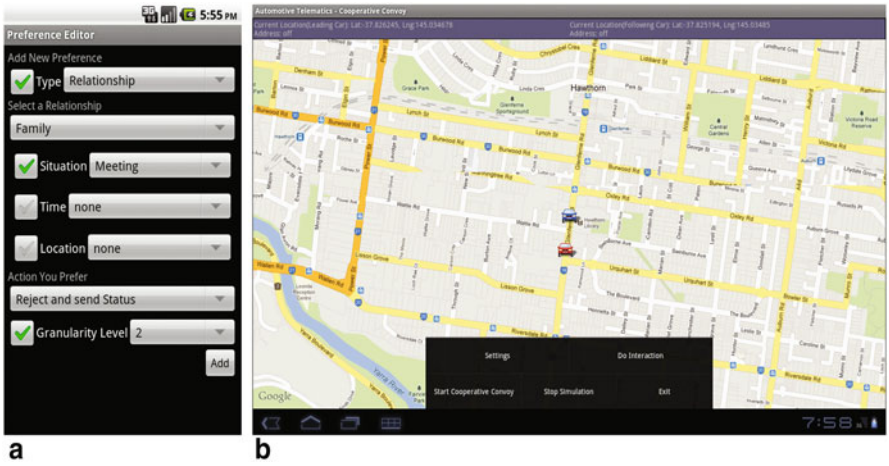

Fig. 19.4 Socially-aware applications **a** SPCall **b** SocioTelimatics

the application allows a person to specify access control polices considering his/her situations at the time of access request and the social relationship with the requester. On the other hand, the application allows a callee to specify her filtering preferences on incoming phone calls such as ring, vibrate, reject, or reject and send situation, considering her current situations and the relationships with the caller. For example, if my situation is *meeting* or *seminar*, and a call comes from *family*, the action is to *reject* and forward my situation at granularity level 2 (Busy). The application exploits social context information provided by the SocioPlatform to support social context-aware behaviour, *i.e.,* filter incoming phone calls and inform situations of the intended callee based on social context information.

SocioPlatform (*SCIM* component) assists the application developer in developing socially-aware phone call applications by collecting and representing the users' social context information from different sources and providing a set of query APIs for the applications to access that information based on the users' privacy preferences. In developing the *SPCall* application we, from the caller perspective, have used the `getSituation(callee)` query API to obtain the situation of an intended callee and then provide that information to the caller. From the callee perspective, to implement the call filtering functionality, we have used the `getAllRelationshipsName(me,inComingCallNum)` and `getSituation(me)` query APIs to obtain the relationships between the caller and callee, and the current situation of the callee. Then, based on the specified filtering preferences in the application, it decides whether to ring, vibrate, reject, or reject and send situation information at a specific granularity. In the case of a "send situation at a specific granularity" decision, the application invokes the `getSituAtGranularity(me,gLevel)` query API to obtain the situation information of the callee at the specified granularity level and then sends it to the

caller. The interested reader is referred to (Kabir et al. 2014b) for a more detailed description.

The *SocioTelematics* application aims to allow two or more vehicle drivers to form a cooperative convoy by supporting their social interactions. Such social interactions are based on predefined agreements and constraints that characterise the *interaction-oriented social relationships* between the players, such as drivers. For example, cars should notify each other of their positions every 10s. In complex and changing environments, such agreements and constraints, and thus interaction relationships are subject to change. Thus, the behaviour of the application needs to be adapted to cope with the changes. The application uses the *runtime environment* and *adaptation management* functionalities of SocioPlatform to facilitate interactions and to cope with the changes in requirements and environments.

The SocioPlatform (*SIM* component) makes it easy to develop this application based on their supposed interaction-oriented social relationships and without worrying about the underlying message communication (*i.e.,* social interactions) and the evaluation of the messages, as these interaction-oriented social relationships are modeled and represented in DSIMs and PSIMs, the runtime support and adaptation of these models are externalized to and managed by the SIM component of the SocioPlatform. Moreover, the runtime adaptation capability provided by the platform allows the application to respond to changes in requirements and environmental factors, without any change in the application code. The interested reader is referred to (Kabir et al. 2012, 2014a) for a more detailed description.

19.7 Related Work and Discussion

19.7.1 Platforms for Managing Context Information

Much research in the area of context-aware software systems has investigated the development of context management infrastructure, so as to reduce the complexity of engineering such systems. It advocates pushing as much as possible the acquisition, management and dissemination of context information from the application into a context management infrastructure. Dey et al. (2001) developed a basic framework to support acquisition and interpretation of context information from sensors. Hong and Landay (2001) advocated using a service infrastructure approach to deploy context-aware applications. In this approach, the tasks of gathering, processing and managing context information are encapsulated as services that are accessible to any context-aware devices and applications. While having such supporting infrastructure is important, we argue that the current infrastructure is highly restrictive in addressing the dynamicity and complexity of social context.

Some efforts, such as Kourtellis et al. (2010) and Xing et al. (2011), have already recognized the need to externalize the social context management functionalities and have taken steps towards systematically managing users' social context information. Prometheus (Kourtellis et al. 2010) collects user's social data from different OSNs

and 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. Like Prometheus, PocketSocial (Xing et al. 2011) also collects social data from different sources. But unlike Prometheus, 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.

Our work significantly differs from the above noted approaches in that it not only collects users' social relationship information (both object- and people-centric) from multiple sources and stores it in richer ontologies, but also considers the owners' status information and their semantics, allowing information representation and derivation at different levels of abstraction and consequently facilitating fine-grained access control and query processing.

19.7.2 Platforms for Managing Interactions and Adaptation

Much research has been carried out into middleware support for runtime adaptation in context-aware systems (*e.g.,* MADAM (Geihs et al. 2009) and 3PC (Handte et al. 2012)) and service-oriented systems (*e.g.,* MUSIC (Rouvoy et al. 2009) and MOSES (Cardellini et al. 2012)). These middleware solutions mainly target the tasks of individual users/applications and have focused on reconfiguring applications' settings (rather than *interaction relationships*) based on physical context information (*e.g.,* place, time)/quality of service requirements (*e.g.,* performance, reliability), rather than interaction relationships. Moreover, their proposed runtime models are application-specific and cannot be used to model interaction relationships among collaborative users.

In contrast to these solutions, our social interaction management component targets interaction-centric socially-aware applications, and focuses on executing adaptation by explicitly modelling and realising interaction relationships using a social interaction model and providing an organiser interface to change such model. On the other hand, we do not address the monitoring of environment changes (*i.e.,* physical context information), acquiring and analysing such physical context information to make adaptation decisions. In that sense, our adaptation management approach is not a substitute for existing middleware solutions that manage physical context information, rather can be built on top of those solutions as appropriate, in order to manage social interactions and runtime adaptation in interaction-centric socially-aware applications.

19.8 Conclusion

In this chapter, we have presented SocioPlatform to provide high-level platform support for developing socially-aware applications. The platform implements a set of adapters to acquire social data from Google Calendar and different online social networks such as Facebook, LinkedIn and Twitter, and stores the consolidated social context information in an ontology-based knowledge base. It provides a number of functionalities including management and querying of social context information, an environment for executing social interaction models and managing their runtime adaptation, and a set of APIs for developers to build socially-aware applications.Overall, the platform hides the complexity of managing social context, and thus provides better support for the development of socially-aware applications.

References

Bauer, M.W., Gaskell, G.: Towards a paradigm for research on social representations. J. Theory Soc. Behav. **29**(2), 163–186 (1999). doi:10.1111/1468-5914.00096. http://dx.doi.org/10.1111/1468-5914.00096

Biamino, G.: Modeling social contexts for pervasive computing environments. In: IEEE International Conference on Pervasive Computing and Communication Workshops, pp. 415–420 IEEE Computer Society, Washington, DC, USA (2011)

Cardellini, V., Casalicchio, E., Grassi, V., Iannucci, S., Lo Presti, F., Mirandola, R.: Moses: A framework for qos driven runtime adaptation of service-oriented systems. IEEE Trans. Softw. Eng. **38**(5), 1138–1159 (2012)

Dey, A.K.: Understanding and using context. Pers. Ubiquit. Comput. **5**(1), 4–7 (2001)

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)

Dourish, P.: What we talk about when we talk about context. Pers. Ubiquit. Comput. **8**(1), 19–30 (2004). doi:10.1007/s00779-003-0253-8. http://dx.doi.org/10.1007/s00779-003-0253-8

Endler, M., Skyrme, A., Schuster, D., Springer, T.: Defining situated social context for pervasive social computing. In: IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011, pp. 519–524 (2011). doi:10.1109/PERCOMW.2011.5766945

Eugster, P.T., Garbinato, B., Holzer, A.: Middleware support for context-aware applications. In: Middleware for Network Eccentric and Mobile Applications, pp. 305–322. Springer Berlin Heidelberg(2009)

Ferscha, A.: 20 years past weiser: What's next? IEEE Pervasive Comput. **11**(1), 52–61 (2012)

Geihs, K., Barone, P., Eliassen, F., Floch, J., Fricke, R., Gjorven, E., Hallsteinsen, S., Horn, G., Khan, M.U., Mamelli, A., Papadopoulos, G.A., Paspallis, N., Reichle, R., Stav, E.: A comprehensive solution for application-level adaptation. Softw. Pract. Exp. **39**(4) (2009)

Gummadi, K.P., Mislove, A., Druschel, P.: Exploiting social networks for internet search. In: Proceedings of 5th Workshop on Hot Topics in Networks, pp. 79–84. Irvine, CA (2006)

Han, L., Jyri, S., Ma, J., Yu, K.: Research on context-aware mobile computing. In: 22nd International Conference on Advanced Information Networking and Applications—Workshops, 2008. AINAW 2008, pp. 24–30 IEEE Computer Society, Washington, DC, USA (2008). doi:10.1109/WAINA.2008.115

Handte, M., Schiele, G., Matjuntke, V., Becker, C., Marrón, P.J.: 3pc: system support for adaptive peer-to-peer pervasive computing. ACM Trans. Auton. Adapt. Syst. **7**(1) (2012)

Henricksen, K.: A framework for context-aware pervasive computing applications. Ph.D. thesis, The School of Information Technology and Electrical Engineering, The University of Queensland, Sept (2003)

Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: models and approach. Pervasive Mob. Comput. **2**(1), 37–64 (2006)

Hong, J.I., Landay, J.A.: An infrastructure approach to context-aware computing. Hum. Comput. Interact. **16**(2), 287–303 (2001)

Kabir, M.A.: A framework for social context-aware pervasive computing applications. Ph.D. thesis, Swinburne University of Technology (2013a)

Kabir, M.A.: Modeling, managing and reasoning about social contexts for socially-aware applications. In: IEEE International Conference on Pervasive Computing and Communication Workshops (2013b)

Kabir, M.A., Han, J., Colman, A.: Modeling and coordinating social interactions in pervasive environments. In: Proceedings of the 16th IEEE International Conference on Engineering of Complex Computer Systems, pp. 243–252 IEEE Computer Society, Washington, DC, USA (2011)

Kabir, M.A., Han, J., Colman, A., Yu, J.: Sociotelematics: Leveraging interaction-relationships in developing telematics systems to support cooperative convoys. In: Proceedings of 9th International Conference on Ubiquitous Intelligence and Computing, pp. 40–47 IEEE Computer Society, Washington, DC, USA (2012)

Kabir, M., Han, J., Colman, A., Yu, J.: Scaas: A platform for managing adaptation in collaborative pervasive applications. In: Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., Leenheer, P., Dou, D. (eds.) 21st International Conference on Cooperative Information Systems (CoopIS 2013). Lecture Notes in Computer Science, vol. 8185, pp. 149–166. Springer, Berlin (2013a)

Kabir, M., Han, J., Yu, J., Colman, A.: User-centric social context information management: an ontology-based approach and platform. Pers. Ubiquit. Comput. 18(5), 1061–1083. (2014)

Kabir, M.A., Han, J., Colman, A.: Sociotelematics: Harnessing social interaction-relationships in developing automotive applications. Pervasive Mob. Comput. 14, 129–146 (2014)

Kapuruge, M., Colman, A., King, J.: ROAD4WS—Extending apache axis2 for adaptive service compositions. In: Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference, pp. 183–192 IEEE Computer Society, Washington, DC, USA (2011)

Khalil, A., Connelly, K.: Context-aware telephony: Privacy preferences and sharing patterns. In: Proceedings of the 20th Conference on Computer Supported Cooperative Work, pp. 469–478 ACM New York, NY, USA (2006)

Kourtellis, N., Finnis, J., Anderson, P., Blackburn, J., Borcea, C., Iamnitchi, A.: Prometheus: user-controlled p2p social data management for socially-aware applications. In: Gupta, I., Mascolo, C. (eds.) Middleware 2010. Lecture Notes in Computer Science, vol. 6452, pp. 212–231. Springer, Berlin (2010)

Li, J., Dabek, F.: F2F: reliable storage in open networks. In: Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS) URL: http://iptps06.cs.ucsb.edu/(2006)

Lovett, T., O'Neill, E., Irwin, J., Pollington, D.: The calendar as a sensor: analysis and improvement using data fusion with social networks and location. In: Proceedings of the 12th ACM International Conference on Ubiquitous Computing, pp. 3–12 ACM New York, NY, USA (2010)

Lu, Y., Tsaparas, P., Ntoulas, A., Polanyi, L.: Exploiting social context for review quality prediction. In: Proceedings of the 19th International Conference on World Wide Web, pp. 691–700. ACM New York, NY, USA (2010)

Lukowicz, P., Pentland, S., Ferscha, A.: From context awareness to socially aware computing. IEEE Pervasive Comput. **11**(1), 32–41 (2012)

Raychoudhury, V., Cao, J., Kumar, M., Zhang, D.: Middleware for pervasive computing: a survey. Pervasive Mob. Comput. **9**(2), 177–200 (2013)

Rosi, A., et al.: Social sensors and pervasive services: Approaches and perspectives. In: Proceedings of the IEEE PerCom Workshops, pp. 525–530 IEEE Computer Society, Washington, DC, USA (2011)

Rouvoy, R., Barone, P., Ding, Y., Eliassen, F., Hallsteinsen, S., Lorenzo, J., Mamelli, A., Scholz, U.: MUSIC: Middleware Support for self-adaptation in ubiquitous and service-oriented environments. Software Engineering for Self-adaptive Systems, pp. 164–182. Springer, Berlin (2009)

Schilit, B.N., Theimer, M.M.: Disseminating active map information to mobile hosts. Netw. Mag. Global Internetwkg. **8**(5), 22–32 (1994). doi:10.1109/65.313011. http://dx.doi.org/10.1109/65.313011

Schmidt, A., Beigl, M., Gellersen, H.W.: There is more to context than location. Comput. Graph **23**(6), 893–901 (1999). doi:10.1016/S0097-8493(99)00120-X.

Schuster, D., Rosi, A., Mamei, M., Springer, T., Endler, M., Zambonelli, F.: Pervasive social context-taxonomy and survey. ACM Trans. Intell. Syst. Technol. (TIST)4(3), Article No. 46 (2012)

Sirin, E., Parsia, B.: Sparql-dl: Sparql query for owl-dl. OWL: Experiences and Directions Workshop (OWLED)Vol 258, CEUR-WS, RWTH Aachen, Germany (2007)

Wang, G., Gallagher, A., Luo, J., Forsyth, D.: Seeing people in social context: Recognizing people and social relationships. In: Proceedings of the 11th European Conference on Computer vision: Part V, ECCV'10, pp. 169–182. Springer, Berlin (2010).

Xing, B., Gronowski, K., Radia, N., Svensson, M., Ton, A.: Pocketsocial: Your distributed social context now in your pocket. In: IEEE International Conference on Pervasive Computing and Communication Workshops, pp. 322–324 IEEE Computer Society, Washington, DC, USA (2011)

Zheng, Y., Li, L., Ogata, H., Yano, Y.: Support online social interaction with context-awareness. Int. J. Contin. Eng. Educ. Life Long Learn. **17**(2), 160–177 (2007)