Social Context: Supporting Interaction Awareness in Ubiquitous Environments

Minh H. Tran, Jun Han, *Member, IEEE Computer Society*, Alan Colman

***Abstract*—In ubiquitous computing environments, certain entities (or actors) often need to interact with each other in achieving a joint goal in a dynamically changing context. To perform such interactions in a seamless manner, the actors need to be aware of not only their physical context (e.g. location) but also their changing relationships with respect to the particular task or goal. The latter interaction-oriented context, which we refer to as *social context*, has significant impacts on the way actors manage their adaptive behavior. However, very little research has focused on supporting such social context in ubiquitous environments. This paper presents our novel approach to modeling and realizing social context. Social context is modeled as a managed composition of loosely-coupled roles with their interaction relationships expressed as contracts. In addition, it is modeled from an individual actor's perspective to allow for possible differences in the actors' perception of the relationships. The social contexts of an actor are externalized from the actor itself to achieve easy management of the actors' adaptive behavior concerning interaction. A layered system architecture is introduced to realize the approach and demonstrate the development of automotive telematics systems that are physically and socially context-aware.**

***Index Terms*—Context-awareness, adaptation, social context, pervasive computing, software architecture, SOA**

I. Introduction

"T*he most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it*" [27] Weiser's vision of ubiquitous computing 18 years ago still stands as a challenge for today's technology. Using heterogeneous devices and infrastructure, applications need to seamlessly support user access to information and user collaboration in a ubiquitous manner. One of the challenges lies in the development of context-aware applications that are able to adapt to changes in the operating environment and user requirements.

In these early days of ubiquitous computing, context-aware applications are stand-alone applications that react to contextual cues picked up from the environment. Such context cues or *context facts* form the *physical context* relevant to the application behavior, including location, identity, time, temperature, battery level, and so on. The recent proliferation of networking and mobile technologies opens the way for context-aware *actors* to interact with each other to achieve a joint goal that is not attainable by a single actor alone. Those actors such as human users, services and autonomous agents, may be independent, heterogeneous, distributed and mobile. Thus, supporting their interactions in a seamless manner to achieve the joint goal while still maintaining their own objectives poses two main challenges.

M. H. Tran (corresponding author), J. Han and A. Colman are with Faculty of Information and Communication Technologies, Swinburne University of Technologies, PO Box 218, Hawthorn, Vic 3122, Australia (phone: +61-3-92148394; fax: +61-3-98190823; email: {mtran,jhan,acolman}@swin.edu.au).

First, in addition to physical context, context-aware actors need to take into account *social context* that mediates the changing interactions between actors. Like physical context, interaction-oriented social context is dynamic and plays an important role in forming the situation in which actors operate. But unlike physical context, social context is not simply acquired directly from physical and virtual sensors. It requires an understanding of constructed relationships, obligations and constraints underlying the interactions between collaborating actors. Many aspects of social context, such as its topology, interaction constraints and non-functional requirements, may need to change in response to the changes of cooperative goals, relationships between actors and acquired physical context facts. Thus, it is also necessary to model the inter-dependence between social context and physical context.

The heterogeneity of ubiquitous environments means that context-aware actors could operate in different contexts with their own perspectives and will not necessarily perceive a given context in a same way. Each actor could have its own partial and perspective view of the social context that reflects the actor's objectives and its relevant role in the interaction. Most existing research on context awareness is based on an implicit assumption of a common and unified representation of context facts (e.g., ontology). Only few researchers such as Benerecetti et al. [3] view context as a subjective concept and model it relatively from the actor's perspective based on its goal. Furthermore, research from related disciplines, such as CSCW [16], has shown that the rationale for actors to participate in a group could be different with respect to their objectives. However, current research on context awareness does not address this level of complexity in context modeling.

Second, the architecture of context-aware systems needs to provide adequate support for the development and deployment of explicit models of social context. Conventional architectures of context-aware actors can be seen as having three layers (see Fig. 1). At Layer 1, physical and virtual sensors are controlled to capture various types of data. This raw data is then propagated to a middle layer (Layer 2) which transforms sensed data into meaningful context facts using abstract physical context models, interpreters and aggregators. At the top level (Layer 3), context-aware applications acquire context facts from Layer 2 using context query languages (e.g., SQL-, RDF- or XML-based languages [8]). The applications adapt their behavior in response to changes of physical context. This layering paradigm allows separating system components into levels of abstractions to reduce the complexity of system development. These layers are implemented and managed separately, and the implementation details of a lower layer are hidden from the upper layer.

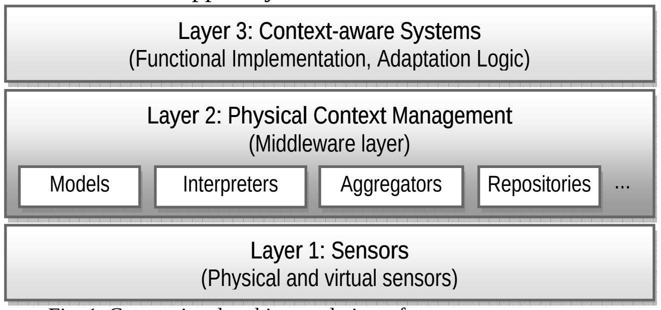

Layer 3: Context-aware Systems
(Functional Implementation, Adaptation Logic)
Layer 2: Physical Context Management
(Middleware layer)
Models
Interpreters
Aggregators
Repositories
...
Layer 1: Sensors
(Physical and virtual sensors)

Fig. 1. Conventional architectural view of context-aware systems

Such a conventional approach to context aware architectures has shortcoming in modeling social context. Although physical context is explicitly modeled at Layer 2, constructed relationships between actors, their interaction constraints and adaptation logic are not modeled explicitly at Layer 3. The relationships and interaction constraints are often directly hardwired in the application implementation making the applications hard to maintain and change.

In order to achieve scalability and greater interaction-oriented adaptability in such environments, the implementation of complex context inference and adaptation logic should be externalized from context-aware actors and managed separately. In this paper, we present an innovative approach to address the above challenges by providing: a systematic and explicit model of social context, and a layered system architecture realizing the model and enabling the development and deployment of physically and socially context-aware systems.

The paper is structured as follows. Section 2 presents motivating scenarios from the context-aware automotive telematics domain. Section 3 analyzes the requirements for modeling and realizing social context. Sections 4 and 5 present our approach to, respectively, modeling social context and defining a layered architecture in a way that addresses those requirements. Section 6 presents a prototype implementation of a context-aware automotive telematics system using our approach. Section 7 reviews related research. Section 8 concludes the paper and highlights future work.

II. Motivating Scenario

This section motivates the research by showing a number of situations where context-aware automotive telematics is used to support interaction between a vehicle and other heterogeneous actors. Such interaction is commonly referred to as *V2X* (see Fig. 2), where *X* represents users' portable devices, other vehicles, road-side infrastructure or information services.

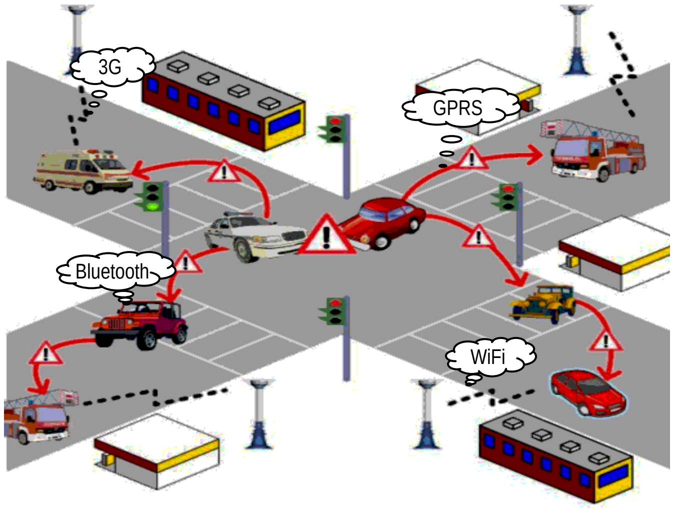


Fig. 2. V2X interactions.

Let us consider that a group of people rent *two* cars and drive from Melbourne to Sydney, Australia. The two cars are equipped with context-aware telematics systems (or telematics for short) that allow V2X interactions as follows:

Vehicle-to-portable device: The passengers exchange music and video between their portable devices (e.g., mobiles, mp3 players, laptop, etc.) and in-vehicle infotainment systems.

Vehicle-to-vehicle: The telematics systems allow cars to sense and talk to each other. Neighboring cars exchange information (e.g., travel distance, speed, etc.) to improve road safety and avoid congestion. In particular, the two cars as mentioned above could form a cooperative convoy of which one car plays the role of leading car and the other following car. The leading car chooses a travel route and the following car follows it. During the trip, the cars frequently keep each other updated of their travel distance and notify the other of any mechanical breakdown.

Vehicle-to-infrastructure: The telematics interact with road-side infrastructure to exchange information about traffic and road condition (e.g., speed limit, accident occurrence, road work delay, and road surface condition). This information is used to improve safety and enhance driver experience.

Vehicle-to-service: While on the road, the drivers use a number of services provided by the car rental company and other service providers (e.g., road side assistance, emergency service, fleet monitoring, travel guide, and fuel watch).

The abovementioned interactions illustrate the dynamic relationships between heterogeneous actors, including users' portable devices, cars, roadside units, service providers, and so on. The relationships could be formed in either an ad-hoc manner (e.g., between a user's portable device and an in-car infotainment system through Bluetooth, and between neighboring cars through Dedicated Short Range Communication (DSRC)), or in a dynamically changed and controlled manner (e.g., between two cars in a cooperative convoy). Those actors could dynamically join or leave a group. Their functional behavior and non-functional attributes could also be dynamically adjusted.

III. Requirements Analysis

We identify two sets of requirements that are associated with the two research challenges highlighted in Section 1.

Requirements for social context modeling: *Social context needs to explicitly capture constructed relationships and interaction constraints between actors. This set of constructed relationships and constraints needs to be managed, and modeled subjectively from an actor's perspective.*

Social context is the set of norms, rules, obligations and understandings that influence an individual's action with respect to a group in a particular situation. Individuals need an understanding of the given social context in order to interact effectively with others in that context. Social contexts can be designed or can be emergent. A social context with goals can be viewed as an organization. Organizational theory addresses the structuring of organizations in order to more effectively achieve those goals (e.g., [18]). As such, organizations are typically viewed as a managed network of roles that decompose the abstract functions of the organization into descriptions that can be performed by role-players (e.g., people, subsystems, other organizations).

We adopt this view of organization as a composition structure of dynamic *relationships* between roles in order to model social context in software systems. Organizational roles are defined in terms of their relationships with other organizational roles and resources. These roles are loosely-coupled elements whose interaction relationships need to be *managed* and coordinated to meet changing requirements and changing environments. In addition, these roles are first class entities and as such they are separate from the actors who play those roles. In mobile and ubiquitous systems this separation between well-defined roles and actors is beneficial given the dynamic nature of the system's composition and the common situation where actors are unavailable.

We further extend this software composition as an organization to support the *subjectivity* of social context modeling. In addition to sharing any common view of group interaction, an actor may have their own perception of the group with respective to their roles. Social context models need to accommodate this. Thus, we model social context from an actor's perspective. This subjective view also determines the non-functional constraints that the actor agrees to provide and demands from the others. Furthermore, a social context is owned by the actor from whom the perspective is modeled (i.e., the actor has an ultimate control of its social contexts). While there may exist no centralized global view of interaction, the models of interaction held by different individuals need to be compatible for cooperative goals to be achieved. In the absence of any unified view of interaction, social context models therefore need to be able to be shared and negotiated between parties to the interaction.

Requirements for system architecture: *The architecture of context-aware systems needs to externalize the management of social context from the implementation of actors. The architecture also needs to support the adaptability of social context, and needs to be easily deployable.*

As actors in ubiquitous environments become open, distributed and autonomous, one key approach to supporting adaptation is to provide a management capability. This management subsystem is responsible for managing the relationships between actors as well as coordinating their adaptive behavior. However, in the conventional architecture view such management capability is often directly hard-wired into the actors. Although using techniques such as configuration and rules supports decoupling the implementation to a certain extent, the tasks of developing context-aware systems are still burdensome. This is because the conventional architecture view fails to separate the implementation of functional actors from the management of their relationships and adaptive behavior. To overcome this problem, the architecture needs to manage social context in a similar manner to it handles physical context. As shown in Layer 2 of Fig. 1, the physical context management layer provides abstractions allowing the applications to acquire context facts without understanding the details of how those facts are collected. Similarly, the architecture needs to *externalize* the management of social context from the actors themselves.

In addition, the architecture needs to support the *adaptability* of social context. Context-aware actors could adapt their own behavior through some forms of reflection. For example, the actors maintain the abstraction of their social contexts. Through explicit models of social context, actors could dynamically change interaction constraints that they have formed with other actors, and the behavior that they adhere to. Furthermore, the architecture needs to allow social context models to be easily deployable and configurable (i.e., *deployability*) to meet changes in user requirements and environment conditions.

IV. Social Context Modeling

Addressing the requirements for social context modeling, we follow three basic principles: first-order representation of *interaction relationships* between loosely coupled actors, *separation* of functional and management operations, and *subjectivity* of the context. First, social context is a structured composition of roles whose interaction relationships are expressed through contracts, and actors are modeled separately from roles they play but are bound to the roles. Second, interactions between roles are coordinated and controlled through a separate management subsystem. Third, social context represents an actor's subjective view of the interaction relationships.

A. Conceptual Model of Social Context

Following the meta-model for creating adaptive software organizations described in [6], we model social context as a self-managed composite comprising four key elements: *functional role, contract, player* and *organizer role* as seen in

Fig. 3. A context-aware actor could own many social contexts. Each social context is composed of functional roles, contracts and a single organizer role. A contract expresses interdependencies between two functional roles. A context-aware actor (as a player) performs functions defined by a functional or organizer role.

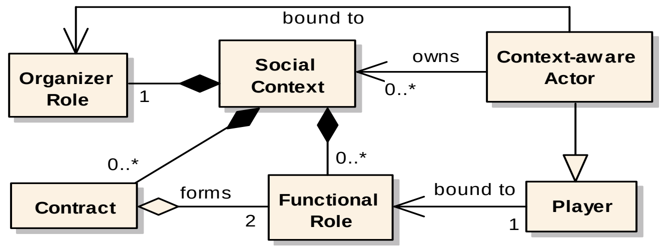


Fig. 3. A meta-model of social context.

Using these concepts, Fig. 4 presents a model of the convoy social context viewed from the leading car's perspective. The model encapsulates interactions that the leading car has with other actors in the cooperative convoy. The social context comprises five functional roles (*LeadingCar*, *FollowingCar*, *NeighboringCar*, *Infrastructure* and *RentalCompany*), four contracts (*C_1, ...C_4*) and an organizer role.

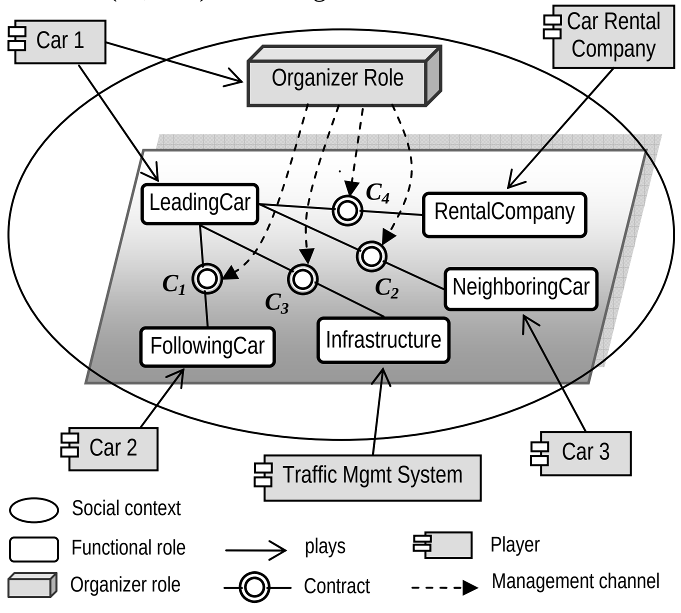


Fig. 4. A model of cooperative convoy social context.

Functional Role:

Functional roles represent expected functional interactions of participating actors with respect to the social context. For example, the *LeadingCar* and *FollowingCar* roles represent the two cars engaged in the cooperative convoy; and the *NeighboringCar* role represents other surrounding cars that are located within a close proximity to the leading car. These functional roles are, in essence, place holders or abstract functional definitions for corresponding players. A role's definition is an aggregation of permitted and obligated operations specified in contracts that associate the role with other functional roles. Functional roles are *internal* to the social context composite in contrast to *external* players (introduced later in subsection 3). The roles' definitions are exposed allowing players to be bound to them at runtime.

Contract:

Interdependencies between functional roles are expressed through contracts. The contracts are central to a social context as they perform a number of key functions, including forming and managing the topology, mediating interactions, triggering the acquisition of context facts and monitoring performance with respect to interaction obligations. Each contract specifies the interaction relationships between two roles, including terms related to atomic interactions carried out by the roles, their obligations, temporal constraints of interactions, and so on. As illustrated in Table I, the general form of a contract includes the following:

TABLE I: AN EXAMPLE OF THE CONTRACT BETWEEN TWO CARS.

```
Contract ID  C1: ConvoyLeaderFollower;
Parties      A: LeadingCar;
             B: FollowingCar;
Physical Context Fact Providers
  A_onBoardUnit: anyURI;
  B_onBoardUnit: anyURI;
  Timer: anyURI;
  RoadAdvisory: anyURI;
Social Context Fact Providers
  convoy: C1.StateTracker; //built-in provider
Interaction Clauses
  i1: {convoyAvailable, AtoB };
  i2: {joinRequest, BtoA, joinResponse };
  i3: {notifyPosition};
  i4: {leaveConvoy};
  i5: {routeUpdate, AtoB, routeAccept};
  i6: {notifyMechanicalIssue};
Conversation Clauses (Temporal Constraints):
  c1: {i1 precedes i2 globally};
  c2: {i2 leadsto i5 globally};
Obligations
  o1: {i2, Timer, duration, < , 30, seconds};
  o2: {i3, Timer, periodic, =, 60, seconds};
Context Rules:
  r1: {Set Maximum Desired Distance:
       when  i3.OnNotificationReceived;
       if    ?convoy.state = ACTIVE &&
             A_onBoardUnit.Raining = TRUE;
       do    A.setMaxDesiredDistance(100)};
  r2: {Notify Mechanical Issue:
       when  A_onBoardUnit.OnNotificationReceived;
       if    ?convoy.state = ACTIVE &&
             A_onBoardUnit.TroubleCode = TRUE;
       do    A.i6};
```

- Contract's *identifier* and abbreviated qualifier (e.g., "C1: ConvoyLeaderFollower").
- Entities that include contracted *parties* (i.e., party A or B), any *physical context fact providers* (e.g. the internal A_onBoardUnit or external RoadAdvisory identified by unique URIs) and *social context fact providers* (e.g., the internal StateTracker).

- *Interaction clauses* that are permitted atomic message exchanges between the contracted parties. A clause's definition includes an identifying message *signature*, a *direction* of the message (i.e., AtoB, BtoA or either) and the message exchange *pattern* (i.e., one-way or request-response). If the direction is not specified, either party could send the message. If a response signature is not specified then the message is one-way. For example, i1 specifies a one-way message convoyAvailable that is sent from A to B. i2 specifies a request-response interaction that involves B sends a joinRequest message to A, and A sends a joinResponse message to B.
- *Conversation clauses* that are acceptable sequences of interactions defined by temporal constraints in the Interaction Rule Specification (IRS) language [14]. For example, c1 specifies that a message indicating a convoy is available must be received before (i.e., *precedes*) a request to join the convoy.
- *Obligations* that can be attached to interactions. In particular, real-time temporal obligations are related to a requirement to send a message in response to some other message, as is the case with *request-response* interaction clauses or *leadsto* conversation clauses. Obligations use performance providers (e.g. timer) to help evaluate if an obligation has been met. For example, o1 states that the leading car must respond to a joinRequest message with a joinResponse within 30 seconds. o2 states that the cars must send each other their positions periodically every 60 seconds.
- *Context rules* define Event-Condition-Action (ECA) rules [17] that evaluate a social situation. The *events* that can trigger this evaluation can be the receipt of a message, a timed event or the firing of another rule in a chain of rules. A social situation that is evaluated by the rule can be a combination of social and physical context facts. The context facts that make up a social situation *condition* are provided by context providers. As social context is a model of relationships expressed in the contracts, social context facts are some abstraction of the state of activities across those contracts. An example of internal social context provider could be a task state machine that identifies task instances from a sequence of interactions and provides abstract representation of the state of those tasks (e.g. convoy.state = SUSPENDED). Physical context providers sense external conditions (in our example weather conditions or accidents) and provide operations for querying that information. These providers may provide information of any state of interest in the environment (e.g. traffic conditions) or any entity (e.g. the motor vehicle). The *action* involved in such rules can result in modifying/setting of operational parameters in the parties of the contract (e.g. setMaxDesiredDistance(x)) or firing of other rules such as those in general clauses that alter the state of the contract. In the example, r1 states that when the social state of the convoy is active and it is raining then the maximum distance between the vehicles should be set to 100 meters. r2 states that when a notification is received from a physical context fact provider A_onBoardUnit, if the convoy is active and A_onBoardUnit notifies any diagnostic trouble code, then the leading car needs to send a notification message i6: {notifyMechanicalIssue} to the following car.

Player:

We define the concept of players to separate the abstract concept of roles in social context from the actual actors that play the roles. Players are external to the social context. Examples of players include human users, autonomous agents, composite services, devices or databases. Context awareness can be added to players by binding them to a social context that models at runtime the relationships between players, and senses the states of relevant interactions and physical context through its contracts. To be able to respond to changes in physical and social contexts, a player needs to be sensitive to those changes by polling or through notification.

Organizer Role:

Another key principle of our modeling approach is the self-managed capability of social context. A context-aware actor manages its own social context. This is achieved through an organizer role. The organizer role is internal to a social context and responsible for managing the social context's topology, regulating the social context by creating and changing contracts, and binding players to functional roles. The organizer role has the capability to change the topology by creating and removing functional roles and contracts, and by adding and revoking conditions specified in the contracts. At runtime, the organizer role could accept, reject and terminate bindings between functional roles and players. The principle of separation between a role and player is also applied to the organizer role, though the role is played by a context-aware actor who owns the social context. The role exposes a management interface that contains operational and contractual methods for manipulating the composition of the social context, monitoring the performance of contracted roles, and negotiating binding with external players. The reader is referred to [15] for an in-depth discussion of a management interface that is required for a self-managed composite.

B. Subjectivity of Social Context

One of the key principles and novelty of our modeling approach is the *subjective* view of social context. This allows two collaborating actors to have different perceptions of the collaboration that are reflected on their roles and constructed relationships with other actors in the group.

Take a scenario of two cars in the cooperative convoy as an example, both cars need to interact with each other and with other actors including *Infrastructure* (IF), *NeighboringCar* (NC) and *RentalCompany* (RC). But specifically for the leading car, in addition to these roles it needs to interact with a travel guide service (i.e., the *TravelGuide* role highlighted in Fig. 5). This requirement is to fulfill one of its obligations, i.e., obtaining a travel route for the convoy.

As shown in Fig. 5, SC1 and SC2 are social context representing relationships between the two cars modeled from Car 1 and Car 2's perspectives (omitting the organizer roles and external players bound to the IF, NC and RC roles for

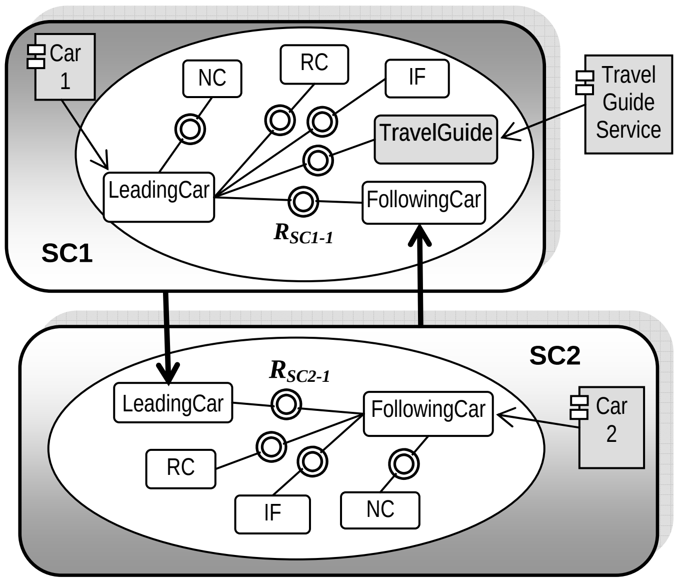


Fig. 5. Two cars interact through their social contexts.

simplicity). Interactions between the cars are mediated by these social contexts in the sense that both cars include the representation of the other in their own social context. In SC1, the LeadingCar role is played by Car 1 whilst the FollowingCar role is played by a *contextualized* Car 2 (i.e., a composition of Car 2 and SC2). Similarly, in SC2 the FollowingCar role is played by Car 2, and the LeadingCar role is played by a contextualized Car 1 (i.e., a composition of Car 1 and SC1). In order to establish the binding between the two contextualized actors, an agreement needs to be reached. For example, conditions specified in the contract R_{SC1-1} of SC1 are consistent with those defined in R_{SC2-1} of SC2.

V. ARCHITECTURE SUPPORTING SOCIAL CONTEXT

In the previous section, we have presented a new modeling approach to represent social context that captures both functional and non-functional aspects of interactions between actors. In this section, we present a system architecture that we have developed for realizing such explicit models of social context. The principles of our architecture are the *externalization* of the social context management, *adaptability* of social context, and ease of deployment (i.e., *deployability*).

A. Externalization of Social Context Management

We extend the conventional architecture view of context-aware systems shown in Fig. 1 to externalize the social contexts and their management. As a result, the architecture that we have developed consists of four main layers (Fig. 6).

In comparison to Fig. 1, *Layer 3* (highlighted in Fig. 6) is an additional layer that is responsible for the management of social context and adaptation. This layer is able to support social contexts of heterogeneous actors. Multiple social context models are deployed and controlled independently as runtime self-managed composites. A context-aware actor could have one or more social contexts. Interaction between the actor and its social context models is loosely coupled and based on a

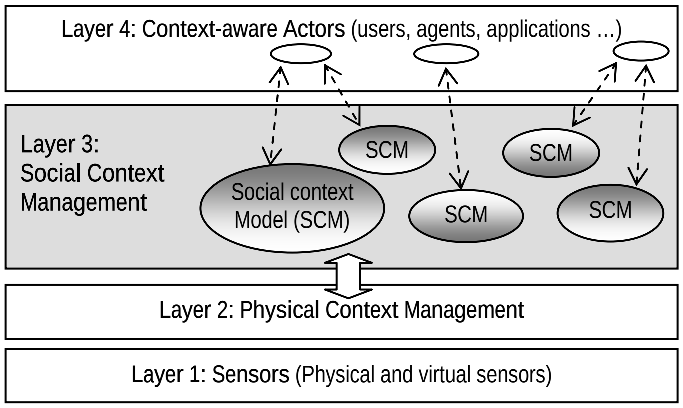

Layer 4: Context-aware Actors (users, agents, applications …)
Layer 3: Social Context Management
SCM
SCM
Social context Model (SCM)
SCM
SCM
Layer 2: Physical Context Management
Layer 1: Sensors (Physical and virtual sensors)

Fig. 6. Architecture for social context and adaptation.

document-based messaging style and *event-driven* interaction. Well-defined messages (e.g., SOAP messages) are exchanged between the actor and the social context composites. Typically, a message contains a message header and a body. The header provides information about the characteristics of the massage (e.g., the target destination, policy, message signature, etc.) that can be used by both the actor and the composite. The body contains data that is specific to operations of the actor. A runtime social context composite acts as a message router that (1) receives messages from one player, (2) evaluates conditions specified in associated contracts, and (3) passes the messages to another player. These processes are triggered by events such as message received, notification of context fact change, notification of performance level, timer, and so on.

Fig. 7 illustrates the processes that a message is sent from Player A to Player B via a social context consisting of Role A, Role B and Contract C. First, Player A sends a message M to Role A. Upon arrival of the message, Role A might need to check the security setting to ensure that only authorized players could send requests. As Role A could form more than one contract with other roles of the social context, it needs to match the message against the appropriate contract using the rules contained with the runtime social context composite. When the message M is received by Contract C, this event triggers a number of actions such as assessing interaction constraints, acquiring context facts, and evaluating performance. Contract C then forwards[1] M to Role B which then passes the message to Player B.

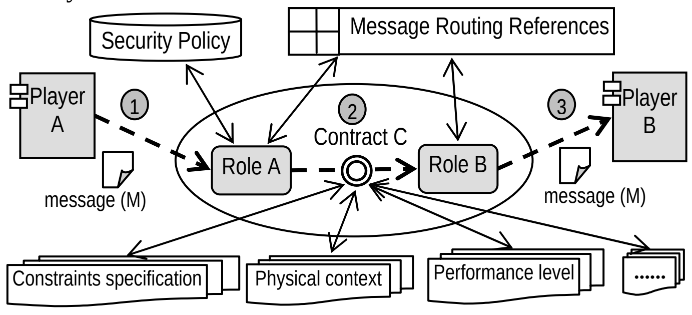


Fig. 7. Message-based interaction via social context.

[1] Note that additional information could be added into the header of the initial message M.

B. *Social Context-mediated Adaptation*

As the physical context and interaction relationships between actors dynamically change, the relevant social contexts need to adapt in response to those changes. The adaptation of social context is performed by the organizer of the social context. Social context supports adaptation in the following ways:

- Changes of topology: Addition and removal of functional roles and contracts is carried out. For instance, in the convoy social context, a third vehicle could join the context while the convoy is already on the way; or a break-down vehicle might leave the convoy before reaching the destination; these situations lead to changes relationships between the roles and contracts.
- Changes of contracts: Interaction constraints, obligations and conditions specified in a contract may change. For instance, a desired distance between two cars reduces, say from 300m to 100m when the road is wet. Contracted roles are informed of such changes in the contract.
- Changes of binding between roles and players: The same functional role can be played by different players at different times. The binding between the role and the players is dynamic. For instance, the Infrastructure role could be played by different traffic management systems in the convoy at different times as the vehicle moves from one jurisdiction to another.

C. *Deployability of Social Context*

The objective of externalizing the management of social context and adaptation logic from the actor is to achieve an ease of deployment (referred to as *deployablilty*). Not only can running social contexts be easily configured, but new social contexts can also be deployed to enhance behavior of context-aware actors. Moreover, this architecture also allows services to be built without prior knowledge of the social context to which they will be bound.

Prior to this research, we have developed the Role-Oriented Adaptive Design (ROAD) framework [5, 6] which is a modeling and implementation framework for adaptive software architectures. ROAD is a well-established framework that provides relevant constructs that can be extended for modeling and deploying social context. In addition, ROAD provides capability for monitoring and managing social context at runtime (e.g., altering the topology and interaction constraints). We adopt ROAD as the underlying framework for the design and deployment of social context for context-aware actors. Fig. 8 shows the process of designing and deploying social context using the ROAD framework and supporting toolkits.

ROADdesigner (Fig. 9) is a graphical modeling tool supporting the design of social context models. ROADdesigner is implemented as a plug-in for Eclipse. The software developer uses ROADdesigner to build a social context model by dragging and dropping model components (i.e., roles and contracts) onto a canvas representing the social context composite. Properties and constraints are added to the

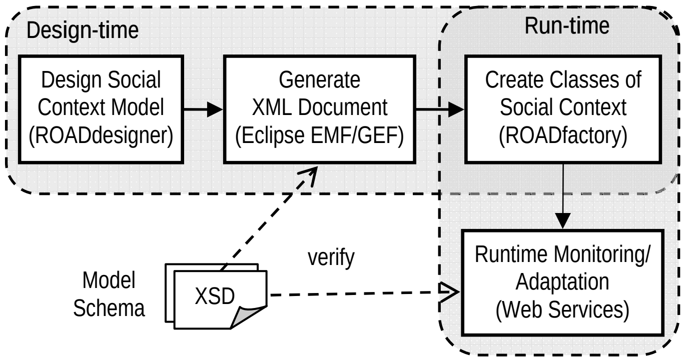
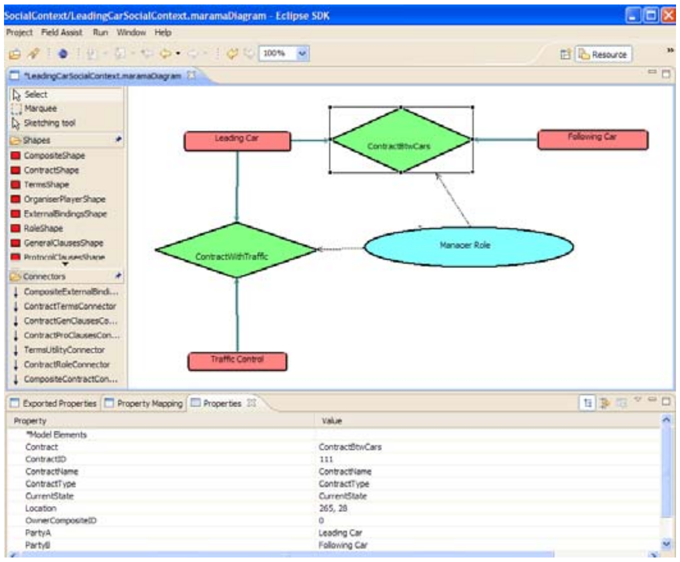

Design-time
Run-time
Design Social Context Model (ROADdesigner)
Generate XML Document (Eclipse EMF/GEF)
Create Classes of Social Context (ROADfactory)
Runtime Monitoring/ Adaptation (Web Services)
verify
Model Schema
XSD

Fig. 8. Design and deployment processes of social context.

contracts. Endpoint references to candidate actors can also be specified at design time. The well-formedness of the model is enforced by a pre-defined schema. If the validation is successful, the social context model could be translated into an XML document, using the Eclipse modeling frameworks (i.e., EMF/GEF). ROADfactory can then create runtime ROAD composites (e.g., classes for roles and contracts, etc.), using JAXB (Java Architecture for XML Binding). ROADfactory integrates the Drools[2] engine to define and fire ECA rules.



Property	Value
*Model Elements	
Contract	ContractBtwCars
ContractID	111
ContractName	ContractName
ContractType	ContractType
CurrentState	CurrentState
Location	265, 28
OwnerCompositeID	0
PartyA	Leading Car
PartyB	Following Car

Fig. 9. ROADdesigner supports social context modeling.

Each social context is implemented as a runtime self-managed ROAD composite that is able to (1) handle requests received from actors; (2) check security settings for authorized access; (3) allocate requests into a message queue; (4) forward messages to roles; (5) evaluate conditions specified in the contracts and (6) send responses to relevant actors, as illustrated in Fig. 10. Currently, social context composites are deployed into a Web Service environment. Social context composites are deployed as Web services to a Web container (e.g., Apache Axis Web Container). A social context composite is able to bind actors (Web services) which meet specific functional and non-functional requirements, to the functional roles of the social context. Interaction between social context

[2] http://www.jboss.org/drools/

composites and those actors is supported by exchanging SOAP messages. Any changes (mostly non-functional requirements) could be made at run-time, resulting in an updated composite. Importantly, changes to the topology of social context need to conform to the social context schema.

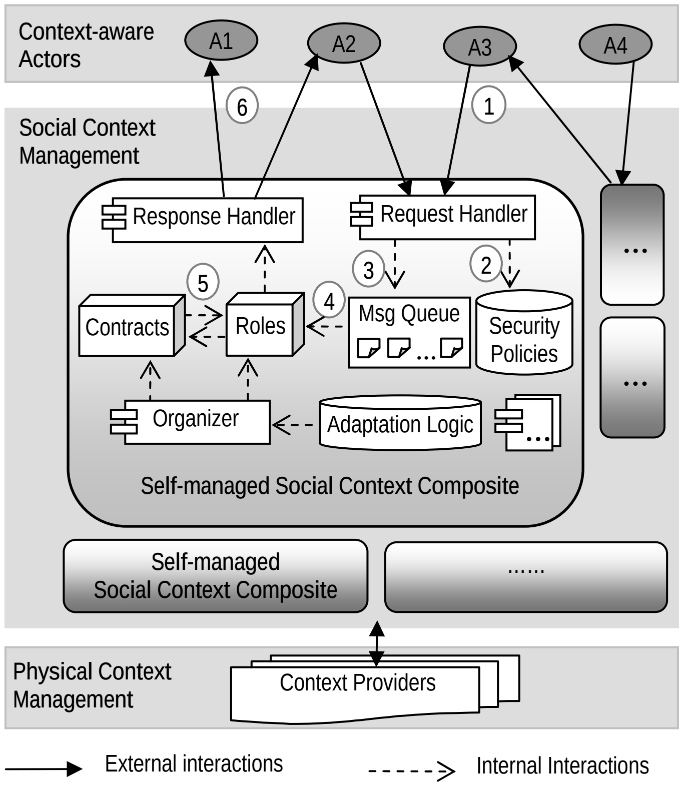
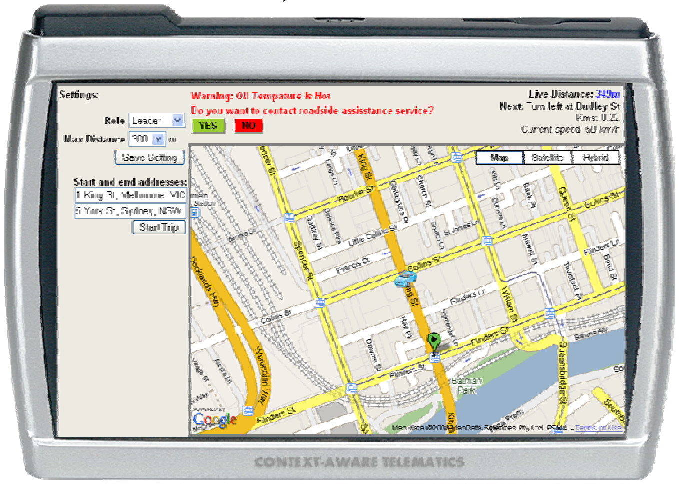


Fig. 10. Architecture of self-managed social context composites.

VI. Prototype Implementation

We have developed a Context-aware Automotive Telematics (CAT) prototype to demonstrate how the proposed social context model and architecture could be applied in building context-aware applications, as seen in Fig. 11. CAT supports coordination between two cars of a cooperative convoy, and interaction between the cars and service providers. CAT is implemented as a Web-based application that uses the Google Maps APIs to display a car's geographic position. CAT acquires physical context facts from external Web services (WS). We implemented four WSs to simulate (1) information about distance between two cars, (2) information about road condition and accident, (3) information about onboard diagnostic, and (4) information about the nearest available garage that is provided by a road-side assistance system.

CAT was developed based on the concept of social context that we have presented in this paper. Social context models for each the two cars were constructed. The XML documents representing those models were used to initiate and populate runtime roles and contracts classes. At run-time, CAT acquires context facts from the WSs. Based on conditions defined in the contract, appropriate information is overlaid on the CAT screen. For example, if a distance is greater than the maximum desired distance, say 300m, a warning is shown, recommending the driver to slow down. In the situation of hazardous condition (e.g., a wet road), CAT recommends the drivers to reduce the maximum distance, to say, 100m. In addition, CAT allows non-functional attributes can be adjusted at runtime (e.g., the driver of the leading car can change the desired maximum distance between to cars, and so on).


Settings:
Role Leader
Max Distance 300 m
Save Setting
Start and end addresses:
1 King St, Melbourne VIC
5 York St, Sydney, NSW
Start Trip

Warning: Oil Tempature is Hot
Do you want to contact roadside assistance service?
YES NO

Live Distance: 349m
Next: Turn left at Dudley St
Kms: 0.22
Current speed 50 km/h

Fig. 11. Prototype of context-aware automotive telematics.

VII. Related Work and Comparative Analysis

Within the scope of this paper, our review of related research particularly focuses on studies that are related to the two research challenges: (1) social context modeling, and (2) system architectures for context-aware systems. In this section, we also present a comparative analysis of the relevant approaches to highlight the contributions of our research.

A. Social Context Modeling

To reduce the complexity of engineering context-aware applications, one major research focus is on context modeling. There have been a number of context modeling techniques available in the literature, such as a graphical approach [12], a mark-up scheme approach [9], an object-oriented model [22], logic-based models [20], and ontology-based models [4, 26]. However, such approaches were developed to model physical context (e.g., representing context facts and relationships between context facts). They provide limited support for modeling social context (i.e., interaction-oriented context). For example, key-value models are simple and easy to use in distributed systems but highly limited in supporting sophisticated structure. Mark-up scheme models are useful in expressing a hierarchical structure of attributes, but limited in defining interaction constraints and dependencies. An ontology-based approach is useful in representing the global knowledge about the relationships between context facts (e.g., using the *is-a* association). However, to the best of our knowledge, there is no study on the use of ontology in supporting the subjectivity of context models. The reader is referred to Strang and Linnhoff-Popien [24] for more detail descriptions of those context modeling approaches.

Strang and Linnhoff-Popien define a set of requirements (e.g., distributed composition, partial validation, richness and quality of information, etc.) that are particularly important for *physical context* modeling. Complement to these requirements, we have defined a set of requirements that are particularly important to *social context* modeling, including: interaction *relationships* (rel), the *separation* of functional and management concerns (sep), and *subjectivity* (sub), as discussed earlier. Table II analyses the extent to which relevant context modeling approaches can be used to model social context, with respect to these three requirements.

TABLE II: COMPARISON OF CONTEXT MODELING APPROACHES WITH RESPECT TO REQUIREMENTS FOR SOCIAL CONTEXT MODELING
(***key:*** ++ comprehensive, + partial, – none)

Approaches - Requirements	**rel**	**sep**	**sub**
Key-value model	–	–	–
Graphical model	+	+	–
Object-oriented model	+	+	++
Mark-up model	+	+	–
Ontology-based model	++	+	+
Logic-based model	+	+	+
Interaction-oriented model (social context)	++	++	++

B. *Architecture for Context-Aware Systems*

Another major research focus in context-aware computing is on architectures for developing context-aware applications. The aim of the architectures is to push as much as possible the acquisition, management and dissemination of context information into a context infrastructure. Dey et al. [7] developed a basic framework to support acquisition and interpretation of context information from sensors by using toolkits. Hong and Landay [13] 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. Becker et al. developed the peer-to-peer pervasive computing (3PC) framework which consists of both a middleware (BASE) [2] and a component model (PCOM) [1]. According to 3PC, a system is viewed as a tree of components. Adaptation is seen as switching of sub-trees using search and selection based on the description of component interfaces and requirements. While having such supporting infrastructure is important, it provides limited support for managing the dynamicity and complexity of social context. The management of interactions and adaptive behavior is still built-in the applications themselves. Moreover, relationships between entities, interaction constraints and adaption are not modeled explicitly. Their implementation is hard-wired directly into the applications. Henricksen and Indulska [10] presented a software architecture that separates an adaption layer from an application layer. But, the adaptation layer merely deals with reasoning context facts and preferences. No adaption logic relating to changes of constructed relationships and interaction constraints is provided. ICrafter [19] is a service framework that supports user access to heterogeneous services in interactive workspaces (i.e., physically co-located technology-rich ubiquitous computing environments). ICrafter supports the deployment of services to such workspaces and dynamically generates appropriate user interfaces to those services for user devices. Gaia [21] is another service framework for context-aware systems that coordinates services and devices in smart spaces. Although Gaia is useful in dynamically aggregating services and applications in the context of smart spaces, such active spaces are typically too constrained (i.e., Gaia provides limited support for scalability and mobility)

Supporting social context in pervasive systems has gained attentions from the research community. Zimmermann et al. [28] proposed a framework for context-aware systems that supports personalization. The framework includes a semantic layer that captures the current situation of a user’s interactions, including social dependencies. The semantic layer contains a sub-layer for entity relationships. However, this framework presents very limited discussion on how this sub-layer is modeled and supported. Wang et al. [25] examined the role of social group as a source of context information in pervasive environment. The authors developed a context-aware group membership scheme, as an example of social context, to support group members’ perceptions of how devices can be used to support group interaction. Their system was reported to be feasible, but the modeling approach and architecture is application-specific and lacks design implications for other aspects of social context. Schmidt and Terrenghi [23] studied the relationships between ubiquitous systems and physical artifacts. The findings were reported useful in designing several domestic display appliances. However, the study is limited in addressing the design requirements catering for constructed relationships between context-aware systems.

TABLE III: COMPARISON OF ARCHITECTURES FOR CONTEXT-AWARE SYSTEMS (***key***: ++ comprehensive, + partial, – none)

Architectures - Requirements	**ext**	**adp**	**dep**
Dey et al.’s Context Toolkits [7]	+	–	–
Becker et al.’s 3PC/BASE [1, 2]	+	+	+
Ponnekanti et al.’s Icrafter [19]	+	+	++
Henricksen et al.’s PACE [11]	++	+	+
Roman et al.’s Gaia [21]	+	+	+
Our layered architecture	++	++	++

As discussed earlier, conventional system architectures are useful in reducing the complexity of developing context-aware systems. They support a strict separation between the acquisition of context data from sensors and the use of the context for adaptation by context-aware actors. However, the conventional architectures are limited in separating the management of interaction-based social context and adaptation from the implementation of the actors themselves. Table III compares the fulfillment of relevant architectures with respect to three requirements: *externalization* of management from the actors (ext), *adaptability* of a social context model (adp), and *deployability* (dep).

VIII. Conclusions and Future Work

This paper has presented an innovative approach to developing context-aware systems, particularly in supporting their adaptation in response to changes in the *relationships* between the relevant actors (e.g., human users, applications, and agents). We have introduced the concept of social context to model the interactions between actors. Social context captures relationships and interaction obligations that a context-aware actor has with other actors with respective to their joint goals, from the individual actor's perspective. A social context is modeled as an organized *composition* of interrelated *functional roles* whose interdependencies are expressed through *contracts*. These contracts define functional operations and non-functional requirements that obligate the contracted roles. Each social context is managed by its own *organizer role*. Both functional roles and the organizer role are played by external players (i.e., the corresponding actors or other entities) at run-time via functional and management interfaces. As such, a context-aware actor owns and manages its social contexts and uses them as proxies to interact with others. We have also introduced a layered architecture that explicitly externalizes the management and adaptation of social context from the actors. We have demonstrated the realization and application of our approach in implementing context-aware automotive telematics systems.

As future work, we will focus on other aspects of social context management, including sharing and reusing of social contexts, privacy, and management of multiple social contexts of an actor. We will also enhance the functionalities of software toolkits for modeling and deploying social context, and conduct quantitative evaluations of the approach.

Acknowledgment

This research was supported by the Australian Cooperative Research Centre for Advanced Automotive Technology (AutoCRC).

References

[1] C. Becker, M. Handte, G. Schiele, and K. Rothermel, "PCOM - a component system for pervasive computing," in *Proc. of 2nd Conference on Pervasive Computing and Communication*, 2004, pp. 67-76.

[2] C. Becker, G. Schiele, H. Gubbels, and K. Rothermel, "BASE: A Micro-Broker-Based Middleware for Pervasive Computing," in *Proc. of 1st International Conference on Pervasive Computing and Communications*, 2003, pp. 443- 451.

[3] M. Benerecetti, P. Bouquet, and M. Bonifacio, "Distributed Context-Aware Systems," *Human-Computer Interaction*, vol. 16(2-4), pp. 213-228, 2001.

[4] H. Chen, T. Finin, and A. Joshi, "An Ontology for Context-aware Pervasive Computing Environments," *The Knowledge Engineering Review*, vol. 18(3), pp. 197–207, 2003.

[5] A. Colman and J. Han, "Roles, Players and Adaptive Organisations," *Applied Ontology: An Interdisciplinary Journal of Ontological Analysis and Conceptual Modeling*, vol. 2, pp. 105-206, 2007.

[6] A. Colman and J. Han, "Using Role-based Coordination to Achieve Software Adaptability," *Science of Computer Programming*, vol. 64(2), pp. 223-245, 2007.

[7] A. K. Dey, G. D. Abowd, and D. Salber, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," *Human-Computer Interaction*, vol. 16(2-4), pp. 97-166, 2001.

[8] P. D. Haghighi, A. Zaslavsky, and S. Krishnaswamy, "An Evaluation of Query Languages for Context-Aware Computing," in *Proc. of 17th International Conference on Database and Expert Systems Applications*, Krakow, Poland, 2007, pp. 455-462.

[9] A. Held, S. Buchholz, and A. Schill, "Modeling of Context Information for Pervasive Computing Applications," in *Proc. of 2nd Conference on Pervasive Computing and Communications Workshops*, Orlando, Florida, 2004, pp. 43- 47.

[10] K. Henricksen and J. Indulska, "A Software Engineering Framework for Context-Aware Pervasive Computing," in *Proc. of 2nd Conference on Pervasive Computing and Communications*, 2004.

[11] K. Henricksen, J. Indulska, T. McFadden, and S. Balasubramaniam, "Middleware for Distributed Context-Aware Systems," *On The Move To Meaningful Internet Systems 2005*, vol. 3760, pp. 846-863, 2005.

[12] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Modeling Context Information in Pervasive Computing Systems," in *Proc. of 1st International Conference on Pervasive Computing*, 2002, pp. 79-117.

[13] J. I. Hong and J. A. Landay, "An Infrastructure Approach to Context-aware Computing," *Human-Computer Interaction*, vol. 16(2-4), pp. 287-303, 2001.

[14] Y. Jin and J. Han, "Consistency and Interoperability Checking for Component Interaction Rules," in *Proc. of 12th Asia-Pacific Software Engineering Conference*, Taipei, Taiwan, 2005, pp. 595-602.

[15] J. King and A. Colman, "A Multi Faceted Management Interface for Web Services," in *Proc. of the Australian Software Engineering Conference*, Gold Coast, Australia, 2009, pp 191-199.

[16] J. Lipnack and J. Stamps, *Virtual Teams: People Working Across Boundaries with Technology*. New York, US: John Wiley & Sons, 2000.

[17] D. McCarthy and U. Dayal, "The Architecture of An Active Database Management System " in *Proc. of ACM SIGMOD International Conference on Management of Data*, Portland, Oregon, 1989, pp. 215-224.

[18] H. Mintzberg, *Structure in fives: designing effective organizations*. Englewood-Cliffs, New Jersey: Prentice Hall, 1983.

[19] S. R. Ponnekanti, B. Lee, A. Fox, P. Hanrahan, and T. Winograd, "ICrafter: A Service Framework for Ubiquitous Computing Environments," in *Proc. of International Conference on Ubiquitous Computing*, Atlanta Georgia, 2001, pp. 56-75.

[20] A. Ranganathan and R. H. Campbell, "An Infrastructure for Context-awareness based on First Order Logic," *Personal and Ubiquitous Computing*, vol. 7(6), pp. 353–364, 2003.

[21] M. Román, C. Hess, R. Cerqueira, K. Nahrsted, and R. H. Campbell, "A Middleware Infrastructure for Active Spaces," *Pervasive Computing*, vol. 1(4), pp. 74-83, 2001.

[22] A. Schmidt, M. Beigl, and H.-W. Gellersen, "There is more to context than location," *Computers and Graphics*, vol. 23(6), pp. 893-901, 1999.

[23] A. Schmidt and L. Terrenghi, "Methods and Guidelines for the Design and Development of Domestic Ubiquitous Computing Applications," in *Proc .of 5th IEEE International Conference on Pervasive Computing and Communications*, 2007, pp. 97-107.

[24] T. Strang and C. Linnhoff-Popien, "A Context Modeling Survey," in *Workshop on Advanced Context Modelling, Reasoning and Management, as part of the Sixth International Conference on Ubiquitous Computing*, Nottingham, England, 2004.

[25] B. Wang, J. Bodily, and S. K. S. Gupta, "Supporting persistent social groups in ubiquitous computing environments using context-aware ephemeral group service," in *Proc. of 2nd Conference on Pervasive Computing and Communications*, 2004, pp. 287-296.

[26] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, "Ontology Based Context Modeling and Reasoning using OWL," in *Proc. of 2nd Conference on Pervasive Computing and Communications Workshops*, Orlando, Florida, 2004.

[27] M. Weiser, "Computer for the 21st century," *Scientific American*, vol. 256(3), pp. 94-104, 1991.

[28] A. Zimmermann, M. Specht, and A. Lorenz, "Personalization and Context Management," *User Modeling and User-Adapted Interaction*, vol. 15(3-4), pp. 275-302, 2005.