Privacy-Aware Autonomous Agents for Pervasive Healthcare

Monica Tentori and Jesus Favela, *CICESE*

Marcela D. Rodríguez, *Autonomous University of Baja California*

Pervasive technology in hospital work raises important privacy concerns. Autonomous agents can help developers design privacy-aware systems that handle the threats raised by pervasive technology.

Hospitals are convenient settings for deploying pervasive computing technology, but they also raise important privacy concerns. Hospital work imposes significant demands on staff, including high availability, careful attention to patients, confidentiality, rapid response to emergencies, and constant coordination with colleagues. These demands shape the way hospital workers experience and understand privacy. In addition, healthcare professionals experience a high level of mobility because they must collaborate with colleagues and access information and artifacts distributed throughout the premises. Hospital workers, therefore, require ubiquitous access to real-time patient data to make critical-care decisions. Some researchers envision a hospital as a pervasive computing environment that integrates hospital information systems into a site in which embedded processors, computers, sensors, and digital communications are inexpensive, ubiquitous commodities.[1]

Through context awareness, such an environment can tailor itself to user preferences and perform tasks according to the physical space's nature. Moreover, the more knowledge an application has of each user and his or her context, the better it can adapt itself to assist that user. Paradoxically, the more an application knows the user, the greater the threat to that user's privacy.[2] Nevertheless, the design of pervasive applications for hospitals has generally overlooked privacy issues. The problem is that developers have little support for designing software and creating interactions that can truly help users manage their privacy.

The privacy a user demands and expects is context dependent,[3] based on the environment's requirements, persisting as a fundamental feature of the substrate into which pervasive computing is threaded. By using the capabilities of autonomous agents, designers can create privacy-aware applications in a scalable, context-reactive, and flexible manner. Scalability here means supporting the incorporation of new agents with their own privacy policies into the system. Context reactivity means allowing the adaptation of a pervasive application on the basis of dynamic context changes. Flexibility means allowing the specification and management of different types of contextual information depending on the environment's requirements.

Autonomous agents' capabilities can thus provide healthcare environments with the means to adapt pervasive applications' behavior to the user's particular privacy conditions and demands, resulting in an agent-based privacy-aware system. To exemplify how autonomous agents can support the development of such a system, we extended the Simple Agent Library for Smart Ambients (SALSA) agent framework.[4] We incorporated customizable privacy-aware mechanisms into SALSA that adapt the application according to the users' context to

Related Work in Privacy-Aware Systems

The pervasive computing literature addresses privacy issues—as does, to a lesser degree, agent-based systems design. However, despite the topic's obvious importance, relatively little systems-oriented research addresses privacy protection in multiagent architectures.

Marc Langheinrich's privacy awareness system (pawS) lets data collectors announce and implement data use policies. It also lets data users control how much of their personal information is stored, used, and removed from the system.[1] Unfortunately, although privacy mechanisms triggered upon information capture have been useful in client-server applications, they aren't suitable for agent-based systems. In the latter, the agent representing the user is the first to detect that user's captured information. Therefore, the threat of an invasion of privacy doesn't depend on the capture of such information but rather on the communication of this information to the environment's other agents. Hence, agent-based pervasive applications require triggering the privacy-aware mechanisms during communication of the information.

On the other hand, Ginger Myles, Adrian Friday, and Nigel Davies developed a system that gives users fine-grained control over the release of their location information.[2] This system protects the information shared once an application has requested it. The problem is that, unlike pawS, this system stores and manages information that the user hasn't agreed to share. The privacy-aware mechanisms we incorporated into SALSA filter the user's personal information before sharing it with the broker and enforce privacy conditions when some entity requests such information. Although enforcing privacy for information requests and communication lets data owners control their information, this approach creates an imbalance between the information that data owners are willing to share and the data collectors' requirements for service delivery. Researchers call this imbalance the *principle of minimum asymmetry*.[3] Privacy-aware mechanisms could handle this problem by increasing the information the data collectors share with the data owners to provide feedback and symmetry.

PRICONIA,[4] an agent-based framework based on a consumer-centered access control mechanism, allows negotiation of membership on the basis of consumer demands and user privacy conditions. By following a priority delivery mechanism, PRICONIA uses a portal agent to inform users of the information an application is handling. Although PRICONIA provides user feedback and reasonable control over the information flow, both the priority delivery and the consumer-centered access control mechanisms are isolated from the context in which the system captures, accesses, and shares the user's personal information. The privacy-aware agents we propose deal with the principle of minimum asymmetry by letting agents negotiate their quality of privacy (QoP) level, considering the user's context in enforcing privacy.

The Confab toolkit uses an entity's contextual information such as location, activity, or name, along with privacy tags to create a context tuple that an infospace represents.[5] The user stores these infospaces, and infoservers manage them. Thus, applications can retrieve and manipulate such infospaces. Using an access control mechanism to access the context tuple stored in the retrieved infospace, infoservers enforce privacy conditions. However, in this approach, users can't control their context tuple. Therefore, if a user's context tuple doesn't meet a service's demands, the infoserver denies the user access, with no way to change his or her context tuple to meet the requirements the service demands or to somehow negotiate access. Because privacy management is a dynamic response to a circumstance rather than a static enforcement of rules,[6] flexibility is required. Our approach lets agents change their QoP level as necessary, establish contracts with other agents, and make special exceptions if required.

References

1. M. Langheinrich, "A Privacy Awareness System for Ubiquitous Computing Environments," *Proc. 4th Int'l Conf. Ubiquitous Computing* (Ubicomp 02), Springer, 2002, pp. 237–245.
2. G. Myles, A. Friday, and N. Davies, "Preserving Privacy in Environments with Location-Based Applications," *IEEE Pervasive Computing*, vol. 2, no. 1, 2003, pp. 56–64.
3. X. Jiang, J.I. Hong, and J.A. Landay, "Approximate Information Flows: Socially-Based Modeling of Privacy in Ubiquitous Computing," *Proc. 4th Int'l Conf. Ubiquitous Computing* (Ubicomp 02), Springer, 2002, pp. 176–193.
4. S. Miyagawa, S. Yamasaki, and E. Uchiyama, "An Agent-Based Information Aggregation Framework with Privacy and Priority Control for Daily Life Support," *Proc. 1st Int'l Workshop Privacy and Security in Agent-Based Collaborative Environments* (PSACE 06), Springer, 2006, pp. 91–105.
5. J.I. Hong and J.A. Landay, "An Architecture for Privacy-Sensitive Ubiquitous Computing," *Proc. 2nd Int'l Conf. Mobile Systems, Applications, and Services*, ACM Press, 2004, pp. 177–189.
6. L. Palen and P. Dourish, "Unpacking 'Privacy' for a Networked World," *Proc. SIGCHI Conf. Human Factors in Computing Systems*, ACM Press, 2003, pp. 129–136.

satisfy their privacy needs. We illustrate the privacy-aware facilities incorporated into SALSA by describing the implementation of an agent-based pervasive hospital application. This application provides relevant information to hospital workers on the basis of contextual information such as their locations and roles, and it lets them communicate through contextual messages. This application raised privacy concerns among potential users, which we addressed using privacy-aware agents. (The "Related Work in Privacy-Aware Systems" sidebar discusses other attempts at managing privacy in pervasive computing environments.)

Using autonomous agents for privacy management

For three months, we conducted a workplace study in a public hospital. The study included systematic observation, long interviews, and a scenario-driven evaluation. We conducted this study to assess the privacy issues hospital workers face in their everyday work, how they deal with them, and how these issues influence their decision making. We also wanted to understand how hospital workers' perception of privacy changes when they envision their hospital using pervasive technologies. Our results indicate that hospital workers relax or enforce their privacy demands depending on the situation and on the value of the application's services. From a systems design perspective, pervasive applications must meet different users' privacy perspectives and be able to enforce or relax users' privacy demands depending on the situation.

Similar to the way a person manages privacy, autonomous agents make decisions, perceive different types of contextual information, and react accordingly while acting autonomously and proactively. These capabilities let autonomous agents adapt pervasive applications' behavior on the basis of each user's particular privacy conditions and demands.

The results of this study helped us identify privacy requirements based on user privacy needs. To complement these requirements, we analyzed several design guidelines and design implications reported in the literature to gain an understanding of developer needs.[3,5–7] We identified the following requirements.

A common language for users and agents

Users and systems perceive privacy differently. Users need an upfront qualitative clarification of what the technology offers and what information is necessary to access such benefits. On the other hand, the technology enforces privacy by using a set of policies that define how the pervasive application must adapt its behavior to respect the user's privacy demands. So, privacy exists at two levels: the user's qualitative perception and the technology-managed quantitative parameters. Addressing both views is challenging because they're expressed differently, depending on the application logic. Thus, a flexible common language is necessary to define and manage, at both the user and system levels, the privacy that the user demands.

We introduced the term of *quality of privacy*, by analogy from the concept of quality of service. QoS is the probability that a computer network meets a given traffic contract. QoP is the probability that a pervasive environment meets a given privacy contract. Thus, a user can demand a certain QoP level from the pervasive environment using a qualitative measure—for example, logging into the system as an anonymous user. On the other hand, the system can map the perception of anonymity in the form of policies representing the privacy contract. In this case, the pervasive application adapts its behavior to the user's context, satisfying the QoP level agreed to by the application and the user.

A way to communicate privacy events

Agents might need to communicate privacy events to users and other agents. In contrast to a static enforcement of rules, these events let agents dynamically respond to a change in context.[7] For example, whenever an agent changes its QoP, a notification goes to all agents connected in the environment. When a particular agent's context changes, the pervasive environment must adapt its behavior. Thus, a privacy-aware protocol is necessary.

Awareness and control of the information flow context

An agent-based pervasive system must enforce privacy when an agent communicates or requests information,[6,8] providing feedback and reasonable control for sharing, accessing, and communicating the users' personal information.[3] Agents must be able to enforce privacy before the system communicates their information to the environment and before it grants access. However, an inflexible approach under this scheme could lead to an information flow asymmetry, which would prevent agents from accessing essential information needed for their correct execution.[5] Thus, agents must follow a *negotiation-based access control* so that they're aware of the information that services in the environment request.

Negotiation of privacy enforcement under special circumstances

Agents might need to make special exceptions, depending on the situation. For example, although removing medical records from the hospital isn't permissible, sometimes physicians let interns remove specific documents for didactic practices. Hence, privacy violations might be permissible if both parties agree. Agents must be flexible to allow these negotiations. Similar to the way persons establish contracts in real life, agents must have a way to negotiate a contract through a set of clauses.

Privacy-aware autonomous agents

Considering the advantages that autonomous agents offer, we created a software infrastructure for privacy-aware autonomous agents to develop pervasive applications. Several agent-based middleware platforms, such as JADE (Java Agent Development Framework), SALSA, and the GAIA operating system (a middleware infrastructure to enable active spaces), enable the creation and implementation of multiagent pervasive systems. These frameworks have at least partially addressed security issues, but not privacy. We decided to design and implement privacy-aware autonomous agents by incorporating mechanisms into SALSA middleware that adapt the contextual information satisfying a desired QoP level.[4] We selected SALSA because its communication language is flexible enough to let programmers represent and manage different types of contextual information. However, other agent platforms could incorporate such capabilities into their designs for privacy-aware pervasive applications.

The SALSA agent framework contains a library of abstract classes that enable developers to implement the agent's components for perceiving, reasoning, and acting and to control the agent's life cycle (which implements the agent's behavior).[4] When a developer creates an agent, SALSA's registering component registers it in an agent directory. Later, SALSA's acting component activates the agent so that it starts perceiving information from its environment in different ways. During the reasoning state, the agent evaluates the perceived information, which could require applying a simple or complex reasoning algorithm to interpret or transform this information into derived contextual information. Depending on the reasoning component's results, the agent executes an action. This action might involve communicating with other agents, moving its own code to another platform, continuing to perceive information, or terminating its execution (being deactivated and killed). An agent in the suspended state is alive but not performing any activity. For example, if an agent is waiting for information it requested from another agent, it changes from the communicating state to the suspended state. Then, when the other agent contacts this agent, its state returns to communicating.

SALSA also provides a communication platform so that users can detect the presence of other users and agents that offer services relevant to their activities. SALSA lets agents convey information and negotiate services with other agents or request that they execute an action. A communication channel, or agent

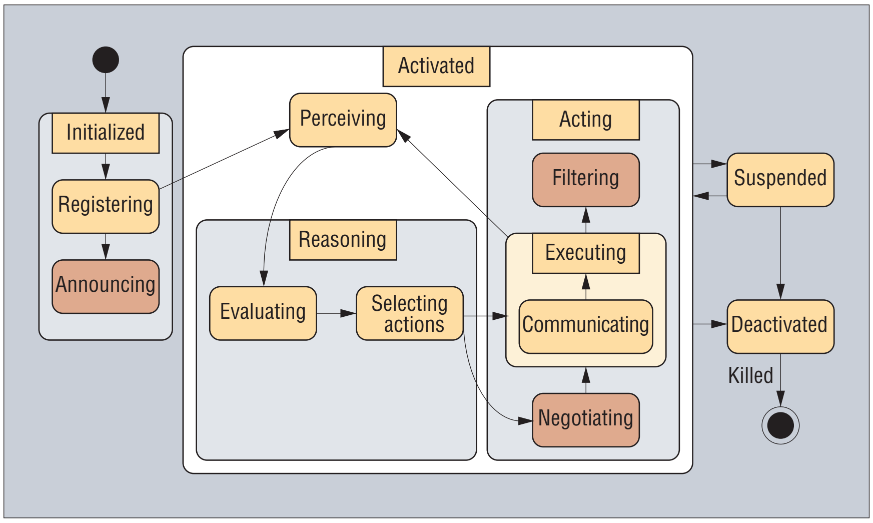


Figure 1. Life cycle of a privacy-aware SALSA agent.

broker, handles communication among ubiquitous devices, services, and users (all of which are represented by agents). A protocol provides an expressive language for exchanging different types of objects between agents (such as perceived context information), between agents and users (such as events generated by user actions); and between agents and services (such as a service's state). When an agent perceives information, a SALSA event occurs, letting the agent activate the reasoning component. The SALSA communication language is flexible enough so that programmers can specify each message's content or extend the type of messages.

We extended the SALSA agent's life cycle, architecture, and communication platform to incorporate privacy mechanisms into the agents.

Life cycle of privacy-aware SALSA agents

Figure 1 shows a SALSA agent's life cycle, which includes three new states. These states help privacy-aware autonomous agents enforce user privacy.

The *announcing* state is a substate of the initialized state that communicates the user-demanded QoP level to the agent broker. This QoP level is in the form of policies mapped into a level of the IPA (information provided by an agent) and the IRS (information requested by a service). The IPA represents the privacy conditions that a user demands from the pervasive environment specifying the information a user is willing to share. For example, if a user wants to join the system anonymously without sharing his location, his IPA indicates that his identity must be shared as anonymous and his location must be shared as unknown. On the other hand, the IRS represents the privacy conditions that the agent demands from other agents that want to use its services. These conditions are access requirements for each of the agent's services. For example, an agent representing a printer must know a user's identity to let that user print a document. The IRS for the printing service indicates that the other agents must share their identity with the printer agent—that is, the service allows no printing by an anonymous agent. The agent broker enforces this policy mechanism.

Activation of the *negotiating* state could occur when the broker rejects an agent request for accessing an agent's service or information. In this case, the broker informs the agent that the IPA level doesn't match the minimum IRS level required to access the service the agent provides. Hence, the agent could decide to negotiate a contract, stipulating clauses (such as expiration time) that, if accepted by both parties, would permit the delivery of the service. For example, a hospital might restrict access to a medical record to be compliant with HIPAA (the US Health Insurance Portability and Accountability Act). But under special circumstances, accessing a medical record from outside the hospital premises might be permissible. Hence, an agent representing a physician could negotiate a contract with the agent that provides access to the medical record so that the physician could temporarily access that record from outside the hospital premises. This negotiation would be agent to agent, depending on the application logic and the agents' nature to select an adequate negotiation protocol and clauses. Thus, this component is an interface that the programmer should overwrite.

Finally, the *filtering* state accesses an agent's policies to control and adapt the contextual information that the agent communicates. In this state, the agent filters the information it perceives, as well as the information announced to the broker. For example, if a user decides to join the pervasive environment with a certain QoP level, indicating he's joining the system anonymously, this state filters the user's identity, which the agent representing the user knows, treating the user as anonymous.

The three states just discussed trigger these mechanisms in privacy-aware agents before and while communicating information. However, we want to enforce privacy for information requests as well. So, we extended the agent broker with an enforcing mechanism.

Handling privacy through the agent broker

An agent interacts with other agents and users through its proxy to SALSA's agent broker, which handles XML messages. The broker stores the status of users and agents, and notifies other agents subscribed to them of changes. Developers can define new states according to the device or service the agent represents.

The developer then defines an IPA and an IRS for each service provided by each agent in the environment, which the broker stores. When a request message arrives, the broker evaluates the information encoded in that message, and triggers and executes an enforcing mechanism. This mechanism is responsible for matching the IPA announced by the service-requesting agent with the IRS demanded by the service-providing agent. For example, if an agent representing an anonymous user tries to access a medical record, the service-providing agent's IRS doesn't match the IPA announced by the anonymous user. So, the broker's enforcing mechanism rejects the request. The broker then sends the result to the service-requesting agent which could decide to negotiate a contract.

The privacy-aware communication protocol

We extended SALSA's communication protocol with methods and events to negotiate privacy contracts between agents, to enforce privacy requests, and to specify different levels of QoP as policies. We implemented seven methods so that agents could

- announce a QoP level,
- send and enforce notification when a request is rejected,
- send a contract negotiation request,
- notify other agents of a contract's clauses,
- send the status of a contract agreed to by agents,
- send an agent policy, and
- convey to other agents the context of information handled.

For example, an agent uses the sendPolicy(xmlContent, agentID) method to inform another agent of its IPA or IRS policies. When invoked, the instance of the method shown in figure 2 forms an XML message by adding the tags specifying to whom the message is addressed (agentB@aBroker), the type of service requested (policy), and the parameters needed to perform the action (location and identity).

The programmer handles the code for each communication's agent policy because this communication depends on the application logic and the details of the pervasive environment's agent system. This flexibility also supports customizable privacy policies, such as those for organizational requirements, as well as the privacy requirements of a user, service, or device. For example, a hospital might define certain rules to be HIPAA compliant—for instance, that the medical record can't be removed from the hospital. We illustrate the privacy-aware facilities incorporated into SALSA through our redesign of a context-aware mobile communication system aimed at addressing the privacy concerns raised by potential users.

Design of a privacy-aware application

The Context-aware Hospital Information System (CHIS) lets users send messages specifying a set of conditions that must be satisfied before the system delivers such messages. These conditions, such as the recipients' location and role or the status of an artifact, then become the message's delivery context.[1] For example, a physician can send a message to the doctor responsible for a patient in the next shift when the laboratory results become available. In addition, CHIS provides access to information and services according to user context. CHIS considers contextual information, such as the user's identity, role, and location; device used; time; and status of an information artifact (for example, the availability of lab results). So, when a physician carrying a PDA is near one of his patients, the system might display that patient's clinical record. CHIS integrates a location estimation mechanism through which the system provides awareness and determines which information to send and present to the users. CHIS displays, as a list of users or in a floor map, the hospital worker's and resources' location. Hence, using a handheld computer, medical staff can locate patients, other staff members, and resources.

We presented CHIS in a scenario-driven session to 28 hospital staff members—13 physicians, eight nurses, and seven support staff—to evaluate the system's core characteristics and the staff members' intentions to use the system. Although the results showed that the system would be useful in their daily work, potential users expressed several privacy concerns that could become obstacles to the system's adoption. For this reason, we conducted an additional study to specifically evaluate the potential privacy risks of using CHIS.

Privacy risks of the CHIS system

We presented two scenarios to hospital workers to measure their privacy concerns about the system. The workers identified both scenarios as useful, but they also expressed privacy concerns. The first scenario illustrates how an intern can use a handheld device to request laboratory tests, receive context-aware notifications of the results' availability, and visualize the location and name of all hospital workers in the vicinity. The second scenario shows how a supervisor can learn whether an employee has performed a given procedure by looking at where the intern was throughout the day and looking at pictures of him taken at different times during his shift. This scenario has serious privacy implications because the nurse (and potentially other supervisors) can track the intern's location throughout his work shift. Nevertheless, we included such a scenario because the interns' supervisors found it useful.

We organized a scenario-driven evaluation to show both scenarios to 27 medical interns. We asked the participants to situate themselves in a specific role within each scenario. We then asked them to complete a survey with seven Likert-scale assertions to evaluate the privacy threats they perceived from this technology. Finally, we applied these findings to identify the contextual information used to regulate and manage privacy, as well as the potential risks of using CHIS. We grouped the main privacy risks the interns identified into three categories.

```
<message to='printer@aBroker'
        from='agentB@aBroker' >
<x xmlns='x:command'>
    <params><type>policy</type>
        <location>room</location>
        <identity>role</identity>
    </params>
</x>
</message>
```

Figure 2. XML message announcing the agentB@aBroker policy to the printer@aBroker.

Sensing. This risk relates to CHIS' ability to invisibly track hospital workers' locations. Privacy concerns centered on where, when, and how CHIS collects this information. Hospital workers considered some areas (such as a bathroom) to be *personal spaces*—areas where they didn't want to be monitored and therefore wouldn't use the system. Thus, the risk of an invasion of privacy surpassed the benefits the system provided. Similarly, the time of day played a role. For example, during night shifts, the medical interns sleep for a few hours, at which time they don't use the system. In addition, interns were concerned about the device's capability to collect their location information. Although the method CHIS uses to monitor their location doesn't represent a threat, not knowing how and when CHIS collects the data does.

Sharing. CHIS independently shares identity and location information with all hospital workers in the user's vicinity. This raised several privacy concerns about when, where, and with whom CHIS was sharing information. For example, while a medical intern is in the dining room, he'd like to protect his privacy by regulating the precision with which CHIS shares his location or identity. Thus, the need for privacy when using CHIS tends to be location and time sensitive, meaning the information a user is willing to share could change after a certain period of time or depending on the user's location or the time of day. In addition, CHIS doesn't let hospital workers specify availability levels, nor select with whom they'd like to share their location or identity. CHIS emphasizes hospital workers' availability without giving them control over this information. But if a medical intern is in the surgical room, he doesn't want a nurse to interrupt him to sign

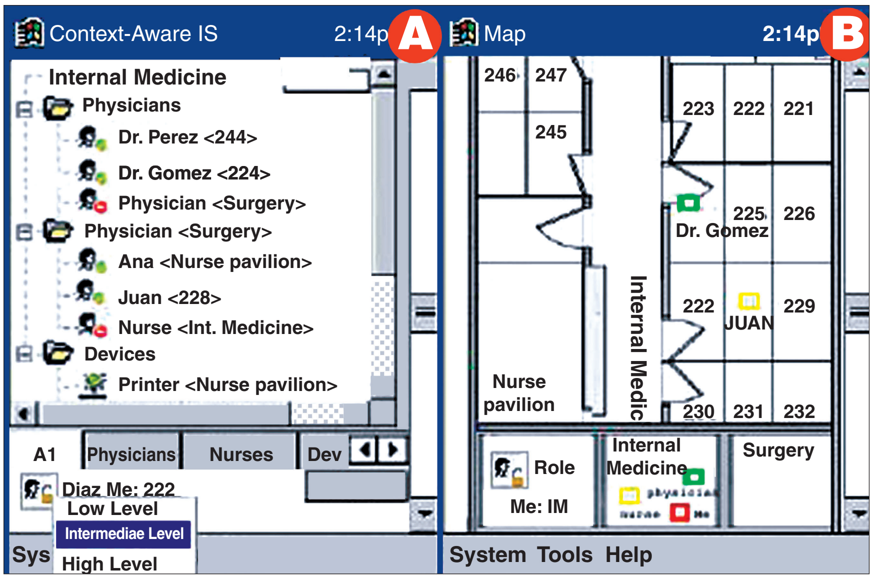


Figure 3. Based on the quality of privacy (QoP) level that a user demands, the Context-aware Hospital Information System (CHIS) provides (a) a list of user locations and services, and (b) a floor map.

a medical note, and he doesn't want context-aware notifications to inform him of the availability of laboratory results.

Storing. Related to the management of the information CHIS stores, privacy concerns emerged regarding the content and persistence of information capture. For example, hospital workers didn't realize that their location information might be stored for long periods of time and could be used to derive secondary contextual information, such as the places they visit frequently or with whom they normally interact. Although CHIS doesn't store location information for long periods of time, it does store contextual messages until they're delivered.

Integrating privacy-aware SALSA agents in CHIS

Because the privacy risks raised by CHIS depend on contextual circumstances such as the user's location, identity, and activity, as well as the persistence of the information captured, we decided to modify CHIS accordingly. We extended CHIS to harness the capabilities provided by privacy-aware SALSA agents. Our solution involved redesigning CHIS to show how considering contextual information to incorporate different QoP levels lets designers adequately and easily manage users' privacy. For instance, suppose Dr. Diaz, a medical intern, joins the hospital information system, demanding a QoP level to communicate his location and identity information by sharing only the room he's in and his role. On the other hand, on the basis of Dr. Diaz' QoP level, CHIS will adapt the information it shows him according to the other agents' QoP levels. As figure 3 shows, a nurse allows only sharing her information by symmetry (meaning both persons see the same thing); hence, Dr. Diaz will know only the floor she's on and her role. In addition, as the privacy-aware bar in figure 3a shows, Dr. Diaz is aware that CHIS is monitoring his information of location and identity; he is also aware of his QoP level and his status (level of availability). By selecting from this privacy-aware bar the icon that represents his QoP level, Dr. Diaz could change this level whenever he wants. For example, if he's in the dining room, he doesn't want to be monitored or have other people infer how long he was there. So, he decides to share his identity as anonymous without presenting his location or activity information by changing his QoP level. Therefore, by allowing different QoP levels, the autonomous agents in CHIS provide a clear value proposition (identifying the amount of information a user needs to share with the application to obtain the benefits that the application's services provide), awareness, and negotiation access control.

To provide such functionality, we redesigned CHIS using privacy-aware SALSA agents. Because we incorporated privacy mechanisms into SALSA's action, perception, and reasoning components, we didn't accomplish this redesign at the architectural level. Rather, we created the QoP levels for users, devices, and services to rule the privacy enforcement in CHIS, and we implemented the actions executed by each agent to enforce privacy.

A CHIS privacy-aware SALSA agent

The privacy that a CHIS user demands and expects concerns location, identity, and activity. Thus, we propose different QoP levels aimed at protecting this contextual information. By extending the context-aware client's interface, a user can specify his or her location, identity, and activity according to three different QoP levels. For example, a user could set the location QoP level to include the room he's in (low level), the floor he's on (intermediate level), or no location information at all (high level). Thus, if a physician doesn't want to be monitored while he's in the dining room, he sets his QoP to high, controlling the location information shared with CHIS. Similarly, we established three different QoP levels for the user's identity and activity, and for the agents who can access this user's information. The system maps these different QoP levels to an IPA policy, as figure 4a indicates.

Figure 4b illustrates the structure of a privacy-aware SALSA agent representing a user in CHIS. The agent inherits the SALSA components for reasoning, action, and perception, along with the privacy-aware mechanisms incorporated into SALSA. We extended the reasoning component so that an agent can trigger privacy events and react to such events generated by SALSA's communication protocol. Extending the reasoning component involved adding rules that allow the execution of privacy actions. For example, if a user changes his QoP, the announcing mechanism triggers an ArrivePresenceEvent. The reasoning component maps the requested QoP level to the corresponding IPA and IRS privacy policies. Using SALSA's filtering privacy mechanism, it then executes the correct action to update the information that SALSA's communicating mechanism communicates to the broker. Similarly, we established different rules in the reasoning component to allow negotiations between agents, such that on the basis of these results CHIS can execute the *negotiate* action. The negotiate action determines the amount of information required for the delivery of a service by informing the agent's IRS policy.

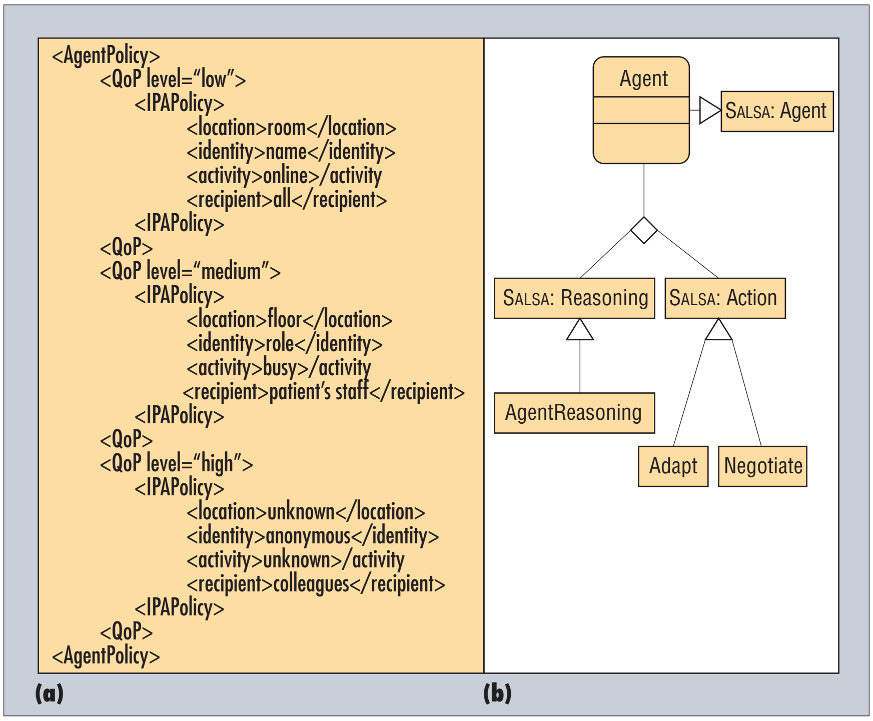
As figure 4 shows, the following mechanisms, which we incorporated in SALSA, simplify privacy management, letting programmers focus on the privacy logic and the QoP levels that users desire:

- the notification of the QoP level,
- the notification of a denied request, and
- the context assigned to a particular privacy event.

Sample application

Figure 5 shows how the CHIS components interact with the privacy-aware mechanisms we incorporated in SALSA by considering the user's context and adapting the application's behavior to the QoP level agreed to by a user agent and a service agent.

When a user joins the system demanding a certain QoP level, the agent representing him maps this QoP level to the corresponding IPA and IRS privacy policies. The user agent's registering mechanism then stores and registers these policies in the agent directory. Consequently, the user agent's filtering mechanism filters the user's information (location, identity, and so on) to respect the user's required QoP level. The user agent's executing component then communicates this information, along with the user agent's presence and IPA and IRS policies, to the broker. Next, the broker broadcasts this information to the agents subscribed in the agent directory. The context-aware privacy client updates the user information displayed by CHIS. When the user agent requests a service, the broker's enforcing mechanism compares the user agent's IPA with the service agent's required IRS. Because the user agent IPA doesn't match the service agent's required IRS, the broker's enforcing mechanism rejects the request. The broker informs the user agent of this result. The user needs this service, so the user agent shares this user's personal information with the service agent to negotiate a contract. The service agent then tells the user agent the conditions of the contract in a set of clauses. If the user agent accepts these conditions, the service can be delivered. The user agent receives these clauses and signs a contract with the service agent, accepting the latter's conditions. Finally, the user agent updates its policies to specify the clauses of the new contract just acquired.



```
<AgentPolicy>
        <QoP level="low">
              <IPAPolicy>
                      <location>room</location>
                      <identity>name</identity>
                      <activity>online>/activity
                      <recipient>all</recipient>
              <IPAPolicy>
        <QoP>
        <QoP level="medium">
              <IPAPolicy>
                      <location>floor</location>
                      <identity>role</identity>
                      <activity>busy>/activity
                      <recipient>patient's staff</recipient>
              <IPAPolicy>
        <QoP>
        <QoP level="high">
              <IPAPolicy>
                      <location>unknown</location>
                      <identity>anonymous</identity>
                      <activity>unknown>/activity
                      <recipient>colleagues</recipient>
              <IPAPolicy>
        <QoP>
<AgentPolicy>
```

Figure 4. Privacy-aware SALSA agent representing a user: (a) IPA (information provided by an agent) policy specifying three QoP levels; (b) class diagram of the agent.

We are developing mechanisms to automatically base the QoP level on the user's context. These mechanisms take into account contextual information such as the user's location and identity, the time of day, the artifacts used, and the presence of colleagues to infer hospital workers' availability and privacy demands. Once a pervasive application has strong evidence of the users' privacy needs, it can adapt itself by selecting the adequate QoP level to proactively enforce hospital workers' privacy. We will integrate these mechanisms into the privacy-aware SALSA agents. ■

Acknowledgments

We thank the personnel at IMSS (Instituto Mexicano del Seguro Social) General Hospital in Enseneda, Baja California, Mexico. This work was founded under contracts CO3-42447 and Conacyt-U-40799, and through scholarship 179381 provided to Monica Tentori.

References

1. M. Munoz et al., "Context-Aware Mobile Communication in Hospitals," *Computer*, vol. 36, no. 9, 2003, pp. 38–46.
2. J.X. Jiang and J.A. Landay, "Modeling Privacy Control in Context-Aware Systems," *IEEE Pervasive Computing*, vol. 1, no. 3, 2002, pp. 59–63.
3. V. Bellotti and A. Sellen, "Design for Privacy in Ubiquitous Computing Environments," *Proc. 3rd European Conf. Computer-Supported Cooperative Work* (ECSCW 93), Kluwer Academic Publishers, 1993, pp. 77–92.
4. M.D. Rodriguez et al., "Agent-Based Ambient Intelligence for Healthcare," *AI Comm.*, vol. 18, no. 3, 2005, pp. 201–216.
5. X. Jiang, J.I. Hong, and J.A. Landay, "Approximate Information Flows: Socially-Based Modeling of Privacy in Ubiquitous Computing," *Proc. 4th Int'l Conf. Ubiquitous Computing* (Ubicomp 02), Springer, 2002, pp. 176–193.
6. M. Langheinrich, "A Privacy Awareness System for Ubiquitous Computing Environments," *Proc. 4th Int'l Conf. Ubiquitous Computing* (Ubicomp 02), Springer, 2002, pp. 237–245.
7. L. Palen and P. Dourish, "Unpacking 'Privacy' for a Networked World," *Proc. SIGCHI Conf. Human Factors in Computing Systems*, ACM Press, 2003, pp. 129–136.
8. G. Myles, A. Friday, and N. Davies, "Preserving Privacy in Environments with Location-Based Applications," *IEEE Pervasive Computing*, vol. 2, no. 1, 2003, pp. 56–64.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.

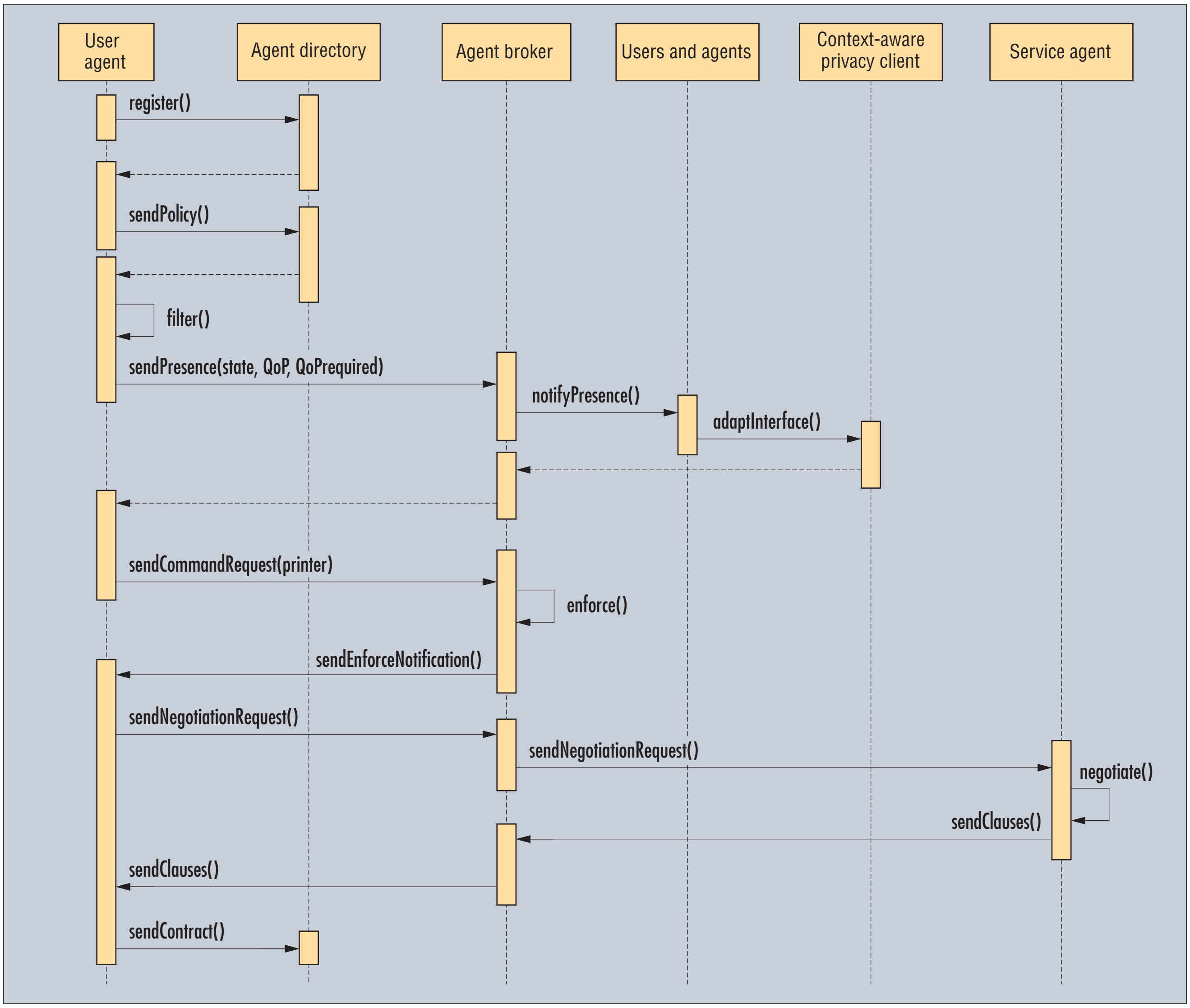
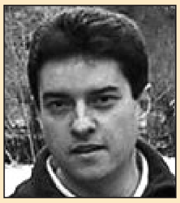


Figure 5. Sequence diagram for negotiating a QoP level for a CHIS application.

The Authors

Monica Tentori is a PhD student in computer science at the Center of Scientific Research and Higher Education of Ensenada (CICESE), Baja California, Mexico. Her research interests include ubiquitous computing, human-computer interaction, and medical informatics. She received an MSc in computer science from CICESE. She is a student member of Sigchi. Contact her at the Computer Science Dept., CICESE, Km. 107 Carretera Tijuana-Ensenada, Ensenada, B.C., 22860, Mexico; mtentori@cicese.mx.



Jesus Favela is a professor of computer science at CICESE, where he leads the Collaborative Systems Laboratory and heads the Department of Computer Science. His research interests include computer-supported collaborative work, ubiquitous computing, and medical informatics. He received a PhD in computer-aided engineering from MIT. He is a member of the ACM and the American Medical Informatics Association. Contact him at the Computer Science Dept., CICESE, Km. 107 Carretera Tijuana-Ensenada, Ensenada, B.C., 22860, Mexico; favela@cicese.mx.

Marcela D. Rodríguez is a professor in computer engineering at the Autonomous University of Baja California. Her research interests include ubiquitous computing, autonomous agents, human-computer interaction, and computer-supported collaborative work. She received a PhD in computer science from CICESE. She is a member of the ACM and Sociedad Mexicana de Ciencia de la Computacion. Contact her at Facultad de Ingeniería, Universidad Autonoma de Baja California, Ave. Alvaro Obregón y Julian Carrillo S/N, Mexicali, B.C., 21100, Mexico; marcerod@uabc.mx.