Developing a Security Protocol for a Distributed Decision Support System in a Healthcare Environment

Liang Xiao
University of Southampton
United Kingdom
lx@ecs.soton.ac.uk

Paul Lewis
University of Southampton
United Kingdom
phl@ecs.soton.ac.uk

Alex Gibb
University of Birmingham
United Kingdom
a.j.gibb@bham.ac.uk

ABSTRACT
In this paper, we describe the unique security issues involved in healthcare domains. These have been addressed to the needs of the HealthAgents project. In the proposed approach, several levels of security have been provided in accordance with Software Engineering principles, ethical regulations for healthcare data, as well as the security requirements usually raised from the distributed clinical settings. The result is the production of a secure and maintainable Multi-Agent System that enables secure communication, uniform home site authentication, and customised resource access authorisation. A security policy rule scheme has been designed for agent interaction modelling. This separates the functional and non-functional (security) requirements but let security policy constraints integrate into the running of the agents via a unified role notion. Each user/agent can play a function role only when its assigned social rights roles permit the access to resources of various types and geographical locations, as specified in the function role behaviour. The approach is illustrated using a comprehensive secure access case.

Categories and Subject Descriptors
D.2 [**Software Engineering**]; J.3 [**Life and Medical Sciences**]: Health, Medical information systems; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms: Design, Security, Languages

Keywords
Security Model, Healthcare, Distributed Decision Support System

1. INTRODUCTION AND BACKGROUND
Assisting medical diagnosis has been one of the main goals of Software Engineering (SE) and Artificial Intelligence (AI) [6]. One approach, the use of them together as a decision support system (DSS), is of particular interest. A successful medical DSS would aim to improve the healthcare outcomes required by an individual clinician. The process of designing such a system requires not only consideration of the clinician's needs but also access to data and processes that may be geographically distributed, and at the same time ensure interaction with other healthcare professionals [7]. Access to patient data, for example, is complicated further by the patients' healthcare records. A patient can have multiple visits to multiple medical centres for different conditions. This requires that data is shared between a distributed set of hospitals without recourse to a centralised system, i.e. an agreed protocol. The protocol is employed as part of the distributed decision support system (d-DSS) which increases the autonomous interaction between the medical centres. This interaction requires not only that data is not lost, but that the technology is trust worthy and secure [5]. Patient privacy and safety are in danger in any system application if the appropriate technological practices are not in place.

These different attributes to providing a medical d-DSS, that is distributed, bring challenges to security for data and services access that are transmitted over insecure transportation networks amongst hospitals. Challenges also lie in the access policy management and application since each centre needs to maintain its own control over its own resources. Moreover, the access to private patient records must be in compliance to legal and ethical regulations as set in countries where centres are located. All resources and services provided by the system should be protected accordingly to guarantee that decision making support for healthcare diagnosis can be executed as it ought to be.

Several existing Software Engineering technologies have been identified as qualified candidates to meet one perspective or another of the multi-facets security needs in d-DSS: Multi-Agent System, Certificate and Public Key Infrastructure, Role-Based Access Control model, and so on. This research investigates the best practice of building a secure d-DSS with the available SE techniques in our HealthAgents project [10] in line with the SE disciplines and healthcare guidelines.

The rest of the section gives background for security in Software Engineering, the legal and ethical guidelines of security in healthcare, as well as selected approaches in the area. Section 2 provides an overview of our HealthAgents project and its system architecture. Section 3 describes the architecture's conformance to the data protection regulations. Section 4 analyses the HealthAgents's security requirements and describes several cases where security must be enabled in resource access messaging flow. Section 5 offers an overview of the proposed solution, followed by detailed discussion of its secure transportation, authentication mechanism, and authorisation mechanism which is illustrated using a representative secure resource access case. Finally, we conclude in Section 6 by describing the security achievements of the proposed d-DSS from various aspects and our future work.

1.1 Security in Software Engineering
Correctness, maintainability, and security are among the key issues under consideration when the Software Requirements Specification (SRS) of a software system is to be documented describing the

ICSE'08, May 10–18, 2008, Leipzig, Germany.

system's intended functionality and attributes, according to the IEEE standard 830-1998 [1]. Unlike functional requirements which describe the system's expected behaviour, maintainability and security are quality requirements which describe the manner in which the system behaves. The security attribute of software can serve as the requirement of protecting the software from accidental or malicious access, use, modification, destruction, or disclosure [1]. The maintainability attribute of software can serve as the requirement to ensure the ease of maintenance, enhancement, adaptation, or correction of software to satisfy the specified requirements. Requirements originating from security and maintainability may significantly direct or restrict the design. These quality requirements are described in the Software Quality Knowledge Area (KA) as part of the Guide to the Software Engineering Body of Knowledge (SWEBOK) [2]. It serves a baseline on a core body of knowledge on Software Engineering established by the IEEE Computer Society and the ACM.

The distinct software quality attributes abstracted according to KA demand careful study prior to software development. Security, for example, is interrelated with functionality as well as maintainability. This complexity has to be addressed before a good security-enabled software design can be achieved. The normal business functional requirements and non-functional requirements of security require the software to be maintainable, in terms of both the required functionality and the identified security constraints. On one hand, these requirements should be separated rather than intermixed, for improving maintainability or modifiability [1] and supporting continuous software evolution. This separation implies the maintenance of security requirements should be in a separate process from the maintenance of functionality. This principle accords with the idea of separation of concerns. On the other hand, the security issue has an impact over the overall system and, this again, presides over its functionality. Moreover, secure access control is a key aspect of the management of configuration items in software configuration [2], supporting software to function properly. Thus, separately maintained functional and non-functional requirements need at some point to be merged and integrated. Therefore, functionality and security requirements need to be maintainable, separately, and at the same time, the maintenance needs to reflect the effects of security-related change to the functionality. In doing so, the software system under development will be functional, secure, and maintainable.

1.2 Laws and Regulations

The UK Data Protection Act 1998 came into force in 2000. It regulates the processing of data of individuals, including the obtaining, holding, use or disclosure of such information. The data protection principles are as follow.

1 Personal data shall be processed fairly and lawfully (and under certain conditions).

2 Personal data shall be obtained only for one or more specified and lawful purposes, and shall not be further processed in any manner incompatible with that purpose or those purposes.

3 Personal data shall be adequate, relevant and not excessive in relation to the purpose or purposes for which they are processed.

4 Personal data shall be accurate and, where necessary, kept up to date.

5 Personal data processed for any purpose or purposes shall not be kept for longer than is necessary for that purpose or those purposes.

6 Personal data shall be processed in accordance with the rights of data subjects under this Act.

7 Appropriate technical and organisational measures shall be taken against unauthorised or unlawful processing of personal data and against accidental loss or destruction of, or damage to, personal data.

8 Personal data shall not be transferred to a country or territory outside the European Economic Area unless that country or territory ensures an adequate level of protection for the rights and freedoms of data subjects in relation to the processing of personal data.

1.3 Existing Software Engineering Approaches to Security in Healthcare

In the last subsections, the paper reviewed the general SE principles for building secure systems and related ethical data protection regulations. Taking these into account, it is believed that security must be engineered into a system from the starting point to achieve pluggable and maintainable security policies technically and, in addition, provide a lawful approach for data access and transfer. A software system may have its functionality and usability negatively compromised if security is to be added or fixed after its implementation. It is our intention to build a separate yet integrative security sub-system within the overall software design, being a target of direct maintenance and at the same time the result of changes immediately being reflected in the whole system. This will alleviate the burden of post-implementation development phases where, both high cost of maintenance throughout code and limitation of testing are considerable barriers to achieving security.

The agent technology is promising in both the building of a d-DSS for healthcare and ensuring its security. On one hand, agents have the capabilities for representing different services required by the system, providing the backbone to ensure the distribution of data, and offering intelligent answers to the demands of the users. On the other hand, their abstraction of different processes where resources are accessed can be under the security control if appropriate measures are imposed upon them. Several approaches that employ agents in healthcare domains for providing security have been investigated.

The concept of heuristic security agents has been introduced in a scheme [5], in which all calls are intercepted to files, networks, library modules and components, as well as other resources. They are checked against behavioural rules before an "allow" or "deny" decision is made, preventing the entire classes of attacks to healthcare information systems.

Security concern has also been focused upon the private patient information sharing among interconnected hospitals. Secure access of electronic healthcare records (EHR) which may be scattered across healthcare units has been considered in [3]. A scheme is proposed that employs a security agent per site which authenticates users and controls the access to the local resources by looking at the user roles. The dedication of an agent for the full security control of each site suffices for the protection of a simple resource type of patient records from a single point of access. However this approach will expose its insufficiency when 1) multiple resource types are available each corresponding to a responsible party in an individual's site and 2) in addition attempting to share some common services amongst multiple sites and 3) at the same time ensure the differentiated access privileges of each user.

Another approach to the similar problem of exchanging private patient records among distributed hospitals introduces a four-tier architecture, a central access control (CAC) system and multiple local access control (LAC) systems sitting between the client application and hospital information systems [4]. CAC and LAC are Multi-Agent Systems which use authentication agents, encryption agents, and access control agents. Multiple LACs enable hospital managers to maintain their distinct access control policies over patient records. The single CAC serves as a communication hub establishing secure communication network with each LAC so that

data access requests can be forwarded amongst LACs and actual data can be passed amongst them in a secure manner. In this architecture, the security level is determined by the weakest LAC and the central CAC may impose a performance bottleneck and a single point of failure to the entire system.

All the above methods introduce agents or multi-agent systems explicitly for the purpose of access control, security not being considered as part of an integrated software design by software engineers in the first place. It has been shown in the Agent.Hospital framework [8] that it is feasible and beneficial to employ MAS as well as ontology technology for modelling and integrating existing individualised healthcare processes towards distributed decision making processes with improved assistance for enabling diagnosis and subsequent treatment plans for cancer patients. The addition of security-specific agents will impose extra designs for existing healthcare system implementation and requires a runtime communication overhead and in addition the maintenance of the security components. Our hypothesis is that, a Multi-Agent System will be most effective in securing a healthcare information system if its participant agents serve core clinical business functions with associated security measures or policies applicable by the agents as behavioural constraints before their performance of normal functioning behaviour in the clinical setting. In doing so, functionality and security are integrated into a single architecture but security policies can be separately maintained, hence improving the software design and the resultant application.

2. THE HEALTHAGENTS PROJECT

2.1 Project Overview

The HealthAgents system is a distributed DSS that supports diagnosis and prognosis, employs a set of distributed nodes that either store patient case data, build classifiers that are trained upon case data and capable of classifying tumour types, or use classifiers for the diagnosis and prognosis of brain tumours. The magnetic resonance spectroscopy (MRS) data used by the system is built up using anonymous information from child and adult cases. Classifiers are created by the producer nodes that receive requests from the clinicians to generate classifiers for particular tumours. Clinicians with cases will employ classifiers to assist in the diagnosis of patients for particular tumours. The HealthAgents system consists of a variety of agents each charged with a different task. In the real world, the main sites will be located at the University of Birmingham with 50 different contributing centres, at the Universitat Autònoma de Barcelona with 6 centres, and at the Universitat de Valencia with 4 centres.

2.2 Key Components and Architecture

Figure 1 shows a prototype version of the HealthAgents d-DSS. Each clinical node, as part of the inter-networked system, can be either a user where requests for classification of a given case are delivered, or the producers where classifiers are created or retrained based on pattern recognition techniques, or both. In any case, they all contribute their data for the training of classifiers. New classifiers may be produced or existing ones improved when new case sets become available, due to the growth of data in existing centres or new centre participation. When a *clinical user* requests the classification of a case that resides internally, its associated *GUI Agent* will retrieve the patient data from the local hospital database via a *Database Agent*, *local data access policies* being applicable.

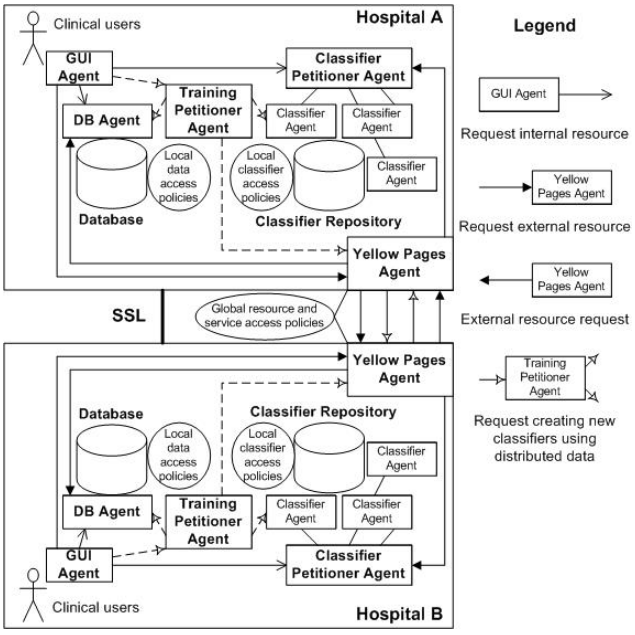

Figure 1. The distributed architecture of the HealthAgents system and its resource access flow control.

Alternatively, if the case under classification resides externally, then the GUI Agent will contact the local *Yellow Pages Agent*, which in turn will contact an external Yellow Pages Agent through which patient data is retrieved via the Database Agent of that hospital, *external data access policies* being applicable. One Yellow Pages Agent resides in each hospital's local node. They synchronise with each other and together maintain a directory of available nodes, agents, as well as the classifiers for the entire HealthAgents network. Their knowledge of the availability and location of resources is useful for answering queries sent from the GUI Agent. *Global resource and service access policies* will apply when 1) cross-centre resource access is requested by an agent, and 2) global services such as the query service provided by the Yellow Pages Agent are requested.

Once the case has been loaded into the GUI application, it may be classified. The local Yellow Pages Agent has registered in it classifiers that can discriminate among tumour classes, including descriptions about their capabilities, reputation, and the training data upon which they have been produced. The clinician may send various questions about the patient's condition including details about the tumour state being aggressive or non-aggressive, the type of cancer, to the Yellow Pages Agent to solve. The Yellow Pages Agent looks up its local registry, contacts external Yellow Pages Agents, and compiles a list of appropriate classifiers. This list is returned to the clinician and the clinician can now send the selected classifiers which can solve questions, accompanied by the patient data that these classifiers can operate upon, to the *Classifier Petitioner Agent*. The Classifier Petitioner Agent will invoke each *Classifier Agent* associated with the classifiers in the list, supplying to it the patient data. *Internal or external classifier access policies* will apply, depending upon the location of classifiers. While this may involve remote classifier access which gives the system a sense of full distribution, in practice, once a classifier is produced a copy might be obtained by every node in the network for local classifier running and better performance.

After the execution of classifiers, classification results are collected by the Classifier Petitioner Agent from multiple classifiers and ranked using statistical data and finally sent back to the clinician. The clinician can now do the diagnosis, supported by the answers and recommendations provided by the system. Eventually, when the diagnosis is finished, the clinician evaluates the classification result produced by the selected classifiers and their reputation updated. The above scenario assumes that classifiers exist to solve the questions. If no such classifier exists, a clinician requests the *Training Petitioner Agent* to create one using data from distributed sites and register the new classifier in the Yellow Pages Agent for later use.

3. REGULATION CONFORMANCE

3.1 Anonymisation and Transference

While complete patient records may be accessed only by hospitals and local nodes, link-anonymised records may be exchanged between a limited number of centres producing classifiers. The term link-anonymised data refers to data from which personal information (e.g. name, address, date of birth) is removed but to which a unique patient identifier is added, preserving patient anonymity but allowing data traceability and maintainability. Furthermore, only limited amounts of data which can be considered as totally anonymised may be accessed outside the closed project network.

Such a scheme constrains the form in which data can be exchanged in the system and the common use of link-anonymised data protects patient privacy. In addition, the direct use of patient case records no matter what forms they may take is largely minimised, and substituted by the use of classifiers that are produced upon cases for decision making. This offers a further level of protection to private data. Cases are normally only known to the classifier producer software agents but not revealed to clinical users. In the tumour classification processes, the produced classifier software agents, as opposed to the specific cases are used for assisting diagnosis.

3.2 Types of Patient Cases and Classifiers and Their Access Principles

Although patient private data is protected by the link-anonymised data scheme and its exposure to users minimised by the classification mechanism, the system still may have to allow the direct access of patient records and this requires some access principles. The age and the gender of patients, for example, can be associated with tumour types and so may be useful for diagnosis. Thus, a contract signed between two clinical centres working closely with each other may allow some cases to be transferred between the two, but not a third party. Also, some classifiers may be trained internally for scientific experiments upon a specific set of data and the creators may not wish them to be accessible to the general public due to their applicability and reliability. These requirements demand the differentiating attributes of HealthAgents resources and their associated access principles.

An anonymised patient case is associated with a status. The status of the patient case can be changed to, for example, *validated.* That is to say the patient diagnosis has been confirmed. The case can be *public*, being accessible by every HealthAgents node, or could remain *private*, being only accessible by its owner node or for producing classifiers. A selection of the validated cases labelled as public at each site can be shared altogether to produce *global* classifiers which are always public. A node can also request the creation of *local* classifiers that are trained uniquely with its own public and private data as defined by the requesting user. Apart from the global and the local classifiers, a node may want to develop *specific* classifiers that are trained with all public cases available in the network in addition to its own private cases, being given a special weight to gain more accurate classification results for this particular site's cases. Again, they can be defined by the requesting users as either public or private. Once a classifier is produced, no matter how it is produced and with what data, all the cases sent over from individual databases for training purposes will be discarded. The case data will only be temporarily stored in any other site apart from its origin.

3.3 Legal and Ethical Obligations

We have described the architecture of our d-DSS, the types of resources being used by it, and their associated access principles in previous sub-sections. We have analysed these against the UK Data Protection Act 1998 regulations listed in Section 1.2. The conformance to the Act can be briefly illustrated as follows.

In the d-DSS, patient case records are only processed for either the diagnosis of that particular patient or for training classifiers, fairly and lawfully, this is in compliance with *Principle 1*. The publicity of a case and its direct access is strictly controlled by the node where the case is stored inside the HealthAgents network and the routine use of such data is replaced by classifiers which are trained upon the data by classifier training software. Thus, cases will not be processed in any manner in contradiction to the specified and lawful purposes of improving disease diagnosis as agreed by the patients and their exposure is minimised. This is in compliance with *Principle 2*. The adequacy, relevance, non-excessiveness, accuracy, and up-to-date status of cases are maintained by clinical centres and wherever possible, link-anonymised data is used for the preservation of patient privacy, this is in compliance with *Principle 3* and *Principle 4*. All cases used for the purpose of training classifiers will be discarded when classifiers are produced and will not be kept for longer than it is necessary, this is in compliance with *Principle 5*. Patients retain the rights of withdrawing their cases and if requested they will be removed from the databases immediately (via the unique patient identifier being added to the link-anonymised data), this is in compliance with *Principle 6*. Each clinical centre enforces the described case access principles and so unauthorised or unlawful processing of personal data or damage to data will be avoided, this is in compliance with *Principle 7*. The HealthAgents project is building a network inside the EU boundary and may allow data transfer outside its network only if it is in a fully anonymised form and protected at an adequate level as being agreed upon, this is in compliance with *Principle 8*.

4. RESOURCE ACCESS CONTROL

Although in the discussion of Section 3.3 we validated the existing infrastructure against the data protection regulations in the UK as stated in Section 1.2, engineering disciplines for security in Section 1.1 must be equally respected in planning and implementing the proper set of security policies for network-wide resource access control. This is because the intended use of data as regulated by principles may be compromised when the system is abused or misused, by unauthorised people in unintended ways. Technical measures must be in place to prevent such access. This requires identifying methods that may result in the breaking of the so called CIA Triad: *confidentiality*, being concerned about unauthorised access to private information; *integrity*, being concerned about the

creation, change, or deletion of data without authorisation; and *availability*, being concerned about the loss of control over the functioning system and its security measures.

Fundamentally, the distributed nature of d-DSS should help to maintain the *integrity* amongst hospitals since individual centres can retain the control over their local patient cases and the policies for sharing them, the responsibility of overall data protection being spread. In addition, the distributed nature of d-DSS should improve *availability* with some built-in fault tolerance. When one node is down, requests for classification service can still be fulfilled due to multiple copies of classifiers being available across centres. Furthermore, the shift to classifier access from patient case access, which is now usually limited to the principle treating doctor and classification software, should help to improve the *confidentiality* of individual patient privacy. Nevertheless, some cases where the CIA Triad may be broken have been identified as follow.

- Theft and disclosure of patient privacy information by a hacker due to insecure transportation network – a confidentiality issue.
- Malicious users may create low quality classifiers – an integrity issue.
- Accidentally, inexperienced users may assign unreasonable reputation values to classifiers, such incorrect alteration of classifier reputation values will mislead diagnosis results – an integrity issue.
- Abuse of system services (Yellow Pages, Classifier Training, etc.) and so make them unavailable or even replace them with malicious alternatives and direct to wrong diagnosis – availability and integrity issues.
- Users from one hospital access data or execute classifiers from another hospital without the proper permission – confidentiality and integrity issues.

In order to avoid such potential security breaches, the existing HealthAgents architecture should tackle some generic security requirements as outlined below.

- Secure encrypted message passing among HealthAgents nodes.
- Local site authentication. Appropriate policy sets application wherever resources are required across centres without requiring extra identification.
- Global resource and service policy sets at the overall HealthAgents level.
- Dynamic site addition to the HealthAgents network and trust relationship management, straightforward new policy sets deployment and minimum intervene to the existing infrastructure.
- Individual policy sets for access authorisation at local sites which retain their independent control over resources reside in their own site and these policies should override the global policy sets wherever a conflict occurs.
- Transparent user interaction without requiring them to be aware of the security measures, their access privileges being dynamically managed and maintained.

Apart from secure communication and authentication, centred in these requirements is the access control over critical system resources. Resource access must be distinguished according to its sensibility and so access policies and handling procedures can be defined accordingly. Common security classification labels have been used widely, i.e. unclassified, sensitive but unclassified, confidential, secret, top secret. In the interest of HealthAgents, access levels can be classified for its resources under protection, in accordance with sensibility and confidentiality levels (from highest to lowest), as the following.

0. Update a private patient record: often only available to the patient's principle physician.

1. Read a private patient record: also available to the producers of specific classifiers.

2. Read a public anonymised patient record: available to classifier producers and under agreements to other hospitals in the HealthAgents network.

3. Create a classifier: available to specific experienced clinicians with sufficient power who may allow the classifier producers to access required anonymised data and later set the publicity of the classifier.

4. Update a classifier reputation: available to experienced clinicians who have executed that classifier upon a case and the accurate diagnosis result is known to them at that moment.

5. Execute a local classifier: often available to local hospitals.

6. Execute a global classifier: available to all hospitals in the HealthAgents network.

7. Invoke a system service (Yellow Pages, etc.): may open even to hospitals outside of the HealthAgents network, this allows them to gain better knowledge of the available resources inside the network so they may want to join in later.

These distinguished levels imply that the subjects of access can be categorised as individual users (level 0, 1 and 4), roles (level 3) and organisations (level 2, 5, 6, and 7). More restrictive permissions may be required. Access at level 5, for instance, may require a general open policy for an organisation to be restricted to an individual role or even a specific person for the use of a particular classifier. In this case, more specific rules in addition to some general ones may be defined. Generalisation is also possible. A patient's principle doctor, for example, may delegate the diagnosis to a specific trusted hospital, in a certain context. All these factors influence the building of a security model and the structure of policy rules. This will be described in Section 5.5.

Message passing: resource request agent→resource manager agent

Resource or result of resource usage passing: *resource*→resource request agent

Policy application: resource manager agent (apply *policy set S* with *operation O* upon *resource R, sensibility level L*)

Follow-up operation: *if success then:*

Figure 2. Symbols for expressing resource access control constructs within our cases.

Resource access levels being specified in such a scheme, based on which security policy rules can be defined, the policies must be applied to enable the HealthAgents access processes. We have outlined such cases in Figure 3 where various access control policy sets (referring to Figure 1) can protect a variety of resources in various control flows. Figure 2 provides the minimum set of symbols for expression of such cases. These cases specify the security needs of major HealthAgents business functions, supported by the envisioned policy sets and associated access sensibility levelling. Based on the formalism, logic expressions will be later employed for formal specification and implementation, illustrated by using a comprehensive case in Section 5.5.

Case 1 Execute a local classifier upon a local case.

GUI Agent A → DB Agent A (apply *Local data access policies A* with *read* upon *local case data, sensibility level 2*) and GUI Agent A → Classifier Petitioner Agent A (*apply Local classifier access policies A* with *execute* upon *local classifier, sensibility level 5*) *if success then: classification results* → GUI Agent A

Case 2 Execute an existing external classifier upon an already loaded case.

GUI Agent A → Yellow Pages Agent A (apply *Global resource and service policies* with *invoke* upon *query service, sensibility level 7*) → Yellow Pages Agent B → Classifier Petitioner Agent B (apply *Local classifier access policies B* with *execute* upon *global classifier, sensibility level 6*) *if success*

then: classification results → Classifier Petitioner Agent B (results being ranked) → Yellow Pages Agent B → Yellow Pages Agent A → GUI Agent A

Case 3 Update a patient case record from a local site and the reputation of an employed classifier after diagnosis.

GUI Agent A → DB Agent A (apply *Local data access policies A* with *update* upon *local case data, sensibility level 0*) and GUI Agent A → Yellow Pages Agent A (apply *Global resource and service policies* with *update* upon *classifier reputation, sensibility level 4*) *if success then: success* → GUI Agent A

Case 4 Build a new classifier using data sets distributed in another site.

GUI Agent A → Training Petitioner Agent A (apply *Local classifier building service policies A* with *invoke* upon *classifier building service, sensibility level 7*) → Yellow Pages Agent A → Yellow Pages Agent B → DB Agent B (apply *Local data access policies B* with *read* upon *local case data, sensibility level 2*) *if success then: case data* → Yellow Pages Agent B → Yellow Pages Agent A → Training Petitioner Agent A and *if success then: produced classifier* → Yellow Pages Agent A and *if success then: success* → GUI Agent A

Figure 3. Major security-critical cases in HealthAgents.

5. THE SECURITY SOLUTION

Security must cover three aspects for HealthAgents to fulfil the security requirements together. First of all, *communication* amongst clinical centres must be secured. This means that the contents of the messages being transported in the HealthAgents network which might contain patient privacy information or diagnosis results should not be intercepted or modified by eavesdroppers. One widely established techniques to resolve this problem is SSL. Java supports SSL connections among nodes by its key and certificate management tool. This same tool can be applied to set up secure links among distributed JADE containers. JADE-S, the extension package of JADE towards security also supports message-based signing and encrypting. This level of security ensures that messages passing in the network are safe but without concern of who is attempting to interact with the system. *Authentication* is, therefore, required at the next level of security. Only users with particular user names and passwords should be able to access the system. The Java Authentication and Authorisation Service (JAAS) provides a framework for user-based authentication. This is inherited by JADE-S. If it is assumed the previous two levels of security are in place, this will guarantee that only authenticated users can access the HealthAgents system and the communication among its distributed nodes is secure. Yet, users should only be able to access what they are allowed to with given permissions and nothing else. The last level of security should constrain the access control of the system, and only permit authorised operations to be performed upon critical system resources according to the security policies set by the administrative authority, as already discussed in the previous section. JADE-S inherits JAAS in this aspect for MAS and it is insufficient. This is because *authorisation* is business-dependent, but the actions JADE-S can permit or reject are specifically concerned with agents running in the JADE platforms, i.e. to create, send messages, or kill agents. Table 1 outlines these security levels.

Table 1. Three major security levels

	Principles	Protection	Techniques
Secure communication (to be discussed in Section 5.1)	All messages passing in the network should be securely signed and encrypted.	Messages in transmission are kept secrete and unaltered, ensuring confidentiality and integrity.	SSL, Public Key Infrastructure, JADE-S
Authentication (to be discussed in Section 5.2)	Users will be allowed to enter the system only if their identities are recognised.	The one who claims to be of an identity has indeed that identity. No one can pretend to be someone else.	JAAS
Authorisation (to be discussed in Section 5.3)	When resources are being requested, security policy rules, as set globally in the network, locally in hospitals, or individually by clinicians will be applied against the particular identity.	Users can access or perform operations upon critical system resources only if they have been authorised to do so, their access permissions being bound with their identities recognised during authentication.	Access control model and policy rules

An example of combining these security levels is shown in Figure 4. A clinician from hospital 1 retrieves and classifies a case from hospital 2 using a classifier from hospital 4. In this scenario, he/she must be authenticated in hospital 1 before access to the local network, after which all messages passing through for the classification purpose in the interconnected HealthAgents network will be encrypted and the case requested from hospital 2 will be classified using the classifier requested from hospital 4, both access to resources being authorised.

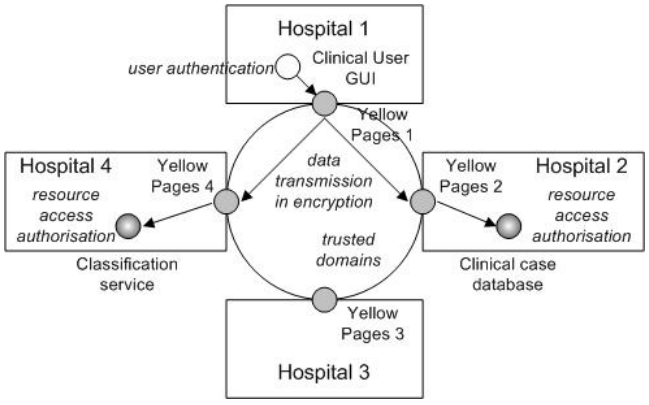

Figure 4. Overview of the security architectural levels using a cross-hospital resource access scenario.

5.1 Secure Message Transportation

Using the public key infrastructure (PKI) and digital signature, a secure communication protocol would be that the sender encrypts a message with a private key where the message is implicitly signed and on receipt of the message the receiver decrypts the message with the sender's public key where its signature is verified. JADE-S provides in its API a security helper and signature and encryption services. Apart from these, we make use of Yellow Pages Agents for storing and managing public keys and establishing trust relationships. In a conversation involving multiple parties, only those agents who have been formally recognised and registered in the Yellow Pages after their starting up will be regarded trustworthy and Yellow Pages Agents are responsible for acknowledging the trusted parties in the network. Thus, they play the role of Certificate Authority (CA) in the sense that they assure the trustworthiness of communicating parties. Being an integral part of the framework, the use of Yellow Pages Agents for secure communication enables easier management and simpler communication. Figure 6 shows the class diagram of the agents we have developed in this level.

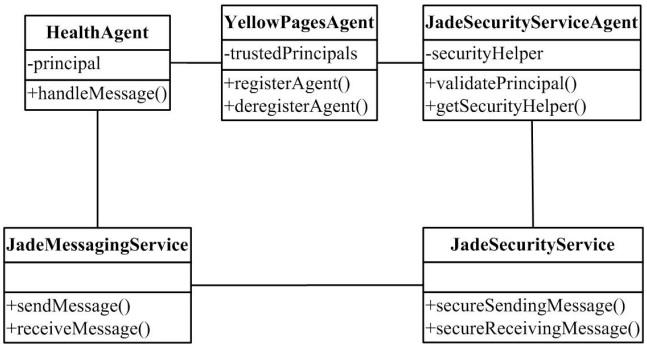

Figure 6. A HealthAgent achieves secure message sending and receiving via two facilitating agents and two supporting services.

a) In the beginning a *HealthAgent* in the network starts up, it needs to register itself in the *YellowPagesAgent*. The message of *registerAgent* must be signed so its identity can be checked and if recognised its *principal* (public key) will be added to the list of *trustedPrincipals* the YellowPagesAgent maintains.

b) Then the HealthAgent uses the *handleMessage* in communication with other agents which are trusted within the network. It uses *JadeMessagingService* for *sendMessage* and *receiveMessage*, which involve a message encryption and decryption process performed in *JadeSecurityService*.

c) When this HealthAgent attempts to send a message, JadeSecurityService will use its *secureSendingMessage* to check if the principal of the message receiver is in the trust list and if so, it will *sign* and *encrypt* the message and send it on. Otherwise no message will be sent since (even secure) communication with agents outside of the network will endanger the system.

d) When this HealthAgent receives a message, JadeSecurityService will use its *secureReceivingMessage* to check if the principal of the message sender is in the trust list and if so, it will *decrypt* the message and reply in signed and encrypted messages. Otherwise, if the message has not been signed or the signature is not recognised then the message will be discarded.

e) In both above situations, we need *JadeSecurityServiceAgent validatePrincipal* of the communicating agent against the trustedPrincipals of YellowPagesAgent for JadeSecurityService at runtime. It maintains in its *securityHelper* internally the trusted principals (*addTrustedPrincipal*) and provides it to JadeSecurityService so it can check the principals of message senders and receivers (*getPrincipal*) against those trusted (*getTrustedPrincipal*) in secureSendingMessage and secureReceivingMessage. Moreover, this helper also encrypts messages (*setUseEncryption*) in the process of secureSendingMessage and gets message signatures and decrypts messages (*getUseSignature* and *getUseEncryption*) in the process of secureReceivingMessage.

5.2 The Authentication Mechanism

Establishing secure message passing among communicating parties, we can be sure that, in the scenario given in Figure 4, a Clinical User GUI Agent, a DB Agent, and a Classifier Petitioner Agent are all within the trusted domains and their communication will be secure. Note that this does not take into account who logs in and initialises such a conversation involving these parties for classification purposes and that user must be authenticated at the GUI Agent side, otherwise the established secure communication does not protect the system and become meaningless.

A web-based GUI has been developed in HealthAgents for loading cases, performing classification, and presenting results. The associated GUI as currently in the prototype development stage assumes a single identical account for all user login and we intend to incorporate the JAAS authentication model into it for authentication. In the model, a user's identity should be confirmed in authentication, represented by a *subject* seen in the model. A *principal* is granted to the user after his/her identity is verified during the authentication, being associated with a set of *credentials*. Such a principal is bound with a user identity to the GUI, while the principal of a GUI Agent is bound with an agent identity to the HealthAgents network as described in the previous subsection.

A *LoginModule* performs the authentication, typically by prompting for and verifying a subject of his/her username and password. Several module implementations have been provided by JAAS and JADE-S and a special *SimpleLoginModule* allows very basic authentication. Alternative LoginModules can be loaded as configured in a *Configuration* file, being consulted by a *LoginContext* that can be instantiated from the GUI. LoginContext invokes a *login* method of the loaded LoginModule for authentication of subjects and upon success will associate principals and credentials with them. Principals of a subject can be later retrieved by invoking its *getPrincipals* method. JAAS policies can be configured for subjects and grant them authorised permissions following authentication. These can be later enforced by invoking *doAs(subject,action)* method, achieving the effect of having an action run as the subject. Those permissions, however, are centred on file or code access and are not of concern in HealthAgents. We will discuss in the next section the fine-grained access control mechanism to resources valued by HealthAgents based on the JAAS-authenticated subject principals.

5.3 The Authorisation Mechanism

Secure communication is guaranteed by identifying the principal of communicating sites and agents, and the secure interaction between users and the system is guaranteed by identifying the principal of the login users. Following these, as shown in Figure 4, the authenticated user should be able to access resources that he/she has been authorised. Authorising access to resources can be based on the identity of a principal, which may be mapped to unique roles for easier access control administration. Since the user principal is obtained in the previous authentication level, it should be passed on and encapsulated in following messages and available to the entire conversation (for classification, etc.). Then associated roles and groups can be looked up and applicable security policy rules enforced where the distributed system resources are accessed. Having been authenticated at the home site, the same principal will be reused for uniform authorisation across the centres. This supports successive security levels and provides a transparent user experience.

In our previous work [9], we have developed a Security Model and an associated Policy Rule Model. Briefly, they borrow the role permission association from Role-Based Access Control [11], avoid its weaknesses, and extend it towards a seamless integration with the role playing pattern from Agent-Oriented Software Engineering. The security model sitting in MAS won't let agents fulfil regular functional requirements unless security requirements are met. A role plays its functional duty if and only if its social constraints are satisfied. This scheme combines the social security role together with computation function role into an integrated role notion. This, therefore, achieves the separation of functional and non-functional

requirements for easier management and maintenance but at the same time the two parts are integrated in the running system with unified agent playing behaviour according to the combined specification. The fundamental access permission rules take the following form with four major dimensions.

{Subject (Id, Role, Organisation), Resource (Id, Type), Access Operation (Op), Access Context (Co)}

This provides fine-grained access permission configuration based on individuals, roles, and organisations. A resource access request message can be identified to its origin and mapped to the roles that subject plays. Role based policies are easier for management but identity based policies allow customisation and exception. Policies can be defined in both forms. In HealthAgents, we have case records, classifiers, services (Yellow Pages, etc.), and their access must be protected by policies. Access operations should be distinguished for resources. One clinician may be able to execute a classifier but not update its reputation as differentiated in sensibility levels as discussed previously.

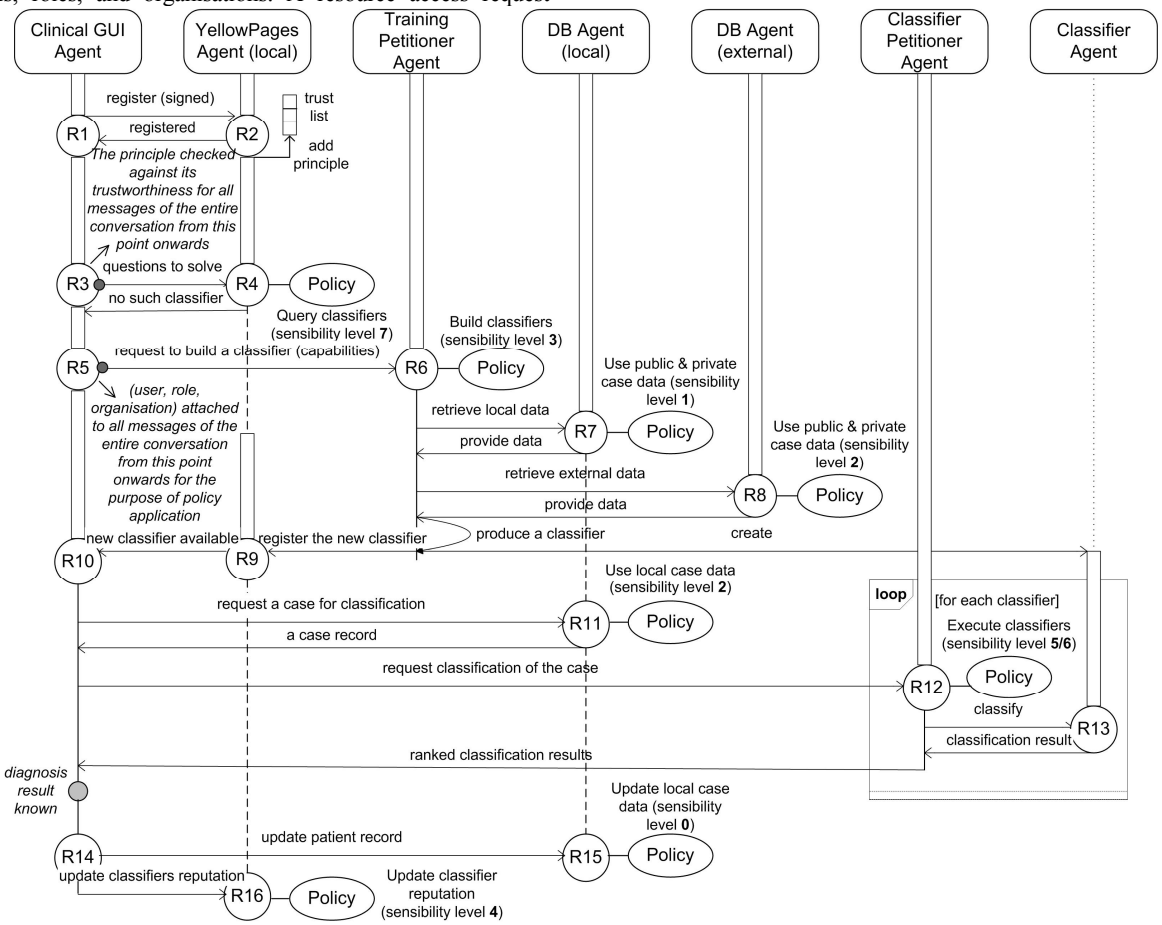

Figure 7. Agent interaction model with security policy set application in HealthAgents.

A context provides the flexibility to the model such as 1) allowing in particular situations certain specially delegated access in the name of a particular role; 2) providing justification of the special access; and 3) constraining the valid time period associated with the access.

Several security-critical access cases have been described in Section 4 and now a more comprehensive one involving all of them is used to demonstrate the application of the above policy rule scheme to meet the requirements described in that section. The case is a representative of most of the HealthAgents business functions as well as resource access flows. Briefly, the scenario is that a new hospital joins the HealthAgents network with a new MAS setup in that site, new clinician users wish to perform classification upon cases from there, and they do so by creating new classifiers for the purpose. The role interaction model (referring to Figure 7) can be described as follows, referring to Figure 1 for HealthAgents architecture and previous sections for supporting layers of secure communication and authentication.

- The new clinician is authenticated by JAAS via the local GUI Agent and his/her principal is bound with the interface for the entire interactive session (R1)
- The GUI Agent registers this new node via the YellowPagesAgent which recognises its identity (prior to this the local hospital manager may have to acknowledge the participation of the new site to the HealthAgents network administrator through conventional phone calls, R1 and R2)
- The YellowPagesAgent adds this new node to the trusted node list (R2)
- The GUI Agent at that node can start to communicate in the HealthAgents network and now it wants to perform a classification upon a local case (R3)
- The GUI Agent searches the YellowPagesAgent for available classifiers by sending questions to solve as the first message it initialises for a new conversation (R3 and R4)
- The YellowPagesAgent has its principal registered and it is in the trusted list so all ongoing communication in this conversation with all other agents will be allowed and all these messages will be signed and encrypted (R4)

- The YellowPagesAgent checks this GUI Agent against the permission of using its Yellow Pages query service and will perform the query to its registered classifiers but unfortunately no such classifier is available (R3 and R4)
- The GUI Agent requires the building of a new specific classifier (referring to Section 3.2 for definition) using distributed data sets (R5 and R6)
- The TrainingPetitionerAgent applies a local policy repository and allows the request operation of building a new classifier (R6)
- Relevant public cases as well as local private cases from the request site will be sent to the building site for the production of the new classifier and data access policy rules will be applied before the data is sent from each site (R6 and R7, R6 and R8)
- A new classifier is produced and registered to the YellowPagesAgent, a copy becoming available to the original request site (R6 and R9)
- The clinician now wants to execute the new classifier upon the case when being informed of the availability of the classifier (R9 and R10)
- The local policy rules on the use of the classifier and the particular case will be applied against this specific clinician and he/she will be allowed to do the operation (R10 and R11, R10 and R12, R12 and R13)
- Decision making support is received from the results of the classification and a diagnosis will be made later on (R12 and R14)
- When an actual diagnosis result is known, the clinician wants to update the classifier reputation and the case he/she just diagnosed and the local policy rules on both operations will be applied against the clinician and he/she will be allowed to do so eventually (R14 and R15, R14 and R16)

Figure 7 shows an interaction model that captures the interactive behaviour of involving agents each playing their respective roles, subject to the satisfaction of associated security policy constraints. The descriptive interaction behaviour which consists of message passing and constraint solving have been defined in Lightweight Coordination Calculus (LCC) [12] that can be transmitted, interpreted, and executed by agents in the network. The LCC language has been developed in the OpenKnowledge project [13] and it uses logic expression to regulate the message exchange protocols among participant peers each of which plays a particular role that dictates its particular message passing pattern in protocols. The following LCC clauses describe the fundamental interaction pattern for resource access control.

```
a(resource_request, RRID) ::
  request(Resource, Operation, Context) ⇒ a(resource_manager, RMID)
a(resource_manager, RMID) ::
  request(Resource, Operation, Context) ⇐ a(resource_request, RRID) ←
grantPermission(RRID, Resource, Operation, Context, Policies) then (
    response(Grant_yes) ⇒ a(resource_request, RRID) or
    response(Resource_result) ⇒ a(resource_request, RRID) ←
getOperationResult(Resource, Operation, Access_result) )
```

Briefly, a(resource_request, RRID) :: Def_{RRID} and a(resource_manager, RMID) :: Def_{RMID} denotes that agents RRID and RMID play the roles of resource_request and resource_manager respectively as defined in the definitions follow. Def_{RRID} has a single and Def_{RMID} has a composite message passing behaviour constructed using the following forms: Def_a *then* Def_b (Def_a satisfied before Def_b), Def_a *or* Def_b (either Def_a or Def_b satisfied), or Def_a *par* Def_b (both Def_a and Def_b satisfied). In the Def, $M_l \Rightarrow A_m$ denotes that a message M_l is sent to agent A_m while $M_l \Leftarrow A_m$ denotes that a message M_l is received from agent A_m. In the above role definitions, a message of resource access request is sent from the agent that plays the request role to the agent that plays the manager role. This appears equivalently as the first clause of two definitions as a message being sent or (expected to be) received for two communicating agents. Upon receipt of this message, the resource manager agent applies appropriate security policies and responds by sending back a message either saying the request has been granted (or rejected) or by providing the actual resources (or the results of their usage) being requested. In the Def, $\leftarrow Cons_n$ denotes that a constraint must be satisfied (as some running code) before the clause prior to it. Two constraints have been used in the second role definition, one being used for policy rule sets evaluation and enforcement and another being used for the computation following the use of required resource sets. It has been assumed that all operations will be granted in the above interaction model for simplicity.

In the following, we give the actual LCC clauses as the specification of the interaction model shown in Figure 7, concentrating on the resource access control procedures of case classification as well as case record and classifier reputation updating afterwards, in the interest of conciseness. The clinician role playing behaviour for resource access includes classification (R10) and updating of case record and classifier (R14). Its role changes when an accurate diagnosis result is known.

```
/* R10: classify a local case using the new classifier just produced */
a(clinician_classify, CID) ::
  classifierAvailable(C) ⇐ a(yellowpages_register, YPID) then
  requestCaseRecordByID(I) ⇒ a(database, DBID) then
  caseRecord (R) ⇐ a(database, DBID) then
  requestClassification(R, C) ⇒ a(classifier_petitioner, CPID) then
  classificationResults(S) ⇐ a(classifier_petitioner, CPID) then
  a(clinician_followingdiagnosis, CID)
/* R14: update case record and classifier reputation following diagnosis */
a(clinician_followingdiagnosis, CID) ::
  ( updateCaseRecordByID(I) ⇒ a(database_update, DBID) then
    caseRecordUpdated(Y) ⇐ a (database_update, DBID) )
  par
  ( updateClassifier(I) ⇒ a(classifier_petitioner, CPID) then
    classifierUpdated(Y) ⇐ a (classifier_petitioner, CPID) )
```

A construct *a(role, id)* can represent a clinician with a unique identity who wants to play a certain function role, being associated with certain constraints. Only when the social rights roles assigned to that identity are permitted to access all resources involved in the function role playing behaviour, the compound role of that clinician can successfully complete the required requirements. To that end, his/her access must be controlled by the database agents and classifier petitioner agents before permissions are granted and functions carried out. The local database role playing behaviour for resource access control includes database issues a case record (R11) and updates the same record (R15). Different access control policy sets will be enforced in two situations.

```
/* R11: send a case record for classification */
a(database_download, DBID) ::
  requestCaseRecordByID(I) ⇐ a(clinician_classify, CID) ←
grantPermission(CID, I, Read, Normal_classify_from_local_site,
Local_database_read_policy_set) then
  caseRecord(R) ⇒ a(clinician_classify, CID) ← getCaseRecordByID(I, R)
then
  a(database_update, DBID)
/* R15: update a case record after classification */
a(database_update, DBID) ::
```

updateCaseRecordByID(I) ⇐ a(clinician_followingdiagnosis, CID) ← grantPermission(CID, I, Update, Normal_update_from_local_site, Local_database_update_policy_set) then

caseRecordUpdated (Y) ⇒ a(clinician_followingdiagnosis, CID)

Being in compliance with the security policy schemes previously discussed, in every resource access request, the dimensions of (Resource, Operation, and Context) should be attached in addition to the identity of the requester. This identify can be extracted from the message being sent from the sender. Appropriate policy sets will be applied by respective resource manager agents (YellowPages Agent, Database Agents, Classifier Petitioner Agent, etc.). The constraint construct as part of the LCC language provides a solution that integrates the security constraints into the agent interaction protocols. These must be evaluated satisfactorily with a Boolean value of true returned once a resource request message is received and only then a response message can be sent back. A grantPermission method will be provided in the system that will be invoked for security policy application.

```
grantPermission(ID RRID, Resource r, Operation o, Context c, PolicySet p) {
  logger.setAccessAudit(RRID, r, o, c, getTimestamp());
  return applyPolicies(RRID, getRoleByID(RRID), r, o, c, p); …… }
```

This offers audit points where each access can be later traced back, hence the audit-ability of sensitive resource access being enabled. The running and execution of LCC specification for agent interaction is supported by the OpenKnowledge kernel.

6. CONCLUSIONS

The unique security issues involved in healthcare domains have been discussed in this paper. The practical solution of these security issues have been addressed to the needs of the HealthAgents project. We believe a sustainable security solution should be provided in accordance with Software Engineering principles, and conform to ethical regulations for healthcare data, as well as fulfil the security requirements usually raised from distributed Decision Support System (d-DSS) due to the nature of clinical settings.

Our work in these directions includes the design and development of security architecture in three levels. Various Software Engineering techniques are employed. We have developed or are in the process of developing in our system the secure communication to enable the protection of data transmission; authentication for user identity recognition; authorisation for fully customised resource access control. Using a security policy rule scheme and applying it in the interaction model for the HealthAgents MAS, we separate the functional and non-functional (security) requirements but let security policies integrate into the running of the agents in a distributed network via a unified role notion. Security policies enable easy and separate maintenance tasks across centres since they can be independently defined and maintained in each individual site but their application is yet under a unified access control scheme for resources with diverse types and locations. These make our security model adaptive. When a new hospital joins, new policy sets can be defined locally by the hospital managers. When its resources are required from other sites these policies will be applied by responsible manager agents residing in that site uniformly, conforming to the regulations set in that site. When its users require the access to resources from other sites, the external policies will be applied in the same manner where users and their assigned roles determine their access privileges. Once any policy rule is changed the effect is immediate to all roles or individuals associated with the rule. Policies are automatically deployed and immediately available, requiring no coding and the minimum administrative overhead. The implementation work to fully achieve these goals is going on in our HealthAgents and OpenKnowledge projects. The work so far has established the basis for providing a comprehensive security model for distributed healthcare systems.

Acknowledgements

This work is supported under the HealthAgents and OpenKnowledge STREP projects funded by EU Framework 6 under Grants: IST-FP6-027214 and IST-FP6-027253.

7. REFERENCES

[1] The Institute of Electrical and Electronics Engineers. 1998. IEEE recommended practice for software requirements specifications. IEEE Std 830-1998.

[2] Bourque, P., Dupuis, R., Abran, A., Moore, J.W. and Tripp, L.L. 2005. Guide to the Software Engineering Body of Knowledge: 2004 Edition – SWEBOK. IEEE Computer Society.

[3] Gritzalis, D. and Lambrinoudakis, C. 2004. A security architecture for interconnecting health information systems. International Journal of Medical Informatics. 73, 3, 305-309. Elsevier.

[4] Choe, J. and Yoo, S. Web-based secure access from multiple patient repositories. International Journal of Medical Informatics. Elsevier. In press.

[5] Keese, J. and Motzo, L. 2005. Pro-active approach to malware for healthcare information and imaging systems. International Congress Series. 1281, 943-947. Elsevier.

[6] Szolovits, P. and Pauker, S.G. 1993. Categorical and Probabilistic Reasoning in Medicine Revisited. Artificial Intelligence. 59, 167-180.

[7] Coiera, E. 1994. Question the Assumptions. Knowledge and Decisions in Health Telematics - The Next Decade. 61-66. IOS Press.

[8] Kirn, S., Heine, C., Herrler, R. and Krempels, K.H. 2004. Agent.Hospital: A Framework for Clinical Applications in Agentcities. In Applications of Software Agent Technology in the Health Care Domain. 67-85. Birkhauser.

[9] Xiao, L., Peet, A., Lewis, P., Dashmapatra, S., Sáez, C., Croitoru, M., Vicente, J., Gonzalez-Velez, H. and Lluch i Ariet, M. 2007. An Adaptive Security Model for Multi-agent Systems and Application to a Clinical Trials Environment. In Proceedings of the 31st IEEE Annual International Computer Software and Applications Conference (COMPSAC'07) Volume II. 261-266. IEEE Computer Society.

[10] HealthAgents. http://www.healthagents.net/.

[11] Sandhu, R.S., Coyne, E.J., Feinstein, H.L. and Youman, C.E. 1996. Role-Based Access Control Models. Computer. 29, 2, 38-47. IEEE Computer Society Press.

[12] Robertson, D. 2005. A lightweight coordination calculus for agent systems. 183-197. LNCS 3476. Springer.

[13] Robertson, D. *et al.* 2006. Open Knowledge: Semantic Webs Through Peer-to-Peer Interaction. OpenKnowledge Manifesto. http://www.openk.org/.