

# Role Based Access Control for a CSCW Environment in the Healthcare Domain

Student Name: Michael Peacock

Supervisor Name: Prof. David Budgen

Submitted as part of the degree of Software Engineering to the  
Board of Examiners in the Department of Computer Science, Durham University.

## *Abstract*

**Context / Background:** In the health-care industry case meetings commonly involve sharing information between colleagues. Traditionally this is done by sharing physical documents and notes, but is problematic because it relies entirely at the discretion of the document “owners” when their decisions may not always be appropriate.

**Aims:** To create a role-based access model for a Computer Supported Collaborative Workspace which allows team members to share case notes and keep an audit trail of such collaborations. This role-based access model is sophisticated enough to allow dynamic permissions based on a users role within teams they are members of as well as other factors.

**Method:** Using the Model-View-Controller architecture I developed a model which supports these dynamic permissions fully, with this architecture the permissions system is controlled by a model making the system easy to improve or restructure. An expert review of the system was conducted using experts in the healthcare domain.

**Results:** The results showed that the system was intuitive, simple to use and had potential for further applications, even applications which were not the intended purpose, but still within the healthcare domain.

**Conclusions:** Role-Based access control in a CSCW environment can greatly benefit the healthcare industry particularly with respect to case meetings. Guidance would need to be provided, by the industry, in terms of when and what to share as the application allows them to easily cooperate, it is often difficult for some users to decide how and when to use the collaboration options. Further feedback from a wider range of potential users should be obtained and analysed so that the system can be improved.

**Keywords** – role based access control, health care, permissions, mvc, computer supported collaborative working, dynamic, information control.

## I. INTRODUCTION

Within the healthcare industry, meetings are often held involving teams or members from numerous teams to discuss a particular case, called **case meetings**. Typically case meetings are attended by members of different autonomous agencies, such as social workers, medical practitioners, clinicians, general practitioners, financial support services, child services and even teachers, counsellors and parole officers, depending on the context of the meeting.

Hardstone, et al. looked at one such example of these teams: Community Mental Health Teams, which were set up to provide community mental health services involving psychiatric nurses, social workers and other relevant healthcare professionals (2004).

These teams introduced a number of concerns such as problems with accountability, lack of managerial clarity, cross organization hierarchical differences and unclear roles. These concerns are often believed, to some extent, to have been caused by the inability to effectively share information. It is difficult to share this information, because it is unclear amongst team members, who should be permitted to see certain information because they may be from different organizations or departments and thus other team members may not be aware of how the access control policies from the two organisations should be combined.

Computer based Multi-organization multi-team collaboration requires an advanced method to represent the roles and permissions available to the collaborators. Factors which need to be effectively taken into account include:

- An individual's job.
- Various teams an individual is a member of.
- Organizations an individual works for
- Delegated permissions, where a team member may grant an individual access to a document or where an individual is acting on behalf of another individual such as vacation cover.
- Time based permissions; certain clinicians may have additional levels of authority at specific times, such as being in charge of a ward one evening a week.
- Specific permissions related to individual documents or meetings.

This is especially important in the Health Care context, where certain information about individuals may be confidential, such as notes from social care workers or counsellors.

In order to investigate potential solutions to this problem, I have developed a Computer Supported Collaborative Working environment, suitable for use in the Health Care industry which implements Role Based Access Control as a means of providing dynamic access to data and system operations for those whose roles allow. Documents and data are stored in a database linked to the system and an audit trail is kept of user actions.

Since the participants may be from different organizations, and have different roles and levels of responsibility given to them, they may not all have the authority to view or share certain types of information.

Despite the difficulties in a computer based system, the only alternative to a computer based system is paper based, which is very problematic, particularly within a meeting scenario. Paper based document sharing requires documents to be manually copied, and requires no accountability for the transfer of information from one user to another. Secondly paper documents can, more easily, be lost, stolen or seen by unauthorised users.

## **II. RELATED WORK**

### **A. CSCW**

In 1984, to describe a common research interest in understanding how technology could assist with the way people worked; Cashman and Grief coined the term Computer Supported Co-operative Work (Grudin, 1994). This research area covers a range of methods across various

disciplines to investigate the use of technology and computers to support and enhance collaborative working, ranging from simple collaborative tools such as wiki based systems, to pure CSCW systems, such as the open source BSCW.

### ***B. IBHIS Broker***

IBHIS is an Integration Broker for Heterogeneous Information Sources, which was developed to help draw together information from a number of potentially distributed autonomous organizations, which is a major challenge particularly to the healthcare industry and healthcare teams (Kotsiopoulos et al, 2003).

By being a service oriented system, the broker can collect information from various sources within the healthcare domain, such as social services, medical records, and mental health team records is fed in to the broker and delivered to the end user from a single source.

The IBHIS broker's limitation lies with its access control. Because the information stored from the system is from various autonomous agencies, the broker only filters the information based on the users' organisation and their access rights within this organisation. A protection proxy is utilised by the broker to filter the entire set of information obtained, down to a subset of information relevant to the end user.

Budgen, Rigby, and Brereton discussed that this proxy would need to be extended with the access permissions associated with the individuals and teams involved in a particular case meeting, and that recording the entire permission sets should be maintained as a single record, as opposed to distributed across the various information sources (2006).

This requirement of maintaining a single record for the permission sets in a single location could be suitably implemented using some form of role based access control.

### ***C. RBAC***

Role based access control is a method of restricting and managing access to a computer system based on a users role. Herein lies the problem however, and that is determining exactly what is meant by a role.

Zang, Oh and Sandu believe that the notion of a role is an organizational concept (2003) and that there are generally "business rules" in many large organizations which relate to access control policy.

These business roles obviously play an important role in conceptualizing specific roles from within an organization, but the complexities involved in some of these business roles, can still make it difficult. Within the healthcare industry, we have both simplistic business roles which can easily be conceptualized into a role; there are also more difficult situations.

Between certain levels of health care professionals, such as a general nurse being different to a psychiatric nurse, we have a clear distinction of roles this distinction becomes less clear with roles such as nurses with prescribing powers, or health visitors who may need access to medical records of family members too (Bennett, K., Rigby, M. and Budgen, D 2006).

### ***D. Mechanisms for CSCW***

There are a number of mechanisms which enable and support computer supported cooperative working, including table top display systems, and the BSCW application.

Real time collaboration with multiple users in the same location can be facilitated by the use of table top display systems, and while this requires participants to be in the same

location, it provides an alternative look upon collaboration within a meeting and document sharing scenario.

Table top display systems such as the Lumisight Table can provide different images to the different users around the table (Kakehi et al, 2005), as well as allowing each of the users to perform tasks and control the display via touch and physical objects interacting with the display.

The Basic Support for Cooperative Work project is suite of web based tools supporting cooperative work, providing shared workspaces to allow users to share files through small repositories and discuss topics in a threaded discussion system. In addition to storing files, it also stores links to documents stored elsewhere (Bentley et al 1997).

Multiple users can all interact with these systems simultaneously, and could easily pass virtual documents to other users, pull out relevant photographs or video for review and discuss medial cases using this form of system. Such close proximity interaction raises issues with data security, as it would be relatively easy for users to see documents which others are looking at – even if it is a document which they should not be allowed to see. This however is a limitation with any same-location solution.

### ***E. Alternative permission models***

As well as the role based access control permission model, there are a number of other permission models available.

UNIX implements its own permission model, which provides permissions for the file owner, the group associated with the file and users who are neither the owner nor a member of the group associated with it. Each of these three classifications have three permissions which are, read, write and execute, granting the right to read a file, write to a file and execute a file respectively. The permissions on a file are encapsulated into a string, which defines the type the file is, and the three permissions for each of the three classifications.

Service-enabled data access control is a particular access control method which for which deployment within a system of distributed data has been researched. In particular there is a proposed hybrid access control model, described as service-enabled data access control, to work alongside the IBHIS broker discussed earlier.

Turner, Brereton, and Budgen created the service-enabled data access control model with the concept of a role as the core of the model but combines features from various other permission models to attempt and meet the security needs of an integrated broker (2006).

While the UNIX permission model lacks flexibility, it provides adequate features for its intended purpose; the classification of “other” which it has is an interesting one, as it provides a set of permissions specifically for users who are not the owner or member of the group, of the file.

## **III. SOLUTION**

The solution was to create a proof-of-concept application, to demonstrate the benefits of using role based access control, particularly within the scenario of a case team meeting with members with varying roles, levels of authority, and from different autonomous agencies.

### A. Model-View-Controller

By utilizing an MVC style architecture, the elements of the system are easily separated into models, which are the logic of the application, views which are the user interface of the application and controllers which respond to user interactions with the views, such as a mouse click, and interfaces with the model to provide a response to the action (Thomas, D., and Hansson, DH, 2005).

This separation of entities ensures that the logic is independent from the user interface and from the controller, and can easily be interchanged with new logic as the needs of the application change. More rapid development is also possible with this approach, allowing for a permission model to be prototyped, and if not successful simply “thrown away” and a new model created in its place.

Furthermore, this allows multiple views to be created, such as a special version for users with visual impairments, a version with an optimized workflow for managerial staff or a version suitable for mobile devices so that information can easily be accessed from pocket PC’s (although such mobile access introduces concerns of its own). Views are also able to relate to specific roles, so when a user with the permission to chair a meeting logs in, the system may use a different set of views to a user with no permission to chair a meeting.

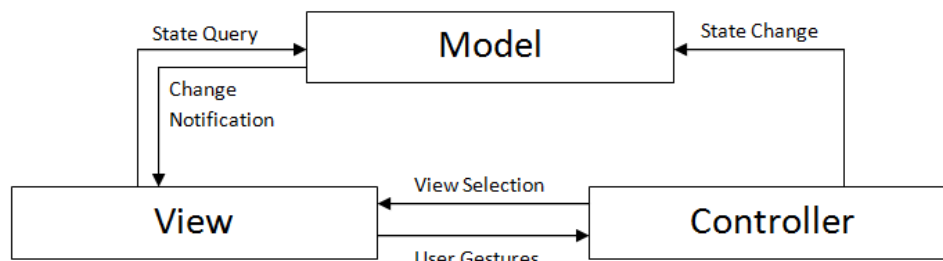


Figure 1. MVC Style Architecture

### B. Web based solution

Although written when dynamic user interaction with the web was in its infancy, Bentley, Hortsman, and Trevor investigated the World Wide Web as an enabling technology for CSCW and found it to be a suitable platform to enable cooperation (1996).

As this system is a multi-user system, with real time collaboration, it needs to be accessible to multiple users concurrently, potentially at multiple geographic locations, and thus it needs to be distributed.

With distributed systems, much effort can be spent on the distribution side of the implementation, and correcting the numerous problems which can occur as a result, to reduce this burden the system was developed as a web-based application, whereby distribution issues are mostly already handled by the web browser, the web server and the HTTP protocols. Specific advantages of utilising a web based solution include:

- Web browsers are built into most computers, including mobile devices, making it simple to access the system using almost any computer
- The system can be used outside of an organisations IT infrastructure
- HTML provides a simple, light weight user interface

- Support for interaction with data and file submissions

Internet based development provides a wide range of programming languages and technologies which can be used, including ASP.NET, PHP, Python, Ruby on Rails and ColdFusion, all of which offer their own benefits and complications. Of the available languages, PHP and Ruby on Rails were the main options under consideration because of PHP's widespread use and the fact that Ruby on Rails had been attracting more interest from commerce and academia.

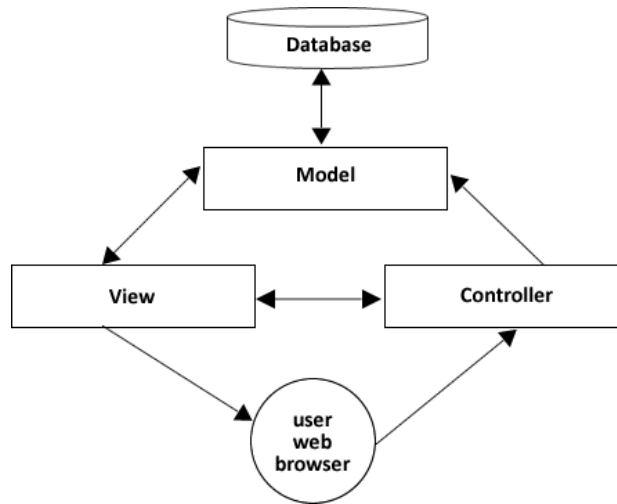
**TABLE 1.** COMPARISON OF PHP AND RUBY ON RAILS.

<b>Factors</b>	<b>PHP</b>	<b>Ruby on Rails</b>
MVC	Not out of the box	Out of the box
My personal experience	Several years	No experience with
Object Oriented	Yes	Yes
Database Connectivity	Yes	Yes
Suitable for web	Yes	Yes

The main difference between the two languages in terms of suitability for the project is primarily personal experience and MVC capabilities; Ruby on Rails is by nature an MVC framework for Ruby (Thomas, D., and Hansson, DH, 2005) where as PHP either would require an MVC framework or some custom development to give an MVC structure to the code. On the other hand, the use of Ruby on Rails introduces additional learning time and, despite its rapid development approach, a learning curve to learn the language as it is structurally and syntactically different from PHP.

Despite being a web based, distributed system it is intended that meetings which are held which utilize the system are held in real time with the aid of additional communication channels such as video conference or conference call systems, and that the system is used only as a facilitator of the meeting, and a guardian of the data. Further security would be needed over these communication channels, such as encryption and recording, in order to ensure that sensitive information is not shared audibly by team members – bypassing the permission model.

Due to the nature of web based applications, the MVC architecture of the system can be expanded to more accurately reflect how the system works, how the user interacts and how the data is stored.



**Figure 2.** web-based MVC application

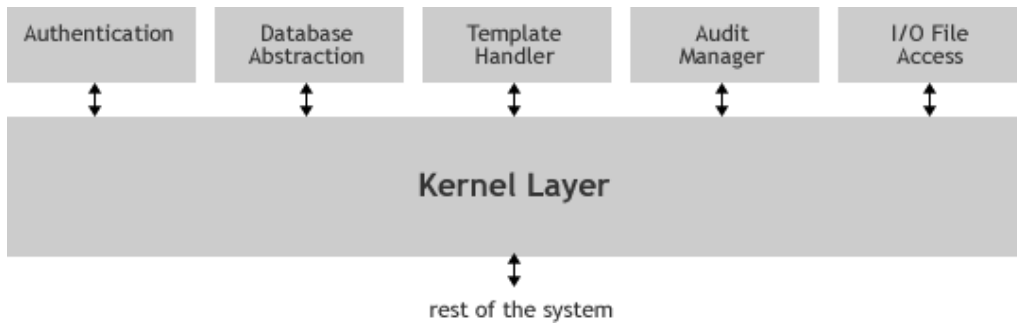
The data which is utilised by the application, such as documents, user details, patient records and roles is stored in a database; this database primarily interacts with the model. Views are composed of hyper text mark-up language, which is then rendered through the users' web browser, user input such as data entry or mouse clicks within the application are handled by the browser and then passed onto the controller which processes the user interaction. To some extent, the database itself acts as the model, which is why only a permission model is required; features such as document access require the permission model, as well as utilising the database to provide the document, and the controller to handle user interactions with the document - such as creating a document, opening a document or sharing a document.

### ***C. Kernel***

Although the Model-view-controller architecture is great for separating the logic, processing and user interface sections, it does not make adequate provisions for common tasks which all of the models or controllers need to perform, and so an additional layer was needed - a kernel. The kernel provides a single point of access to various core classes the system requires including:

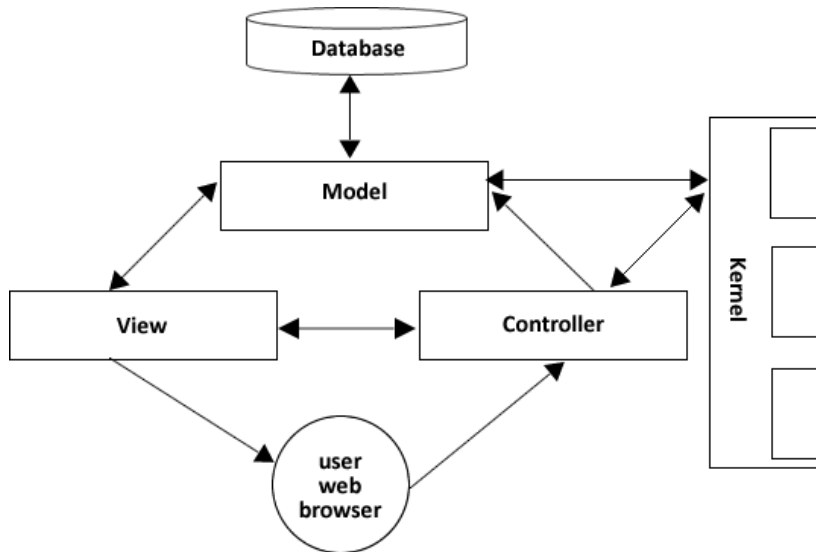
- Database Abstraction to provide functions for accessing the database, managing the data, and caching data where appropriate.
- Template Management to take the various views (or templates) and apply the information to them
- User Authentication - processing user login and logout requests, and ensuring that users are logged into the system.
- Audit Management - creating an audit log of operations and file views as well as creating a "human readable" version of log files for analysis.

The features within the kernel can all interact with each other, by using the kernel object as a communication backbone, this is important as the authentication component needs to access the database to lookup the users authentication credentials, similarly, the audit trail manager must store audit logs into the database.



**Figure 3.** The kernel layer, providing centralized access to common functionality

Both the model and the controller within the application have access to the kernel, this enabled the model to interact with the database and ensure a user is logged in successfully and subsequently obtain basic user credentials, such as username, and user id from the authentication handler and of course generate views to the user via the template handler. The controller can also interact with the database should it need to, it can process user logins using the authentication handler. Figure 4 below illustrates the combination of the kernel, with the MVC architecture of a web based application.



**Figure 4.** The kernel layer, combined with the MVC style for a web based application

#### ***D. Audit Trail***

With any computer based system involving authentication, access control and privileged information, a record of activities is essential, with dynamic access controls where information can be shared easily, or where access to restricted information can be overridden at the click of a button, a detailed log is imperative.

Accountability and justification are vital to the Audit trail, to prevent abuse of the sharing and override functions and safeguard the sensitive material held by the system. Certain actions, such as sharing a document with another user or submitting a document to a meeting require justification to be entered into the system before the operation will process. Some more urgent actions, such as requesting an override on a document, something which is intended for emergency use only this will process without justification, but will later request

justification from the user by inviting them to complete an Audit Log Form detailing the reasoning behind their actions. Managerial personnel can then monitor the audit trail and follow up on inconsistencies or potential policy abuse concerns.

If we take a child protection situation, Budgen, Rigby, Brereton, and Turner discuss situations where such an override would be required. Brokers, such as the IBHIS project provide individuals from different organisations some information on seemingly minor incidents, but this is shown in isolation. Bringing these smaller pieces of information together could show a different situation, such as a serious child abuse case (2007). Without team members being able to step back from their current task and reveal all relevant information to see a clearer picture, such problems can go un-noticed. Of course, on its own the override would not be effective; it would still require the skill of the team members to think there could be a wider picture to a particular situation.

One limitation of the implemented audit trail system is when an operation involves multiple users, for instance if a user was to share a document with another user, this would currently be stored as two entries in the audit trail, one as the user sharing the document, and another as a user accepting or rejecting the shared document. Table 2 below illustrates that only one related element is assigned to the log along with the user. Ideally this should all be encapsulated in a single log entry making it easier to trace activities.

**TABLE 2.** OUTLINE OF DATA STORED IN THE AUDIT TRAIL.

<b>Data stored</b>	<b>Description</b>
ID	A unique identifier for the log entry
User ID	The user performing the action which is being logged
Timestamp	The time the operation was performed
Content Type	The type of content in question, such as a document, meeting or patient
Operation	The operation performed, such as view, create, etc.
Related Element	Identifier of the document, meeting or patient in question (combined with the content type, to find the exact data)
Reasoning	The reason for a particular action, generally only stored for emergency overrides.

A minor scalability issue also exists within the Audit Trail system, in that lists of operations and actions that can be performed, and subsequently stored in the logs, are hard coded into the systems files. It would be better if this was abstracted to database entries, allowing the system to be expanded more easily and also to reduce dependence on variables in the code when storing database records and generating human readable audit trails from the database records.

### ***E. Singleton Pattern***

The introduction of a kernel into the system produces a potential problem if multiple instances of an object are instantiated, causing partial sets of information to be held within different instances, particularly problematic with respect to database access. To prevent this the singleton pattern is used within the Kernel object to ensure the sole instance of the class is a normal class, and such that only one instance can ever be created (Gamma et al 1994).

References to the kernel object are then passed to other objects within the system so they can access the kernel and share information within.

### ***F. Security***

With any software application, security is essential, with web based applications it is even more so, risks can be introduced by Internet browsers, data input, server software, network attacks and database engine attacks all of which have entry points, being part of a web based application.

Three main security considerations were taken into account during the development process: Protected methods, data sanitization, and short expiring time based sessions to control logins.

Protected methods provide a basic form of security from an unauthorized web application on the same server trying to access code and process data, although this is common practice with many programming languages, within PHP this has only been supported since version 5 which is only now starting to see widespread deployment across the Internet. Data passed into the system is sanitized either to ensure a value is an integer, to ensure that data is escaped preventing MySQL injection attacks on the server which could provide an unauthorized user access to privileged information or to a privileged user account, and to ensure there is no code in the data which could be presented to other users in an attempt to “hijack” their login to the system. The final main security consideration taken into account during the development process was short expiring time based sessions, ensuring that inactive users were logged out after a short period of activity, the reason for this is twofold, if a user leaves their computer for a short period of time they would need to login again – preventing another user using the system, and also because of the method user authentication is managed on web based systems.

Each time a page is reloaded or a button is clicked, the web browser is making a new connection with the server and needs to prove that the user is still logged in, the time based session acts as an invisible password which is sent on each request, and updated each time it is sent with a new time key.

Although commonly seen as standard practice with web applications, being incorporated by many commercial systems, it is important to point out how user credentials are kept secure. Passwords are passed through a one-way hashing algorithm, and the resulting hash is stored in the database as the user’s password. When a user then logs in, the password they submit is passed through the same algorithm and compared to the password in the database for that user.

## IV. RESULTS

Successful implementation resulted in an intuitive, web based application implementing role based access controls in a computer supported collaborative working environment, suitable for use in the healthcare domain.



**Figure 5.** screen shot of the applications front end

### A. *Permission Model*

On initialization, the permission model builds up a record of the users permissions and privileges, both in terms of system operations and tasks which they are permitted to perform, and also types of document they are allowed to view.

Documents are generally assigned to a patient, and the first stage is to discover which patients the user is permitted to have any form of access over, this allows the system to generate a partial list of documents that the user is then permitted to see. These documents, together with documents which may have been shared with a particular user are then made available to the user in a centralized document repository within the system. Although the user will be able to see all of these documents, they are not all accessible to the user, depending on the classification of the documents. Further permissions come into play to restrict the access to these documents, based upon the users' role.

Team Roles, individual roles, time based permissions and delegated roles are then gathered for the user and then compared to determine both the overall role of the user within a particular context or situation, and which permissions should take effect.

Permissions for document types and operations are generated through an iterative process: during the first iteration of all of the factors determining the users' role, the permission for that particular document or operation is then recorded, when the system iterates to the next permission for the same operation or document type, this is then assigned a numerical value so that it may be compared to the value of the current permission.

This is an important difference from other role-based access control systems, in which the various permissions required as assigned to a single role which the user is then assigned to, in this case the user can instead be assigned multiple roles, each reflecting their role in particular organizations, contexts or teams. Assigning only a single role to each user would prove extremely problematic in conceptualizing the users' role and encapsulating all of the appropriate permissions into that role.

Only four permission values are available to the system, which for the purposes of establishing this proof-of-concept solution, is sufficient, however if such a solution was implemented within the Health Care industry further comparison stages may be required, as may more complex comparisons. The permission values available are illustrated in the table below.

**TABLE 3. POSSIBLE PERMISSION VALUES**

Key	Description	Numerical Value
d	Disallow	1
a	Allow	2
ds	Disallow "Strict"	3
as	Allow "Strict"	4

The "Strict" allows and disallows were to allow flexibility to members of multiple teams or organizations, such that although all members of social teams would have allow access to social information on patients, there maybe one or two members who are from another organization or team which dictates that under no circumstances may they see that information, in which case they would be given the "disallow strict" permission to ensure that it would override the other teams allow permission.

Formally, the role based access control permission model can be represented by:

- $P = \{ o, v \mid o \in O, v \in V \}$  (permissions are a tuple containing an operation and the corresponding permission for the operation)
- $U_R \subseteq U \times R$  (user to role relation)
- $T_R \subseteq T \times R$  (team to role relation)
- $UT_R \subseteq U \times T$  (user to team relation)

Where  $U$  is a set of users,  $T$  is a set of teams,  $R$  is a set of roles,  $P$  is a set of permissions,  $O$  a set of operations and  $V$  a set of permission values.

## ***B. Meetings***

Case meetings with specialist teams, as discussed earlier, are an area which is a problem in the Health Care domain. The system created not only provides an implementation of Role Based Access Control, but also the capabilities to facilitate meetings with members of these teams.

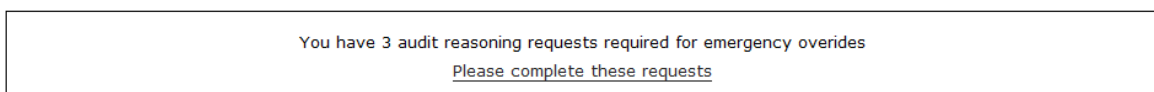
Both distributed team meetings and local team meetings can be held and facilitated using the system, and while the main focus was on distributed team meetings the system is just as suitable for local meetings, although there is one organisational difficulty with local meetings.

With local team meetings it would be relatively easy to circumvent accountability measures, in that a user could just turn their computer screen to another to show them a document. This could be solved by combining the system with CCTV as a protection against such circumvention attempts, in a similar way to distributed meetings would require the audio conversation to be recorded.

### ***C. Audit***

Each operation performed within the system, such as creating a document, creating a meeting, sharing a document and so on, is logged along with a relation to the user performing the action, any document or area in which they are performing this action and the time. As each user continues to use the system, an audit trail is built up detailing the users activities. Generally, this audit trail would only be checked if there was a specific reason to investigate specific users' actions, or to see who had view a particular document.

One exception to this is the emergency overrides which were discussed earlier. Should a situation arise where a user of the system requires access to a document, then they can override the permission model and gain access to the document. This creates a log in the audit trail with a flag to be completed. Until the user completes a reasoning request form, explaining their reasoning for overriding the permission model, a notice will be displayed at the top of each page reminding them to do so, as illustrated below in figure 6. Managerial personnel would also be informed of this, and could chase up the individual in question if required. It may be necessary to limit the number of overrides a user can perform without completing a request form, however this raises issues of its own: a user may, for whatever reason, have not completed reasoning requests for accessing information in a genuine emergency, and then find themselves needing to access another document but finding that they cannot, because they had exceeded a set limit, because they had been too busy to complete the request form.



**Figure 6.** Audit reasoning request notice for outstanding reasoning requests

## **V. EVALUATION**

There were three stages to the evaluation of the system, firstly a plan was established to outline the method of evaluation, then the evaluation was conducted and finally the results and feedback from conducting the evaluation were analysed.

### ***A. Evaluation Plan***

To ensure the effective evaluation of the system, an evaluation plan was created to outline the method of evaluation and the steps involved in producing the evaluation. Due to the nature of the system, and the fact that it is intended for use in a specific, specialised situation within a specific problem domain, users with knowledge of the domain, and in particular case team

meetings, will be able to provide the best possible feedback and evaluation, and determine if the concept has potential in the industry.

Obviously, evaluation by other users can generate good feedback and information, but this would lack detail in specific areas of improvement, potential for the industry and suitability for purpose. It can however, be useful for evaluating non-specific areas of the application.

Taking this into account, an expert review of the system appears to be the best method to evaluate it. Professor Barbara Kitchenham is widely viewed as an expert in software evaluation, and so, to ensure the evaluation of the system was planned correctly, a copy of the evaluation plan was shown to Professor Kitchenham and feedback from this consultation was taken into account. The main point raised by Professor Kitchenham was that although expert review is the most suitable method to evaluate the system, so long as the case meeting scenarios were reviewed as suitable by experts in the domain, then provided the system could cope with the meeting scenario with other users, the system would be a success.

Because of this, when creating the evaluation forms, I added a section relating to the credibility of the meeting scenarios used.

### ***B. Conducting the Evaluation***

Two scenario case team meetings were conducted with four users, including two expert users, distributed across two remote locations. Although the system was used to facilitate the meeting, VOIP software Skype was used to enable real time communication during the course of the meeting.

Each team member of the meeting assumed a different role, and was assigned different login credentials to use the system within the context of the scenario case meetings. Throughout the course of the meeting, feedback and comments were received orally from the participants involved. After the meeting feedback was obtained from a survey, which served two main purposes:

- To evaluate the effectiveness of the meeting scenarios themselves, as without a realistic meeting scenario, the comments received during the meeting would not reflect on the software in its intended use.
- To evaluate individual features of the system, their suitability for use and their potential for further application.

### ***C. Results from Evaluation***

The majority of the feedback from the evaluation was received orally during the course of the scenario meetings, the main points raising from this feedback was:

- Issues relating to the conclusion of a meeting
- The lack of staggering the users attending a particular meeting into stages, and
- The potential application within the context of rare diseases.

In order to conclude a meeting, the chairperson was required to enter notes on the meeting in a “conclusions” box, save those to the meeting and then click a stop meeting button to prevent the meeting from being active. As the concluding of a meeting is essentially a group

process, it was suggested that each member of the meeting should be shown the conclusions and be presented with the option to accept or dispute their contents. This would also enhance the accountability aspect of the system, as if medical actions are taken from the advice based on the meeting which was not appropriate, each member of the team would have had to give their consent for those recommendations or conclusions to have been saved. One other suggestion related to the conclusion of meetings, was enabling each user to see what the chair was writing as they were writing it. Since the system is a web based application, real time text updating could only really be done using asynchronous JavaScript and XML, however if this page was left open, the page would, in the background, continually be communicating with the server, effectively keeping the user logged in - even if the user had left the computer, which could be a security risk.

Meetings facilitated by the software allow all attendees access at the same time, which is intended for small case team meetings. Another suggestion was to expand the system to allow a "staged meeting". For instance, if a case meeting was being held for a child services patient, where staff from social services, the child's local General Practitioner, and staff from the child's school were required to attend, it may be appropriate for the meeting to be staged to allow certain users access only at specific times during the meeting.

Great potential in the software was observed; in particular some feedback indicated the software's potential application in a healthcare niche which was not previously considered, which is related to rare diseases. One of the expert evaluators, who is an expert in healthcare information systems, suggested the systems potential application for diagnosis and discussion of rare diseases.

Certain rare diseases can provide complications with common illnesses and ailments, and due to the rarity of these diseases the majority of General Practitioners will not have the knowledge required for the case. The system could be adapted such that the general practitioner had a case conference with an expert on the disease, utilising the software to store the expert's notes on the disease and various cases, and the general practitioners notes from the particular case at hand. If and when appropriate, certain documents can be shared and disclosed.

Although a lot of feedback was received, including a number of areas for improvement, this further illustrates the potential of the system. The fact that it was able to generate further ideas shows that it was engaging, and demonstrated the concept of dynamic, role based access control aiding a computer supported cooperative working environment, particularly suitable for case team meetings.

Results from the surveys completed by the expert reviewers show that the meeting scenarios were realistic (scoring 9 out of 10, with 10 being the most realistic) despite one of the scenarios being seen as difficult. The system was deemed to be easy to use, intuitive and with potential and suitability for use within the healthcare domain.

## **VI. CONCLUSION**

Permissions models, such as the UNIX permission model simply don't provide the flexibility offered by Role-Based Access Control. Williams and Graves note that it is "difficult to do more complicated access lists" with UNIX because you "cannot express that groups of users can read and other groups of users can write" (2002). While role based access control is not as complicated as the hybrid S-DAC model, or some of the other hybrid models, it does provide enough flexibility for the permission model itself, although if the environment was

service based, obtaining its information from a range of sources, further security protections would be beneficial.

The main issue with this kind of system is difficulties within the intended organization(s) in which they are implemented, in this case the healthcare domain. While it was shown to have potential, it leaves a lot up to the organisations in question to iron out more of the details for how it should be used, which is due to the fact this is a proof of concept system. Informality is taken away from case meetings by the introduction of enforced accountability, which leaves it up to the organisation to effectively determine, and train their staff, how and when documents should be shared or accessed.

Should such a system be utilised by the healthcare service, I believe a more service orientated approach would be of great benefit. This system set about using web services as a means of using and distributing itself to distributed teams, however if it was to also have service based interfaces which would allow it to interact with other applications and obtain its data from the various autonomous agencies who would use the system, then it could easily be linked into new organisations which may require integration, letting organisations using the system be interchangeable.

This project aimed to create a computer supported cooperative working environment which implements role based access control, in order to investigate the suitability of such a system within the healthcare domain.

Although only a proof of concept, the system not only shows that there is potential for such a system in this domain, it also shows there are many more applications available, and that combined with other technologies would be very effective in tackling healthcare case meeting cooperation issues.

Reflecting on related work, I've found that both the IBHIS project and the table top display mechanism are both systems which can greatly compliment a CSCW which implements role based access control. IBHIS provides a service oriented architecture which allows it to effectively broker information from various autonomous sources as outlined above, which would provide the CSCW with a powerful document repository. The table top displays, while not suitable for local meetings, would provide an intuitive interface for managing documents which would be suitable if each user of the system had their own table top display, although such a system would not be compatible with a web based system, but would be compatible with a role based access control system.

## REFERENCES

- Bennett, K., Rigby, M. and Budgen, D., "Role based access control – a solution with its own challenges", *IEEE-Proc.Softw.*, Vol 153, 1, Feb. 2006.
- Bentley, R., Appelt, W., Busbach, U., Hinrichs, E., Kerr, D., Sikkel, K., Trevor, J., and Woetzel, G., "*Basic Support for Cooperative Work on the World Wide Web*", *International Journal of Human Computer Studies: Special issue on Novel Applications of the WWW*, 2007.
- Bentley, R., Horseman, T., and Trevor, J., "The World Wide Web as Enabling Technology for CSCW: The case for BSCW", *Computer Supporter Collaborative Work*, 6(2-3): 111-134, Jun. 1997.
- Budgen, D. Rigby, M and Brereton, P, "Using an Information Broker in Healthcare: Issues for Cooperative Working", Unpublished.
- Budgen, D., Rigby, M., Brereton, P., and Turner, M., "*A Data Integration Broker for Healthcare Systems*", *IEE Computer*, 40(4), 34-41, Apr. 2007.
- Hardstone, G. Hartswood, M. Proctor, R. Slack, R. Voss, A and Rees, G, "Supporting Informality: Team Working and Integrated Care Records", *CSCW*, Nov. 2004.
- Gamma, E. Helm, R. Johnson, R. and Vlissides, J, *Design Patterns: Elements of Reusable Object-Orientated Software*. Addison Wesley. 1994.
- Grudin, J., "Computer Supported Cooperative work: history and focus", *IEE Computer*, 27 (5) pp19-26, 1994.
- Takehi, Y., Iida, M., Shirai, Y., Matsushita, M., and Ohguro, T., "Lumisight Table: An Interactive View-Dependent Tabletop Display", *IEEE Computer*, Feb. 2005.
- Kotsiopoulos, I., Keane, J., Turner, M., Layzell, P., and Zhu, F., (2003), "IBHIS: Integration broker for heterogeneous information sources", *Computer Software and Applications Conference*, 378-384.
- Thomas D and Hansson, DH, *Agile Web Development with Rails*, Pragmatic Bookshelf, 2005, pp
- Turner, M., Brereton, P., and Budgen, D., "*Service-enabled access control for distributed data*", *IEE Proc.-Softw*, 153(1), Feb. 2006.
- Williams, R., and Graves, R., "Limitations of UNIX permissions", 2002, <http://my.brandeis.edu/wp/display/132/385.wimpy>, accessed November 2007.
- Zhang, X., Oh, S., and Sandhu, R., "PBDM: A Flexible Delegation Model in RBAC", *Symposium of Access Control Models and Technologies*, 149-147, 2003.