Islamic University – Gaza
Deanery of Post Graduate Studies
Faculty of Information Technology
Information Technology Department

Enhanced Context Aware Role Based Access Control Framework for Pervasive Environment

Submitted By

Mohammed O. Al-akhras

Supervised By

Dr. Tawfiq S. Barhoom

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology

1435 هـ - 2014 م

Abstract

Mobile and network based applications usage becomes more wide. In such environments resources become more vulnerable for malicious access and illegal utilization. this encourages researchers to develop security mechanisms for protecting resources from illegal access and use. Security has two main parts firstly is authentication, secondly is authorization which presented mechanisms to access resources, such mechanisms used to control access and level of interaction between the end user and the systems which contain these resources based on specific criteria. The control of resources and its mechanisms presented researchers to new challenges, where plenty of information which could be used about or around the user, which described as contextual information. Utilization of contextual information is considered very useful for improving access decision making process against system resources to be more effective in service providing the large number of end users.

As consequence we selected a role based access control model. The selected model makes decisions based on context information sensed and collected from user environment. We implement prototype based on theoretical idea previously published then focus our study on enhancing context utilization and framework performance in practical approach, through studying the process of making decision based on context information validity. Permissions grant and revocation for users will be controlled upon how much context information sustain its value, monitoring will be conducted to the context information used within the predefine access control policy during permission grant or revocation, my research contribution focuses on enhancing the distribution and management process of context information over users by using the proxy, which works as guard to enforce policy for short term context information. In case any change breaks access control policy rules, then the proxy on user device will automatically send revocation/grant request based on change made for context information related to the user in his local environment.

Mainly research contribution depends on using proxy, which acts as an observer. The proxy pulls context information from a middleware data sources and logs any changes occur. The middleware works as a context information provider, the proxy will grab policy rules related to that end user, and will grab roles granted to the user during initial session request. If any change is made to the context related to the available policy rules, the proxy will evaluate it on user device and utilize available resources on the device. The proxy will make grant/revocation to permissions based on rules satisfaction, if any change made to the permission states will issue new grant/revocation access request to make the web service database up-to-date. Such enhancement will highly increase system response and enhance grant/revocation of permission to that end user.

الملخص

استخدام التطبيقات المرتبطة بالشبكات اصبح واسعا و منتشر، مع ذلك أيضا اصبحت المصادر اكثر عرضة للوصول و الاستغلال الخبيث، فهذا السبب كحافز شجع الباحثين على تطوير آليات لحماية هذه المصادر من الوصول و الاستخدام غير المشروع، وكذلك ايضا إن الوصول للمعلومات و المصادر يحتاج الى آليات للتحكم في الوصول و مستوى التفاعل بين المستخدم النهائي و الأنظمة و التي تحتوي على هذه المصادر بناءا على معاير محددة.

التحكم بالمصادر و ألياتها تضع الباحثين أمام تحديات جديدة، حيث يوجد اليوم وفرة في المعلومات التي من الممكن جمعها عن المستخدم و حول المستخدم، والتي تُوصف و تلخص على انها معلومات الاتساق او معلومات السياق، و التي يعتبر من المفيد جدا استغلالها لتحسين اتخاذ القرار للوصول للمصادر و بشكل فعال. كنتيجة لما سبق فقد قمنا باختيار نموذج نظري للتحكم في الوصول المبني على الوظائف او الأدوار، النموذج المذكور يقوم باتخاذ القرار بناء على معلومات من السياق يتم جسها و جمعها من بيئة المستخدم، لقد قمنا اولا ببناء و تطوير اطار عمل مبني على الفكرة النظرية المقدمة في بحث سابق ومن ثم التركيز في الدراسة على تحسين استغلال معلومات الاتساق و تحسين الأداء بطريقة عملية، من خلال دراسة عملية اتخاذ القرار بناء على صلاحية معلومات الاتساق. حيث يتم التحكم في آليات منح و سحب الصلاحيات من خلال دراسة حالة و قيمة كل معلومة سياق مستخدمة في قواعد و سياسة التحكم في الوصول ، مساهمتنا تتضمن تحسين توزيع و ادارة عملية جمع معلومات الاتساق لجميع المستخدمين من خلال الوسيط، والذي سيعمل كحارس يقوم بفرض السياسة بناءا على معلومات الاتساق قصيرة المدى، في حالة حدوث تغيير لمعلومات الاتساق سيقوم الوسيط بالتحقق ما إذا كان هذا التغيير يكسر قواعد سياسة التحكم في الوصول، حينها سيقوم الوسيط على جهاز المستخدم بارسال طلب سحب او منح بناءا على التغير الحاصل لمعلومات الاتساق ضمن بيئة المستخدم.

مساهمتنا تعتمد بشكل اساسي على استخدام الوسيط، حيث سيقوم بجلب السياسات المتعلقة بالمستخدم، كذلك سيقوم بجلب الأدوار الممنوحة له خلال عملية التقييم التي تتم لمرة واحدة فقط، وهي خلال طلب الجلسة لأول مرة، ومن ثم في حال حدوث أي تغير على معلومات السياق المرتبطة بالسياسة و قواعدها سيقوم الوسيط بتقييمها على الجهاز الخاص بالمستخدم مستغلا الموارد المتوفرة،ومن ثم سيقوم بمنح او سحب السماحيات منه بناء على تحققها ام لا، مما سيزيد من سرعة استجابة النظام بشكل كبير، من جهة و سيقوم بتحسين آليات منح و سحب السماحيات من المستخدم من جهة أخرى.

Dedication

To my kind parents and my wife,

To my kind uncles

To everyone give, only for giving

Acknowledgments

Day after day, I have increasing feeling that any personal achievement can't be made without help and support of a great people and beautiful minds, such people usually don't wait to take, they give only, to whom I really thank them for their endless giving.

Thanks of people is part of Allah thanks, I can't continue without saying thank you, firstly to my research supervisor Dr. Tawfiq S. Barhoom, really his invaluable help, his advice and guidance helps me a lot to complete my thesis as required. Also I would thank information technology staff along my study.

I would also like to thank my job supervisor Husam Diab and Muyassar El-amia for their support to me, really their open heart and patience and their morale support helps me a lot to continue. Also I express my gratitude to college of Dar Al Dawa and Humanities represented by Dr. Ali Y. Yaqoubi, during my tenure.

Really words can't express my gratitude to my brothers and my sister, the hope in their eyes always encourages me to continue, also thanks to my uncles especially Nabil and Akram for their continuous support and love. Also I would have to thank everyone supported me directly or indirectly.

Contents

List of Tables

List of Figures

List of Abbreviations

NIST	National Institute Standards & Technology
RBAC	Role Based Access Control
ACL	Access Control List
DAC	Discretionary Access Control
GRBAC	Generalized Role Based Access control
UA	User Assignment
STC	Short Term Context
LTC	Long Term Context
ACM	Access Control Manager
CAUAM	Context Aware User Assignment Manager
CAPAM	Context Aware Permission Assignment Manager
SCM	State Check Machine
PCE	Pervasive Computing Environment
XACML	Extensible Access Control Markup Language
SoD	Separation of Duties
SA	Session Agent
PEP	Policy Evaluation Point
PDP	Policy Decision Point
PIP	Policy Information Point
PAP	Policy Administration Point
ABAC	Attribute Based Access Control

Definitions:

Pervasive environment

An environment contains various computing devices with variant scale of computation power and abilities, which connected using network media like wireless or any other suitable media.

Ubiquitous environment

Synonym name for pervasive environment.

Context Information

1. Any piece of information about person or entity that changes in its value leads to a change in system behavior [33].
2. Any piece of information, the change of its value leads to a change in the behavior of a certain service being delivered, or its output representation to the end user

1 Introduction

Pervasive environment means that processing of information becomes integrated with everyday life activities, where computers involved in human life will not stop them to practice daily life unlike desktop computers which enforce users to stop any other daily life activity to use them.

The emergence of such environment which called pervasive computing or ubiquitous computing as coined by Mark Wesier [24]. Where surrounding ambience becomes smarter, our actions and existent become noticed and measured by computers and sensors which provide computer applications with information about us.

Such environment poses new concerns that should be taken into account, one of the major concerns is security where user resources in such environment is vulnerable for unauthorized access. Data and services confidentiality and integrity is now more under risk, also the opportunity to quickly join for unknown users to the networks become easy.

One of the accompanied concepts of pervasive computing is the use of context awareness. Emergence of context aware applications and services where an increasing demand for such applications.

Context aware applications utilize implicit information to adapt system or application behavior. Researchers produced several definitions for context, Bill Schilit [2] referred to as location, people interact with and objects and any change to these objects. Later Bill Schilit [3] added to the definition new parameters like network connectivity, communication costs, bandwidth, resources or social situation.

While aspects and parameters for context cannot be enumerated due to situation change, so as a consequence Gregory Abowd [1] defined context as follows: "any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant or the interaction between a user and an application, including the user and applications themselves".

As mentioned before security is a major concern, especially how to control access to resources and services which defined as access control mechanisms used to control user authorization.

One of the recently wide spread techniques is Role Based Access Control (RBAC), which conforms companies hierarchical structure makes it more desirable. Advancement in pervasive computing makes traditional RBAC [15] issued by National Institute Standards and Technology (NIST) has limitations, where an extension that take context information into account becomes an urgent need.

Many enhancement approaches are adopted, where their model design is based on:

1. Flat RBAC model for enhancement [7].
2. Constrained RBAC model for enhancement [6].
3. Non RBAC models: which build and implement access control based on mechanisms other than role based such as Access Control List (ACL) or Discretionary Access Control (DAC).

The different contributions provided and described don’t provide complete image for framework enhancement and development for pervasive computing. For example context information validity, which means how much the measurement of context information, will be compliant with real value, so the decision made based on the value of specific context information will be valid or not. So as a result of the change of context information value will lead to a change in permission state.

We will introduce enhanced framework which will combine some techniques from existing models and presented another, the framework will enhance permission revocation and restoration based on context information validity.

1.1 Problem statement

NIST RBAC model, didn’t take into account context information when making authorization decision. Currently available extensions based on context do not enhance revocation techniques that are convenient for pervasive environment.

1.2 Objectives

Here we will introduce the main objectives that we set to solve the problem we describe; also we split it into set of goals that covers all the work intended to be done in this area.

- **Main Objective:**

Improve context aware access control framework extends RBAC issued by NIST, especially context information utilization and validation, for pervasive environment.

- **Specific Objectives:**

1. Listing current context aware role based access control models convenient for pervasive environment.
2. Studying currently available context based RBAC frameworks architectures and design.

3. Selecting and adopting a discovery technique for context information convenient to pervasive environment framework.
4. Selecting representation for context information mechanism, to increase effectiveness, easy access and decision making in RBAC framework.
5. Choosing validity check of context information for dynamic permission grant and revocation for pervasive environment users.
6. Design extendable context sensitive role based access control architecture, compatible with pervasive environment.
7. Developing and implementing context aware role based access architecture based on design process.
8. Evaluate framework applicability and success to serve in pervasive environment from several perspectives mainly performance and security.

1.3 Importance of the thesis

The significance of our work yields from the increasing demand and adoption for ubiquitous computing, domains and environments day after a day become more aware of our actions and existence, such awareness provide large amount of unutilized information that helps application to change behavior to be more effective and efficient in providing services.

Security becomes more crucial concern, where services and resources become more vulnerable for unauthorized and illegal access; as a result new security mechanisms that aware of the environment and the existence of information needed to protect the resources and services which available over various networking and communication techniques.

One of the major aspects of non utilization of context information that could describe information related to the user, such information could help system administrator to enhance system ability to control access on the resources and services, in terms of existence of new information about users or their surrounding environment.

The enhanced framework will take into account providing context aware access control mechanism based on traditional RBAC models, which by its nature conforms companies' structure, so makes it more desirable.

Thesis gain significance for the following reasons:

- The enhanced framework will focus on improving the process of granting and revoking permissions to users based on context information conditions validity, if context information included in conditions changed that will affect the grant/revocation decision that previously given.

- Provides technical specification for designing and implementing context aware RBAC framework.
- Investigating enhancement steps for improving performance in order to improve responsiveness.
- Enhancing the validity of context information where involved in policy.

1.4 Scope and limitations

This section describes efforts will be covered, also describes efforts will not be addressed and will be out of my study scope.

• Scope

This section describes the efforts made and the tasks will be covered:

1. Introduce an extension framework for Role RBAC model issued by NIST.
2. The extended model in the framework handles access control mechanism, through utilizing context information censored from the environment or through context information providers.
3. Enhanced role based access control permission grant and revocation mechanism, to be more efficient and more responsive while context information values change.
4. Prototype of suggested Context aware Role Based Access Control framework, to prove theoretical enhancement guidelines made.
5. Testing the prototype of the framework in a pervasive environment, using different scenarios for such testing.

• Limitations

Here we describe research limitations, and what will be further work expectation than the research study met and covers:

1. The framework will not provide novel authentication mechanism, the authentication will be done based on previous work made in this field that fit pervasive environment restrictions.
2. The expected platforms for applying implementation and testing activities in my study for web service part are JDK tool kit, and android platform for mobile.
3. The framework will not take into account any environment other than pervasive computing environment.
4. We will not consider mechanisms and researches in authorization and controlling access, where context information and RBAC is not the center of the research.

1.5 Methodology

We will introduce the intended steps to be done in order to address our objective and goals, the methodology will group steps into three major tasks:

1.5.1 Research and Theoretical Approach

1. Conduct searching and analysis for current Role based access control models issued by NIST and finding out currently available extensions that support or utilize context information in their behavior and decision making.
2. Studying and analyzing selected models architecture and check its applicability and compliance with pervasive computing environment based on its aspects and restrictions.
3. Suggest enhancement alternatives for these limitations, and avoidance technique for expected failures, mainly in terms of performance and security.

1.5.2 Context information representation analysis

1. Study context information representation alternatives and mechanisms, and how each contributes in framework effectiveness.
2. Study how validation of context information could be achieved on different available frameworks and models of RBAC, and analyze its applicability for pervasive environment.
3. Selecting based on scoring sheet between representations of context information listed before.

1.5.3 Architecture design and development

1. Build architectural design that take into account enhancement guidelines and recommendations for the new context aware role based access control.
2. Finding out currently available and open source implementations modules for Role Based Access Control issued by NIST for reuse.
3. Selecting among the implemented frameworks based on object oriented commitment and used language.
4. Break down development and implementation process of context aware role based access control framework.
5. List and apply available testing mechanisms and procedures for such frameworks and such environment.
6. Design scenario based testing mechanism for most common and generic situations to prove design efficiency and effectiveness.

1.5.4 Evaluation and final justification

Evaluation will be considered based on three essential corners performance, security, and heterogeneity:

- **Performance:**
  1. Evaluate and statistically compare results.
  2. Use different devices with different computation power in simulated pervasive environment.

- **Security:**
  1. Simulating attacks situations and high load on the framework.
  2. Check applicability of Injection false context information.

1.5.5 Software Development Life Cycle

The software life cycle used in development of the framework is prototyping, reasons behind the selection:

- Prototyping reduce time to find defects in early stages of development.
- User interaction is increased, before implementation done.
- The requirement of the model is not well expressed and understood due to novelty of idea being developed, so the usage of high risk aware techniques will reduce time overhead and any other cost factor results from ambiguity.

1.6 Resources and Tools

Such research study needs mainly for various tools and resources to measure and evaluate results, so will list needed items:

- **Resources:**
  1. IUG Library.
  2. Supervisor.
  3. Internet Connection.
  4. High speed Computer.
  5. Other research groups working in the same area.

- **Tools:**
    1. Eclipse
    2. Open source Java compiler
    3. SDK tool

1.7 Structure of thesis

Thesis organization as the following, Chapter two will present an overview for the current research problems also will focus on providing detailed description of the current model being selected for enhancement and implementation, chapter three will study the related work, chapter four will explain the structure and main components of our enhancement made on model and improvement made on the framework, chapter five will address implementation issues and environment development selections, chapter six will evaluate the experiments made using client agent finally chapter seven will discuss and present conclusion and future work in this research area.

2 Theoretical Foundation

The context awareness gained popularity since 90s, where Active Badges System [28] which provides information about its holder location, the information stored in central database based on a network, which used by XEROX to route calls to the holder of badge through the nearest telephone device. Context awareness today found in many domains for example tourism, entertainment and commerce, for example tourism doamin needs location identification to adapt application of tourism behavior upon it. Also on shopping environments for example SmartSight Tourist Assistant [29].

Also context awareness embedded in project management environments [30] where system developed to react to changes of resources to manage tasks in offices. Also games exploited context awareness [31] in controlling by players move using hand gesture or any other part. Another last example is CityWide Performance mixed reality where users of the system [32] move around city where the 3D model events connected to physical city. As we see before that context awareness start to increase its existence in many life applications which in turn enhance service providing to the end user, very critical part of all applications is protecting resources and information based on wider range of security rules and preferences using different known and unknown context attributes and parameters that affect on system resources legal access.

As an integration for these systems which utilizes context information we try to develop and improve based on previous step using role based access control systems in order to manage access to computing devices resources.

2.1 Context aware Computing

New computing paradigms, where context information handling, usage, and management get more focus when building computing structures:

2.1.1 Context Definition:

As we have seen context in English language has the following meaning on the following dictionary list:

- Longman: suitable, events, or information that is related to something and help you to understand it.
- Oxford: circumstances that forms the setting for an event, statement or idea and in terms of which it can be fully understood.

The research in context aware role based access control for pervasive environment, includes multiple disciplines, mainly how context information acquisition could be made to provide the system to be able to make right access decision.

Also the research should take into account the restrictions of pervasive environment, where such framework designed to be adequate and operable in such important environment especially under the heavy orientation towards pervasive environment.

In the first section of chapter 2 we will describe the researching efforts done to enhance and improve research in each category independently of interdisciplinary title that we have.

In the second section we will describe the research achievements and enhancements on access control disciplines, and in the third section of chapter 2 we will describe RBAC model, in the fourth section we will address pervasive environments definitions and studying shortly the growing emergence of pervasive environment and analyzing restrictions related to such environment that affect application development. In the fifth section we will study XACML structure and its importance. Sixth section will provide a description to the CAP model. Finally section seventh provide a conclusion.

2.1.2 Context aware challenges

The main challenges needs to be overcome in order to enhance modeling of context aware role based access control. As we know that applications that utilizes context awareness features in its execution and service providing to the end user are complex and with special requirements in building, in comparison with others that not support context.

[33] Identifies requirements as follows

- Suitable context model describes relationships between different types and facilitate inference and abstraction of context information.
- Quality of information model is necessary to allow reasoning about quality parameters of each context type and values such as accuracy of location information.
- Suitable infrastructure for context information acquisition and management, where separating the process of acquisition from the process of consuming and use.

2.1.3 Context aware computing

The transition from standalone applications to pervasive environment yields more significance and demand on context awareness where hidden and previously neglected information became needed to make applications adaptable and accommodate its behavior and results upon such information, especially after sensors costs that measures such information become incredibly lower.

Schilit and Theimer [2] defines context as location, identities of nearby people, objects and changes to these objects, after that they add network connectivity, resources, cost of communication as well as user related information like user social situation, physical properties as lightening, noise and tempreture. Ward [34] express context as the state of the setting in which the application is operating. Schmited [35] define it as knowledge about users and information technology devices state including around, situation and location. [1] Express the important aspects of context cannot be enclosed where situations change between different applications. They introduce general formal definition:

"Any piece of information that can be used to characterize the situation of an entity. Where an entity is a person, place or object that is considered relevant or the interaction between users and application including the user and the application themselves" [1].

They considered context as the entire collection of entities and their properties that can enter into a meaningful relationship with users during interaction with applications including user and application themselves affect the behavior.

So context aware systems is defined by Anind Dey [33] as: "*A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task*"

2.1.4 Context Models

Context modeling considered the subject of many recent studies and researches, modeling of context information helps in making it handled by applications that utilize them in their business logic, where difference in modeling schemas for context information make it hard to exchange context information between different applications and services, So well defined context and common structure helps in exchanging such information to make systems more extensible and interoperable. [33] Provides infrastructure support building context aware applications.

2.1.5 Common Modeling structures:

1. Key-value pairs: store context information as data item linked to value [3] model location in such way. This approach considered simple where matching is very easy but complex information representations could not be created to its simplicity.
2. Logical approach: such representation of context information as expressions for example entity relationship, [36] introduced COBRA-ONT an ontology supports pervasive context aware systems expressed based on web ontology language (OWL) [37] using logical predicates for describing any properties.
3. Object Oriented: in this method context information handled as states of the objects, accessed using methods, Project TEA [38] an example, also Active Object Model [39] for location context, more modularity of context data, higher complex access.
4. Markup Schemas: using annotations with each context information in describing of what role values play, [40] Sticke-e note context information represented as tags and values as their fields, also SGML Standard Generalized Markup Language, also [38] introduced simple XML based protocol for exchanging contextual information.

2.1.6 Context Information Quality

While context information depends on real world entities and events, so such information could be invalid or not true due to change in the real world environment where these attributes and parameters are mapped to. Also sensors that measure this information could fail to any reason in measuring which leads that context information loses quality due to such incorrectness, inconsistency or incompleteness. As consequence context information quality model includes the following metrics [41]:

1. Freshness: indicates last update for context information.

2. Confidence: to which extent value is correct.

Anind Dey [33] describes quality in terms of the following:

1. Accuracy: which means is the information stored is still correct or match.

2. Coverage and resolution: sets of possible values for context attributes and change required for context attribute to change respectively.

3. Frequency: how often the information needs to be updated.

4. Reliability: to which extent the application tolerant to sensors failures.

5. Timeliness: the time the application allows between the actual context change and the notification to the application.

2.1.7 Context aware applications classification:

Researchers' taxonomies the features characteristics of context aware applications where two attempts made in this context:

[3] Introduced a classification based on two orthogonal factors

- Identifies whether the task is to get information or execute.
- Determines whether the task is executed manually or automatically.

[3] Classification of context application features is:

- Proximate selection: means applications that retrieve information for the user manually based on context information available.
- Automatic contextual reconfiguration: means applications that retrieve information for the users automatically based on available context.
- Contextual command: applications that execute commands for the user manually based on available context.
- Context-triggered actions: applications that execute commands for the user automatically based on available context information.

Pascoe [40] features classification

- Contextual sensing: means the ability to detect context information, similar in category as proximate selection.
- Contextual adaption: the ability to execute and modify a service automatically based on context, which linked directly to Schilit's context-triggered actions.
- Contextual resource discovery: allows context aware applications to locate resources and services relevant to user's context, mapped to automatic contextual reconfiguration.
- Contextual augmentation: the ability to link data with user context.

Anind Dey commented on Pasco and Schilit classifications, that both of them exploit resources to the user’s context, the ability to execute commands based on context and the ability to display relevant information to the user. Pascoe adds also the ability to display also context and not just information for further user selection, like showing a user location in addition to the list of printers and making users able to select. As criticism Pascoe’s taxonomy doesn’t support the presentation of commands relevant to user’s context (contextual commands in Schilit taxonomy).

[33] Proposed a new category combines the ideas from these two taxonomies and takes into account the three major differences:

- Presentation of information and services to a user.
- Automatic execution of a service.
- Tagging of context to information for later retrieval.

As we see previously that Pascoe added the ability to associate digital data with user’s context.

[33] Considered an application to be context-aware if it uses contextual information to provide relevant information and services to the user.

2.1.8 Context Usage Difficulties

The properties that context has which leads to difficulties in usability [48]:

1. Context information acquisition done by non-traditional devices like Global Positioning Systems (GPS), or using image processing techniques to detect users.
2. Context information acquisition could have a range of uncertainty like detection using image processing techniques.
3. Context aware application functionality, based on specific related process or situation, which makes even general functions hardly to use without modification.
4. Context information should be converted to fit higher level of use where GPS coordinates will not make sense as returning the street name or nearest buildings, so such information needs more processing and abstraction.
5. Context information change is frequent due to its relevancy to real time environments, so it should be up-to-date.

2.1.9 Middleware measuring context information

To facilitate the development of context aware applications and foster the usability in such emerging field, also such middleware decrease development overhead, [33] listed the important features that each context middleware should support.

1. Context specification: the middleware should include clear description for the context information being sensed, and how such specific context information could make system change its behavior.

2. Discovery: the process of identifying sensors and sources for such context information sources, like using APIs or any other metric to measure such information.

3. Context acquisition: mechanism to notify the application consumes context information value should be located.

4. Context Interpretation: needs a mechanism to change and interpret low level context information such as location coordinates to be readable by end users, like linking coordinates to roads, building and any more readable information, in short should be multi-layer of abstraction.

5. Context storage: presenting a way to represent such context information in specific format that facilitate its retrieval and usage for further processing such comparison and history change tracking.

6. Transparent distributed communications: mechanisms to synchronize multi-sources for context information, to prevent any interference and contradiction. And such sources should be transparent for applications.

7. Availability: context information should be available for further use, where some of them have totally independent providers.

2.2 Access Control and Role Based Access Control (RBAC)

[42] “access control constraints what a user can do directly, as well as what programs executing on behalf of the users are allowed to do, in this way access control seeks to prevent activity that could lead to breach of security”. This chapter presents overview of access control concepts and techniques developed, also the motivation in access control, categories and types of access controls, also explain role based access control (RBAC) development steps until its final architecture, discussing the limitations of RBAC, finally the relevance of access control discipline to the thesis research.

2.2.1 Concepts

We need to list the main concepts that related directly to access control process especially and to this research domain in general:

1. Object: is an entity which itself is a piece of information, for example it could be database record, file, network node, processor, video device and many others.
2. Subject: an entity that change the systems status due to an action initiated by.
3. Operation: active process invoked by the subject.
4. Permission: authorization to perform an action on the system, usually combination of object and operation.
5. Access Control List: list associated with an object that specifies who are able to access its information from the set of subjects.
6. Separation of Duty (SOD): no enough number of permissions for system misuse by subjects, such separation enforced dynamically or statically.
7. Static SOD: this done by defining conflict roles, where can't be done by one user.
8. Dynamic SOD: such constraints prevents applying conflicted permissions during the session.
9. Policies: defines and represents the requirements that specify how objects should be accessed, by whom and when.
10. Access Control models: handles user access requests, and apply a mechanism to enforce the policy issued be vendor or system administration.

2.2.2 Role Based Access Control Model Evolution

One of the major flexible and efficient [5] access control models proposed is Role Based Access Control – RBAC, RBAC evolution done over several stages and levels, the first version issued was named flat RBAC, then issued Hierarchical RBAC and constrained RBAC.

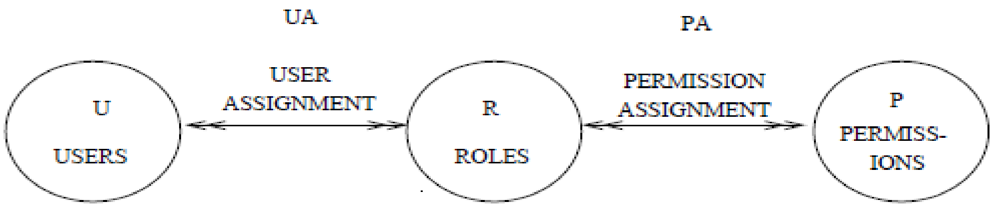


Figure 2-1 Flat RBAC [15]

Figure 2-1 represents the parts of the simple RBAC or flat RBAC which includes the following:

Roles: represents the organization structured functions and jobs.

Permissions: Are the actions and rights on a specific objects or a resource.

Users: to gain access for any object should be assigned roles, where each role has some permission based on the role mapped function.

User Assignment (UA): the process of assigning role set for each a user.

Permission Assignment: the process of assigning permission set for each a role.

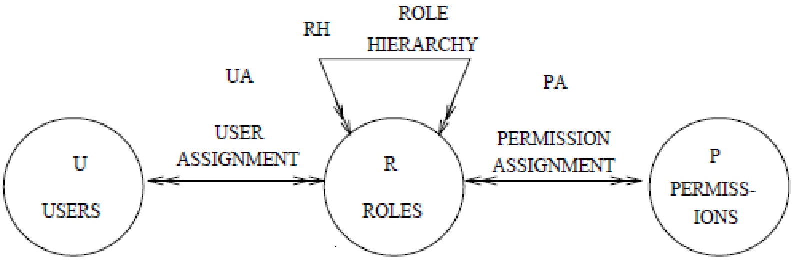


Figure 2-2 Hierarichal RBAC [15]

In hierarchical RBAC model as seen in Figure 2-2, the enhancement made over flat RBAC is roles represented in hierarchy form.

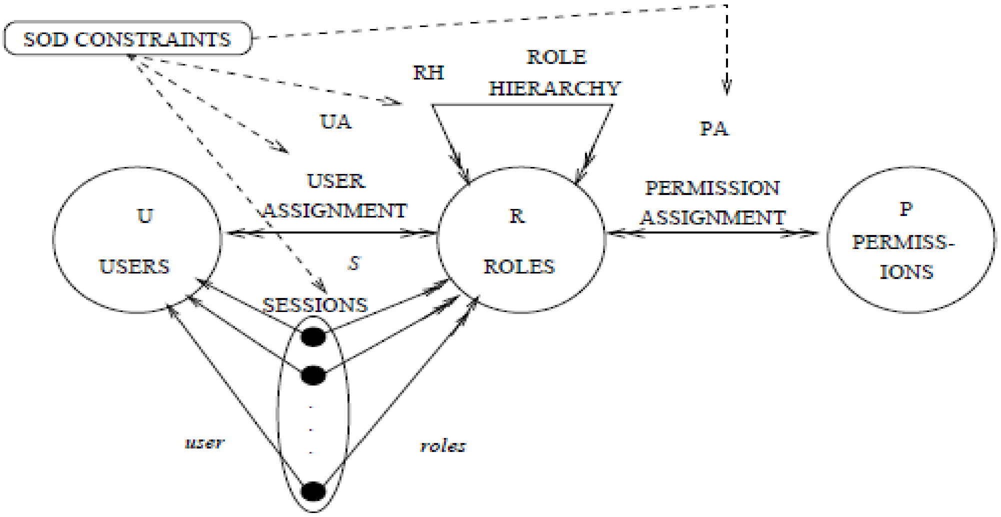


Figure 2-3 Constrained RBAC [15]

Constrained RBAC as shown in Figure 2-3, introduced the constraint principles where the user assignment, permission assignment, sessions and role hierarchy could be controlled and restricted using these conditions, which basically established for separation of duties.

Also the increasing existence of pervasive environment make searching for access control that utilize context information in making decision and dynamically changed upon context situations.

While this area which combine several disciplines and researching areas become more challenging, where researchers and developers introduced several models and frameworks related contextual nature as an extension to role based access controls since its emergence as a standard issued by NIST [15].

2.2.3 Policies

Categorized into two main types are:

Discretionary Access Control (DAC): in this type of access control permissions left to object owner or any user has authorization to control access on this object. DAC has drawbacks:

- Information can be copied from one object to another, without owner permission.
- No restriction apply on usage of information when user has received it
- The privileges to access object granted by object owner and not a system-wide policy reflects organization policy.

Non-Discretionary Access Control: means that the access rules not established based on the discretion of the user, but through administrative action.

1. Mandatory Access Control: means that access decision done be a centralized authority and owner can’t change these access rights, for example we have a user with Secret label classification should not be allowed to read a file with a label of Top Secret.

2. Role-based access control: access decision made based on roles assigned to users for example as part of an organization, where access permissions to objects and resources are linked to the role the user is being granted. Such management is very effective for enforcing enterprise-specific polices. We will study RBAC with more details later in section 2.3.

3. Temporal constraints: formal access statements that involved time-based constraints to access resources and objects, which required for limiting access based on time frame. For example in the bank employee should possess Teller role only during bank working hours therefore time constraint is required for such cases.

2.2.4 Access Control Techniques Capabilities and Limitations:

We will describe the capabilities and limitations for most common techniques.

2.2.4.1 Access Control List (ACL):

Considered most common and straightforward type used under discretionary Access Control (DAC) category, associates an operation on object to a list of subjects who can access the object.

- Capabilities:
  1. Fast mechanism.
  2. Easy to implement.
- Limitations:
  1. Linking users to privileges and permissions inappropriate, where usually users change, and their duties could change over time.
  2. Large amount of objects and resources make it hard to manage.

2.2.4.2 Role Based Access Control:

- Capabilities:
  1. Fully compatible with structured administrative domains, where permissions distributed based on duties and roles.
  2. Easy to manage, event with numerous amount of elements such as users and objects.

1. Limitations: Implementation complexity, in comparison to ACL.

2.3 Role Based Access Control:

2.3.1 Description:

RBAC is a non-discretionary access control mechanisms, it doesn't depend on individuals and object to match permissions, but it depends heavily on domain functionality represented in a specific job description, as we know always users change and the roles stay, so each user to get access to a resource should be assigned to a role that match his real job.

2.3.2 Organization

RBAC Model has core components and constraining components:

- Core components are:
    - User: usually is the person intended to benefit and interact with the system.
    - Roles: represents specific jobs within the system being implemented.
    - Operations: represents the actions and processing done over the service or any resource, called in general object.
    - Objects: represents any resource or service.
    - User Assignments: represents a mapping function that link users to roles.
    - Session: represents the interaction hold between the model and the user within a continuous period of time.
- Constraint Components are:
    - Role Hierarchies.
    - Separation of duties.
        - Static
        - Dynamic

2.3.3 Levels

RBAC Levels represents the complexity steps added to RBAC model from its first occurrence.

Table 2-1 : RBAC Model Levels

Model Level	Hierarchies	Constraints
RBAC0	No	No
RBAC1	Yes	No
RBAC2	No	Yes
RBAC3	Yes	Yes

2.3.4 Functional Specification

Represents the main operations the model has categorized as follows:

- Administrative Operations: for elements defined in RBAC previously, like user element, role, constraint and so on.
    - Create
    - Delete
    - Maintain
- System Level Functions:
    - Create session
    - Activate role / deactivate role.
    - Access decision enforcement.

2.4 Pervasive Computing Environment (PCE)

Pervasive environment computing become more involved in daily life, where such computing paradigms enhance utilization of information sensing, as a consequence enhancement to the services provided by computers to the end users, in terms of such significance we need to study and analyze such discipline and its requirements to operate as well as the challenges that limits its evolution and higher usage and demand.

2.4.1 Definition

Pervasive computing gains the last decade very high focus due to evolution in hardware, software, and networking. The change of the life style where devices and systems pervade most of our life behavior leads to merge technology usage with everyday life, more than twenty years ago Mark Weiser in 1991 said: "*The most profound technologies* *are those that disappear, they weave themselves into the fabric of everyday life until they are indistinguishable from it.* ”[27]

2.4.2 Pervasive Computing requirements

Any pervasive environment depends mainly to operate on the following components [27]:

1- Devices

Since advent of personal computing in 70’s, the devices spread and its numbers every year multiplies to cover all over the world. Such devices collect large amount of data about its users using sensors.

2- Networking

The communication infrastructure today expanded; also advance in technologies to transfer large data becomes available and meats the anticipated demand.

3- Middleware

Any computing environment in order to operate needs middleware, middleware will be as interface between network and application used by end user.

4- Applications

Applications will utilize the middleware capabilities to implement and realize its advantages in different life domains, such as education, healthcare, transportation, access control and authorization or any domain in our life.

2.4.3 Challenges and Research approach

Pervasive computing environment requires certain factors to be realized [27]:

1- Heterogeneity

The pervasive computing environment must bridge the gap in difference of user’s experience, and environments readiness.

2- Scalability

Due to proliferation of devices manufacturers, the need to write application once and run every where is required; such value will solve such problem.

3- Interoperability

The components of pervasive computing already available needs to be integrated with each other, components cooperation leads for better service providing.

4- Invisibility

The system that decreases the amount or the need of interaction increases its invisibility, so system ability to auto-configure and self-healing, leads to invisibility which is important to weave any service to be indistinguishable.

2.4.4 Research relevancy

The improved context aware framework designed to operate in a pervasive environment, so the need to analyze and study the concept and challenges that surrounds this field is very important to suggest alternatives and solution to our problem.

2.5 XACML: Extensible Access Control Markup Language

XACML refer to eXtensible Access Control Markup Language. Standard access control policy language implemented in XML, describe how to access requests according to the rules defined in policies based on business logic could be evaluated.

The XACML model supports and encourages the separation of the access decision from the point of use. XACML promote a common terminology and interoperability between access control implementations done by multiple vendors. So XACML is primarily an Attribute Based Access Control system (ABAC), where attributes associated with a user or action or resources are inputs into the decision. Role-based access control (RBAC) can also be implemented in XACML as a specialization of ABAC.

XACML shows how we can provides a representation for access request and how it should be formed, also provides with a representation for the policy structure, where XACML could be used as a standard mechanism within our framework or in future steps.

XACML has two models are language model and data-flow model, Figure 2-4 explains the mechanism of enforcing policy against requests made to get decision as output:

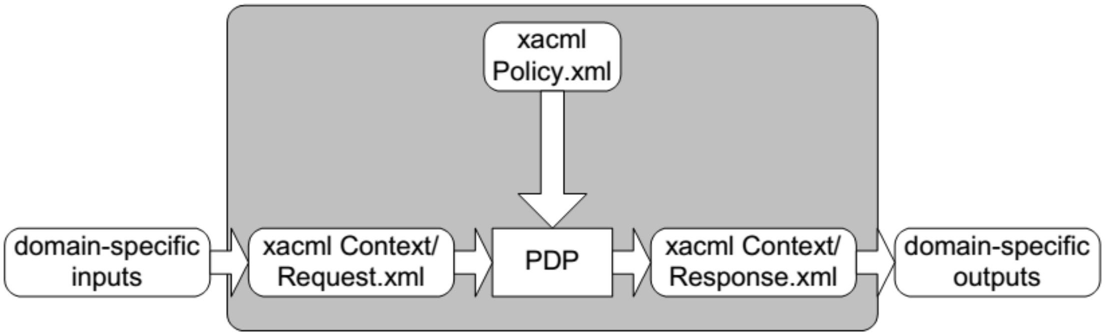


Figure 2-4 XACML Flow Scenario [49]

2.5.1 Language Model:

Describes the main components that represents the language:

2.5.1.1 Model Components

XACML is structured into three main levels of elements:

- **Rule**: an elementary unit of the policy, the rule evaluated on basis of its contents, where rule parts as follows:

  1. Target: sets of requests to which the rule is intended to apply, which could be a resource or subject.
  2. Effect: indicates intended consequence of true evaluation, for the rule “Permit” or “Deny”.
  3. Condition: a Boolean expression that refines the applicability of the rule beyond the predicates implied by the target.
  4. Obligation: expressions added by the writer of policy to the rule.
  5. Advice: expressions added to the rule such expressions could ignore safely by the PEP.

- **Policy**: rules are not exchanged among system entities, where PAP combines rules in a policy, which combines for main components[49]:

  1. Target.
  2. Rule-combining algorithm-identifier.

3. Set of rules.
4. Obligation.
5. Advice.

- **Policy set**: A Policy can contain any number of Rule elements, components are[49]:
    1. Target.
    2. Policy-combining algorithm-identifier.
    3. Set of policies.
    4. Obligation.
    5. Advice.

2.5.2 Dataflow model

Represents the steps of applying and evaluating access request, through standardized components to make common behavior for development and introducing access control mechanism as seen in Figure 2-5:

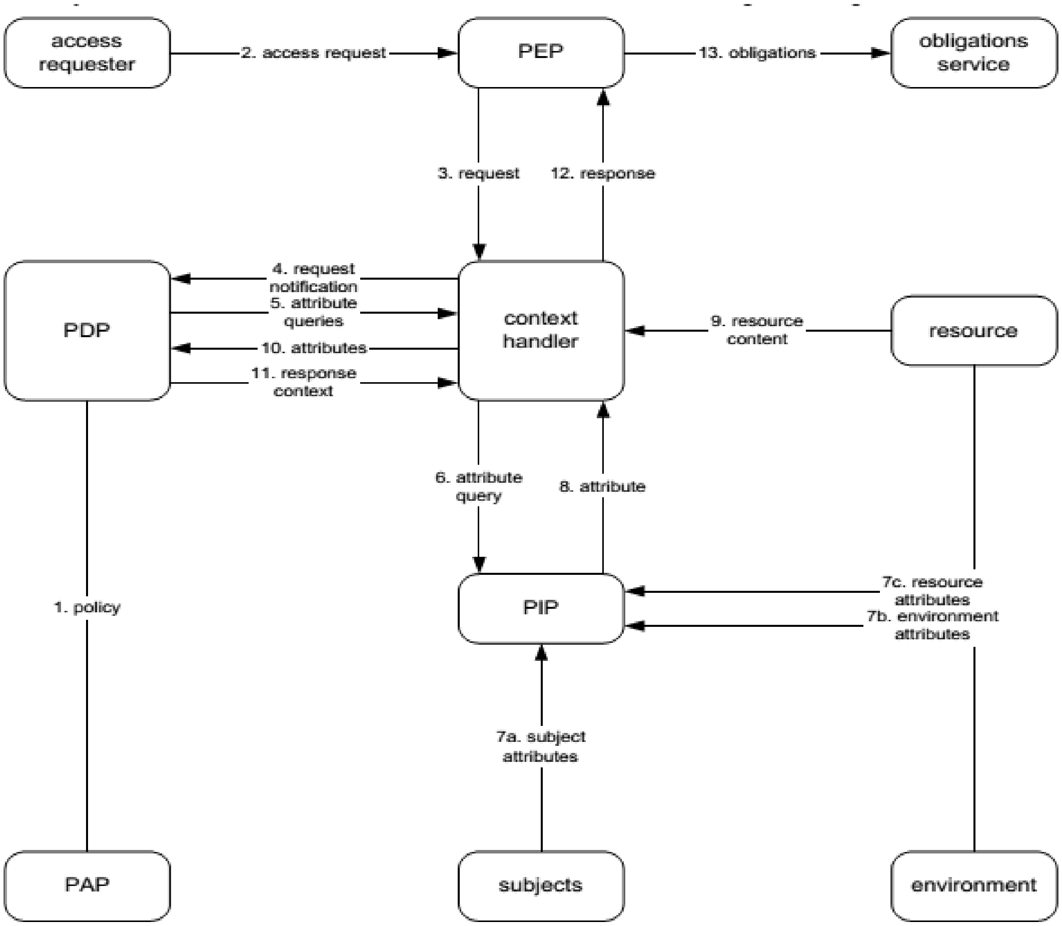


Figure 2-5 Data-flow Diagram [49]

Operating Steps [49]:

1. PAPs: write policies and policy sets to be available to the PDP.
2. Access requester sends the request for access to PEP.
3. PEP sends the request access to context handler, including attributes of the subject, resource, action, and environment.
4. Context handler constructs an XACML request context and sends it to the PDP.
5. PDP requests any additional subject, resource, action, environment and other categories from context handler.
6. Context handler requests attributes from PIP.

7. PIP obtains the requested attributes.

8. PIP returns the requested attributes to the context handler.

9. Optionally, context handler includes the resource in the context.

10. The context handler sends the requested attributes and resources to the PDP, where PDP evaluates the policy.

11. PDP returns the response context to the context handler.

12. Context handler translates the response context and returns it to PEP.

13. PEP ensures to fulfill obligations.

14. If access permitted then PEP permits access to the resource, otherwise denied.

2.5.3 Attributes & Categories

Both Rules and Requests use Subjects, Resources and Actions [49].

1. A Subject element is the entity requesting access. A Subject has one or more Attributes.

2. The Resource element is a data, service or system component, has a single Attribute.

3. An Action element defines the type of access requested on the Resource, have one or more Attributes.

4. An Environment element can optionally provide additional information.

2.6 Conclusion:

Our research includes various research areas, such interdisciplinary research requires from us to provide a description for such disciplines to make the idea and research contribution more clear and obvious for preliminary readers. Context awareness as an emerging discipline where definition still controversial point between researchers and how such definition evolutes over time, also we address the challenge and drawbacks in the front of such fields especially those related to context representation and management of context information acquisition and management, another important discipline includes is access control mechanisms such as RBAC, were RBAC address organizations structures. Pervasive environment is vital concept were become more applicable after emergence and evolution of new advance techniques in networking and computer interaction which simplify its application in real world, another important tool for generating policy and enforcing it is XACML which acts as a common terminology which foster interoperability between multiple vendors of access control techniques.

3 Related Works

Researchers use context awareness in their suggestions for new applications paradigm, also access control mechanisms is one of disciplines affected by this new paradigm, efforts made in improving access control mechanisms to be enriched with context aware, using our research perspective we categories efforts into RBAC based and Non-RBAC based, so we will revise shortly Non-RBAC then RBAC based research in detailed manner.

3.1 Non-RBAC Context Aware Access control Mechanisms:

Access control methods utilized context information, which don't use RBAC as basis for controlling access:

3.1.1 Semantic based approach:

Presented semantic context aware access control framework for PCE [45], designed semantic context-aware policy model adopts ontology and rules to express context information and context-aware access control policies. Mainly focuses on non-organizational bodies and spontaneous interaction scenarios and on enhancing policy dynamicity and adaption, so the need to focus and solve access control for centralized business that has its own resources needed to be protected and controlled from unauthorized users and also considered as pervasive environment.

3.1.2 Web service based approach:

Such approach depends mainly on using web services to control access of users to objects and resources, [47] presented architecture that utilizes web service to enforce policies for controlling access.

3.2 RBAC based Access control Mechanisms:

The first attempt to utilize RBAC in contextual manner done by [10] they provide a model to create and access information about homed residential and resources within the home called Generalized RBAC it depends on environmental roles in addition to traditional roles provided by RBAC about subject and object, in a short the RBAC notion of a role is generalized to capture the state of the environment. Presented a model for securing future applications, which uses generalized approach in handling context information, the model incorporate the notion of roles and environment roles with notion of subject roles. Where homes equipped with high technologies needs high knowledge about information security, which usually not found by most of home residents. Systems should make it very simple to define and manage security policies; also another challenge security mechanism should be usable and non-intrusive.

For example the days in the week split into two groups holidays role and weekdays role or based on locations upstairs, downstairs and guestroom. Permission Assignment in GRBAC done not only based on subject roles but also on active environmental roles.

Providing such roles for environmental state is considered heavy load on pervasive environment systems. Also I think applying such example to be generalized on more wide situations where large number of user than a home, which make such system more complicated to maintain high accuracy measurements. Also the authentication of user requests for access doesn't introduced, which considered very important module to authenticate users interacting with the system, and preventing any deception or any identity spoofing.

[25] Introduced a mechanism that uses state machine matrix SCM to grant or deny access privileges based on context.

The additions for traditional RBAC are:

- State Checking Matrix: handles context information like location, time, and others.
- State Checking agent: handles roles subset for each user.
- Context aware agent: handles permissions subset for each role.

[7] Introduced a model as an extension to RBAC, where roles dynamically assigned to users depending on their context, where they used two state machine for each user one for representing assigned subset of roles and permission assignment hierarchy and both of the subset changed dynamically depending on context change where monitored and transferred to central authority by context agent.

Generating such state machine for each user in pervasive environment especially if the resources or services being targeted by user not exist on large servers or central computing power, in other words if the resource or the service exist of limited power devices with existence of large request form large number devices, this model will be at the expense of responsiveness of the system.

[23] Introduced a formal model for context sensitive access control, where Reference Monitor responsible for making decision.

The proposed architecture doesn't concentrate on how:

- Integration or extending new context factors could be done.
- Also how much such a model could be convenient to pervasive environment?

[9] Introduced formalized definition for managing dynamic roles and permissions assignment, also three major components responsible for three major operations as follows:

- Access Control Manager-ACM: responsible for processing access control request.
- Context aware User Assignment Manager-CAUAM: provides roles assignment based on context requirement defined in each table.
- Context aware Permission Assignment Manager- CAPAM: provides permission assignment based on context requirements, also provides personalized access control via utilizing user preferences information stored in user profile repository.

[22] Presented a model for context aware RBAC in pervasive computing applications, the model uses context information in role admission policies also how application behave when context condition fails to hold.

Also they provide framework for context applications based on the model. [22] Provide personal role permission, which is different from role member to another, unlike NIST model where permissions for a role are the same for each user.

Personalization for permissions done as the following example, if a person want to print a document the permission grant will depend on different factors mainly user preferences like preferring specific printer as well as context information as location.

Also permission specified in this model on objects which in turn classified to private and shared. The model uses context guard to check validity of the context condition. There is available context guard for each role operation individually.

Such context guard which will be for each operation will forms high load when large amount operation available, which could affect dramatically the performance.

[16] Introduced context aware access model based on RBAC for pervasive computing environment called CAP, the context information is grouped into long term (LTC) and short term (STC) context information. CAP introduced as a solution for RBAC

drawbacks for handling unknown users that join pervasive network through using dynamic user role assignment and using context information for dynamic permission activation for roles as well , but this model as stated by authors in next step research still [6] has drawbacks as follows:

- Fetch many context information values to make a decision some of them may not be used, which causes overhead at execution time.
- Doesn't support role hierarchy.
- Uses limited combination of context conditions for assigning roles or activating permissions.

[6] Introduced enhanced version of CAP called iCAP, and tries to overcome some of the limitations in CAP, as noticed author's transformed describing context from a 4-tuple to triple<contexttype,contextrelater,value>,where previous published paper introduced as follows: <entity,contexttype,contextrelater,value>. Also the iCAP now handles roles hierarchy when assigning roles which includes inheritance.

Contribution added in iCAP is how to handle and assign role hierarchy which includes inheritance feature among roles, which in turn allow transferring permission set from parent to child. The iCAP does not provide a mechanism for permission revocation when the condition changed.

The need for authentication component to identify or authenticate users in such environment where unknown users or malicious user has the ability to join such environment and try to access resources and services with no right.

[20] Introduced an extension for RBAC model, assumed that a problem could be occur while trying when a user tries to access some resources from to domains, as a consequence the extension model introduced, on the base which user can perform what operations on which object for what purpose, the paper argues that purpose should be identified in order to grant user to access resources and services in multi-domain environment. This argues for RBAC could be added even for traditional RBAC version, and don't provide any improvement for context management or handling by RBAC.

[13] Present a context-aware access control model that provides a representation of user and service accesses based on gathered context, where activity concept added to the traditional NIST RBAC model

[4] Introduced a model called Context Based Team Access Control, which integrate RBAC model and the TMAC [14] where team notion introduced which represent group of users with specific goal and context information assigned to teams, a new entity called Teams added to entities set defined in RBAC model issued by NIST, simply its main core

idea states using context information to handle access requests. Also activation require selection for a role based on RBAC principles and team, then permissions filters according to teams context where the user is participate in.

Environmental roles [46] depends to control user request which enriched by contextual information on using roles other than traditional roles related to functionality and responsibility of users, but on roles holds environmental definitions such “high CPU load” role which activated and revoked based on policy or context information such as business hour, weekdays and so on.

3.3 Conclusion:

The contributions listed previously most of them focus on the representation of context information within a model, where a critical need for enhancing the process of permission revocation and restoration based on change of context information value or availability. So limited number of research topics listed describes how could implementation and application of framework to be employed that helps to be used as a core for different use purpose systems within pervasive environment. Many contributions made to enhance role based access control system that’s utilizes context information support to change and accommodate decision made upon it.

4 Framework Design

"The framework will allow application designers to expend less effort on the details that are common across all context-aware applications and focus their energies on the main goal of these applications" as described by Anind Dey [18] . The framework will consist from three main components as follows:

- Sensor information acquisition middleware: which built by third party.
- Server Side Engine which represented by:
  - Webservice: which recieves users requests through proxy.
  - Database Management System: responsible to manage our database.
- Proxy: will be on users devices and works as a guard for monitoring, notifiying and enforcing policy.

Our framework will send to client required context information set with change factor for each one, then the client will check if the previously required context information breaks this threshold or factor, as a consequence will notify the web service in order to revoke/grant request to the permissions based on this context information.

The selected framework split context information into two main different groups based on change frequency, the categorization for context information evaluated based on session time average, were such approach helps to improve the performance and minimize communication overhead, at the same time we attempt to enhance the framework through decreasing the communication overhead by controlling and evaluating the validity of each context information based on change flexibility, where such change threshold doesn't make authorization decision validity false. Design will help us to define the framework components, and simplify the process of implementation, through analyzing and studying components using different charts and diagrams that facilitate build process.

4.1 Framework Architecture

Mainly the framework depends on a web service in communication and messaging between different platform client applications and the backend; the web service completely handles and manages authorization process of clients issued requests as shown in Figure 4-1.

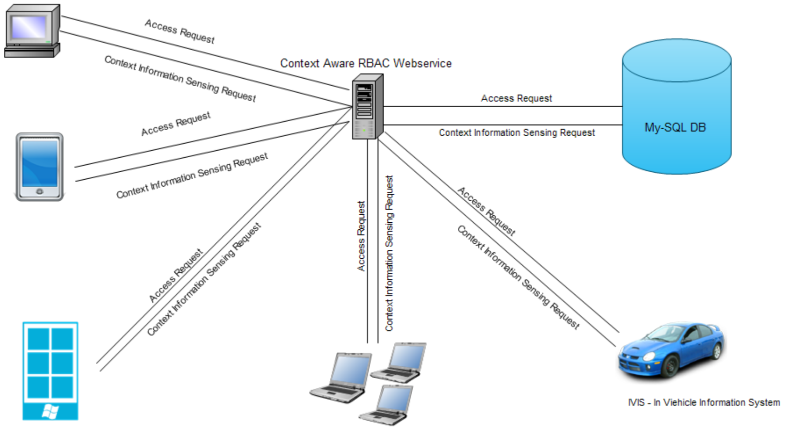

Access Request
Context Information Sensing Request
Context Aware RBAC Webservice
Access Request
Context Information Sensing Request
My-SQL DB
Access Request
Context Information Sensing Request
Access Request
Context Information Sensing Request
Access Request
Context Information Sensing Request
Access Request
Context Information Sensing Request
IVIS - In Viehicle Information System

Figure 4-1 Framework Architectural Diagram

4.2 Model selection

The selection of CAP model to make enhancement as framework due to the following reasons:

- Simplicity of the structure: where the component functionality clear and division of component reduce processing time and separate tasks and support encapsulation.
- The model handles context information based on change to its value, that's would decrease network overhead and improve performance.
- The division of context information into long term and short term reduces dramatically the processing time to handle and control access, which compatible with environment such as pervasive environment.

4.3 CAP Model: Contribution basis

Many security systems proposed for securing information and resources, under continuous advancement in communication technologies which helps to expand the concept of pervasive computing environment where users possess several devices while daily life practiced, where these devices communicate with each other.

A major discipline in security cannot be ignored is access control, which mean how we could control user access for services and resources rounding us. While the pervasive environment getting deeper side by side daily communication advancement pushes researchers to take another important factor to control access on objects which could be a resource or a service is the context information.

Context information represents any information could be used to change the system behavior while running depending on such information measurement and change.

Pervasive Computing: [17] refer to as building a global computing environment where seamless and invisible access to computing resources is provided to the user, where acquirng context knowledge from the environment and providing dyaninc proactive services to the user.

[26] Suggest SOA technology for realizing the vision of ubiquitous computing, trough incorporating services with individuals' daily life and activities. New suggestion to realize pervaisve environment through services.

Context Aware Computing: Context aware applications face different problems specially scalable and extensible applications that adapt with context. Another critical and challenge within this area is context modeling which means how to represent context information to be more convenient for use and share between applications.

4.3.1 CAP Model Description

The model adopted by [16] is a context aware access control model for pervasive computing environments, where roles assigned to users in dynamically based on context information labeled as long-term context information, and permissions granted to those users according to another context information group labeled as short-term context information, this proposed model called CAP for controlling access to resources in pervasive computing environment.

4.3.1.1 CAP model has eight main elements*:*

1- Users: represent the set of users or subjects uses the resources of the system or the computing environment, each user assigned roles in each session based on change on context information.

2- Roles: represents the set of previously initialized jobs in the system, iCAP implements the hierarchy order with support for inheritance.

3- Permissions: represents the set of access rights on objects during session active time.

4- Sessions: represents the set activities done during continuous access time to computing environment.

5- Long Term Context (LTC) Set: represents the set of information or context attributes that don't change during session time, the estimation of this set is based on variables Ts which indicates the average of session lifetime, and M as integer value, where LTCs is the context information that doesn't change during the period M.Ts. So M usually chosen high, to ensure that change is insignificance for LTCs.

6- Short Term Context (STC) Set: represents the set of short-term context information which change frequently in a time less than the average time of the session lifetime, usually that applied to the following information
   a. Location
   b. Time
   c. CPU time
   d. Network load
7- Role-Assignment Condition (RAC): a mapping between roles and long term context information, where in order to obtain specific role, the long term context information listed constraints values should satisfied to obtain the role.
8- Role Permission Condition (RPC): mapping between role and the permission that based on the change of the short term conditions, if the condition that takes short term information as a criteria to grant permissions to specific role.

The following relational diagram in Figure 4-2, represents the components of the Role Based Access Control (RBAC) model.

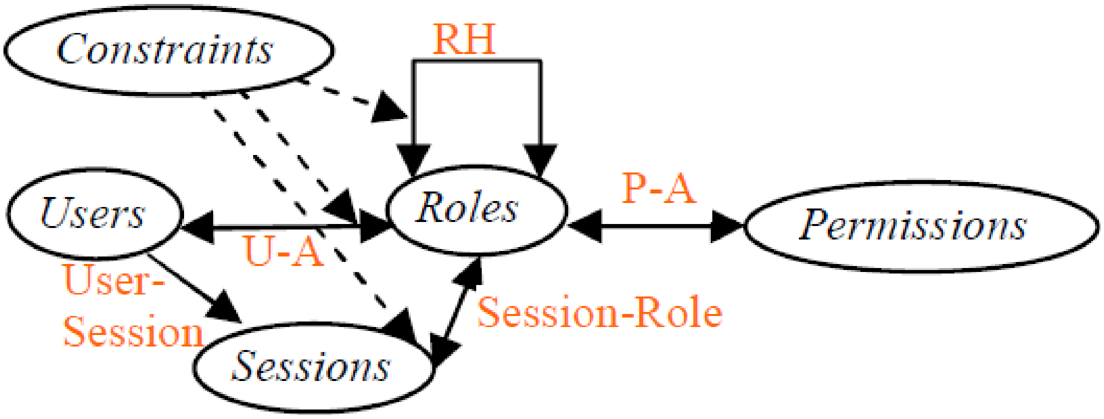


Figure 4-2 RBAC relational diagram [16]

4.3.1.2 CAP Model Formal definition*:*

context information split into two groups the first is Long-term context information and the second is short-term context information, as names implies the first group stores context information that's change infrequent basis where time period between changes is long relatively or not happened over time, the later is short-term context information where changes occurs frequently.

Context information represented as triple predicate as the following:

- Context information type: represents the context information, which could be location, time, situation or any other related information that could change system behavior upon change of its value.
- Context relater: represents the logical operator that will be used in comparison or checking satisfaction of the predicate.
- Value: represents the context information value being compared.

So in terms of the above description of the predicate which will take the simplified term:

(<contextInfoAttribute>,<Relater>,<contextInfoValue>)

An example: (StudentMark,>,60) means if the Student mark is larger than 60 the predicate which represents the context condition is satisfied.

U is user set, R role set, Pr permission set, S sessions, CAP assigns roles to users based on Long Term Context Information satisfaction.

$\text{EntCxSet} = \text{EntSet} \times \text{CtxSet}$

$\text{CtxSet} \subseteq \text{CtxTypeSet} \times \text{CtxRelaterSet} \times \text{CtxValSet}$

$\text{CtxSet} = \text{LTC-Set} \cup \text{STC-Set}$

$\text{LTC-Set} = \text{E-LTC-Set} \cup \text{U-LTC-Set}$

$\text{STC-Set} = \text{E-STC-Set} \cup \text{U-STC-Set}$

E-LTC-Set: is the set of environmental long term context information.

U-LTC-Set is the set of user long term set

Role Assignment Condition RAC maps a subset of LTC-Set to each role, where LTC-set contains Environmental and User centric LTCs.

$\text{RAC: Rx} \rightarrow \text{P(U-LTC-Set)} \times \text{P(E-LTC-Set)}$

$\text{S-U: S} \rightarrow \text{U}$

S-U Maps each session to a user.

$\text{S-R: S} \rightarrow \text{P(R)}$

S-R maps each session to set of roles.

Where roles assigned to a user session dynamically, when session started, STC-Set used to assign permissions to a role which defined statically in the model which called Role Permission Condition.

Role Permission Condition (RPC): a mapping function which assigns STC-Set user and environmental context information to permissions.

RPC: $R \times Pr \rightarrow P(U\text{-}LTC\text{-}Set) \times P(E\text{-}LTC\text{-}Set)$.

When the users start a session and roles assigned to it, iCAP obtains permissions to the session roles based on repository called Session Permission Assignment (SPA).

SPA: $S \rightarrow P(Pr \times P(U\text{-}LTC\text{-}Set) \times P(E\text{-}LTC\text{-}Set))$

4.3.2 CAP Model Architecture:

[16] CAP has two main parts domain authority and session agent as shown in Figure 4-3, session agent created when the user starts or enter new session:

4.3.2.1 Domain Authority

Collects long term contexts and responsible for assigning roles to the user in the beginning of the session based on long term context conditions related to the role, this part responsible for filling S-R in the session according to RPC, also appoints SPA and then sent it to the Session Agent (SA) to manage access, domain authority consist of the following components:

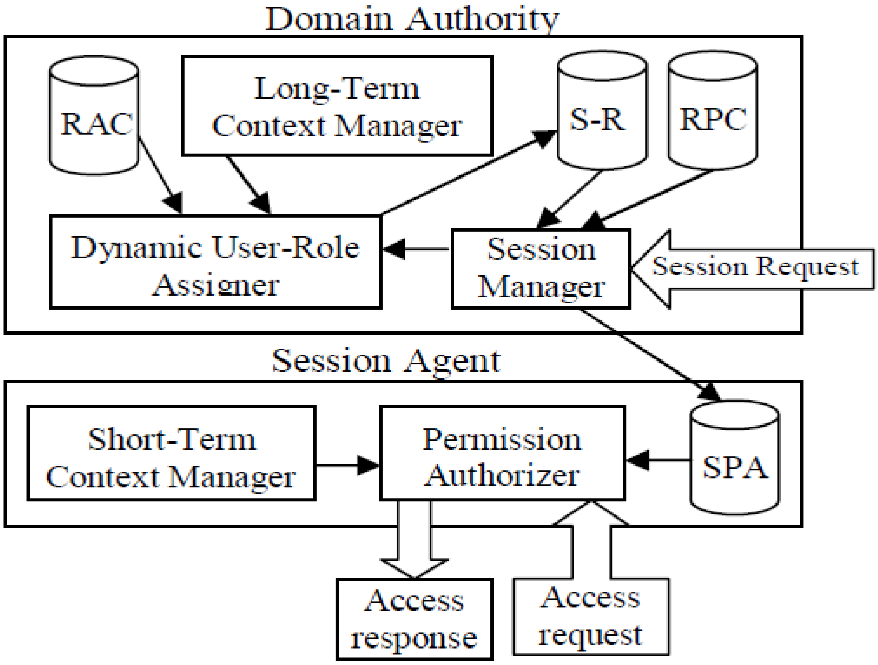


Figure 4-3 CAP Architecture Diagram [16]

- Long-Term Context Manager: collect long term context information from sensors, and then convert it to predicate formula to be stored.
- Session Manager: responsible for handling session requests from users, also assigns session agent and session to a user.
- Dynamic User Role Assigner: assigns roles to the user session based on role assignment conditions then fills S-R storage.

4.3.2.2 Session Agent

Collects short-term context information and evaluates user’s access requests according to SPA, if Request authorization function accepts request then permission granted to the request issuer otherwise rejected, the main components in session agent:

- Short-term Context Manager: collect short term context information.
- Permission Authorizer: makes a decision about users access requests based on role permissions in the session.

4.4 Framework components

As shown in Figure 4-5, the improved context aware framework as described in previous contributions and studies split into a three main components: sensor middleware, domain authority and session agent:

4.4.1 Context Information Provider Middleware

As explained before context information gathering and data collection depends on a component could be provided through third party, the middleware should have the ability to be extended to add new mechanisms to manage and collect any information could be considered as context information for specific domain.

I tried to utilize the various sensors for sensing context information periodically and check the conditions validity after any change made, the middleware developed by third party and support gathering predefined context information remotely via web service or could be done locally.

The middleware is an open source [50]. After intensive review and analysis for its work and used techniques, I found that the developers collect the sensors data and store them locally periodically before flush the memory after frequent upload to the web service based on previous preferences, also such upload could be disabled where any interaction to the web service is done via improved CAP proxy.

The collected data stored as Content Provider class which permits us to collect them using Observer class, where any change made to the SQLlite database the observer in any other application for the file will be notified to fire specific action as we intended to do.

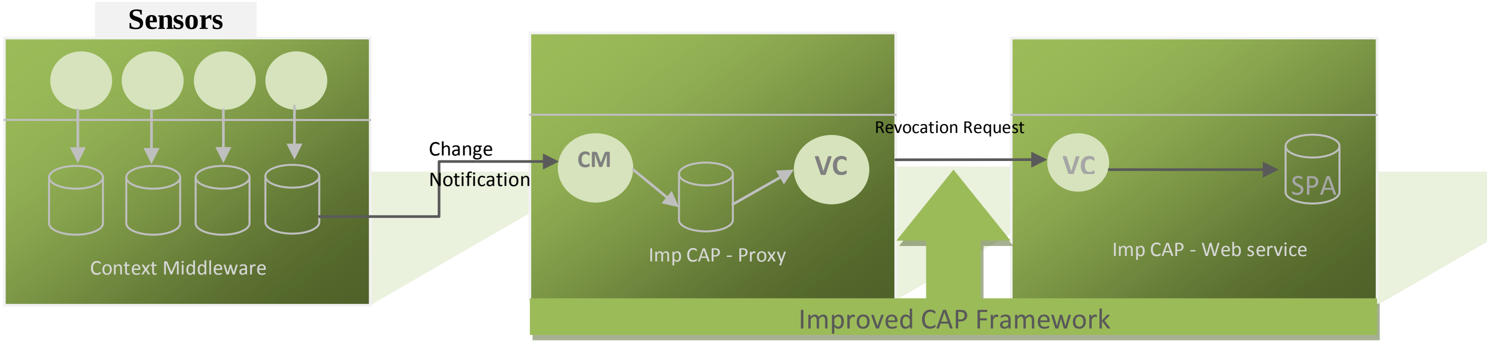


Figure 4-4 Middleware and Improved CAP framework interaction

As shown in the Figure 4-4 Context manager receive context sensor change notification, and send its value to the validity controller w.0hich in turn check the change value if not exceeds the condition interval or values, in case of invalidity revoke request will be sent for the related permissions to prevent any illegal user access or usage for such request.

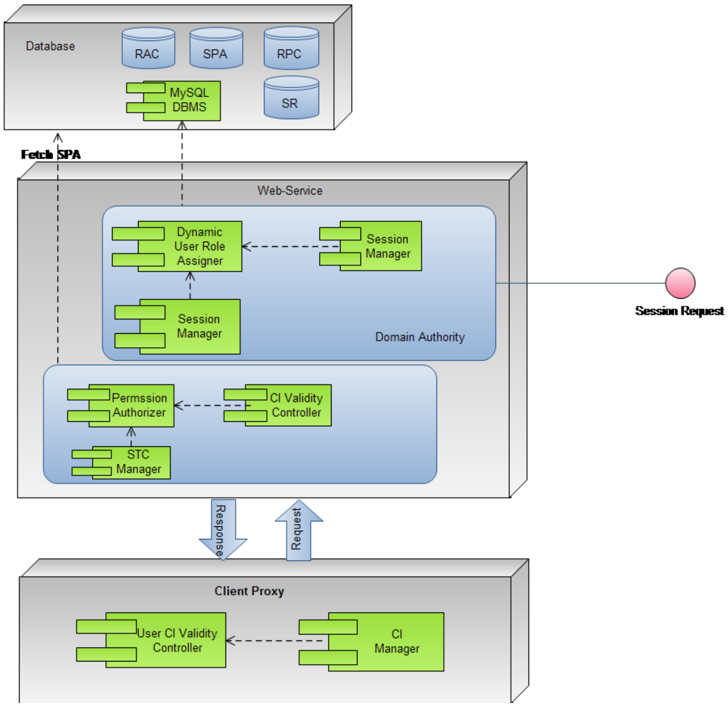


Figure 4-5 Context Aware RBAC Framework Architecture

4.4.2 Domain authority:

contains session manager which is responsible for handling session requests issued by clients, long term manager which handles and manages context information acquisition and distribution and dynamic user role assigner handles and enforce assignment of roles to users according to long term context information LTCI and role assignment conditions RAC then fills session role S-R database.

4.4.3 Session Agent:

Contains short term context manager which is responsible for managing context information acquisition and distribution which classified as short term, permission authorizer make decisions about user access requests based on granted permission for the session which belongs to, stored in the dynamic database SPA, also session agent contains new component called environment context information validity controller which also responsible for controlling validity of context information such as environment related such as time, permission authorizer will check its validity before making decision.

4.4.4 Client Proxy:

Client proxy is responsible for enforcing and ensuring the integrity of context information measured through secure communication channel to prevent fraud attempts from malicious users, client agent will contain context information manager responsible for acquisition and distribution of measured context information, and user context information validity controller which will check that any context information change and is less than change threshold or not to issue event to the web service to revaluate access request decision made.

4.4.4.1 User Context Information Validity Controller*:*

Responsible for handling and monitoring context information validity based on value change and how that change affect permission condition validity being guard, while the number of conditions being guard for each user individually, will dramatically degrade required performance to server side framework.

4.4.4.2 User Context Information Manager

Collects context information and format it to be processed by validity controller, such module needs to be pluggable module that enables adding new tools and APIs to broaden its ability to measure and sensing user and environmental context information, with reasoning mechanisms.

4.5 Improved framework scenarios

Scenario help us to make the usage of the framework in various domains, also will use scenario in proving framework efficiency and applicability in such environments and domains where pervasive environment is applicable.

Definition: "Context information is any piece of information, the change in its value leads to a change in the behavior of the service being delivered or its output representation to the end user"

Example 1: As indicator for our definition, that in the scenario 1, user trying to read registration regulations list, if the student visually disabled, system converts output into voice mode, else use textual mode, as noticed user health status considered as context information, also such context information comes as independent of main interaction or aims of the service.

Example 2: student wants to pay money to be able to register, based on load balancing to teller offices redirect student to right office, such information office load considered as context information.

What we want to prove that we can't consider that all information of the application as context information, where most of them is a part of the business logic of that domain, and subject to rules of that domain, where change is only in results and not on the way of its representation or behavior.

So at the stage of policy creation, we should take as consideration the definition when selecting context information, which will dramatically enclose context information that being monitored by controllers, as a consequence performance will be better due to this best practice with better enhancement to quality of service being presented to end users.

4.5.1 Scenario 1: University registration system

First year student joins university network using cell phone try to display course management system catalogue course, in order to do any of the following tasks which represented as permissions as sown in Table 4-1:

Table 4-1 Permissions Set - University Registration System

Permissions	
Object	Action
Course	ShowActivites
Course	Register
Course	AcceptJoin
Course	Assign
Course	GetPre-requisite
Course	GetLevelPre-requisite
Assignment	Add
Assignment	Edit
Assignment	Answer
Assignment	Delete
Student	Register
Announcement	Add
Announcement	Edit
Announcement	Delete
SchoolMap	Show

Figure 4-5 shows how a student could display details and course activities needs to be registered within one of the sections of that course, if she/he is not registered yet then the system checks registration transactions close date, if not passed then check his balance, if no enough credit to register then based on student location redirect her/him to nearest teller office, in order to add credit to be able to register and display course activities. As shown in Table 4-1 list the main actions in the scenario, also Table 4-2 shows expected roles within the domain, also Table 4-3 shows the interaction between the users represented as student in this scenario and the framework:

Table 4-2 Available Roles University Registration System

Roles
Student (S)
Teacher (I)

Table 4-3 Long Term Context Information - University Registration System

LTC	Value	Description
ID	1200140052	
Term (Tr)	1st	
Department (Dt)	CSc	

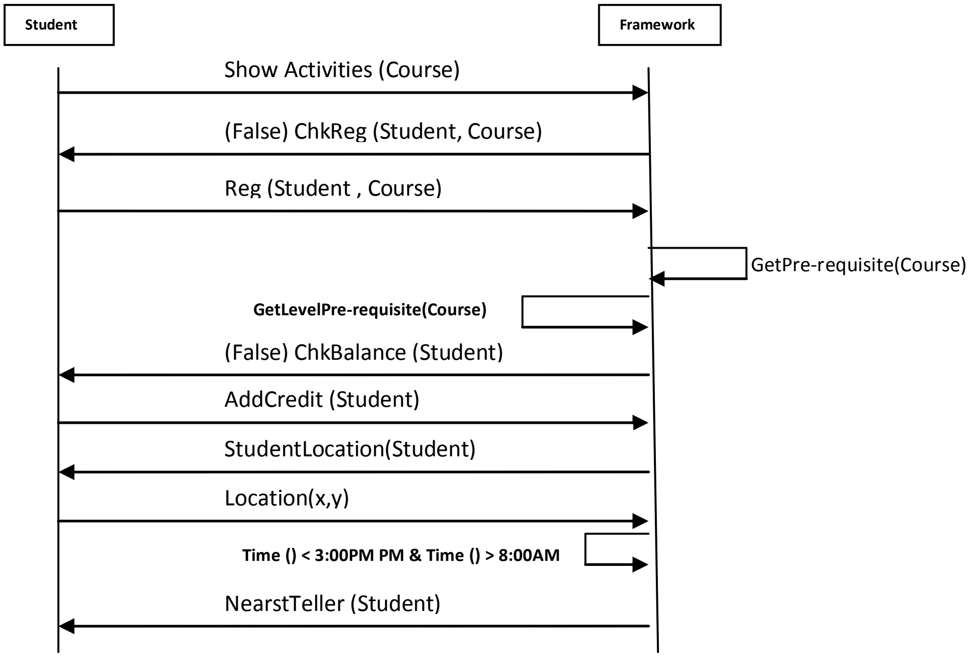


Figure 4-6 Student Scenario Sequence

Table 4-4 Short Term Context Information - University Registration System

STC	Value	Description
Time (T)	1:00 PM	Current time.
Location (L)	31.1102,32.3221	Current location of the user.
Course Pre-requisite (CPr)	NA	Requested course for register, if s/he has pre requisite courses.
Student Level (SL)	2	The student level related to credited hour that he finished.
Student Balance (SB)	900	Represents the financial status, if has money in his account.

The Role Assignment Conditions is shown in Table 4-5 responsible for validating access request based on conditions provided as part of access control policy regards the resources and provided services to any domain. Based on the conditions in RAC access response will be change if the context information provided changed, so malicious users could send to the framework web service invalid context information leads to incorrect access right to a resource or a service being controlled.

Table 4-5 Role Assignment Conditions (RAC) - University Registration System

Role	Conditions represented by predicate logic		
	Object	Relater	Value
Student	Student-ID	=	1200140052
Teacher	Teacher-ID	=	3200140019

So as consequence we assume that server side framework can't operate correctly, where user side context information is very important to access decision function, from here we conclude to urgent need to client side agent protect and provide trusted channel for frame work service to provide it with valid context information.

Table 4-6 Role Permssion Condition (RPC) - University Registration System

Role	Conditions represented by predicate logic		
	Object	Relater	Value
Student	Student-x-loc	>	31.5145474
	Student-x-loc	<	31.5138911
	Student-y-loc	>	34.4408026
	Student-y-loc	<	34.4407328
	Time	>	8:00
	Time	<	15:00
Teacher	Teacher-x-loc	>	31.5145474
	Teacher-x-loc	<	31.5138911
	Teacher-y-loc	>	34.4408026
	Teacher-y-loc	<	34.4407328

Role Permission Condition (RPC) as shown in Table 4-6, built in order to validate permissions granted to the user, some of these permissions granted based on RPC needed to be checked and monitored to ensure validity of the access control.

In order to enhance performance, the permissions granted based on conditions related to context information issued from users devices, will be loaded to user context information validity controller component in order to be monitored, any change leads to invalidity controller will evaluate change against user available RPC then issues grant/revocation to the web service, such mechanism will benefit from pervasive computing environment properties as wide range of devices will break down computation for each device independently and ensure at the same time high quality service with trusted channel available between framework web service and users devices. Table 4-7 shows the mapping of user with sessions during interaction.

Table 4-7 Session User Mapping - University Registration System

id	Session	User	Description
1	96e4f2ec-ca4c-4d61-a239-40ddd687a37e	1200140052	
2	0e7125af-fadb-4fd5-ac25-1cf095775340	3200140019	
3	7cf0e6ea-8db4-4683-a66e-bf67270bcbb3	2200140019	

Table 4-8 Session Permission Assignment - University Registration System

Session-id	Permissions	
	Object	Action
1	Course	ShowActivites
1	Course	Register
1	Assignment	Answer

Session Permission Assignment as show in Table 4-8, is a dynamic dataset will stores the permissions granted based on the collected context information, such dataset will change dynamically by permission authorizer, each time the change occur on any context information being followed by controller at client side or web service side.

4.5.2 Scenario 2: Health care system

Patients visiting health care center will provide their own identity number to client agent in order to be involved within the system processing, as soon as s/he joined health care center environment, the system will automatically based on family record will assign her/him to meet the right doctor with order number based on her/his arrival and join time, as part of the doctors diagnose task requires the patient to get x-ray for specific body part, patient will enters the x-ray department to take x-ray picture at this moment s/he takes order number, based on order number and patient identity, radiologist will review doctors request and take action, then provide feedback to the patient case. First visit patient benefit from health center services needs to be registered within information system using client agent, as a consequence new generated patient id placed queue to meet a doctor based on load balancing parameter for each doctor room, As shown Table 4-9 explains the scenario as actions, Table 4-10 shows expected roles within the domain, also Table 4-11 shows the interaction between the user represented as student in this scenario and the framework.

Table 4-9 Permissions Set – Health Care System

Permissions	
Object	Action
Patient	Register
Patient	OrderEnter
Patient	WardClassify
Patient	RequestDrugs
Doctor	diseaseDiagnose
Doctor	WritePrescription
Doctor	RedirectPatient
Radiologist	ImageAcquisition
Radiologist	ImageReview
Radiologist	UpdatePateintRecord
Laboratory	BloodTest
Nurse	PateintBioOpsMeasure
Pharmacist	RegisterDrugs
CenterMap	DisplayMap
CenterMap	IdentifyLocation

The first context information change the system behavior in feedback is doctors load balance. As shown in Figure 4-7, explains the scenario.

Table 4-10 Available Roles – Health Care System

Roles
Patient (P)
Doctor (D)
Radiologist (R)
Pharmacist(Ph)
Medical Officer(MO)

Table 4-11 Long term Context Information – Health Care System

LTC	Value	Description
PatientID	901225569	
Age	25	
NCD-status	YES	Non Communicable Disease like Hypertension, diabetes mellitus, cardiology

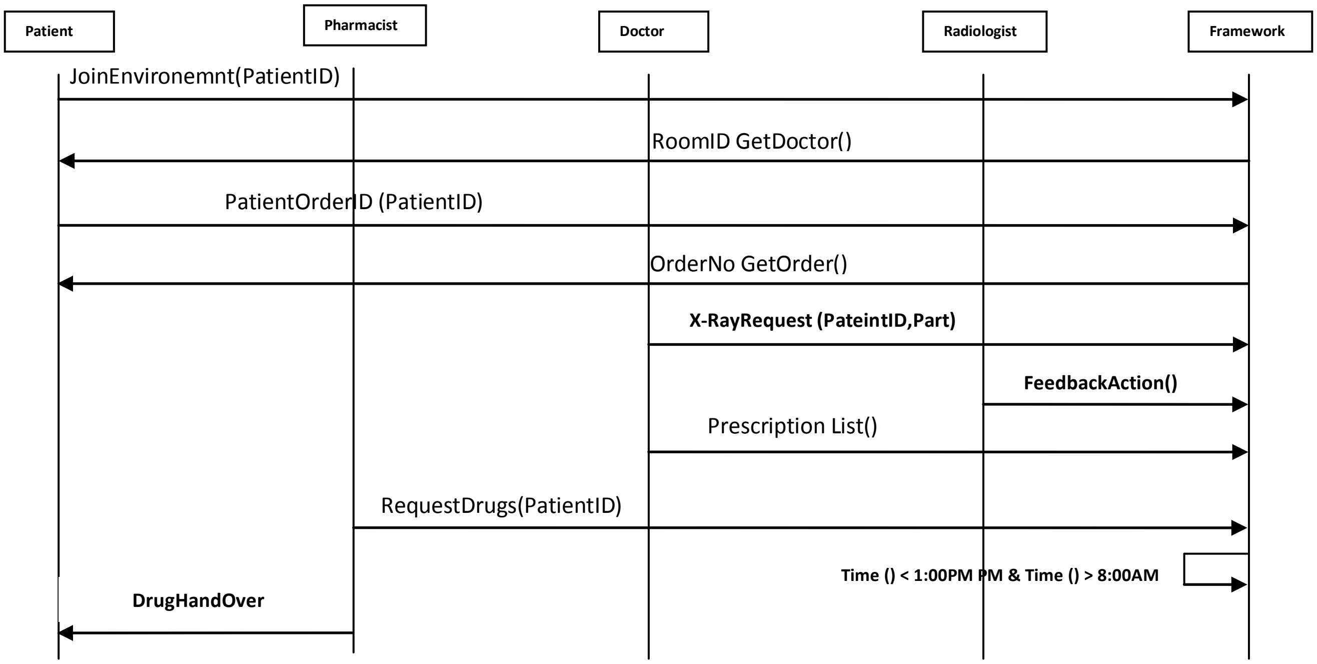


Figure 4-7 Health center Sequence – Health Care System

Table 4-12 Short Term Context Information – Health Care System

STC	Value	Descr
Time (T)	1:00 PM	Current time.
Location (L)	31.1221,32.1231	Current location of the user.

The Role Assignment Conditions as shown in Table 4-13 responsible for validating access request based on conditions provided as part of access control policy regards the resources and provided services to any domain.

Based on the conditions in RAC access response will be change if the context information provided changed, so malicious users could send to the framework web service invalid context information leads to incorrect access right to a resource or a service being controlled. So as consequence we assume that server side framework can't operate correctly especially we user side context information is very important to access decision function, from here we conclude to urgent need to client side agent or proxy protect and provide trusted channel for frame work service.

Table 4-13 : Role Assignment Conditions (RAC) – Health Care System

Role	Conditions represented by predicate logic		
	Object	Relater	Value
Patient	Patient-ID	=	901225569
Doctor	Doctor-ID	=	945151691
Pharmacist	**Pharmacist-ID**	**=**	804171821

Table 4-14 Role Permssion Condition (RPC) – Health Care System

Role	Conditions represented by predicate logic		
	Object	Relater	Value
Patient	Patient-x-loc	>	31.5145474
	Patient-x-loc	<	31.5138911
	Patient-y-loc	>	34.4408026
	Patient-y-loc	<	34.4407328
	Time	>	8:00
	Time	**<**	15:00
Doctor	Doctor-x-loc	>	31.5145474
	Doctor -x-loc	<	31.5138911
	Doctor -y-loc	>	34.4408026
	Doctor -y-loc	<	34.4407328

Role Permission Condition as show in Table 4-14, built in order to validate permissions granted to the user, some of these permissions granted based on RPC needed to be checked and monitored to ensure validity of the access control.

In order to enhance performance, the permissions granted based on conditions related to context information issued from users devices will be loaded to user context information validity controller component in order to be monitored, any change leads to invalidity controller will notify the framework web service to re-evaluate the decision, such mechanism will benefit from pervasive computing environment properties as wide range of devices will break down computation for each device independently and ensure at the same time high quality service we trusted channel available between framework web service and users devices, Table 4-15 shows the mapping of user with sessions during interaction.

Table 4-15 Session User Mapping – Health Care System

id	Session	User	Description
1	96e4f2ec-ca4c-4d61-a239-40ddd687a37e	901225569	
2	0e7125af-fadb-4fd5-ac25-1cf095775340	945151691	
3	7cf0e6ea-8db4-4683-a66e-bf67270bcbb3	804171821	

Table 4-16 Session Permission Assignment – Health Care System

Session-id	Permissions	
	Object	Action
1	Patient	OrderEntry
1	Doctor	WritePrescription
1	RequestDrugs	RequestDrugs

Session Permission Assignment as shown in Table 4-16 is a dynamic dataset will stores the permissions granted based on the collected context information, such dataset will change dynamically by permission authorizer, each time the change occur on any context information being followed by controller at client side or web service side.

4.6 Conclusion

Through design process we draw complete picture about the system components and their main functionality, also we present the contribution made to context aware role based access control framework, then we define the structure of the framework after improvement made, the design chapter also have scenario based validation, to prove applicability in different domains and business environments.

5 Framework Implementation

Chapter five will discuss the main issues related to the process of building enhanced context aware framework in terms of interdisciplinary techniques, also we will enrich implementation chapter with charts and algorithms, the importance of our work yields from, transferring the framework before and after enhancement from theoretical state into experimental approach, where statistics about performance will be analyzed.

5.1 Functionalities of Framework Components

The framework main tasks divided into two parts are web service part and proxy part; we will define the most effective tasks done as code and flow chart:

5.1.1 Proxy:

Represents the end point users interact with the system through, we will list operations done on devices in order control access request made and control granted privileges or permissions:

5.1.1.1 Observing change for context information

Observing tasks used to follow up any change could be made to the context information being collected periodically, in order to check validity of the conditions against RPC.

Algorithm:

```
  Function ObserveCI_X ()    Begin
 Set flag =false,   context_var = getContextVal(),
 context_rpc_list = getUserRPC()
 S = context_rpc_list.size()
 For i=1 to S   Begin
 flag =CheckValidity(context_var, context_rpc_list[i],permid)
 End
 If flag is false Then
 RevokeifExist(permid)
Else
GrantIfNotExist(permid)
 END
```

Figure 5-1 Observing change for context information

Figure 5-2 shows how the Proxy monitors any change could happen to the user context information, which included within RPC rules, then the user-related RPC which already loaded to local proxy database on the device will be re-evaluated locally to check its validity after any change, so evaluation will identify which permission could be continue granted or revoked locally, once action applied to the local SPA database then the web service repository will affected, Figure 5-3 show the flow chart for the process.

Flowchart:

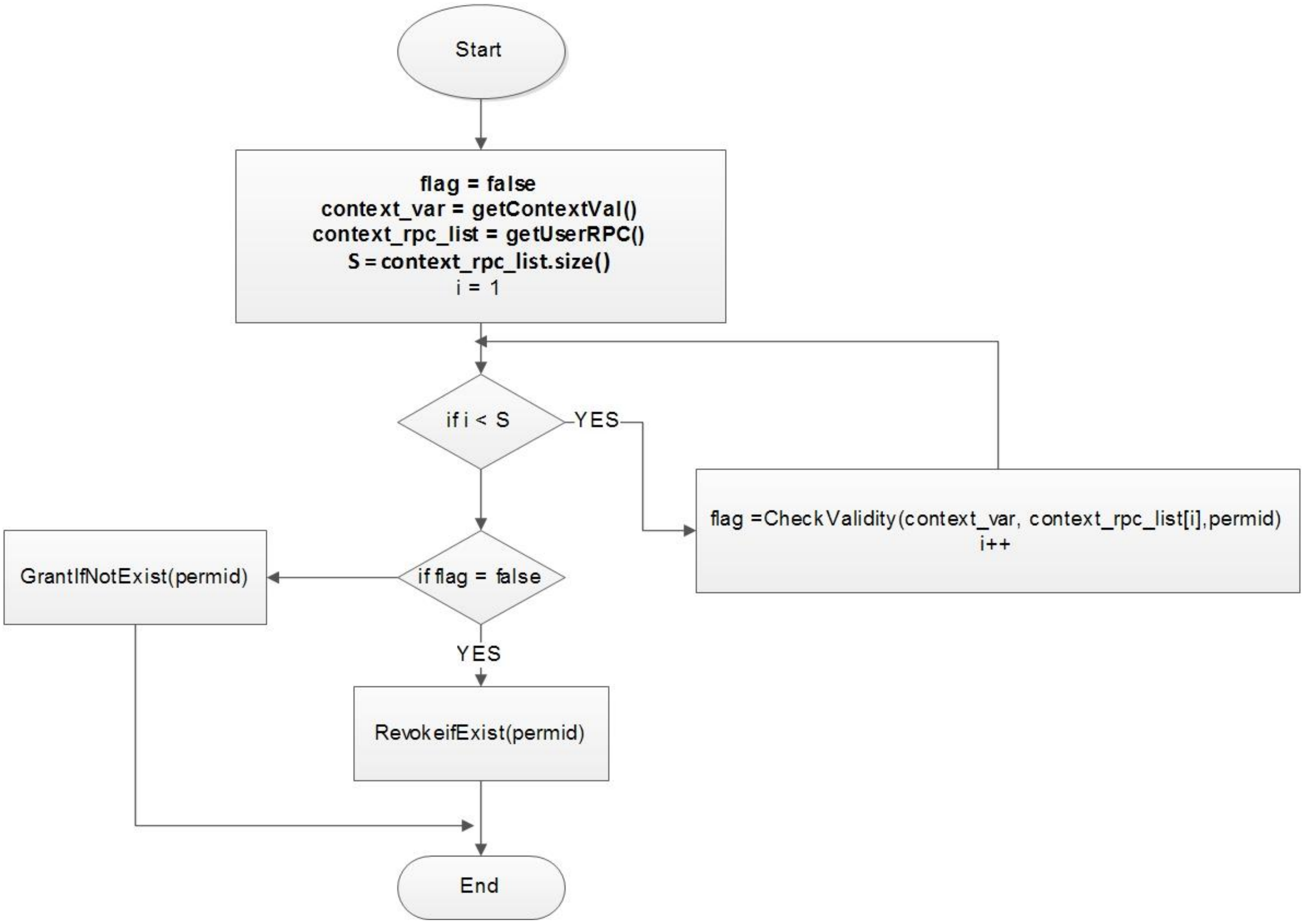

Start
flag = false
context_var = getContextVal()
context_rpc_list = getUserRPC()
S = context_rpc_list.size()
i = 1
if i < S
YES
flag =CheckValidity(context_var, context_rpc_list[i],permid)
i++
if flag = false
GrantIfNotExist(permid)
YES
RevokeifExist(permid)
End

Figure 5-2 Observing change for context information flowchart

5.1.2 Web service:

5.1.2.1 Initial Session Request:

Represents the first request made be the user represented by the proxy, which asks session manager to establish new session to start interaction:

Algorithm:

```
Begin

Set flag = false

flg = establishsession(userid)

If flag == true Then

flg = fillSR(userid)

return flg

end
```

Figure 5-3 initial session request algorithm

Code:

```
boolean  flg = false;

 DynamicURAssigner duassigner = new DynamicURAssigner()

 SessionManager smanager = new SessionManager()

 flg = smanager.EstablishSession(userid, ipaddr)

  duassigner.fillSR(userid)

 return flg
```

Figure 5-4 initial session request code in the webservice

Flowchart:

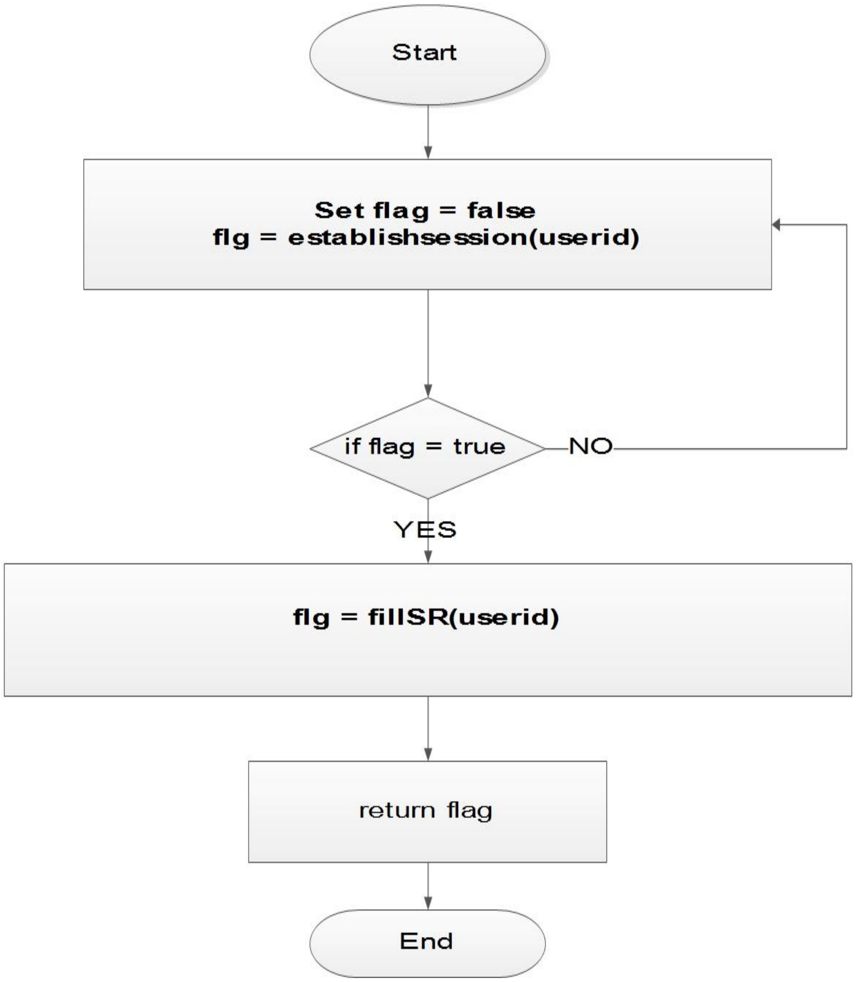


Figure 5-5 initial session request flow chart

As shown in Figure 5-4 if the request success in establish session for the user, then will fill session role (SR) database, which will link user through the session to expected roles based on evaluation of the RAC. Figure 5-5 shows the used java code to implement the method on the web service, also Figure 5-6 shows the sequence of the method in flow chart.

5.1.2.2 Loading granted roles – FillSR

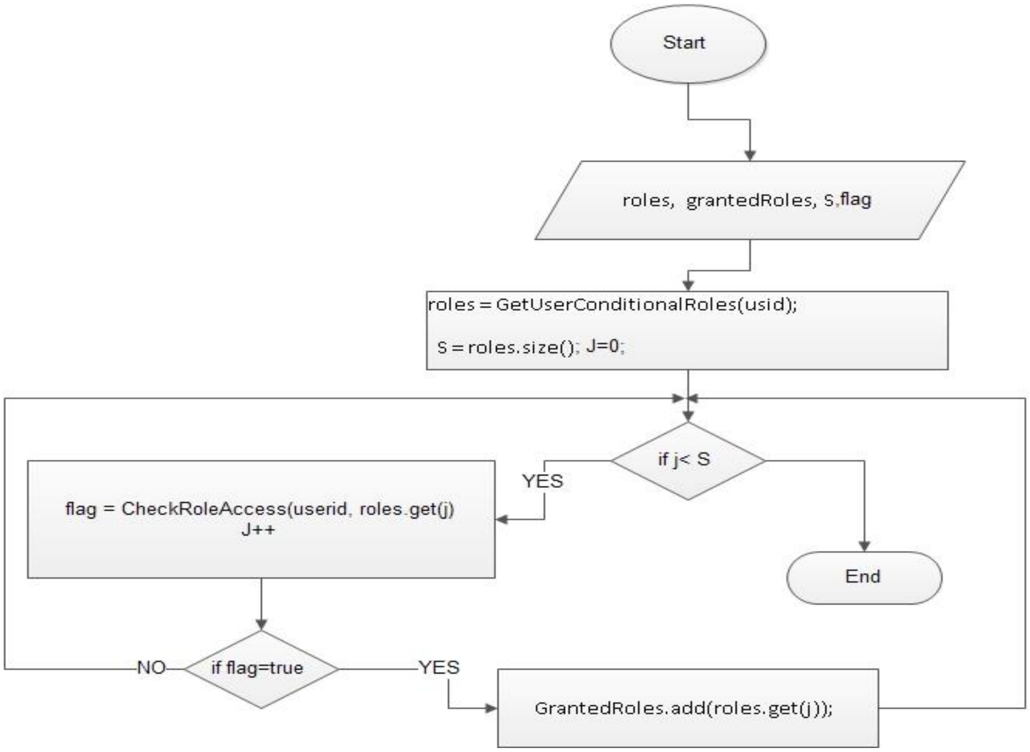
This critical method fills the table of roles for current session into SR database, for the user initiating the request, such process will done only at the beginning of the session, as shown in Figure 5-7

Algorithm:

```
Function FillSR (usid) Begin
roles = GetUserConditionalRoles(usid)
 Set grantedRoles;
 S = roles.size()
For j=1 to S
Begin
if (CheckRoleAccess(userid, roles.get(j - 1)) Then
        GrantedRoles.add(roles.get(j - 1));
End if
End
```

Figure 5-6 FillSR Algorithm

Flowchart:



Start
roles, grantedRoles, S,flag
roles = GetUserConditionalRoles(usid);
S = roles.size(); J=0;
if j< S
YES
flag = CheckRoleAccess(userid, roles.get(j)
J++
End
NO
if flag=true
YES
GrantedRoles.add(roles.get(j));

Figure 5-7 FillSR flowchart

5.1.2.3 Loading granted Permissions – ValidateAllPerms

Such method responsible for checking validity of permissions for each granted role in the previous step, based on RPC in order to use the list of permissions output to fill SPA table for each user based on user identification number for each session request.

Algorithm:

```
Function ValidateAllPerms (sessionid) Begin

Allperms = GetPerm4AssignedRoles (sessionid)

 Set ValidPerms;

 S = perms.size()

For i=1 to S Begin

if (CheckPermValidity (Allperms.get(i), v_userid)) Then

            ValidPermsids.add(Allperms.get(i));

End if  End
```

Figure 5-8 Validate Permissions Algorithm - ValidateAllPerms

Flowchart:

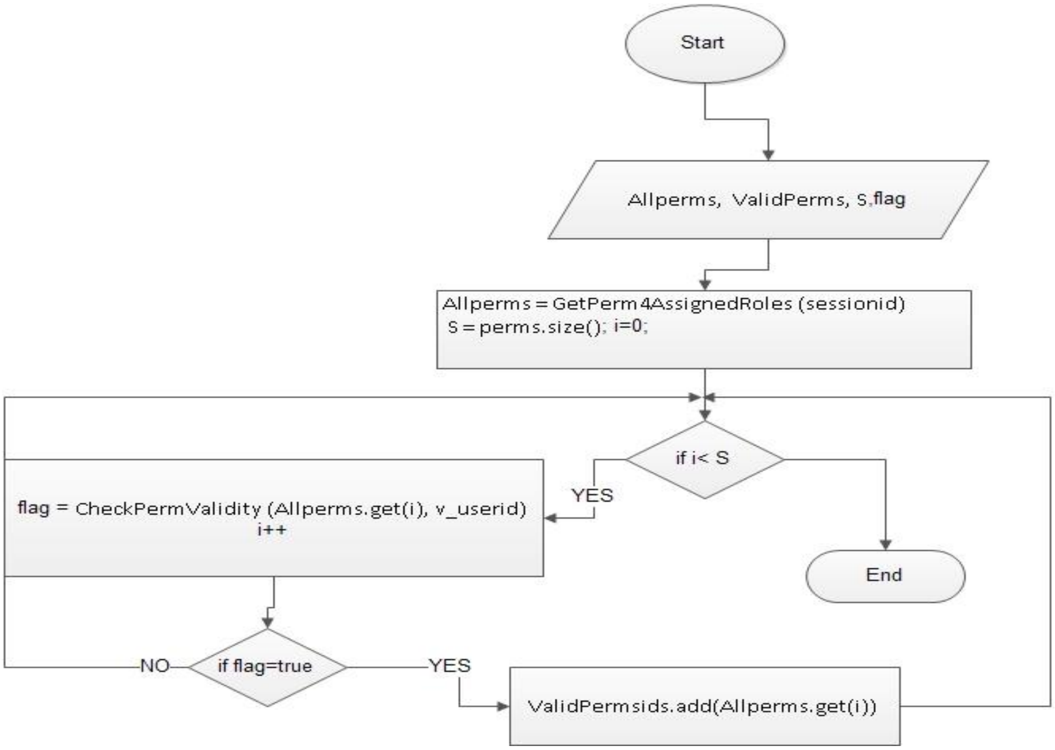

Start
Allperms, ValidPerms, S,flag
Allperms = GetPerm4AssignedRoles (sessionid)
S = perms.size(); i=0;
if i< S
YES
flag = CheckPermValidity (Allperms.get(i), v_userid)
i++
End
NO
if flag=true
YES
ValidPermsids.add(Allperms.get(i))

Figure 5-9 Validate Permissions flowchart - ValidateAllPerms

5.1.2.4 Loading conditions needed for gaining role permissions – FillCondsByRole:

This function is a mapping function that restores required conditions, the user needs to gain specific role.

Algorithm:

```
 Function FillCondsByRole ()   Begin
Set userid
 vic_cons_list instance of CI_Condition;
 context_var = getContextVal()
 context_rac_list = getUserRAC(userid)
 S = context_rac_list.size()
For i=1 to S   Begin
      Create CI_Condition cicond
       cicond.setCiid(context_rac_list [i].rac_ciid);
       cicond.setOperator(context_rac_list [i].rac_operator);
       cicond.setValue(context_rac_list [i].rac_value);
       cicond.setRoleid(context_rac_list [i].rac_roleid);
       cicond.setRuleeffect(context_rac_list [i].rac_rule_effect);
       vic_cons.add(cicond);
    END
return vic_cons
END
```

Figure 5-10 Loading conditions by role algorithm

Figure 5-7 show how web service load RAC rules needed to assign roles based on long term context, such conditions will be evaluated against user long term context information and grant applicable roles, and store them into SR database, Then start another process which represented on starting evaluation for RPC to grant users permissions based on short term context information, as seen in Figure 5-8 the implementation code used for that algorithm, also Figure 5-9 shows the algorithm steps as flow chart.

Code:

```
Vector<CI_Condition> vic_cons = new Vector<CI_Condition>();

    Db_Connection conn = new Db_Connection();

    conn.connect();

    String Sqlquery = "SELECT  " + rac_ciid+ "," + rac_rule_effect     + "," + rac_value

          + "," + rac_roleid    + "," + rac_operator    + " FROM " + Tables.getInstance().rac

          + " WHERE "    + rac_userid + " = " + userid

          + " AND " + rac_roleid + " = " + roleid;

    ResultSet rs = conn.DoQuery(Sqlquery);

    int cnt = 0;

    while (rs.next()) {

       CI_Condition cicond = new CI_Condition();

       cicond.setCiid(rs.getInt(Tables.getInstance().rac_ciid));



       cicond.setOperator(rs.getString(Tables.getInstance().rac_operator));

       cicond.setValue(rs.getString(Tables.getInstance().rac_value));

       cicond.setRoleid(rs.getInt(Tables.getInstance().rac_roleid));

       cicond.setRuleeffect(rs.getString(Tables.getInstance().rac_rule_effect));

       vic_cons.add(cicond);

       cnt++;

    }//end while

    return vic_cons;
```

Figure 5-11 Loading conditions by role - code

Flowchart:

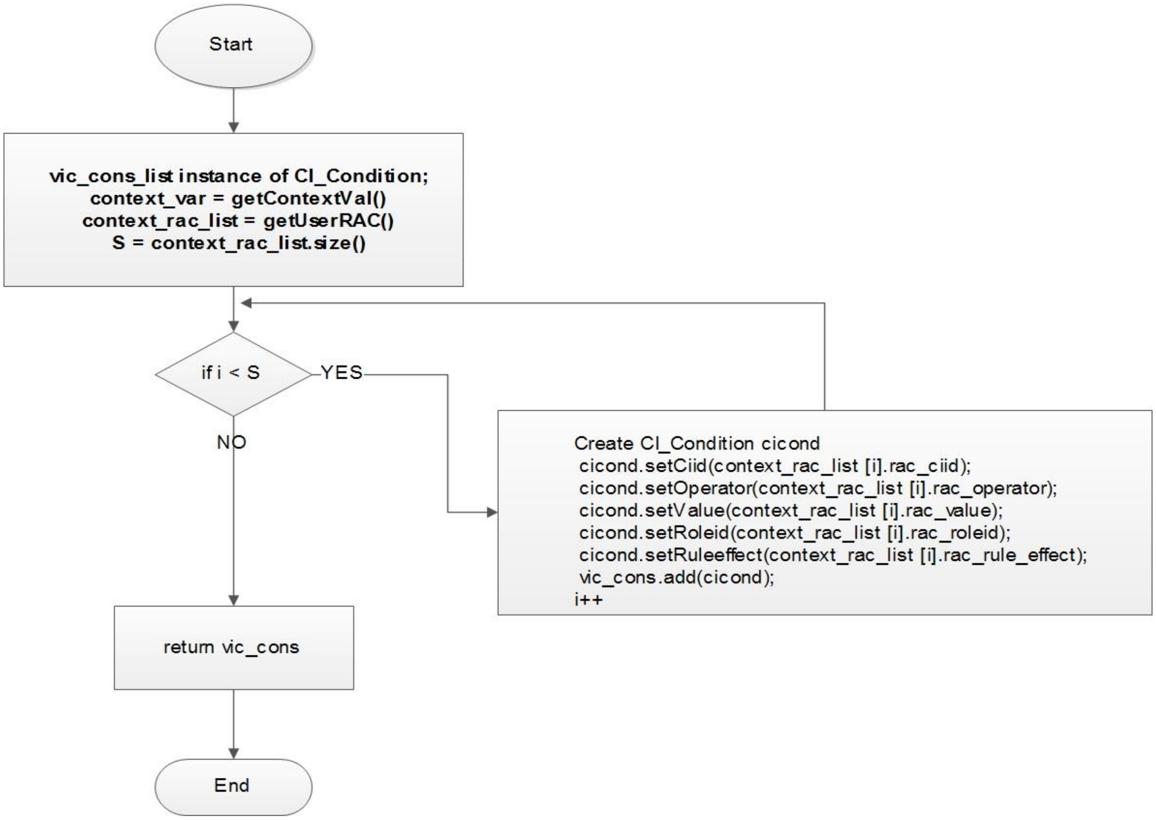

Start
vic_cons_list instance of CI_Condition;
context_var = getContextVal()
context_rac_list = getUserRAC()
S = context_rac_list.size()
if i < S
YES
NO
Create CI_Condition cicond
cicond.setCiid(context_rac_list [i].rac_ciid);
cicond.setOperator(context_rac_list [i].rac_operator);
cicond.setValue(context_rac_list [i].rac_value);
cicond.setRoleid(context_rac_list [i].rac_roleid);
cicond.setRuleeffect(context_rac_list [i].rac_rule_effect);
vic_cons.add(cicond);
i++
return vic_cons
End

Figure 5-12 Flow chart for loading conditions by role

5.2 Development Tools

The main development tools used to build Improved CAP as follows:

- Integrated Development Environment (Netbeans release 7.2.1)
- Eclipse Android Developer Tool
- Android SDK.

5.3 Improved CAP Framework Components

The main components the framework is:

- **Backend**: My-SQL Database stores states of context information censored from clients, also stores dynamic database like Session Permission Assignment SPA, Session Role database S-R, also stores static database records such as Role Assignment Conditions RAC, Role Permission Condition RPC.

- **Server**: represented by the web service that receives client requests, initiate the calculation and applying the control decision on incoming requests from different users connected to system.

- **Proxy**: represents the utility that will forms users request and works as a secure communication channel between the framework and the end users, also used to transfer events occur or related to the different given permissions to the user, where quality of context information and validity based on user preferences and usage privileges. The client implemented to fit Android OS environment:

5.4 Platform Dependency

The web service component in the framework is based on the following technologies:

- Java development environment as main service provider component.
- Requests directed to the framework using XML based, compatible with SOAP requests. To utilize web service techniques on handling such requests.

5.5 Database Design

Figure 4-3 explains using entity relationship diagram the structure of the database for the web service which handles end users requests.

As shown in Figure 4-3 entity relationship diagram, explains the database structure for the web service part, where SR and SPA tables represents dynamic containers for permissions and roles granted or revoked from users during the interaction, RAC and RPC represents the containers of the domain policy defined by the domain administrators. Users table is a container for domain users and session table contains users sessions each time involved the system, permissions table works as a container for defined permissions in a specific domain.

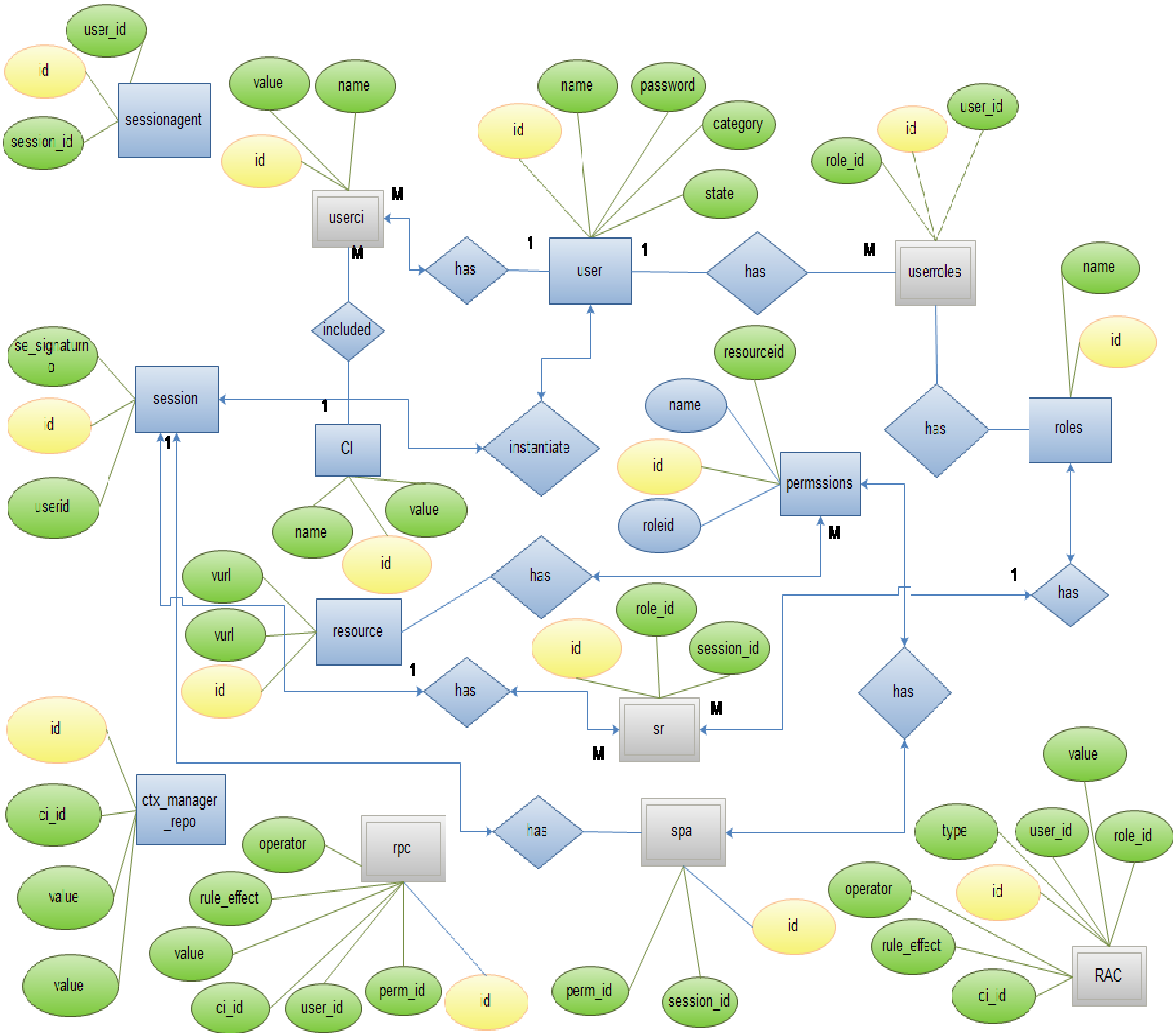


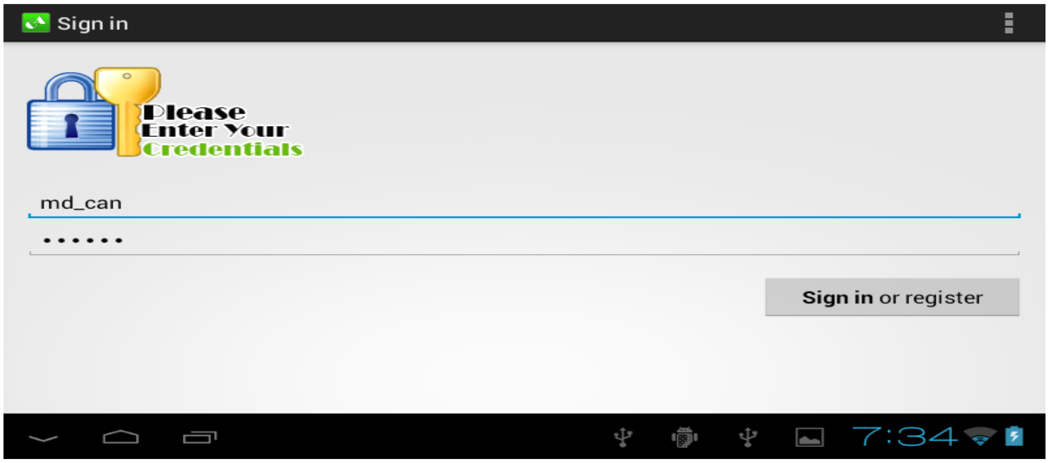
Figure 5-13 RBAC Database ER-Diagram webservice

5.6 Client Agent UI

The following interfaces is part of the client agent interface, where this agent will work as a guard for conditions that its values related directly to context information collected from user context information manager, so at the time the condition become invalid due to change in context information value, the client directly notify controller component within the web service.

Sign in UI

Figure 4-6 interface represents user's authentication step, where users will be able to register and sign in, in order to utilize resources and services.



Sign in

Please Enter Your Credentials

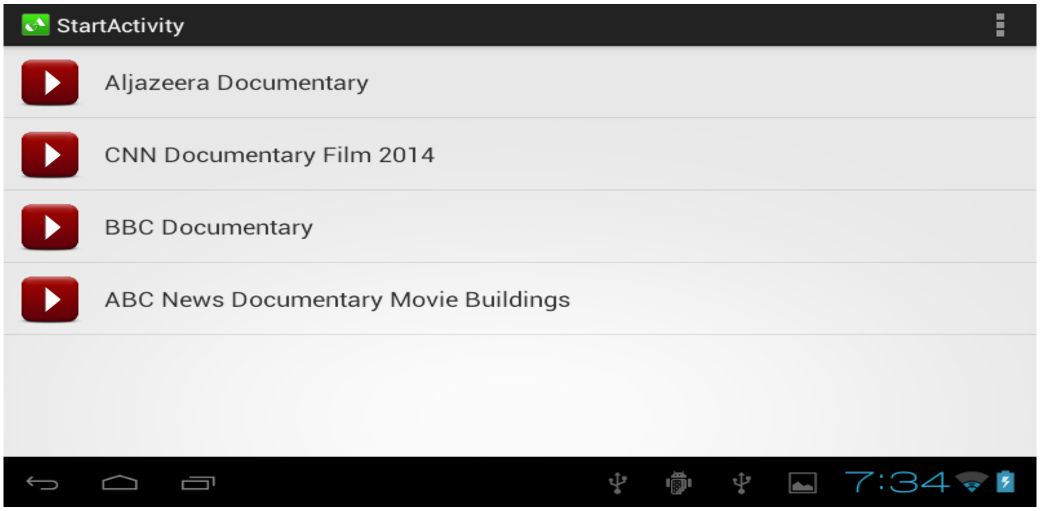
md_can

••••••

Figure 5-14 Login interface

Resources List UI

The following interface in Figure 4-7 lists videos being available to watch, where users able to opt between items.



StartActivity

Aljazeera Documentary

CNN Documentary Film 2014

BBC Documentary

ABC News Documentary Movie Buildings

Figure 5-15 Listing resources menu interface

List Permissions UI

In Figure 4-8 shows action context menu, by continuous press over video items will yields context menu tells you to choose “List Permissions” which in turn list available action could be performed over this resource as shown in Figure 4-9.

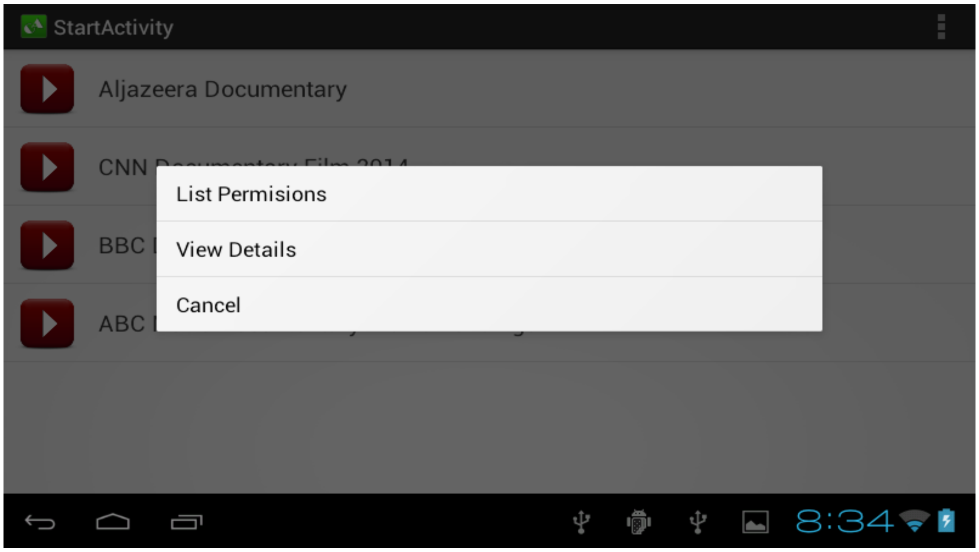


Figure 5-16 Actions context menu interface

List Permissions UI

From the previous menu shown in Figure 4-9 for the selected resource or item, the system based on context information will generate the permissions available for that user on that object or resource.

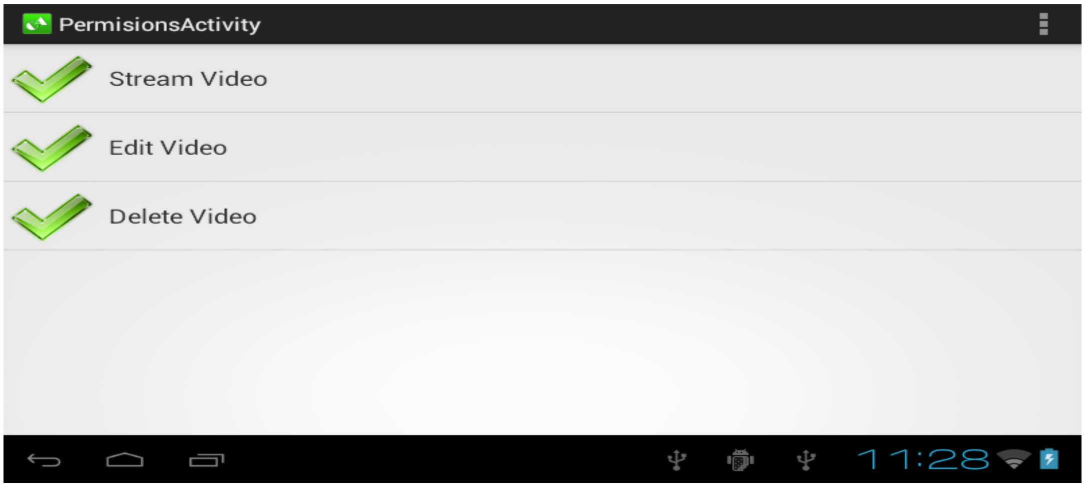


Figure 5-17 Resource permission list interface

5.7 Conclusion

This chapter describes the main implementation issues Study the best model for the development life cycle which will be convenient to accomplish objectives and their specific goals; we try to express the most important functionalities for the system in different representations such as algorithm and flowcharts. The chapters discuss the web service database which used as the backend for storing and managing data. The chapter also introduces the proxy interfaces during interaction to show the framework usage.

6 Framework Testing and Evaluation

This chapter presents the process of evaluating the prototype of the framework for context aware role based access control framework, where quantitative approach adapted to measure performance.

6.1 Experimental setup

We describe in this section what is needed to set experimental environment in order to evaluate the system against evaluation points being under focus. Our framework includes web service implemented using java programming language, and deployed using GlassFish Server 4.0 , the web service used as a back end database MySQL 5.1.37 database management system.

The platform incubate such server applications, is Intel(R) Core™2 Duo CPU P8400 @2.26 GHz, 3072 MB RAM.

Also the experiment includes mobile device with android operating system, has the following specification, 1 GHz dual core Cortex-A9, 800 MB RAM, Wi-Fi 802.11 a/b/g/n dual band, Android OS, v4.4.2 (Kitkat).

All devices are connected using wireless network access point 150 Mbps, Model No: TL-WA701ND.

6.2 Performance

Evaluation of the performance will take response time as criteria to measure system performance, also will measure response time for the most frequent and critical operations and tasks done the web service component, then we will test response time for the proxy component, which in turn calls web service remotely to access services or resources:

6.2.1 Calculation of web service tasks execution time evaluation:

Using JUnit test:

1. Validate User
2. Loading granted roles – FillSR
3. Loading granted Permissions – ValidateAllPerms
4. Loading conditions needed for gaining role permissions – FillCondsByRole

Table 6-1 : Web Service execution time average using JUnit testing

Test Case	Validate User	Session Request	Loading granted Permissions	Loading Local conditions	Total Response Time
1	16	615	85	482	1301
2	10	560	65	417	1152
3	19	636	50	482	1308
4	19	585	43	474	1297
5	13	539	41	508	1212
6	14	526	43	472	1165
7	25	621	53	493	1308
8	11	495	72	413	1090
9	46	501	39	470	1214
10	20	594	55	581	1354
Average (run time)	19.3	567.2	54.6	479.2	1240.1

Table 6-1 represents the results of executing the web service which handle the main evaluation process for access decision made. We didn't include load granted roles, because it's executed once during the session, also our enhancement focus on changes made on short term context where granting roles related to long term context.

6.2.2 Evaluation of proxy execution time:

We evaluate and statistically compare results on the device:

1. Load Local RPC
2. Load Local SPA

Table 6-2 : Samsung Device with Android OS run the framework

Test Case	Validate User	Session Request	Loading granted Permissions	Loading Local conditions	Total Response Time
1	60	805	407	838	3136
2	116	852	315	859	3986
3	119	730	330	828	3207
4	156	948	416	584	3227
5	108	998	525	1277	4802
6	215	935	448	1013	4825
7	101	611	367	764	3230
8	93	572	318	694	3189
9	112	659	145	629	2951
10	99	805	197	776	3163
Average (run time)	117.9	791.5	346.8	826.2	3571.6

As shown in Figure 6-1 based on result collected in Table 6-1, we can see that the session request operation and Loading local conditions operation have high execution time includes the following main operation with high execution time, then we need to analyze session request operation as follows:

1. Establishing a session for the end user.
2. Evaluating the roles assignment based on conditions from preset policy and represented previously in a database table.

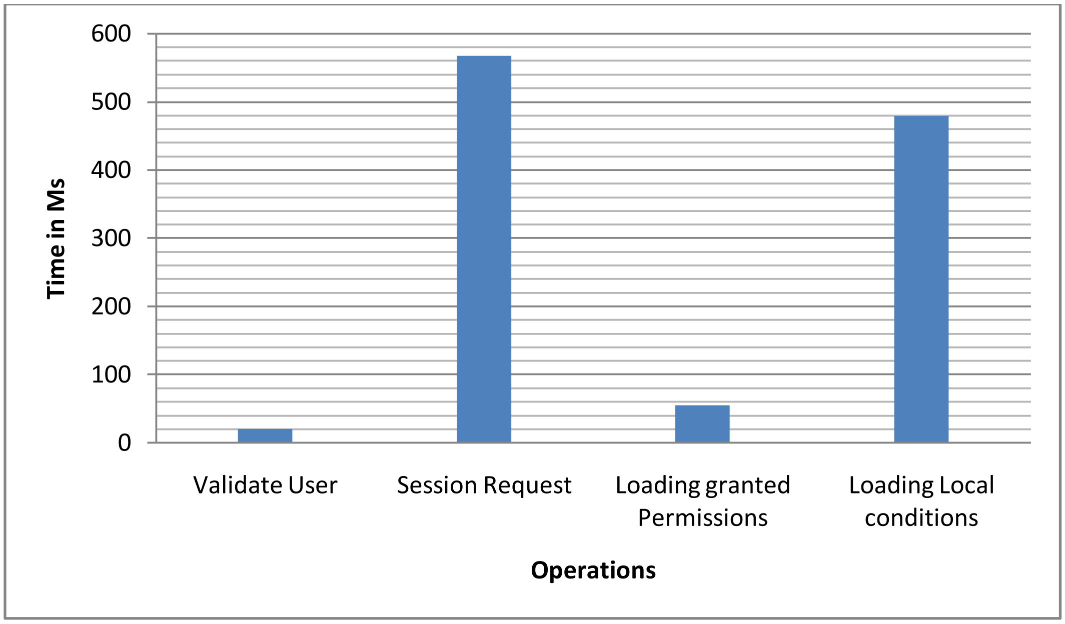


Figure 6-1 operations execution time scale chart

3. Evaluating the permission assignment based on conditions from preset policy and represented inside the database table.
    a. Loading granted Permissions – ValidateAllPerms.
    b. Filling them to the SPA.

To approve our enhancement made by adopting the new approach, let us suppose that we have a session with 100 access request, then the execution frequency for each user as follows:

Table 6-3 : comparison of estimated of tasks execution time frequency during the session

Operation		Estimated Execution Frequency per session	Execution time	Total Execution time
Validate User		1	19.3	19.3
Session Request	**Establishing a session**	1	110	110
	Evaluating the roles assignment based on conditions (FillSR)	1	188	188
	Evaluating the permission assignment (ValidateAllPerms && FillSPA)	100	215	215000
	Loading local permissions	100	54.6	5460

As shown in Table 6-3 the Validate task and FillSR done once at the beginning of the session, which FillSR depends on long term context information as noted before such LTCs doesn't change during the session average time. The most frequent operation which occurs with every change to the context information provided by proxy, such change enforce the framework to revaluate conditions validity, in order to identify which permissions will continue grant or will be revoked.

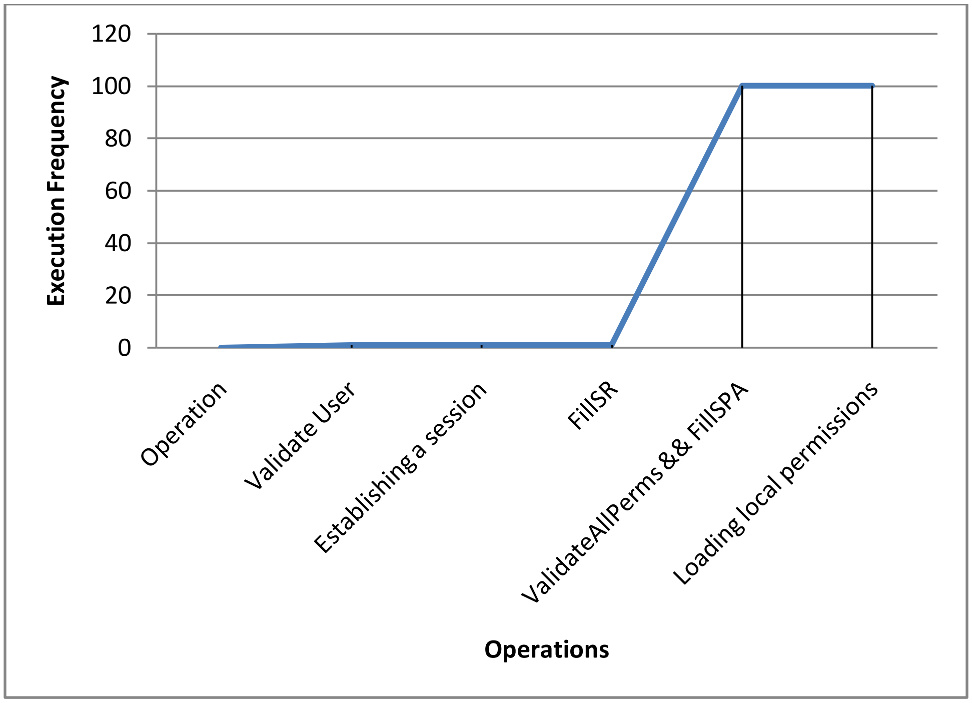


Figure 6-2 Frequency of operations execution during a session

As shown in Figure 6-2, the execution time to the operations ValidateAllPerms and FillSPA which has high frequency in execution almost with each access request, our enhancement made high frequent tasks on the web service to be done on the proxy side for the user, such approach will make heavy load tasks distributed to each user device, so that will enhance the performance, and foster the principle of context information validity, where any change made to user context will be evaluated then as consequence grant or revocation request for permissions will made.

6.3 Security:

We should take into account for matching security CIA triad:

6.3.1 Confidentiality and Integrity:

To address system confidentiality we have two choices:

- Transfer soap request using SSL.
- Using SOAP extension WS-Security.

While the pervasive environment communication resources is expensive and limited, also the computation power is relatively low, where most devices which communicate is low power devices also to decrease battery life impact for portable devices, that leads us to choose less expressive technique to avoid the constraints that we have.

We use SSL as technique to transfer soap messages between the framework parts, through using third party library ksaop2-android, to ensure security matching side by side save system parts performance to be more convenient for pervasive environment.

Also using proxy side to control context information foster integrity, where malicious users will be prevented from injecting or providing the web service with false values for the framework web service which leads for false access control decision.

6.3.2 Availability:

The most risky reasons for falling down or stopping the system from functioning return to the following problems:

6.3.2.1 System failures

Failures such inability to read data sources, or to make comparisons or business logic processing, so we conduct unit testing, to ensure that each functionality works and output results correctly, also we foster during programming testability of the software, like simplicity and independence of modules and components also to improve fault tolerance we use exception handling.

6.3.2.2 Denial of service attacks

Another important drawback for availability is service interruption, as a consequence our system architecture are distributed which in turn simplify and facilitate monitoring of each part, so that’s leads to reduce service bottleneck problem, we used connection pooling for managing connections in order to alleviate any overload made on the system due to user large numbers.

7 Conclusion and Future Works

Context awareness application has increasing significance in various environments and domains; one of these domains is authorization, while authorizations techniques are various we select role based technique as a base for research, such techniques fits the organizational structures, where roles is a matching for persons functional behavior for a specific system or domain.

[16] CAP model selected as reference model to be enhanced and implemented as prototype to accommodate and simulate practical and applicable framework, which could be used in a pervasive environment, where users in such environment connected and disconnected based on their location and many contextual information sensed using surrounding environment or using their own devices connected to the system through it.

In this thesis we investigated and enhanced the model to be applicable to work in pervasive environment, through adding the proxy module, the proxy used to control and enforce the defined policy for a specific domain set by domain administrators, such proxy also responsible for monitoring context information collected by the framework to make access decision, any change occur will notify context manager for the change to make reevaluation for the related permissions or roles granted previously based on such context information.

Also the proxy increasingly enhance the model performance when applied, where the framework server part will not require revaluation for policy rules with each request initiated, instead each device for end user will independently monitor and notify for the change after its occurrence and hold resource utilization if policy rules broken.

The enhanced framework also will enhance system applicability from the security perspective, where malicious change for context information, which intended to change access decision, became impossible.

After design guidelines set, we start development of the framework prototype, and setup the environment required, we tested the environment applicability with multiple devices where the system performance considered convenient for pervasive environment, and works well when using SSL layer for securing SOAP messages.

7.1 Contribution

This thesis has the following principal contributions:

1. Its reviews existing work in context aware computing and its major problems and challenges.

2. Reviews the evolution of role based access control model, and how researchers involved context to be working with.
3. Describes the design and implementation of CAP [16], which transfer theoretical model into a complete framework could be included within any security system.
4. Making an enhancement to CAP [16], represented in enhancing revocation techniques by using proxy component.
5. Making enhancement to the service performance through distributing computation load to the proxy built for each client, which enhance the performance.

7.2 Future Work

The future expansion of enhanced framework has multiple directions, where such of research includes many areas and disciplines need to be enhanced and covered, especially researching the dynamicity of mounting context information from available sensors, focus study for best selection of encryption techniques for messaging which has neutral effect on performance. Also the certainty of context information values should be researched to enhance value measurement with lower battery impact. Also we need applying further study for user privacy preferences to be taken into account when applying the framework policy, another important research issues should be extended this study especially in making the framework has zero configuration to enhance usability in various domains and business environments.

Bibliography

1- Abowd, Gregory D., et al. "Towards a better understanding of context and context-awareness." Handheld and ubiquitous *computing*. Springer Berlin Heidelberg, 1999..

2- Bill Schilit, M.Theimer. “Disseminating Active Map Information to Mobile Hosts” , 1994.

3- Bill Schilit, Norman Adams, Roy Want. “Context-Aware Computing Applications” - 1994.

4- C. K. Georgiadis I. Mavridis, G. Pangalos, and R. K. Thomas “Flexible team-based access control using context”, 2001.

5- D.F.Ferraiolo D.R.Kuhn and R.Chandramouli. “Role-Based Access Control” [Book]. - Artech House Publishers, 2003 (accessed at books.google.com\?id=48AeIhQLWckC)

6- Emami S. S., S. Zokaei. “Context-Senstive Dynamic Role-Based Access Control Model” [Book]. ISecuri, 2010.

7- G. Zhang and M. Parshar. “Context-aware Dynamic Access Control for Pervasive Applications” [Journal],2004.

8- Hong J. and Baker M. “What's New in the Ubicomp Community?”, IEEE JOURNALS & MAGAZINES, 2013. Vol. 12.

9- Jung Hwan Choi Hyunsu Jang and Young Ik Eom. “CA-RBAC: Context Aware RBAC Scheme in Ubiquitos Computing Environments”, 2010.

10- M. Covington M. Moyer, and M. Ahamad. “Generalized role-based access control for securing future applications”, 2000.

11- M. Strembeck G. Neumann. “An integrated approach to engineer and enforce context constraints”, 2004.

12- McDaniel P. “On Context in Authorization Policy”, 2003.

13- Nguyen Tammy. “Context-aware access control in pervasive”, 2005.

14- Patrikakis C. “Security and Privacy in Pervasive Computing”, IEEE, 2007, Vol. 6.

15- R. Sandhu D.F. Ferraiolo, D, R. Kuhn. “The NIST Model for Role Based Access Control: Toward a Unified Standard” , 2000.

16- Sareh Sadat Emami Morteza Amini, Saadan Zokaei. “A Context-Aware Access Control Model for Pervasive Computing Environments”, 2007.

17- Singh Sachin, Puradkar Sushil and Lee Yugyung. “Ubiquitous Computing: Connecting Pervasive Computing through Semantic Web” [Journal]. - [s.l.] : Springer-Verlag, 2005.

18- Dey, Anind K., Daniel Salber and Gregory D. Abowd (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction 16. To appear in 2001.

19- STREMBECK G. NEUMANN and M. “A Scenario-driven role engineering process for functional”, 2002.

20- Tahir, Muhammad Nabeel. "C-RBAC: Contextual role-based access control model." Ubiquitous Computing and Communication Journal 2.3 (2007): 67-74.

21- Thomas R. K. “Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments”, 1997.

22- Tripathi Devdatta Kulkarni and Anand. “Context-Aware Role-based Access Control in Pervasive Computing Systems”, 2008.

23- Weili Han Junjing Zhang, Xiaobo Yao. “Context-sensitive Access Control Model and Implementation”, 2005.

24- Weiser Mark. “The Computer for 21st centuray”, 1991.

25- Y. G. Kim C. J.Mon,D. Jeong,C.Y. Song and D. K. Baik. “Context-aware access control mechanism for ubiquitous applications”, 2003.

26- Zhou Jiehan [et al.] “Pervasive Service Computing: Visions and Challenges” [Conference] // Computer and Information Technology (CIT), 2010 IEEE 10th International Conference. IEEE, 2010.

27- Venkataramana, Y. “Pervasive Computing: Implications, Opportunities and Challenges for the Society”, 2006.

28- Roy W., A. Hopper, Veronica F., and Jonathan G., “The Active Badge Location System, ACM Transactions on Information Systems”, 1992.

29- Jie Y., Weiyi Y., Matthias D., and Alex W., “Smart Sight: A Tourist Assistant System” in Proceedings of the Third International Symposium on Wearable Computers (ISWC), Pittsburgh, Pennsylvania, USA, 1999.

30- Shankar R. P., Brian Lee, Armando F., Pat H.,and Terry W., “ICrafter: A Service Framework for Ubiquitous Computing Environments”. In Proceedings of the Third International Conference on Ubiquitous Computing (UbiComp), pages 56–75, Atlanta, Georgia, USA, 2001.

31- Robert H. and Rupert C., “Movement Awareness for Ubiquitous Game Control”. Personal and Ubiquitous Computing, 6(5-6):407–415, 2002

32- Steve B., John B., Paul C., Luigina C., Martin F., Mike F., Chris G., Tony H., Sten O. H., Shahram I., Tom R., Holger S., andIan T., “Unearthing Virtual History: Using Diverse Interfaces To Reveal Hidden Virtual Worlds”. In Proceedings of the International Conference on Ubiquitous Computing (UbiComp), 2001.

33- Anind K. Dey. “Providing Architectural Support for Building Context-Aware Applications”. PhD thesis, Georgia Institute of Technology, Atlanta, Georgia, USA, 2000.

34- Andy W., Alan J., and Andy H., “A New Location Technique for the Active Office”. IEEE Personal Communications, 4(5):42–47, 1997.

35- Albrecht S., Kofi A. A., Antti T., Urpo T., Kristof V. L., and Walter Van de Velde. “Advanced Interaction in Context”. In Proceedings of the First International Symposium on Handheld and Ubiquitous Computing (HUC), pages 89–101, Karlsruhe, Germany, 1999

36- Harry Chen, Tim Finin, and Anupam Joshi. An Ontology for Context-Aware Pervasive Computing Environments. Knowledge Engineering Review, 18(3):197–207, 2003.

37- Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. DAML+OIL Reference Description. Project Website available at http://www.w3.org/TR/daml+oil-reference, 2001. (13 March 2014).

38- Nick Ryan. ConteXtML: Exchanging Contextual Information between a Mobile Client and the FieldNote Server. Language specifications available at http://www.cs.kent.ac.uk/projects/mobicomp/fnc/ConteXtML.html, 1999.

39- Cheverst, Keith, Keith Mitchell, and Nigel Davies. "Design of an object model for a context sensitive tourist GUIDE." Computers & Graphics 23.6 (1999): 883-891.

40- Pascoe, Jason. "The stick-e note architecture: extending the interface beyond the user." Proceedings of the 2nd international conference on intelligent user interfaces. ACM, 1997.

41- Ebling, Maria, Guerney DH Hunt, and Hui Lei. "Issues for context services for pervasive computing." Middleware 2001 Workshop on Middleware for Mobile Computing, Heidelberg. 2001.

42- Sandhu, Ravi S., and Pierangela Samarati. "Access control: principle and practice." Communications Magazine, IEEE 32.9 40-48, 1994.

43- Ferraiolo, David, Janet Cugini, and D. Richard Kuhn. "Role-based access control (RBAC): Features and motivations." Proceedings of 11th Annual Computer Security Application Conference. 1995.

44- Gomez Laurant, Olk, Eddy. "Context-Aware Access Control Making Access Control Decisions Based on Context Information." Mobile and Ubiquitous Systems - Workshops, 2006. 3rd Annual International Conference on, San Jose, CA, 2006

45- Alessandra Toninelli1, Rebecca Montanari1, Lalana Kagal2, and Ora Lassila. "Context A Semantic Context-Aware Access Control Framework for Secure Collaborations in Pervasive Computing Environments." 5th International Semantic Web Conference, ISWC 2006. 2006

46- Michael J. Covington,Wende Long, Srividhya Srinivasan, Anind K. Dey, Mustaque Ahamad, Gregory D. Abowd."Securing context-aware applications using environment roles". Proceeding SACMAT '01 Proceedings of the sixth ACM symposium on Access control models and technologies, 2001.

47- Claudio Ardgana, Ernesto Damiani, Sabrina Vimercati,Pierangela amarati. A web service architecture for enforcing access control policies".Proceedings of the First International Workshop on Views on Designing Complex Architectures, 2004.

48- Vukovic, Maja, and Peter Robinson. "Context aware service composition." University of Cambridge, Technical Report UCAM-CL-TR-700, 2007.

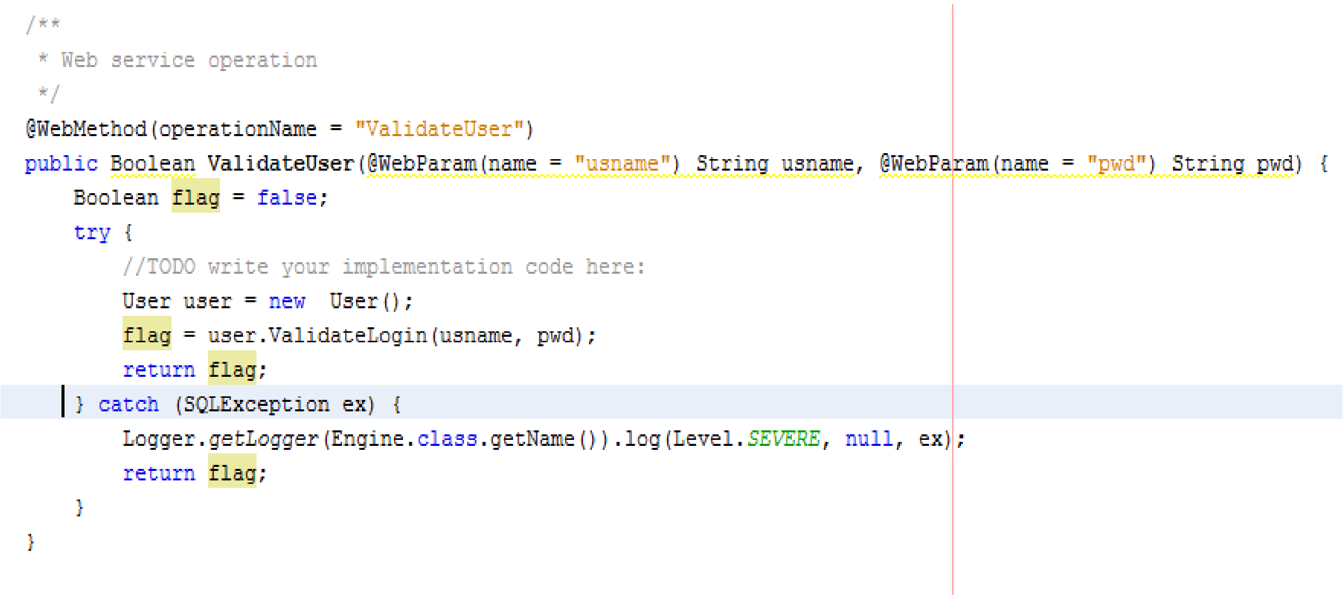
49- OASIS Standard Specififcation, eXtensible Access Control Markup Language (XACML) Version 3.0, 2010.

50- http://www.awareframework.com/home/ last access [28 May 2014]

51- Bardram, Jakob E. "Applications of context-aware computing in hospital work: examples and design principles." Proceedings of the 2004 ACM symposium on applied computing. ACM, 2004.

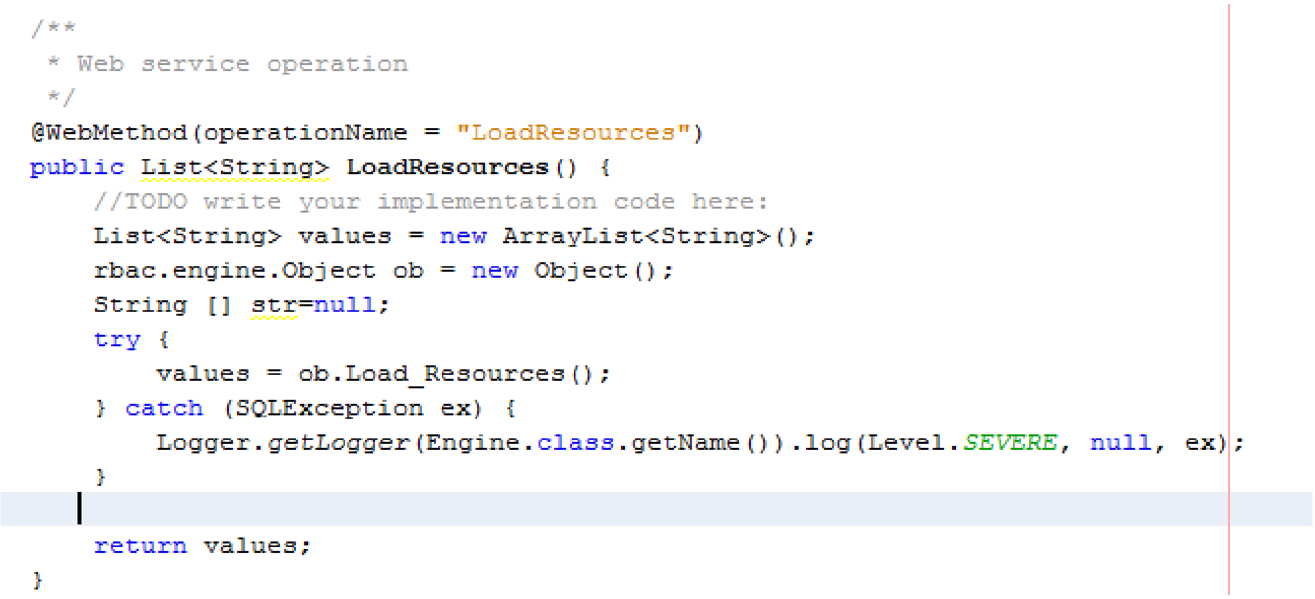
Appendix A

Web service basic source code



```java
/**
 * Web service operation
 */
@WebMethod(operationName = "ValidateUser")
public Boolean ValidateUser(@WebParam(name = "usname") String usname, @WebParam(name = "pwd") String pwd) {
    Boolean flag = false;
    try {
        //TODO write your implementation code here:
        User user = new  User();
        flag = user.ValidateLogin(usname, pwd);
        return flag;
    } catch (SQLException ex) {
        Logger.getLogger(Engine.class.getName()).log(Level.SEVERE, null, ex);
        return flag;
    }
}
```

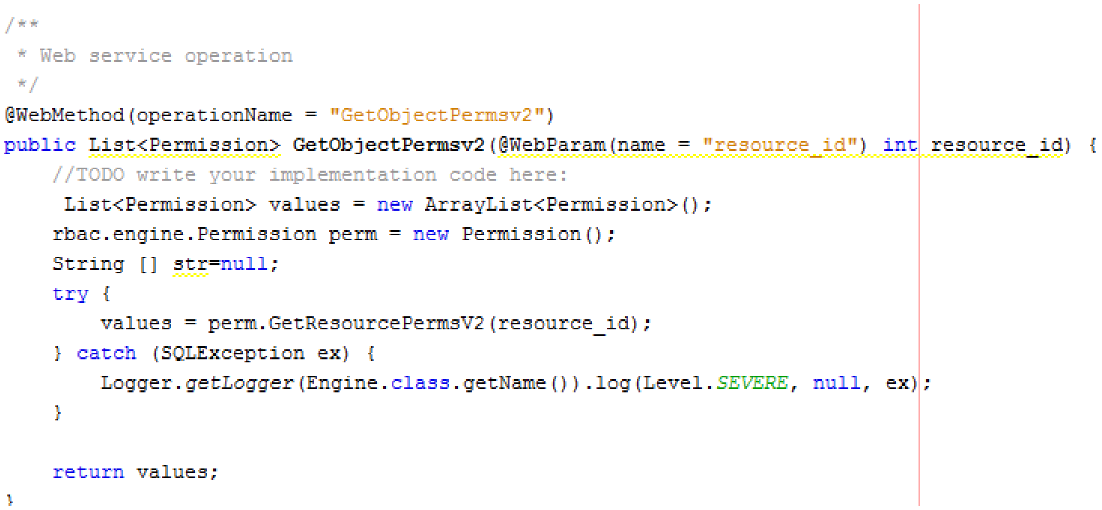
Figure A- 1 User login method



```java
/**
 * Web service operation
 */
@WebMethod(operationName = "LoadResources")
public List<String> LoadResources() {
    //TODO write your implementation code here:
    List<String> values = new ArrayList<String>();
    rbac.engine.Object ob = new Object();
    String [] str=null;
    try {
        values = ob.Load_Resources();
    } catch (SQLException ex) {
        Logger.getLogger(Engine.class.getName()).log(Level.SEVERE, null, ex);
    }

    return values;
}
```

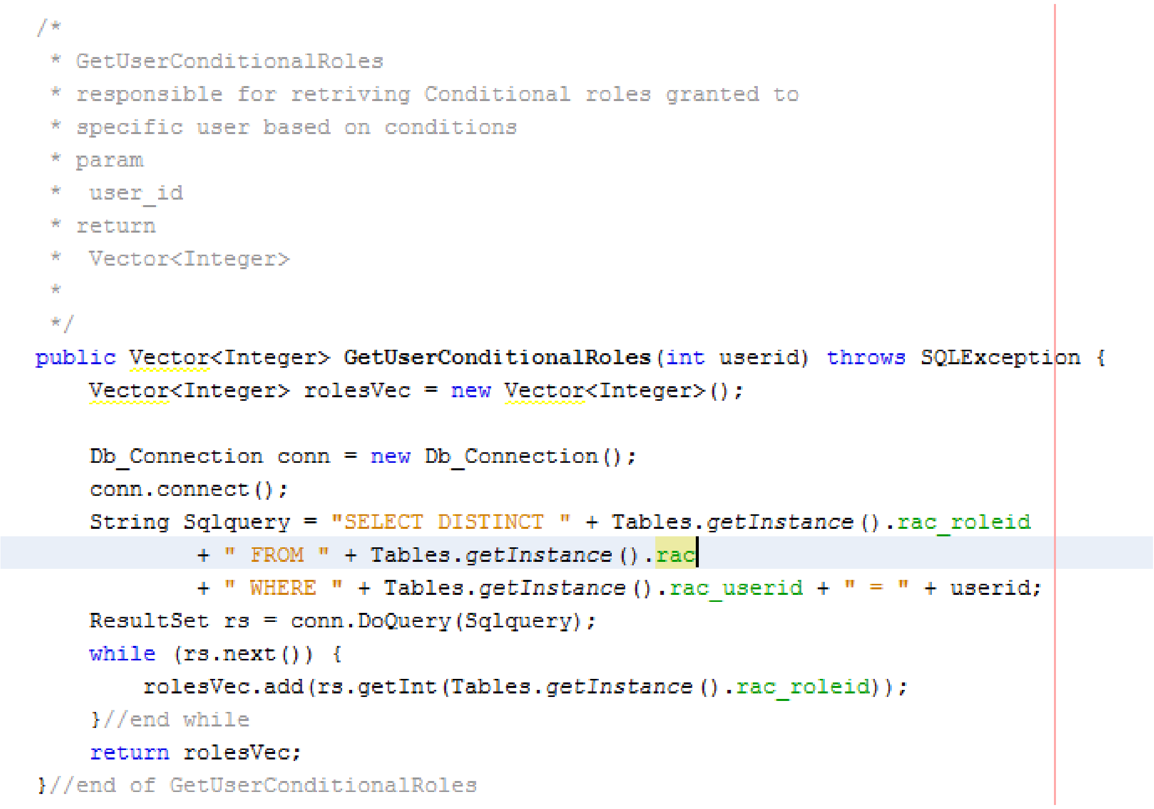
Figure A- 2 : Loading resources



```
/**
 * Web service operation
 */
@WebMethod(operationName = "GetObjectPermsv2")
public List<Permission> GetObjectPermsv2(@WebParam(name = "resource_id") int resource_id) {
    //TODO write your implementation code here:
     List<Permission> values = new ArrayList<Permission>();
    rbac.engine.Permission perm = new Permission();
    String [] str=null;
    try {
        values = perm.GetResourcePermsV2(resource_id);
    } catch (SQLException ex) {
        Logger.getLogger(Engine.class.getName()).log(Level.SEVERE, null, ex);
    }

    return values;
}
```

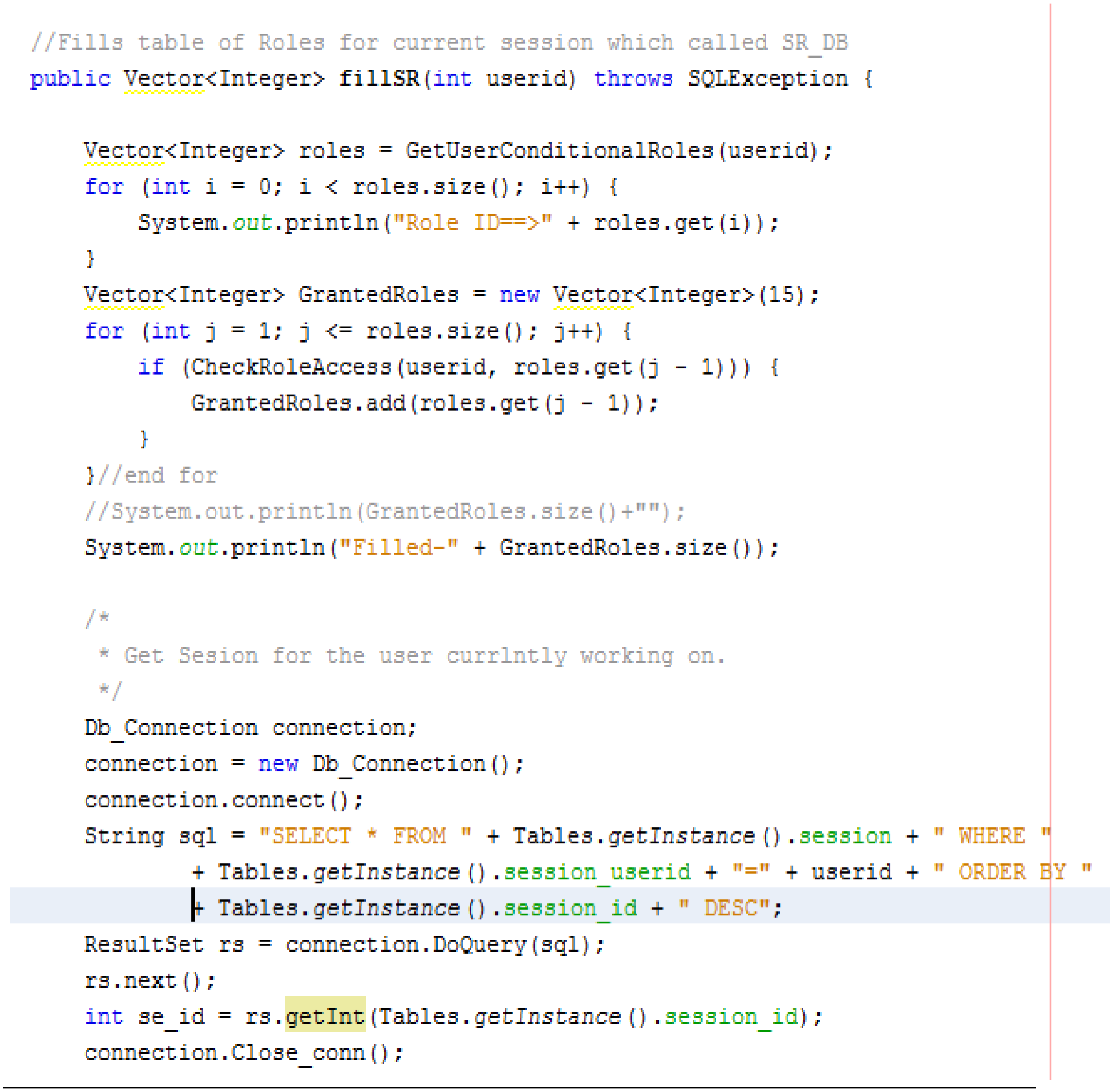
Figure A- 3: Loading permissions



```
/*
 * GetUserConditionalRoles
 * responsible for retriving Conditional roles granted to
 * specific user based on conditions
 * param
 *  user_id
 * return
 *  Vector<Integer>
 *
 */
public Vector<Integer> GetUserConditionalRoles(int userid) throws SQLException {
    Vector<Integer> rolesVec = new Vector<Integer>();

    Db_Connection conn = new Db_Connection();
    conn.connect();
    String Sqlquery = "SELECT DISTINCT " + Tables.getInstance().rac_roleid
            + " FROM " + Tables.getInstance().rac
            + " WHERE " + Tables.getInstance().rac_userid + " = " + userid;
    ResultSet rs = conn.DoQuery(Sqlquery);
    while (rs.next()) {
        rolesVec.add(rs.getInt(Tables.getInstance().rac_roleid));
    }//end while
    return rolesVec;
}//end of GetUserConditionalRoles
```

Figure A- 4: Retrieve conditional roles in RAC

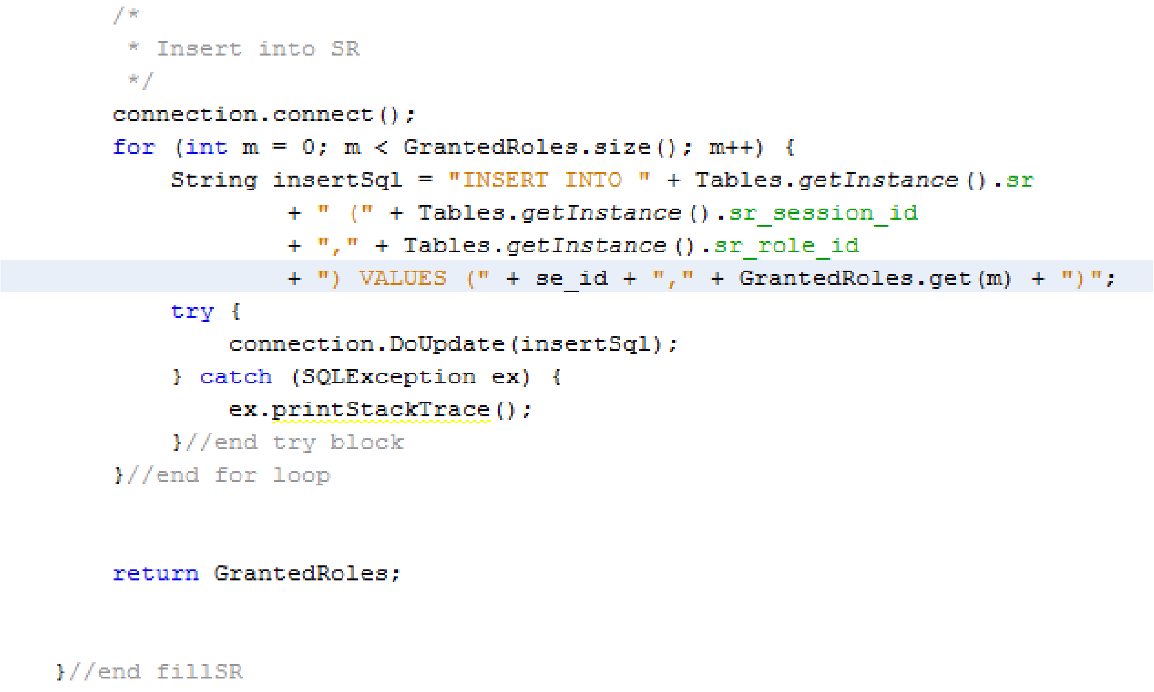


```
//Fills table of Roles for current session which called SR_DB
public Vector<Integer> fillSR(int userid) throws SQLException {

    Vector<Integer> roles = GetUserConditionalRoles(userid);
    for (int i = 0; i < roles.size(); i++) {
        System.out.println("Role ID==>" + roles.get(i));
    }
    Vector<Integer> GrantedRoles = new Vector<Integer>(15);
    for (int j = 1; j <= roles.size(); j++) {
        if (CheckRoleAccess(userid, roles.get(j - 1))) {
            GrantedRoles.add(roles.get(j - 1));
        }
    }//end for
    //System.out.println(GrantedRoles.size()+"");
    System.out.println("Filled-" + GrantedRoles.size());

    /*
     * Get Sesion for the user currlntly working on.
     */
    Db_Connection connection;
    connection = new Db_Connection();
    connection.connect();
    String sql = "SELECT * FROM " + Tables.getInstance().session + " WHERE "
            + Tables.getInstance().session_userid + "=" + userid + " ORDER BY "
            + Tables.getInstance().session_id + " DESC";
    ResultSet rs = connection.DoQuery(sql);
    rs.next();
    int se_id = rs.getInt(Tables.getInstance().session_id);
    connection.Close_conn();
```

Figure A- 5 : Fills Session-Role Table Part 1



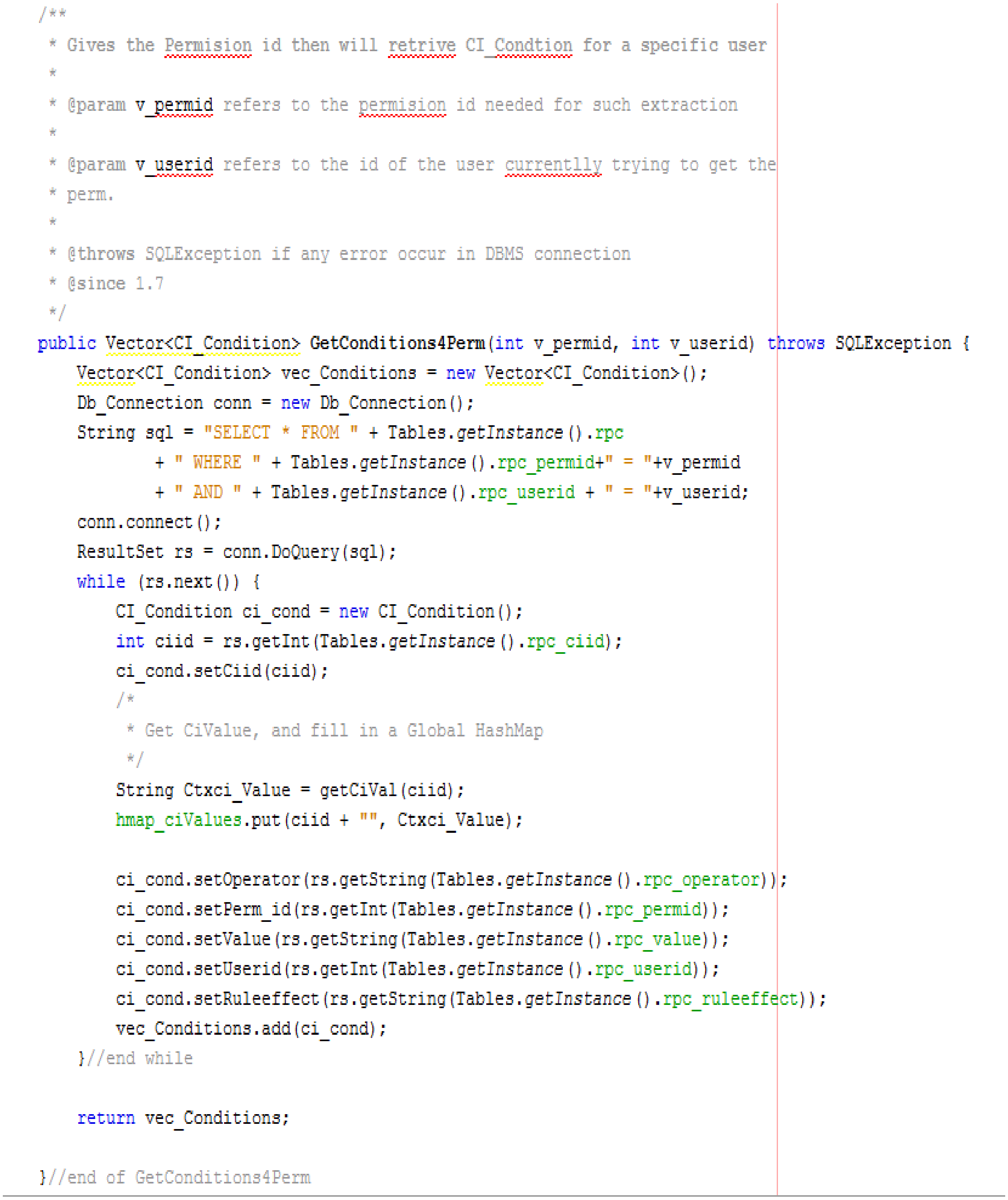
```
        /*
         * Insert into SR
         */
        connection.connect();
        for (int m = 0; m < GrantedRoles.size(); m++) {
            String insertSql = "INSERT INTO " + Tables.getInstance().sr
                    + " (" + Tables.getInstance().sr_session_id
                    + "," + Tables.getInstance().sr_role_id
                    + ") VALUES (" + se_id + "," + GrantedRoles.get(m) + ")";
            try {
                connection.DoUpdate(insertSql);
            } catch (SQLException ex) {
                ex.printStackTrace();
            }//end try block
        }//end for loop


        return GrantedRoles;


    }//end fillSR
```

Figure A- 6 : Fills Session-Role Table Part 2

Permission Authorizer main code:



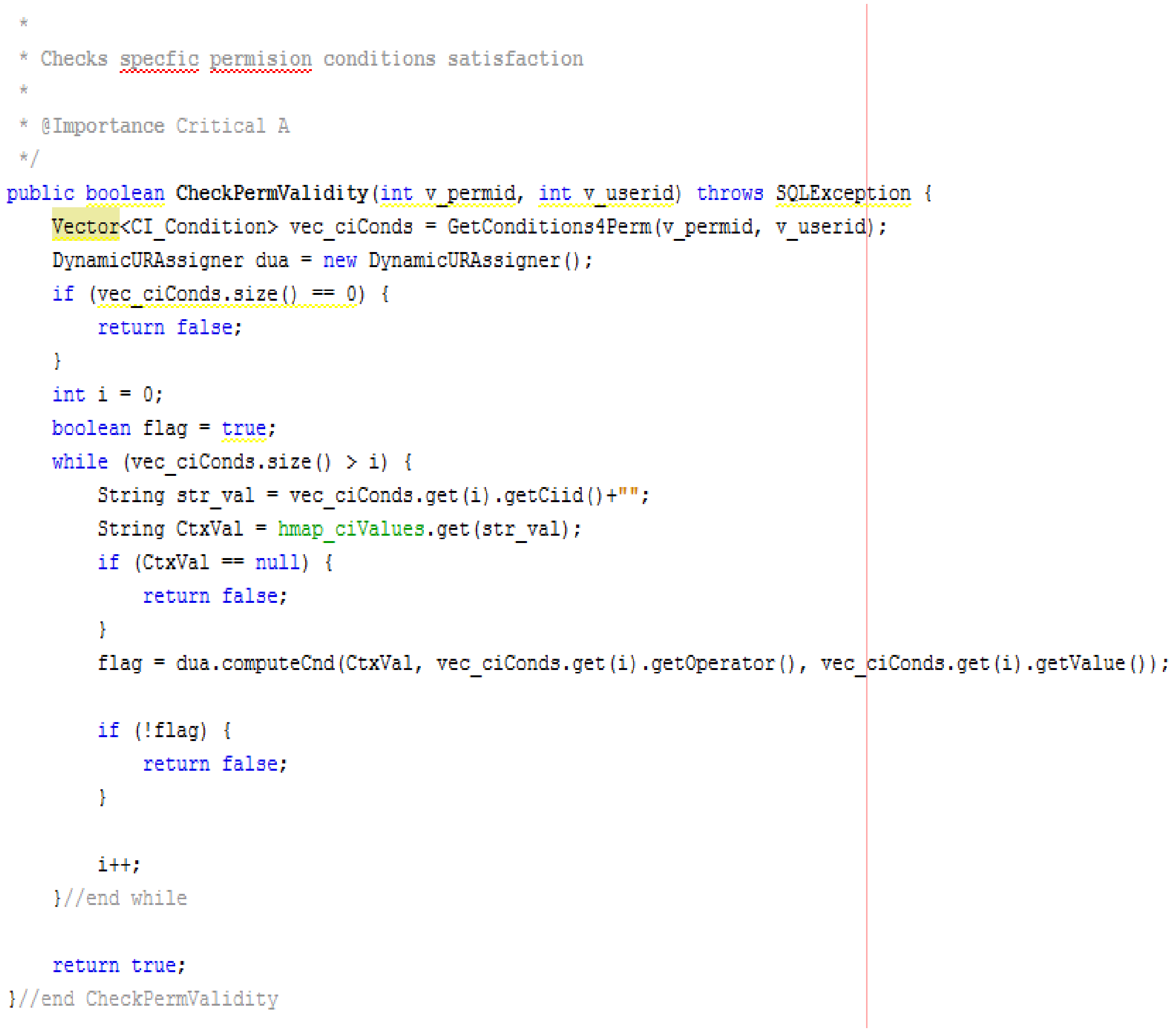
```java
/**
 * Gives the Permision id then will retrive CI_Condtion for a specific user
 *
 * @param v_permid refers to the permision id needed for such extraction
 *
 * @param v_userid refers to the id of the user currentlly trying to get the
 * perm.
 *
 * @throws SQLException if any error occur in DBMS connection
 * @since 1.7
 */
public Vector<CI_Condition> GetConditions4Perm(int v_permid, int v_userid) throws SQLException {
    Vector<CI_Condition> vec_Conditions = new Vector<CI_Condition>();
    Db_Connection conn = new Db_Connection();
    String sql = "SELECT * FROM " + Tables.getInstance().rpc
            + " WHERE " + Tables.getInstance().rpc_permid+" = "+v_permid
            + " AND " + Tables.getInstance().rpc_userid + " = "+v_userid;
    conn.connect();
    ResultSet rs = conn.DoQuery(sql);
    while (rs.next()) {
        CI_Condition ci_cond = new CI_Condition();
        int ciid = rs.getInt(Tables.getInstance().rpc_ciid);
        ci_cond.setCiid(ciid);
        /*
         * Get CiValue, and fill in a Global HashMap
         */
        String Ctxci_Value = getCiVal(ciid);
        hmap_ciValues.put(ciid + "", Ctxci_Value);

        ci_cond.setOperator(rs.getString(Tables.getInstance().rpc_operator));
        ci_cond.setPerm_id(rs.getInt(Tables.getInstance().rpc_permid));
        ci_cond.setValue(rs.getString(Tables.getInstance().rpc_value));
        ci_cond.setUserid(rs.getInt(Tables.getInstance().rpc_userid));
        ci_cond.setRuleeffect(rs.getString(Tables.getInstance().rpc_ruleeffect));
        vec_Conditions.add(ci_cond);
    }//end while

    return vec_Conditions;

}//end of GetConditions4Perm
```

Figure A- 7 : Retrieve conditions for permission from RPC



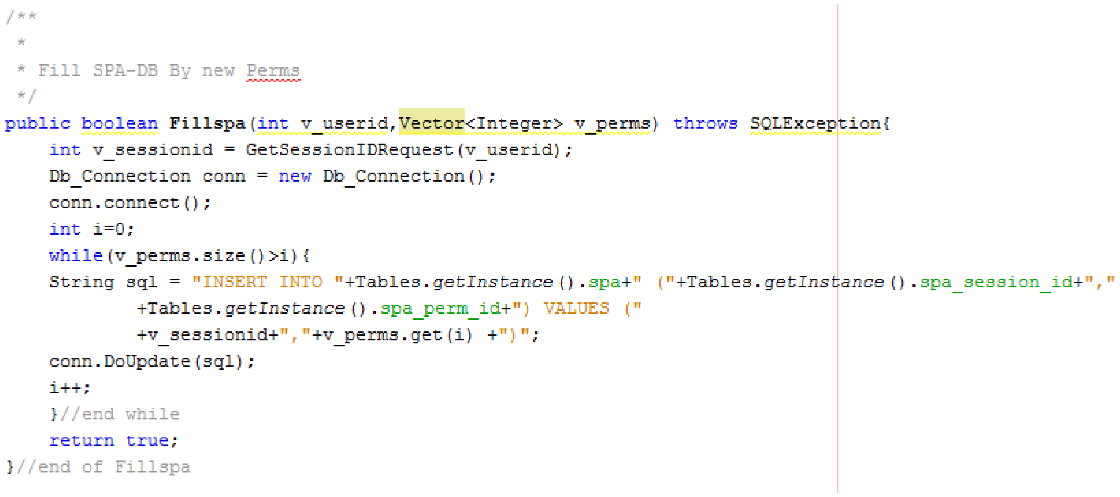
```
  *
  * Checks specfic permision conditions satisfaction
  *
  * @Importance Critical A
  */
 public boolean CheckPermValidity(int v_permid, int v_userid) throws SQLException {
     Vector<CI_Condition> vec_ciConds = GetConditions4Perm(v_permid, v_userid);
     DynamicURAssigner dua = new DynamicURAssigner();
     if (vec_ciConds.size() == 0) {
         return false;
     }
     int i = 0;
     boolean flag = true;
     while (vec_ciConds.size() > i) {
         String str_val = vec_ciConds.get(i).getCiid()+"";
         String CtxVal = hmap_ciValues.get(str_val);
         if (CtxVal == null) {
             return false;
         }
         flag = dua.computeCnd(CtxVal, vec_ciConds.get(i).getOperator(), vec_ciConds.get(i).getValue());

         if (!flag) {
             return false;
         }

         i++;
     }//end while

     return true;
 }//end CheckPermValidity
```

Figure A- 8 : Responsible for checking validity of x permssion for a user based on RPC

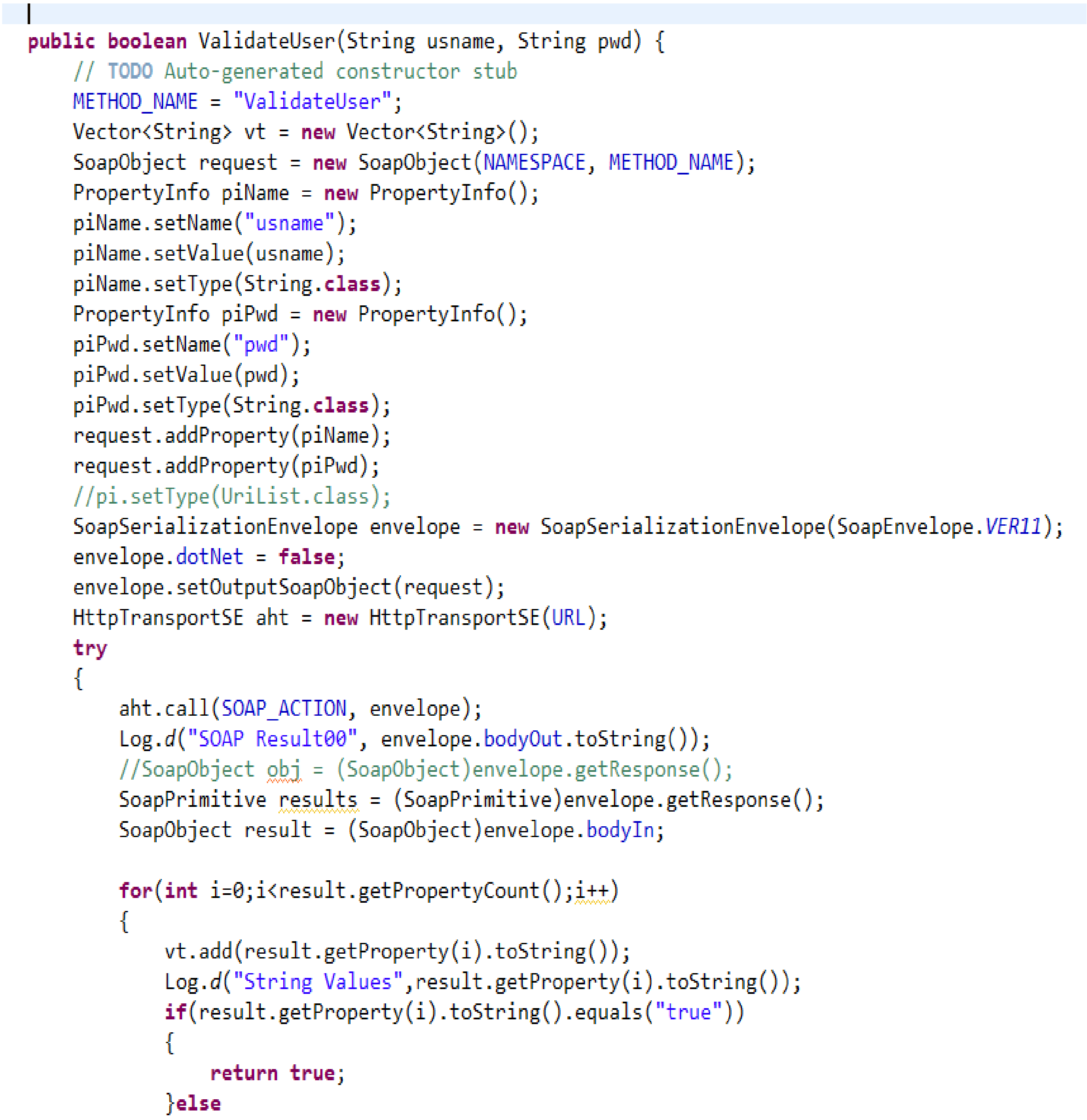


```
/**
 *
 * Fill SPA-DB By new Perms
 */
public boolean Fillspa(int v_userid,Vector<Integer> v_perms) throws SQLException{
    int v_sessionid = GetSessionIDRequest(v_userid);
    Db_Connection conn = new Db_Connection();
    conn.connect();
    int i=0;
    while(v_perms.size()>i){
    String sql = "INSERT INTO "+Tables.getInstance().spa+" ("+Tables.getInstance().spa_session_id+","
            +Tables.getInstance().spa_perm_id+") VALUES ("
            +v_sessionid+","+v_perms.get(i) +")";
    conn.DoUpdate(sql);
    i++;
    }//end while
    return true;
}//end of Fillspa
```

Figure A- 9 : Fills SPA after Evaluation

Appendix B

Proxy basic source code



```
public boolean ValidateUser(String usname, String pwd) {
    // TODO Auto-generated constructor stub
    METHOD_NAME = "ValidateUser";
    Vector<String> vt = new Vector<String>();
    SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
    PropertyInfo piName = new PropertyInfo();
    piName.setName("usname");
    piName.setValue(usname);
    piName.setType(String.class);
    PropertyInfo piPwd = new PropertyInfo();
    piPwd.setName("pwd");
    piPwd.setValue(pwd);
    piPwd.setType(String.class);
    request.addProperty(piName);
    request.addProperty(piPwd);
    //pi.setType(UriList.class);
    SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);
    envelope.dotNet = false;
    envelope.setOutputSoapObject(request);
    HttpTransportSE aht = new HttpTransportSE(URL);
    try
    {
        aht.call(SOAP_ACTION, envelope);
        Log.d("SOAP Result00", envelope.bodyOut.toString());
        //SoapObject obj = (SoapObject)envelope.getResponse();
        SoapPrimitive results = (SoapPrimitive)envelope.getResponse();
        SoapObject result = (SoapObject)envelope.bodyIn;

        for(int i=0;i<result.getPropertyCount();i++)
        {
            vt.add(result.getProperty(i).toString());
            Log.d("String Values",result.getProperty(i).toString());
            if(result.getProperty(i).toString().equals("true"))
            {
                return true;
            }else
```

Figure B- 1 : Call webserivce method ValidateUser