УДК 004.056.5

О.Ю. Сєдушев, Є.В. Буров
Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж

ВИКОРИСТАННЯ ТЕХНОЛОГІЙ ЕКСПЕРТНИХ СИСТЕМ ДЛЯ СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ УПРАВЛІННЯ ПРАВАМИ ДОСТУПУ ДО ІНФОРМАЦІЙНИХ РЕСУРСІВ

Показано інтеграцію моделі управління правами доступу RBAC із технологіями експертних систем для побудови інтелектуальної системи управління правами доступу. Як результат, новостворена система має деякі переваги серед аналогів та підтримує автоматизованість завдяки продукційним правилам бази знань та принципам самонавчання.

Ключові слова: база знань, контроль доступу, експертна система, модель RBAC

In this paper integration of RBAC (Role-based access control) mechanism with an expert system technology for building modern intellectual access control systems is presented. As a result, intellectual access control system is created which if compared to existing systems proposes additionally some automatization based on rules in knowledge base and self-teaching.

Key words: access control, expert system, knowledge base, RBAC model

Вступ

Корпоративні застосування – необхідний в наші дні інструмент підвищення ефективності бізнесу. У великих компаніях співробітники використовують в середньому від двох до десяти, а іноді і більше, бізнес-програм та працюють із великою кількістю корпоративних об'єктів. Усі вони, згідно з сьогоднішніми вимогами безпеки, вимагають аутентифікаційної та авторизаційної інформації.

Велика кількість критичних для бізнесу багатокористувацьких інформаційних систем, збільшення кількості користувачів ведуть до зниження рівня інформаційної безпеки, зростання витрат на адміністрування IT-систем, ускладнення процедур узгодження прав доступу.

Постановка проблеми

У зв'язку з постійно зростаючою кількістю інформаційних систем виникає проблема управління ідентифікаційною інформацією користувачів та правами доступу до інформаційних ресурсів для них. Ця проблема неминуче позначається на інформаційній безпеці підприємства і породжує багато складнощів та ризиків в різних сферах діяльності.

Провідні консалтингові компанії світу визнають, що тематика управління правами доступу до корпоративних інформаційних ресурсів є основоположною у стратегіях інформаційної безпеки компаній і найактуальніша на підприємствах, що мають розвинену IT-інфраструктуру та велику кількість співробітників, таких як банки і фінансові організації, телекомунікаційні компанії, великі холдинги, нафтогазові і енергетичні компанії.

За відсутності механізмів централізованого автоматизованого управління правами доступу співробітників великі компанії можуть зіткнутися з багатьма проблемними факторами, серед яких низький рівень захисту інформації, довготривалість узгодження і надання прав доступу до інформаційних ресурсів та значні трудовитрати на зміну прав доступу працівників.

Аналіз останніх досліджень і публікацій

У комп'ютерній безпеці під поняттям контроль доступу мається на увазі здатність дозволити або відхилити використання (доступ до) специфічного системного ресурсу (об'єкта) специфічному суб'єкту. Під доступом розуміється можливість виконання якоїсь дії над комп'ютерним інформаційним ресурсом, тобто його використання, перегляд чи зміну (оновлення вмісту) [1].

Загальним підходом для усіх моделей управління доступом є поділ множини сутностей, з яких складається система, на множини об'єктів та суб'єктів. Визначення понять об'єкт і суб'єкт

можуть істотно відрізнятися. Варто усвідомлювати, що об'єктами є деякі контейнери з інформацією, а суб'єктами – користувачі, які виконують різноманітні операції над цими об'єктами.

У багатьох організаціях кінцеві користувачі не володіють інформацією чи об'єктами, доступ до яких у них є [2]. Для цих організацій фактичними власниками системних об'єктів, а також програм, які працюють з цими об'єктами, є корпорація або підрозділ корпорації. Контроль часто заснований на функціях працівника, ніж на власності даних.

Сьогодні можна виділити три основні моделі управління доступом до об'єктів:

1) мандатну (Mandatory Access Control - MAC);

2 вибіркову (Discretionary Access Control - DAC);

3) рольову (Role-based Access Control - RBAC).

У рольовій моделі [3] операції, які необхідно виконувати у межах якого-небудь службового обов'язку користувача системи, групуються в набір, що називається роллю. Рішення щодо контролю доступу часто визначені апріорі завдяки ролям, які відіграють користувачі, будучи "частинкою" організації [2]. Кожен користувач системи грає в ній одну або декілька ролей. Виконання користувачем певної дії дозволено, якщо в наборі його ролей наявна потрібна.

У цій моделі у об'єктів немає певних власників. Уся інформація розцінюється як така, що належить організації, яка володіє системою. Відповідно, і ролі користувача усередині системи – це ролі, які він грає в цій організації. Як наслідок, користувачеві неможливо делегувати права на якийсь певний об'єкт. Або у нього є доступ до усіх подібних об'єктів системи, або немає.

Отже, перевагою рольової моделі перед вибірковою є простота адміністрування: призначення користувачів на певні ролі і створення нових ролей не створює жодних труднощів. Це дозволяє розглядати рольову модель як найбільш підходящу для використання у прикладних програмах.

За словами Санді (1996) адміністраторські функції у системах з моделлю RBAC полягають у можливості модифікації таких відношень [4]:

- Користувач – роль (user assignment relations);
- Роль – повноваження (permission assignment relations);
- Роль – роль у рольових ієрархіях (role hierarchy relations).

Роль – це абстрактний опис поведінки та взаємовідносин між працівниками організацій [5]. Це також інтерпретований набір транзакцій (під транзакцією мається на увазі пов'язаність процедури перетворення і доступу зберігання даних), які користувач або група користувачів може виконувати у контексті організації [2]. Вона є доволі стійкою та ефективною. Ролі є складовою частиною ієрархічної рольової структури організації, яка проектується відповідно до вимог бізнес-процесів організації та інтерпретації прав доступу у них.

Згідно з концепцією моделі RBAC взаємна асоціація між користувачами та їхніми правами здійснюється через ролі, які надаються (прив'язуються) користувачам (суб'єктам). Користувач асоціюється з роллю, яка, своєю чергою, зіставлена з правами доступу. Отже, користувач набуває певних прав шляхом отримання ролі. Отже, ролі є певними транзитивними ланками між користувачами та правами доступу.

Політика RBAC заснована на ролях суб'єктів і може конкретизувати політику захисту шляхом побудови ієрархічної рольової структури, яка передбачає усі можливі випадки для надання тих чи інших прав доступу.

У результаті, моделі RBAC забезпечують інтуїтивну підтримку для вираження організаційної політики контролю доступу, особливо у разі проблем захисту у середовищах взаємодіючих систем менеджменту бізнес-процесів (collaborative business process management systems) [6].

Проте, звичайна RBAC-модель містить недолік, який полягає у нездатності специфікувати контроль доступу для окремих користувачів у межах конкретної ролі і недієвості підтримки доступу лише до окремих об'єктів деякої частини інформаційного проекту організації. Для взаємодіючих середовищ це становить велику проблему, адже дуже часто виникає необхідність у наданні специфічних прав користувачу у межах ролі для роботи над об'єктом [7].

Хоча модель RBAC схвально сприйняли дослідники, виявилося, що її доволі важко застосувати до існуючої бізнес-системи будь-якої організації, оскільки та, своєю чергою, не передбачала існування ролей та рольової ієрархії, у результаті чого необхідно проводити реінжиніринг бізнес-процесів організації чи компанії, що, звичайно, позначається на їхньому фінансовому стані. У праці [8] спеціалісти наводять такі проблеми стосовно цього питання:

- існуюча система у загальному випадку не пристосована до використання ролей;
- надзвичайно важко додати функції контролю доступу моделі RBAC до кожного програмного модуля існуючої системи.

Ці самі спеціалісти у своїй публікації намагаються вирішити наведені проблеми. У їхній праці наведена реалізація сервісно-орієнтованого підходу, побудованого на агентах, що дає змогу інтегруватися існуючій системі з RBAC-моделлю. Можна вважати це доволі хорошим відкриттям, адже доволі природно розглядати великі системи у вигляді сервісів та послуг, які вони надають.

Відповідно до інформації у [9], RBAC-модель знайшла своє застосування у двох важливих класах комерційного програмного забезпечення: реляційних СУБД та корпоративних продуктах адміністрування систем безпеки. Мотивації до використання RBAC-моделі у цих двох класах програмного забезпечення є різними. У СУБД-продуктах модель RBAC утворює цілісний інтегрований компонент механізму контролю доступу, і в результаті цього контроль доступу до об'єктів бази даних відбувається під управлінням СУБД-продукту.

У корпоративних продуктах адміністрування систем безпеки модель RBAC використана як абстрактна високорівнева модель для стеження за авторизаціями до ресурсів підприємства у різноманітних гетерогенних системах (операційних системах та прикладних програмах). Автори називають її авторизаційною моделлю підприємства.

Можна знайти й інші сфери застосувань RBAC-моделей. Як відзначив Боззон (Bozzon, 2007), модель RBAC можна використати у системах управління веб-навчанням (Web-based Learning Management Systems – Web-based LMS) [10], які використовуються у світі електронної освіти. У цьому аспекті механізми управління доступом застосовуються до даних, що містяться у індексах пошукових механізмів. Його стаття присвячена інтеграції механізмів RBAC для взаємодії з пошуковим механізмом у Web-based LMS. Ключовою позицією при цьому Боззон вважає реалізацію репозиторію знань з механізмом прав доступу для управління його вмістом.

Ще одним полем застосування RBAC-моделі є глобальні та локальні мережі. Про таке застосування йдеться у праці [11]. Корейські науковці вважають, що для вирішення гетерогенності та комплексності у домашніх мережах допоможе OSGi (Open Service Gateway initiative). Головними проблемами безпеки у домашній мережі є аутентифікація та авторизація користувача, який отримує доступ до мережі. Очевидно, що, як зазначають автори, є деяка подібність у RBAC-моделі та gateway-вході, оскільки вони обидва приймають рішення про надання чи заборону доступу.

Доволі цікавою за темою є праця [12], у якій йдеться про фреймворк для RBAC-моделі у мережевих файлових системах. Автори скорочено назвали його FRAC (Framework for Role-based Access Control). FRAC – це моніторингова служба, що базується на політиці RBAC і контролює потік повідомлень, який циркулює між клієнтами та серверами файлової системи. Автори статті реалізували FRAC для NFS (Network File System).

До того ж модель RBAC має деякі обмеження, які іноді дуже утруднюють її використання. У праці [13] Майоров (2008) чітко та послідовно описує обмеження базової рольової моделі:

1) усі ролі є глобальними;

2) відсутність власника об'єкта;

3) операції належать ролям.

Перше обмеження полягає у тому, що користувач приймає свої ролі по відношенню до усієї системи відразу. Відповідно для системи немає різниці у правах між двома користувачами, що перебувають на однаковій посаді, навіть якщо вони обіймають ці посади у різних підрозділах. Наприклад, будь-який користувач в ролі "начальник відділу" має право управляти будь-яким відділом своєї організації, а це, звичайно, неправильно. Рішенням такого обмеження може бути

розбиття усієї безлічі об’єктів системи на декілька підмножин (доменів) і надання користувачам можливостей грати різні ролі в різних доменах системи. Такий підхід застосовується, наприклад, у бібліотеці Microsoft Authorization Manager.

Друга перешкода полягає у тому, що користувач, який створив об’єкт, немає на нього ніяких виняткових прав. Це цілком прийнятно для систем, що підтримують, наприклад, процес купівлі-продажу, але перестає бути придатним, як тільки документи починають містити якісь авторські матеріали, тобто є об’єктами інтелектуальної власності.

Нарешті, третя проблема стосується угрупування операцій системи у ролі, у межах яких вони виконуються. Це спрощує адміністрування, але це знову правильно не для усіх типів систем. Припустимо, що у системі є десять різних типів об’єктів, для кожного з яких визначена операція “видалити”. Тоді, якщо додати цю операцію в яку-небудь з ролей, то будь-який користувач, що грає цю роль, отримує право видаляти об’єкти будь-якого типу. Очевидно, що це далеко не завжди є бажаною поведінкою.

На нашу думку, до головних недоліків систем, управління правами доступу в котрих грунтується на RBAC-моделі, можна зарахувати такі:

1. Відсутність або складність швидкої динамічної конфігурації відношень користувач-роль, роль-права доступу. Адміністратору доводиться займатися перевіркою безлічі дрібниць, щоб уникнути неправильних призначень ролей користувачам чи прав доступу ролям, а разом з тим негативних наслідків. Усе це сповільнює роботу організації, що позначається на її фінансовому стані.
2. Якщо користувач отримав права доступу до деякого об’єкта на основі деякої ролі, то до складу цих прав входять усі можливі операції, які можна виконати над об’єктом. Вагомим недоліком є, власне, відсутність можливості надання користувачу у розпорядження лише однієї операції, наприклад лише читання інформаційного об’єкта. Ця проблема може викликати певну надлишковість прав доступу у користувача.
3. Непідтримування такого поняття, як автоматичне тимчасове надання чи позбавлення прав доступу. Тобто, можливість звертатися до об’єкта або бути позбавленим доступу до нього лише упродовж певного часу, наприклад з 10.09.2011 до 10.10.2011. А потім гарантування того, що цих прав у користувача, і відповідно ролі, більше не буде, або ж навпаки. Автоматичне тимчасове позбавлення прав доступу зручне для таких випадків, коли працівник пішов у відпустку і є загроза несанкціонованого втручання у його комп’ютер.
4. Повна відсутність автоматизованості у системах управління правами доступу. Адміністратору доводиться пробляти усю пов’язану з управлінням роботу вручну, що займає багато часу та виконується не завжди вчасно. Повністю або частково автоматизована система управління правами доступу могла б виконувати усю роботу замість адміністратора, а також підтримувати журнал подій, що значно спростило б сам процес управління правами доступу.

Формулювання цілі статті

Метою цієї статті є розроблення автоматизованої інтелектуальної інформаційної системи управління правами доступу до корпоративних інформаційних ресурсів, яка використовує модель RBAC, проте вирішує більшість із зазначених вище недоліків та проблем цієї моделі. До того ж головний ухил робиться до забезпечення автоматизованості системи, яка проявляється у здатності системи до поступового самонавчання та прийнятті автоматичних рішень.

Виклад основного матеріалу

Бажану поведінку автоматизованого модуля системи, який відповідає за прийняття автоматичних рішень про надання чи заборону прав доступу до інформаційних об’єктів користувачу, можна описати за допомогою об’єктно-орієнтованого підходу та нотації UML, навівши приклад діаграми станів системи, яка описує важливі аспекти функціонування системи та можливі послідовності станів і переходів, які в сукупності характеризують динамічну поведінку модуля системи впродовж його життєвого циклу (рис. 1).

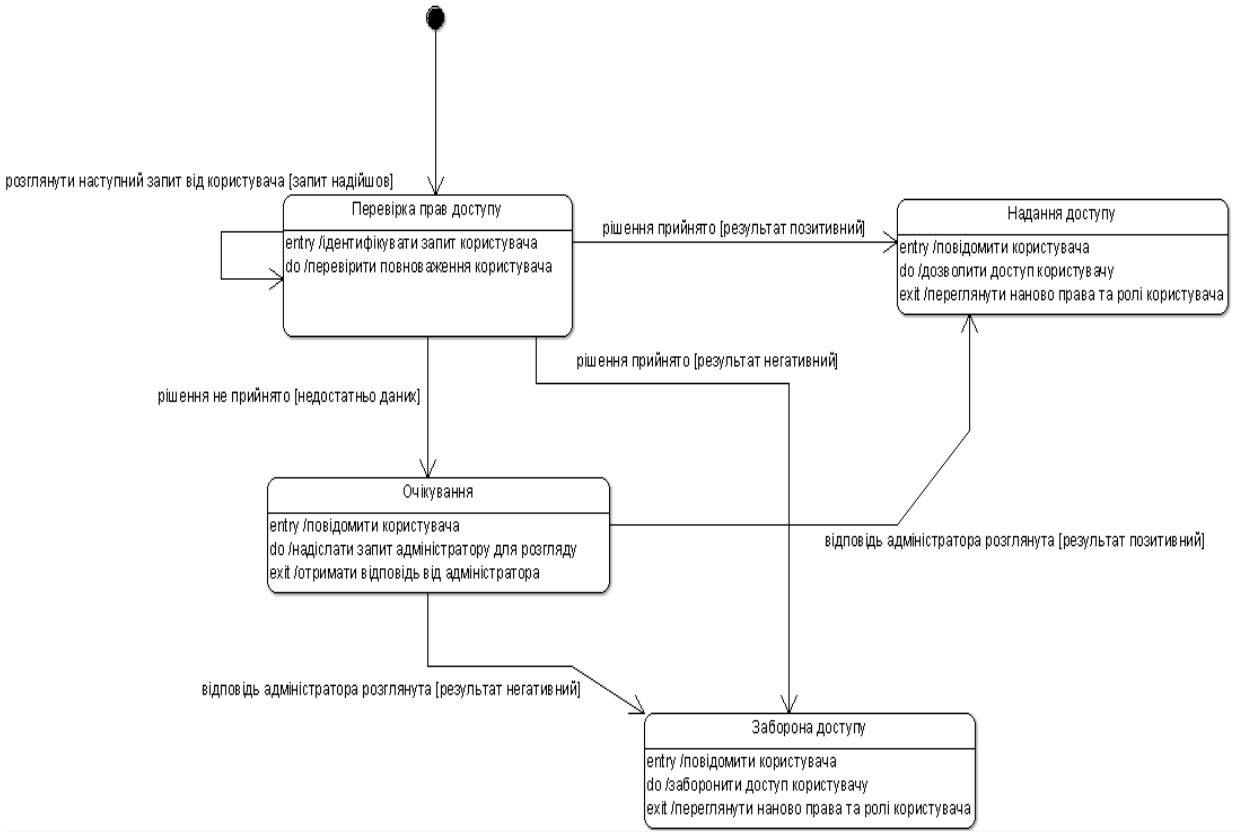

Рис. 1. UML-діаграма станів автоматизованого модуля системи управління правами доступу

Із початкового формального стану система переходить у стан перевірки прав доступу. Цей стан передбачає ідентифікацію отриманого запиту користувача, його розгляд. У цьому стані система управління правами доступу перевіряє повноваження користувача та приймає на основі цієї перевірки певні рішення. Цей стан характеризується рефлексивністю, тобто він може переходити сам у себе. Це відбувається тоді, коли надходить наступний запит від якогось користувача і, отже, його потрібно розглянути та перевірити.

Рішення перевірки може прийняти самостійно система або за участю адміністратора. Якщо рішення не прийнято самостійно, то це означає, що у базі знань системи мало або недостатньо даних про прийняття рішення, тобто її інформаційна складова є неповною по відношенню до деяких аспектів. У результаті система переходить у стан очікування, повідомляє про це користувача і надсилає запит адміністратору для того, щоб рішення прийняв він. Головною особливістю цього стану є апріорі невизначена його тривалість. Це не означає, що система зависла на розгляді одного запиту. Паралельно вона обробляє інші запити, тобто наявно стільки встановлених сесій із користувачами щодо оброблення їхніх запитів, скільки необхідно (своєрідна мультисесійність). Щоб вийти з цього стану (у контексті деякої сесії), система має отримати та розглянути відповідь від адміністратора.

Якщо адміністратор своїм рішенням дозволив користувачу отримати доступ, то система переходить у стан надання доступу, повідомляє про це користувача та дозволяє безперешкодне користування наданими правами доступу. Якщо ж адміністратор своїм рішенням заборонив користувачу отримати доступ, то система переходить у стан заборони доступу, знову ж таки повідомляє про це користувача та не дозволяє користуватися певними запитуваними користувачем інформаційними ресурсами.

Проте система управління правами доступу може самостійно приймати рішення про надання чи заборону доступу. Ці дії є аналогічними до вищеописаних. Єдиною відмінністю є те, що оминається стан очікування і у прийнятті рішення не бере участь адміністратор. Система може прийняти позитивне рішення і надати право доступу, або ж негативне – і заборонити доступ.

Ця модель (діаграма) не враховує (не показує) той факт, що доступ може бути не повним, а обмеженим, тобто користувачу може бути дозволене виконання тільки деяких операцій над інформаційними об'єктами (наприклад, тільки операції перегляду і читання). Цей факт істотно покращує метод контролю доступу на основі ролей RBAC.

Розглянута поведінка системи спонукає до реалізації системи у вигляді демонстраційного прототипу експертної системи, на основі якого можна буде потім прийняти рішення про придатність експертної системи для розв'язання поставлених перед нею задач.

У цій ситуації експертом може виступати адміністратор, якому добре відомий процес надання чи позбавлення прав доступу до інформаційних ресурсів підприємства. Тобто це висококваліфікований фахівець у проблемній області, який має змогу визначити знання, що характеризують проблемну область, а також забезпечити повноту і правильність введених у систему знань.

Розроблена реалізація процесу спілкування експерта з експертною системою наступна. Експерт (тут – адміністратор) описує проблемну область у вигляді сукупності фактів і правил. Факти визначають об'єкти (файли, користувачі, ролі тощо), їхні характеристики і значення, що існують в області експертизи. Правила визначають способи маніпулювання даними, характерні для проблемної області. Експерт, використовуючи компонент отримання знань, наповнює систему знаннями, що дозволяють експертній системі в режимі рішення самостійно (без експерта) розв'язувати задачі з проблемної області, а також самонавчатися (у цьому контексті означає виводити за певними правилами нові знання, використовуючи уже відомі у базі знань) і тим самим йти на шляху до певної автоматизованості, яка позбавить рутинної участі адміністратора у взаємодії із користувачами при наданні чи позбавленні прав доступу до ресурсів мережі організації.

Оскільки йдеться про створення методики автоматизації праці адміністратора (експерта у своїй області) щодо управління правами доступу, доречним є питання про застосування експертних систем у їх класичному розумінні. Тобто, у цьому розумінні експертна система – це комп'ютерна система, що містить знання спеціалістів з деякої проблемної області і яка в межах цієї області здатна приймати самостійні експертні (у даному випадку адміністраторські) рішення. Структурна схема інтелектуальної інформаційної системи управління правами доступу наведена на рис. 2.

База знань – центральна частина експертної системи. У нашому випадку база знань зберігатиметься окремо від експертної системи (на диску або іншому носії інформації) у форматі XML. Збереження та опис бази знань у форматі XML – актуальний та, можливо, найкращий сьогодні варіант з-поміж усіх інших. Все-таки збереження та опис бази знань мовою Prolog сьогодні – це далеко не оптимальний варіант порівняно з минулими десятиліттями.

Можливим недоліком системи можна вважати той факт, що початкові значущі знання набуваються системою неявно шляхом модифікації файла з базою знань експертом чи інженером зі знань за посередництвом стороннього XML-редактора бази знань.

Для представлення знань у базі знань вибрана продукційна модель, елементом якої є продукційне правило. Оскільки час виконання задачі не є критичним показником під час розроблення цієї системи і родовидову ієрархію понять хоч і важко, але можна представити, а модульність та зручність модифікації якнайкраще підходять для поставленої системи, то продукційна модель є доволі хорошим варіантом для вибору. Як альтернативу можна було вибрати фреймову модель.

Щоб в алгоритмі виводу можна було оперувати фактами, значеннями фактів, враховувати їх зв'язок в певному правилі і робити висновки, що відповідають цьому набору фактів, база знань експертної системи представляється у вигляді певних структур:

- список цілей <Targets>
- масив змінних (об'єктів) <Objects>
- масив питань <Questions>
- набір правил <Rules>

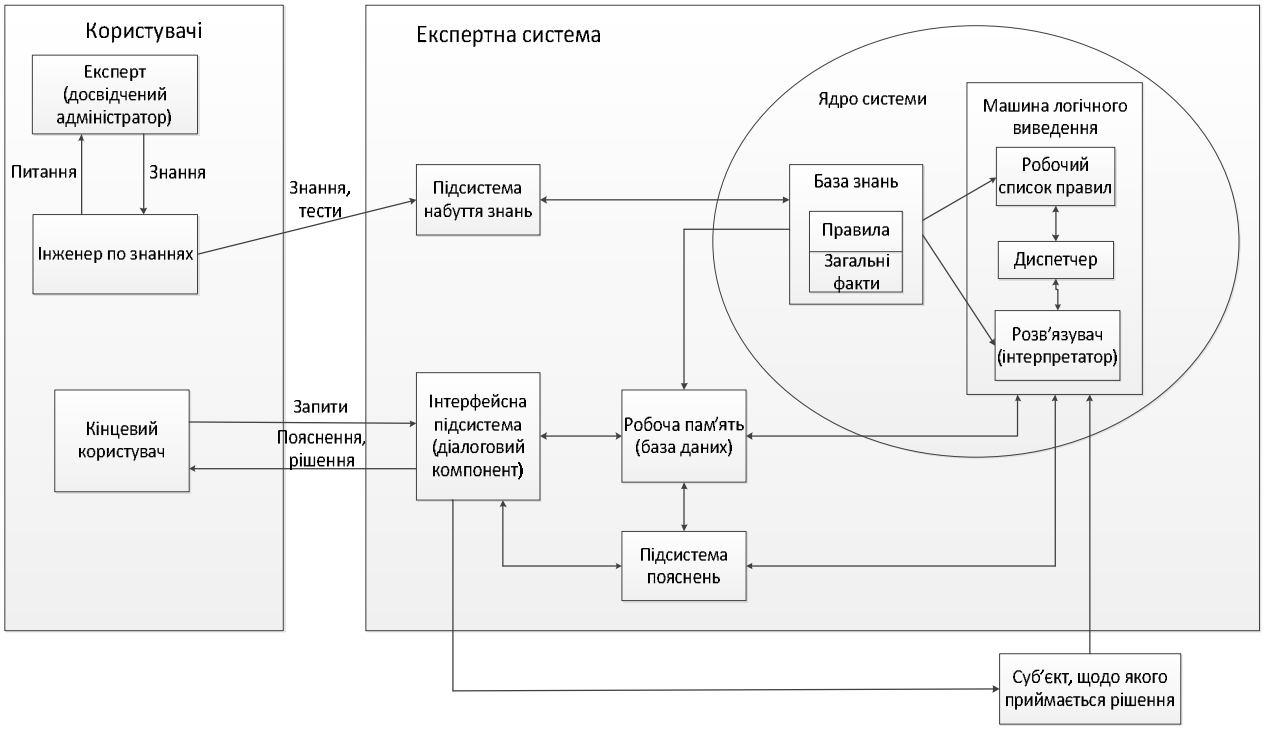

Рис. 2. Структурна схема інтелектуальної інформаційної системи управління правами доступу до інформаційних ресурсів

Список цілей – це ті цілі, які ставляться перед системою для їхнього вирішення. Для прикладу, в базі знань ціль можна записати так:

<Targets>

<Target name=”Можливість_надання_прав_доступу” />

</Targets>

Завдяки простоті внесення змін у базу знань додавання нових цілей не становить труднощів. Головне – це забезпечити відповідну обробку та послідовність дій у системі для досягнення внесеної цілі.

До головних змінних (об’єктів), які фігурують у базі знань, належать співробітники, їхні посади, які можна асоціювати з ролями у виробничому підрозділі, інформаційні ресурси (важливі файли та директорії, які потребують захисту від небажаного втручання), операції над ресурсами, а також дублікати цілей, які структурно зберігають можливі альтернативні відповіді у разі настання заданих цілей. Усі ці змінні відокремлені одна від одної.

Для прикладу, працівника у базу знань можна внести за допомогою такої конструкції:

<Object name=“Працівники”>

<Value>Бренич_Андрій</Value>

</Object>

Прізвища працівників та усі можливі значення інших змінних записані у алфавітному порядку, що строго ідентифікує записи значень у базі знань.

Для прикладу, сукупність посад у базі знань описується так:

<Object name=“Посади(ролі)”>

<Value>Дизайнер</Value>

<Value>Інженер</Value>

<Value>Менеджер_проекту</Value>

</Object>

У всіх системах, що ґрунтуються на експертних технологіях, існує залежність між вхідним потоком даних і даними у базі знань. Під час консультації з системою вхідні дані зівставляються з

даними у базі знань. Результатом зіставлення є негативна або позитивна відповідь. У системі, яка базується на правилах, ствердна відповідь є результатом дії одного з продукційних правил. Ці продукційні правила визначаються вхідними даними. Отож, головні змінні поділені на вхідні (або явні, які задаються користувачем системи) та приховані (або неявні, які ініціалізуються системою на основі вхідних змінних).

До вхідних змінних можна зарахувати вибір співробітника, інформаційного ресурсу та операції над ресурсом. А до прихованих – вибір посади (ролі). Очевидно, що людині, яка буде користуватися створеною системою, чи самій системі, приймаючи самостійні рішення у подальшому, не потрібно знати посаду працівника. Система самостійно визначає його посаду (роль) на основі закладених продукційних правил у базі знань.

У масиві запитань <Questions> зберігаються запитання для користувача системи, які пов'язані з метою отримання від користувача необхідних вхідних знань. Цими запитаннями є такі: "Виберіть співробітника", "Виберіть об'єкт для доступу", "Виберіть можливі операції над об'єктом" та спеціальне запитання, призначене для вибору цілі.

Продукції зберігаються у масиві <Rules>. Заздалегідь визначено, що кількість умов у правилі не є обмеженим. Нехай є N правил подібної структури. Кожне *i*-те правило у базі знань має таку структуру:

<Rule id="i">
<Condition name="Ідентифікатор змінної (ім'я) "value1="Значення змінної" />
[<Condition name="Ідентифікатор змінної (ім'я) "value2="Значення змінної" /> ,
<Consequence name="Ідентифікатор змінної (ім'я) " value3="Значення змінної " />
<Reason text="коментар (пояснення) до правила" />

Тут Rule id – номер правила, Condition name – це факти *i*-того правила, value1, value2 – значення фактів, квадратні дужки означають необов'язковість (оскільки правило може містити від одного до m фактів), Consequence name – назва виведення *i*-того правила, value3 – зміст або значення виводу, Reason text – пояснення, яке вказує на існуюче правило.

Тепер покажемо як система самостійно розпізнає посаду співробітника. Візьмемо для прикладу деяке правило, яке записано у базі знань так:

<Rule id="1">
<Condition name="Вибір_працівника" value="Бренич_Андрій" />
<Consequence name="Вибір_посади" value="Програміст" />
<Reason text="пояснення перше" />
</Rule>

Це правило однозначно ідентифікує посаду (роль) працівника на основі його прізвища та ім'я. Тобто, якщо вхідним значенням під час вибору співробітника був Бренич Андрій, то це правило спрацює і в результаті буде отриманий факт, який засвідчує, що його посада – програміст. Оскільки користувач запитує лише про одну людину (себто про себе), то нам не потрібно турбуватися про те, що посада не прив'язується явно до працівника. У робочій пам'яті будуть зберігатися копії фактів, які так чи інакше пов'язані між собою. Це можливо завдяки створенню окремої сесії для кожного запиту користувача.

Тепер продемонструємо як у базі знань однозначно ідентифікується заборона доступу до файла. Візьмемо за основу таке правило:

<Rule id="13">
<Condition name="Вибір_посади" value="Дизайнер" />
<Condition name="Вибір_об'єкту" value="ClassDiagram.uml" />
<Consequence name="Можливість_надання_прав_доступу" value="Ні,варто_не_надавати" />
<Reason text="пояснення тринадцяте – дизайнер немає відношення до діаграм UML" />
</Rule>

Припустимо, що система визначила посаду працівника та отримала від користувача знання про розгляд об'єкта ClassDiagram.uml. Тоді незалежно від отриманої від користувача можливої операції над файлом (навіть не вдаючись до розгляду операції), система однозначно ідентифікує заборону на отримання запитуваних прав, оскільки дизайнер немає жодного відношення до діаграм. Подібно реалізується і однозначний дозвіл на отримання доступу до файла та виконання над ним потрібних операцій.

Розроблені правила забезпечують також можливість розподіленого за операціями контролю доступу. Покажемо це на прикладі наступного правила:

<Rule id="22">
<Condition name="Вибір_посади" value="Дизайнер" />
<Condition name="Вибір_об'єкту" value="Readme.txt" />
<Condition name="Вибір_операції" value="читання(перегляд)" />
<Consequence name="Можливість_надання_прав_доступу" value="Так,можна_надати" />
<Reason text="пояснення двадцять друге – темпоральний файл з інструкціями для кожного" />
</Rule>

Отже, розроблена логічна структура бази знань є доволі гнучкою та легко піддається модифікаціям.

Що стосується машини логічного виведення, то вона використовує дедуктивний алгоритм з прямим ланцюжком міркувань (він відповідає стратегії "від даних до мети" або стратегії управління даними) з можливістю вибору пошуку рішення вшир або вглиб. Машина логічного виведення (інтерпретатор правил) у розробленій експертній системі виконує дві функції:

1) по-перше, перегляд правил з бази знань;

2) по-друге, застосування правил.

У створеній системі інтерпретатор продукційних правил працює циклічно. У кожному циклі він переглядає усі правила, щоб виявити ті умови, які збігаються з відомими і істинними на цей момент фактами. Після вибору правило спрацьовує, його висновок заноситься в робочу пам'ять, а потім цикл повторюється спочатку.

Протягом кожного циклу можуть бути активізовані і поміщені в робочий список правил багато правил. Крім того, у робочому списку правил залишаються результати активізації правил від попередніх циклів, якщо не відбувається деактивізація цих правил у зв'язку з тим, що їхні ліві частини більше не виконуються. У такий спосіб під час виконання програми кількість активізованих правил у робочому списку правил змінюється.

В одному циклі може спрацювати тільки одне правило. Якщо декілька правил успішно зіставлені з фактами, то така ситуація називається конфліктною. Інтерпретатор виконує вирішення конфліктів шляхом вибору за певним критерієм єдиного правила, залежно від вибору стратегії вирішення конфліктів (вшир або вглиб).

Істотним плюсом системи є те, що вона пояснює усі свої чи адміністраторські рішення завдяки вбудованій підсистемі пояснень. Таким чином ведеться журналювання подій.

Математично (формально) вирішену за допомогою розробленої системи проблему можна описати через набори відповідних сутностей:

- Users (Користувачі) – множина користувачів u
- Roles (Ролі) – множина ролей r
- Permissions (Права доступу) – множина наборів прав доступу p

Тоді користувацько-рольове відношення (UR relation) математично подається так:

$$UR \subseteq Users \times Roles \quad (1)$$

Формальне подання роле-правового відношення (RP relation) таке:

$$RP \subseteq Roles \times Permissions \quad (2)$$

Відношення контролю доступу (AC relation) можна подати як композицію відношень (1) та (2):

$$AC = UR \circ RP \tag{3}$$

Інакше кажучи, AC визначене як множина відповідних пар:

$AC = \{ (u, p) \in Users \times Permissions \mid \exists r \in Roles, (u, r) \in UR \wedge (r, p) \in RP \}$

Крім того, потрібно чітко специфікувати набори прав доступу щодо конкретних файлів (F) та директорій (D). Це максимізує масштабованість системи та привносить певний ефект новизни у сучасну RBAC-модель. Множину файлів та директорій ($F \cup D$) потрібно розцінювати як сукупність складових елементів деякого робочого простору, яка потребує "пильного", конкретно специфікованого нагляду, а не як множину усіх файлів та директорій файлової системи, оскільки це було б значною надлишковістю з погляду адміністратора чи системи управління правами доступу. Тому варто скоригувати (3) і подати її так:

$$AC = UR \circ RP \circ (F \cup D) \tag{4}$$

З погляду системи процес самонавчання у (4) можна зобразити у вигляді зворотної стрілки, суть якої полягає у тому, що система з часом буде здатна самостійно виводити нові знання та приймати адекватні рішення на основі відомих правил та знань, тим самим поповнюючи власну базу знань новими знаннями і здійснюючи ефективний контроль та управління доступом до файлів та їхнього вмісту залежно від запитів користувачів та їхніх реальних прав доступу. Отож:

$$AC = (UR \circ RP \circ (F \cup D))_{\circ} \tag{5}$$

Розроблена система управління правами доступу містить підсистеми, які забезпечують відповідне функціонування системи. До таких підсистем належить підсистема контролю прав доступу, підсистема адміністрування та підсистема інженерії знань. Дані підсистеми зручно показати у вигляді об'єктної (пакетної) діаграми. Ця діаграма зображена на рис. 3.

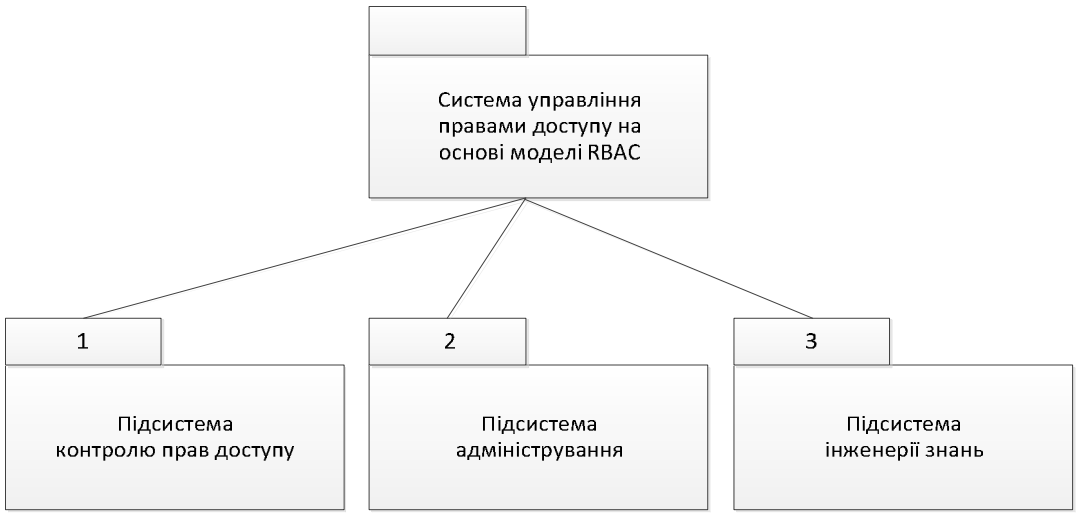

Рис. 3. Об'єктна діаграма розробленої системи управління правами доступу

Висновки

Істотною відмінністю побудованої системи від усіх інших є те, що вона має можливість самонавчатися і з часом може стати автоматизованою (в тому сенсі, що буде позбавлена рутинної участі адміністратора у процесах її функціонування, пов'язаних із прийняттям рішення щодо надання чи позбавлення прав доступу працівників до інформаційних корпоративних ресурсів).

Впровадження цієї системи управління правами доступу дозволить вирішити цілу низку проблем, таких як несанкціонований доступ до корпоративних ресурсів, а також значні трудовитрати на надання та зміну працівникам тимчасових прав доступу до інформаційних об'єктів.

1. Rao K.W. Web-Services Security Architectures using Role-Based Access Control /Rao K.W, Rao M.S, Devi K.M, and all // International Journal of Computer Science and Information Technologies, Vol. 1 (5), 2010. – pp.402–407. 2. Ferraiolo D. Role-Based Access Controls /Ferraiolo D, Kuhn R//Proceedings of 15th National Computer Security Conference, Baltimore, USA, 1992. – pp.554–563. 3. LaPadula L.J. Secure Computer Systems: A Mathematical Model /LaPadula L.J, Bell D.E//MITRE Corporation Technical Report 2547, Vol. 4, 31 May 1973. 4. Sandhu R. Role-based access control model / Sandhu R, Coyee E.J, Ferinstein H.L, Youman C.E// IEEE Computer, 29(2), February 1996. – pp.38–47. 5. Sun Y. PRES – A Practical Flexible RBAC Workflow System /Sun Y, Pan P// Proceedings of the 7th international conference on Electronic commerce, 2005. – pp.653–658. 6. Guo Y. An approach for implementation of RBAC models with context constraint to business process systems /Guo Y, Choi M, Shin S, Bae G // Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16–18, 2006. – pp.566–572. 7. Tolone W. Access Control in Collaborative Systems / Tolone W, Ahn G.J, Pai T // ACM Computing Surveys, Vol.37, No.1. 2005. – pp.29–41. 8. Chen F. Enforcing Role-Based Access Controls in Software Systems with an Agent Based Service Oriented Approach /Chen F, Li S, Yang H/ Software Technology Research Laboratory, De Montfort University, Leicester, England, 2007. – 6 p. 9. Ferraiolo D. Role-based Access Control /Ferraiolo D, Kuhn R, Chandramouli R. – Artech house inc, 2007. – 405 p. 10. Bozzon A. RBAC for the interaction with Search Engines /Bozzon A, Nejdl W, Taddeo A.V// COOPER Workshop in conjunction with ECTEL07 Conference, 17 September 2007. – P.1–5. 11. Chon E. Access Control Police Management Framework based on RBAC in OSGi Service Platform /Chon E, Moon C, Park D// Computer and Information Science Conference, Korea Digital University, Korea, 2007. – pp.1–6. 12. Bohra A. FRAC: Implementing RBAC for Network File Systems / Bohra A, Smaldone S, Iftode L// Department of Computer Science, Rutgers University, New Jersey, USA, 2007. – pp. 1–8. 13. Майоров А.В. Улучшенная ролевая модель управления доступом /Майоров А.В.// 2008. – 8 с.