АСПЕКТНА ДЕКОМПОЗИЦІЯ КОМПОНЕНТІВ ЗАХИСТУ РОЗПОДІЛЕНОЇ СИСТЕМИ ТЕПЛОВОГО ПРОЕКТУВАННЯ

Віталій Яковина, Назарій Мамроха, Максим Сенів

Національний університет "Львівська політехніка"
вул. С. Бандери, 12, Львів, 79013, Україна, тел.: (032) 258-25-78 E-mail: yakovyna@polynet.lviv.ua

Анотація

В цій роботі проведено аналіз ефективності використання аспектно-орієнтованого програмування при побудові компонентів захисту розподіленої системи теплового проектування мікроелектронних пристроїв.

Наведено загальну архітектуру підсистеми захисту та здійснено аспектну декомпозицію системи. Показано, що використання такого підходу дозволяє інкапсулювати функціональність, яка не може бути представлена у вигляді окремої процедури чи компоненти, що дозволяє мінімізувати об'єм та спростити структуру програмного коду та значно спрощує супровід і розширення системи. Кожна вимога до захисту програмного продукту реалізується з мінімальним зв'язуванням, в результаті чого отримується модуль, що містить цю вимогу до системи без зайвих зовнішніх залежностей.

Вступ

При реалізації практично будь-якого програмного продукту, що передбачає розділення функціональності між двома комп'ютерами з передачею даних в мережі (клієнтом і сервером, основним модулем і агентами, тощо), постає задача забезпечення більшості сервісів безпеки – конфіденційності, цілісності, аутентифікації, тощо. З тих чи інших причин використання готових продуктів чи протоколів (таких як ssl, ipsec, kerberos чи інших) може виявитись непридатним для даної конкретної розробки, тоді виникає потреба реалізації власної програмної підсистеми захисту. Однак, при реалізації захисту додатків звичними способами, зокрема з використанням об'єктно-орієнтованого підходу, можна виділити наступні основні недоліки:

- реалізація захисту дуже сильно прив'язана до конкретного додатку;
- оскільки код перевірок входить до коду системи, то неможливо надійно гарантувати, що усі місця розкриття секретних даних супроводжуються такими перевірками;
- код з впровадженими викликами підсистеми захисту, особливо системи авторизації, трудно розширювати і тестувати.

Усі ці та інші проблеми виникають через те, що основна функціональність системи переплітається з функціональністю, що реалізує захист. Схожі проблеми завжди існують при створенні складних інформаційних систем, і відбувається це через те, що доступні об'єктно-орієнтованому програмуванню засоби не дозволяють зручно і просто суміщати функціональності, що перетинаються.

Аспектно-орієнтоване програмування (АОП) є однією з концепцій програмування, яка продовжує розвиток процедурного та об`єктно-орієнтованого програмування. За деякими оцінками [1], близько 70% часу в проектах витрачається на супровід і внесення змін у готовий програмний код. Тому достатньо важливою у найближчій перспективі стає роль АОП і схожих трансформаційних підходів. Порівняно нова технологія уже отримала досить широке розповсюдження, показавши свою ефективність на тестових аплікаціях, однак місце цього підходу в індустрії ПЗ все ще не визначено. Незважаючи на те, що сьогодні це скоріше академічні дослідження, багато компаній, таких як BEA, IBM, Microsoft, а також open source об'єднання, такі як JBoss, ObjectWeb, Eclipse вже працюють в цьому напрямку з помітними результатами.

Метою даної роботи є проведення аналізу ефективності використання аспектно-орієнтованого підходу для побудови системи захисту розподіленої системи теплового проектування мікроелектронних пристроїв.

Основні концепції та аналіз використання аспектно-орієнтованого програмування

Основні ідеї АОП були сформульовані в найбільш явній формі в одній з ранніх статей [2]. АОП – методологія, в основі якої знаходяться ідеї, що зустрічаються в програмуванні вже досить давно. Це суб'єктно-орієнтоване програмування (subject-oriented programming) [3], композиційні фільтри (composition filters) [4, 5], адаптивне програмування (adaptive programming) [6]. АОП тісно пов'язане з ментальним програмуванням (intentional programming), концепції якого викладені в [7]. Іншою близькою ідеологією є так зване породжуюче чи трансформаційне програмування (generative programming, transformational programming) [8].

Аспектно-орієнтоване програмування – це нова технологія, що дозволяє програмістам компонувати програми із блоків, що взаємно перетинаються. АОП уводить в дію поняття аспектів. Вони інкапсулюють особливості поведінки класів, що взаємно впливають один на одного, в складі повторно використовуваних модулів.

На даний час досліджено значну кількість різноманітних шляхів виділення функціональностей у складних системах в окремі модулі [9–11]. АОП є одним із цих рішень, він пропонує засоби виділення цих функціональностей в окремі програмні модулі – аспекти.

Аспектно-орієнтований підхід розглядає програмну систему як набір модулів, кожний із яких виражає аспект – ціль, особливість функціонування системи. Набір модулів, що складають програму, залежить від вимог до програми, особливостей її предметної області. При проектуванні програмної системи розробник вибирає модулі так, щоб кожен із них реалізовував певну функціональну вимогу до системи. Але реалізація деяких вимог до програми часто не може бути локалізована в окремому модулі в рамках процедурного чи об'єктно-орієнтованого підходу, в результаті чого код, що відображає такі функціональні аспекти буде знаходитись у різних модулях. Аспектно-орієнтований підхід дозволяє покращити дизайн системи, забезпечуючи можливість локалізації наскрізної функціональності [12] в спеціальних модулях – аспектах. АОП дозволяє реалізувати окремі концепції у слабо зв'язаному вигляді, комбінуючи такі реалізації, формує кінцеву систему. АОП дозволяє побудувати систему, використовуючи слабо зв'язані розбиті на окремі модулі (аспекти) реалізації загальносистемних вимог.

Розробка в рамках АОП складається із 3 основних кроків:

- Аспектна декомпозиція: розбиття вимог для виділення загальної і наскрізної функціональності. На цьому кроці необхідно виділити функціональність для модульного рівня із наскрізної функціональності системного рівня.
- Реалізація функціональності: реалізувати кожну вимогу окремо.
- Компоновка аспектів: аспектний інтегратор визначає правила створення своїх модулів – аспектів, формуючи кінцеву систему.

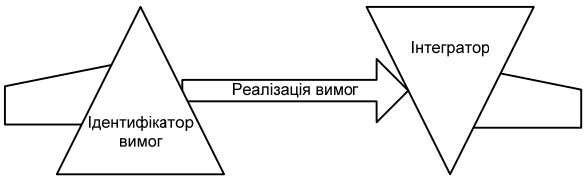

Рис. 1. Фази аспектно-орієнтованої розробки ПЗ

При розробці з використанням АОП концепції реалізуються незалежно одна від одної, оскільки усі існуючі між ними зв'язки (наскрізна функціональність) можуть бути локалізовані в аспектних модулях, що описують протокол взаємодії концепцій.

Аспектом в АОП є системний модуль, в який винесена наскрізна функціональність. Аспектний модуль – це результат аспектної декомпозиції, на етапі якої виявляються будь-які явища, поняття, які можуть бути застосовані до групи компонентів, отриманих після об'єктної декомпозиції. Аспект представляє собою концепцію, що схожа з класом, тільки більш високого рівня абстракції. Аспекти можуть зачіпати багато компонент, використовуючи так звані точки вставки для реалізації регулярних дій, що розосереджені по всьому тексту програми. В аспектному модулі описуються зрізи точок – точки виконання програми, в які вбудовуються інструкції мови, що повинні виконуватись до, після чи замість визначеної точки виконання програми. Ці інструкції є функціональністю, що підтримує взаємодію компонент, крім того в аспектному модулі можуть описуватись ролі компонент, на які може впливати аспект. З точки зору АОП аспект – це сервіс, що зв'язує компоненти системи.

Таким чином можна підсумувати переваги і недоліки аспектно-орієнтованого підходу до побудови програмних продуктів і компонент захисту зокрема. Так, в якості переваг АОП можна виділити:

- Покращення декомпозиції системи на окремі модулі: АОП дозволяє інкапсулювати функціональність, яка не може бути представлена у вигляді окремої процедури чи компоненти. Дозволяє реалізувати кожну вимогу з мінімальним зв'язуванням, в результаті чого отримують модуль, що містить дану вимогу до системи без зайвих зовнішніх залежностей. При такій реалізації модулі містять мінімальну кількість коду, що дублюється, а також це дозволяє уникнути заплутаного коду.
- Спрощення супроводу програмної системи і внесення у неї змін. Оскільки можуть існувати модулі, на які можуть впливати аспекти, то досить легко додати нову функціональність шляхом створення нових аспектів.
- Наявність можливості повторного використання коду [11], що реалізує наскрізну функціональність: наскрізна функціональність в АОП може бути реалізована у вигляді аспектів чи слабо зв'язаних модулів.

Основними недоліками АОП на даний момент є:

- Не повністю пророблена методологія аспектно-орієнтованої розробки програм. На даний момент існує завершений компілятор лише для декількох мов програмування.

- Недостатньо якісна реалізація розширень мов. На даний момент існуючі реалізації аспектно-орієнтованих розширень для різноманітних мов програмування і платформ розрізняються за своїми можливостям Кожна реалізація по-своєму визначає види точок зв'язування, вирішує питання їх опису, що робить складним розуміння загальних принципів АОП.
- Недостатньо пророблений механізм прив'язки аспектів до компонент. В АОП реалізаціях точки зв'язування описуються в термінах програмних конструкцій – класів, методів, полів. При цьому виникає тісний зв'язок між аспектом і компонентом, до якого він застосовується, і порушується одна з основних ідей АОП – незалежність компонентів від застосованих до них аспектів.
- Не повністю дослідженні випадки найбільш ефективного і зручного застосування аспектів.

Таким чином, належне виконання описаних етапів аспектно-орієнтованої розробки модуля захисту розподіленої системи теплового проектування дозволить отримати компактний, універсальний та легко розширюваний код, чого дуже тяжко досягнути використанням традиційних підходів до програмування.

Архітектура та аспектна декомпозиція компонентів захисту

В розроблюваній розподіленій системі теплового проектування мікроелектронних пристроїв передбачається наявність сервера, що містить базу даних компонентів та їх характеристик і розподіляє обчислення між агентами, клієнтської частини, що дозволяє ставити задачу проектування та агентів, що виконують отриману від сервера частину задачі на множині комп'ютерів мережі. Таким чином компоненти захисту повинні забезпечувати аутентифікацію клієнта при доступі до сервера, взаємну аутентифікацію агентів і сервера та конфіденційність даних, що передаються в мережі. Узагальнена схема інформаційних потоків мережі представлена на рис. 2: рис. 2а представляє початкову схему без засобів безпеки, а рис. 2б – схему з використанням модуля захисту (System Defender). На рис. 2в подано узагальнену схему модуля захисту, що повинен забезпечувати аутентифікацію, авторизацію, шифрування та обмін ключами шифрування.

При проектуванні розподіленої системи теплового проектування мікроелектронних пристроїв можуть змінюватись вимоги до рівня безпеки і виникати нові вимоги щодо забезпечення додаткового захисту.

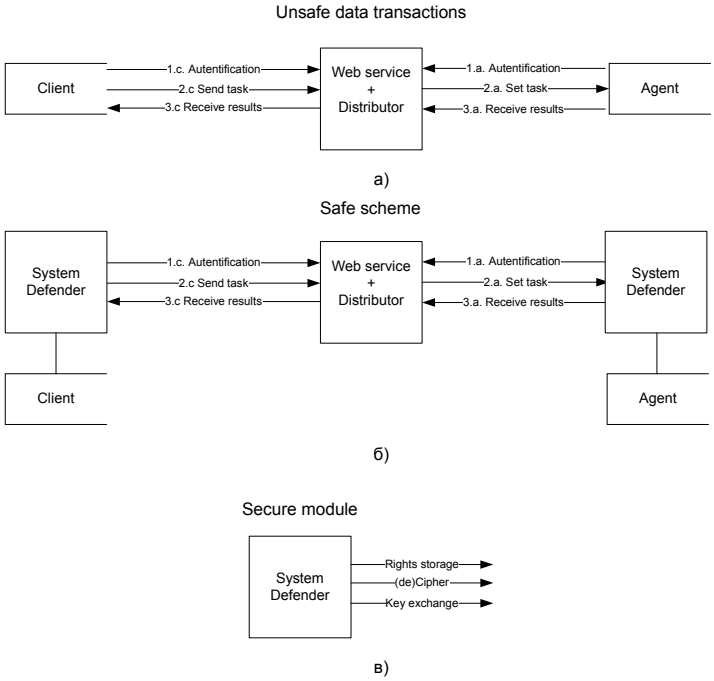

Рис. 2. Узагальнена схема інформаційних потоків компоненти захисту розподіленої системи теплового проектування

Використовуючи об'єктно-орієнтований підхід для створення компонента захисту, було б необхідно при цьому переглядати та вникати у деталі програмної архітектури даної розподіленої системи, реорганізувавши і зробивши в ній необхідні зміни. Оскільки дана система є комплексним веб-додатком це зробити було б досить складно і ресурсоємно, тому для полегшення такої задачі зручно використати парадигму аспектно-орієнтованої розробки. При доданні нових вимог слід лише написати необхідну процедуру перевірки безпеки і вимагати від системи у рамках певного аспекту при виклику заданих методів виконувати дану процедуру.

Таким чином, щоб застосувати аспекти необхідно провести спочатку аспектну декомпозицію. Необхідно виділити новий рівень абстракції поза існуючою програмною архітектурою і задекларувати у ньому функціональність, яка необхідна розроблюваній системі. Також потрібно виділити програмні процедури у існуючому модулі захисту, що обслуговують систему і у рамках аспектів винеси їх в окремі класи.

Згідно поданої на рис.2б схеми при декомпозиції системи теплового проектування можна виділити наступні аспекти захисту:

- аспект аутентифікації;
- аспект авторизації;
- аспект шифрування;
- аспект обміну ключами.

З допомогою поданих вище аспектів можна виділити з наскрізної фунціональністі у системі функціональність модульного рівня захисту, таким чином звільнившись від зовнішніх залежностей у загальній програмній архітектурі розроблюваної системи.

Таким чином робиться більш чіткою і наглядною бізнес-логіка розроблюваної системи, а також отримується чітко розподілена наскрізна функціональність поза базовою моделлю системи.

Висновки

Аналіз можливостей використання АОП для реалізації компонент захисту розподіленої системи теплового проектування завдяки наявності чітко визначених точок виконання програми і відповідних їм аспектів показав, що аспектно-орієнтований підхід надає нові можливості по інтеграції нових вимог в програмну систему в порівнянні з об'єктно-орієнтованим підходом. Якщо виникають нові вимоги до системи на етапі супроводу системи, то такі вимоги легко інтегруються без втрати модульності і модифікації коду компонент, що є дуже актуально на даному етапі розробки. АОП не розглядається як заміна існуючим парадигмам програмування, а виконує роль розширення, що дозволяє забезпечити модуляризацію наскрізної функціональності. На даний момент АОП – єдина методологія, що дозволяє справитись зі складністю, що властива дуже великим системам.

В роботі проведено аспектну декомпозицію компоненти захисту та визначено основні точки виконання програми та функціональність аспектів та інструкції, що повинні виконувати у відповідних точках програми, згідно із заданою функціональністю.

Література:

[1] Rational Software Corporation (http://www.rational.com/)
[2] G. Kiczales, J. Lamping, A. Mendhekar, et al. "Aspect-oriented programming", In proceedings of the European Conference on Object-Oriented Programming (ECOOP). Finland, Springer-Verlag LNCS 1241. June 1997.
[3] Homepage of the Subject-Oriented Programming Project, IBM Thomas J. Watson Research Center, YorktownHeights, New York, (http://www.research.ibm.com/sop/)
[4] Homepage of the TRESE Project, University of Twente, The Netherlands, (http://wwwtrese.cs.utwente.nl/)
[5] M. Aksit, L. Bergmans, and S. Vural, "An Object-Oriented Language-Database Integration Model: The Composition-Filters Approach", In Proceedings of the ECOOP'92 Conference, LNCS 615, Springer-Verlag,1992
[6] K.Leiberherr, "Component Enhancement: An Adaptive Reusability Mechanism for Groups of Collaborating Classes", In Information Processing'92, 12th World Computer Congress, Madrid, Spain, J. van Leeuwen (Ed.), Elsevier, 1992, pp.179-185.
[7] Ch. Simony, "The Death of Computer Languages, The Birth of Intentional Programming", Microsoft Research, 1995, (http://research.microsoft.com/pubs/view.aspx?tr_id=4)
[8] Krzysztof Czarnecki, Ulrich Eisenecker, "Generative Programming: Methods, Tools, and Applications", Addison-Wesley, Paperback, Published June 2000.
[9] С. Стелтинг, О. Маассен, "Применение шаблонов Java. Библиотека профессионала". – Издательство Вильямс, Санкт-Петербург, 2002.
[10] Laddad R., "I want my AOP! Part 1", JavaWorld (2002). Avaliable at http://www.javaworld.com/.