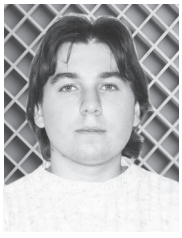
УДК 656.072

КОМПЛЕКСНА СИСТЕМА КОНТРОЛЮ ПАСАЖИРОПОТОКУ В МІСЬКОМУ ТРАНСПОРТІ

Л. О. Левченко, *канд. екон. наук,*
В. В. Ярмола

Вступ

З 80-х років минулого століття науковці, праця яких була пов'язана з комп'ютерною технікою, шукали нові методики створення програмного забезпечення (ПЗ), що дали б змогу підвищити швидкість розробки нових програм і поліпшити надійність результатів цих розробок. Саме в цей період були закладені основи сучасних досягнень у галузі архітектури складних програмних засобів і проведено комплексні дослідження, пов'язані з використанням найрізноманітніших підходів до процесу проектування і розробки програмних систем у цілому. До таких досягнень можна віднести чітке визначення декількох парадигм програмування, що уособлюють основну ідею, яка направляє процес розробки комплексного програмного забезпечення. З одного боку – це концепція об'єктно-орієнтованого програмування, яка на сьогоднішній день стала стандартом де-факто в розробці програмного забезпечення. З іншого – політика професійного застосування існуючих програмних комплексів для досягнення оптимального співвідношення часу і витрат на розробку власної системи.

Мета статті – огляд сучасних парадигм проектування програмних систем і засобів, які забезпечують розробку складних програмних комплексів, а також практичне використання цих методик для створення комплексної системи контролю пасажиропотоку в Київському метрополітені. Особливу увагу приділено специфіці проектування програмного забезпечення і реалізації складних систем у гетерогенному середовищі.

Для цього передусім необхідно розглянути об'єктно-орієнтоване програмування (ООП), пов'язане з поняттям об'єкта, що відображає деякий аспект предметної області разом з його поведінкою і властивостями. Концентруючись на узагальненому інтерфейсі системи і абстрагуючись від деталей роботи вже реалізованих підсистем, стало можливим кардинально підвищити рівень складності як нових, так і вже існуючих систем. Незважаючи на те, що окремі принципи цього підходу вже давно були відомі світовій науці, саме парадигма ООП об'єднала все позитивне відомого підходу з новими ідеями, систематизувавши розрізнені знання в чітку і виважену систему. Слід зауважити, що саме ця парадигма вже більше 15 років є основою складних програмних комплексів у всьому світі, проте вона має деякі недоліки, зокрема це проблема надмірної абстракції, що нівелює простоту сприйняття об'єктів системи через складність організації взаємодії між ними і відповідно – ті відмінності, що виникають у сприйнятті цієї концепції програмістами. Для того, щоб вирішити цю проблему, було розроблено список типових прийомів, що узагальнюють методики, які використовуються під час розробки програмного забезпечення, чітко визначаючи завдання і стратегію використання кожного з них [1]. Проте цей спосіб не є універсальним, крім того, він не вирішує проблем, які не пов'язані безпосередньо з архітектурою (недостатня гнучкість і виразність використовуваної мови програмування або обмеженість її стандартної бібліотеки).

Парадигма повторного використання програмного коду є очевидним наслідком накопичення величезного

обсягу роботи, проведеної іншими програмістами. Теоретичне і практичне обґрунтування такої методики розглянуто в тексті ліцензії GNU GPL, що з юридичної точки зору надає можливість використовувати код, розміщений за даною ліцензією у власних продуктах [2]. Завдяки такій політиці втрачається необхідність розробки тих складових підсистем, що вже були реалізовані. До таких компонентів можна віднести операційні системи, драйвери роботи з зовнішніми приладами, типові бібліотеки, що містять реалізації загальноприйнятих протоколів та інше. Отже, отримуємо змогу зосередити зусилля саме на тих компонентах, які є унікальними для розроблюваної системи, тобто таких, що формують її основну функціональність.

Система контролю пасажиропотоку в Київському метрополітені на рівні окремої станції складається з декількох рівнів:

- перший – включає в себе прикладні пристрої, що взаємодіють з пасажирами і персоналом станції, а саме: турнікети, зчитувачі безконтактних карток, автомати дозапису безконтактних карток (АДБК) і касові апарати. Кожен з цих пристроїв виконує чітко визначену роль і працює на базі власного мікроконтролера з власним унікальним програмним забезпеченням;
- другий – це вузловий сервер, присутній в кожному вестибюлі. Його функція – своєчасне збирання й агрегація інформації про події, що відбуваються на кожному з пристроїв першого рівня, у фіксованому форматі, що буде проаналізований автоматикою вищого рівня, а також реалізація віддаленого керування всіма аспектами роботи окремої станції. Вузловий сервер виконано на базі мікрокомп'ютера, який працює під керуванням операційної системи GNU/Linux на процесорі з архітектурою ARM [3];
- третій – складають комп'ютери, які виконують збирання даних з вузлових серверів і формують статистику пасажиропотоку Київського метрополітену.

Розподілена система обов'язково повинна мати розвинуті засоби комунікації між її складовими. Крім того, при розробці систем такого класу необхідно враховувати такі особливості:

- обмеженість ресурсів системи (тактова частота процесора, обсяг оперативної пам'яті і швидкість передачі даних по каналу зв'язку);
- жорсткі вимоги до швидкості роботи кожного з компонентів і їхньої стійкості до екстремальних ситуацій, які виникають у процесі експлуатації;
- різноманітність типів пристроїв, з яких складається система (мікроконтролери, мікрокомп'ютери і персональні робочі станції – тобто гетерогенність системи);
- значні відмінності в програмному забезпеченні компонентів системи.

Роль протоколів у функціонуванні системи

Протоколом називають послідовність дій, що виконуються двома або більше учасниками для розв'язання певної задачі. Мережевий протокол являє собою набір правил, які визначають принципи взаємодії комп'ютерів у мережі, і є стандартом щодо обміну даних у ній. Протокол також задає загальні правила взаємодії різноманітних програм, мережевих вузлів чи систем і створює таким чином єдиний простір передачі. Використання уніфікованого протоколу дає змогу знизити ступінь гетерогенності системи за рахунок загального інтерфейсу взаємодії між різними пристроями. Проте нераціонально використовувати єдиний протокол для взаємодії з принципово відмінними між собою пристроями системи, оскільки це призводить лише до надмірного розростання специфікації протоколу і не дає жодних позитивних результатів. Концепція протоколу надає можливість, встановивши фіксовану методику взаємодії між вузлами системи, проводити подальший розвиток усього комплексу в рамках, заданих на етапі розробки протоколу. Саме тому протокол є однією з важливих складових надійності і якості роботи системи.

Вузли даної системи фізично розподілені по вестибюлю окремої станції метрополітену й об'єднані за допомогою двох окремих каналів зв'язку: мережі Ethernet, яка з'єднує вузловий сервер з АДБК і третім рівнем системи, і мережі Control Area Network (CAN), яка поєднує з вузловим сервером усі інші пристрої першого рівня. У той час, як перед першим рівнем ставиться завдання забезпечувати безперебійне виконання своєї основної функції (обслуговування пасажирів) і надавати мережевий інтерфейс, що дає змогу зчитувати інформацію про події на цьому пристрої, перед вузловим сервером ставиться стратегічна задача, що полягає у взаємодії з кожним типом пристроїв, наявних на станції, і наданні зручного доступу до агрегованих подій пристроям третього рівня. На основі цих мереж і типових для них протоколів (TCP/IP і CAN protocol відповідно) було розроблено власні прикладні протоколи.

Враховуючи особливості розподілу пристроїв і мереж на станції і вимоги, пов'язані зі швидкістю роботи всієї системи, яка має працювати в реальному часі, обслуговуючи сотні тисяч пасажирів у день, протоколи взаємодії між пристроями є одним із чинників, що визначають загальну потужність усього комплексу з обслуговування пасажирів.

В основу протоколів взаємодії цієї системи покладено поняття команди. Команда – це структурна одиниця даних, що передається по каналу зв'язку між сервером і клієнтом і містить повну інформацію про дію, яку має виконати сервер у відповідь на цю команду.

Під час проектування системи було враховано обмеженість ресурсів пристроїв першого рівня, тому було прийнято рішення створити двійковий протокол з фіксованою довжиною команди, що, з одного боку, мало забезпечувати швидкість обробки таких команд, а з іншого – розширення діапазону доступних команд без зміни протоколу взаємодії.

Вузловий сервер як основний компонент системи окремої станції реалізує цілу низку протоколів, завданням яких є забезпечення взаємодії з пристроями першого рівня, а також з пристроями третього рівня. Для них існує більш високорівневий текстовий протокол, за допомогою якого відбувається керування усіма аспектами роботи окремої станції, зокрема проводиться оновлення програмного забезпечення, встановлення часу на станціях метрополітену тощо .

Методики реалізації

Під час розробки програмних продуктів досить часто неможливо повною мірою дотримуватися єдиної парадигми програмування. Прикладом такої ситуації є наявність протиріч серед вимог до системи або обмеження, продиктовані обраними засобами реалізації. Типовим розв'язком цих протиріч є поєднання декількох методик і знаходження компромісу між ними з метою досягнення оптимального рішення.

Розробляючи даний програмний комплекс, також довелося прийняти ряд компромісних рішень задля оптимального використання ресурсів усієї системи. Так, програмне забезпечення пристроїв першого рівня розроблялося без використання сторонніх бібліотек і всієї повноти ООП, оскільки це знизило б швидкість роботи пристрою і призвело б до невиправданих витрат його ресурсів. Парадигма ООП, яка добре зарекомендувала себе під час розробки складних систем, не змогла забезпечити достатню швидкість роботи прошивок пристроїв пасажирської автоматики. При цьому вона була успішно використана при програмуванні програмного забезпечення вузлових серверів. Для розробки ПЗ вузлового сервера також широко використовувалися готові рішення, що поставляються разом з операційною системою GNU/Linux: мережевий стек, як основа для описаних вище протоколів прикладного рівня, і набір прикладного програмного забезпечення, що спрощує діагностику і контроль роботи станції метрополітену [4].

Відповідно до поставленого перед системою завдання, що полягає в забезпеченні надійності і швидкості обслуговування пасажиропотоку, проектування програмного забезпечення вузлового сервера як ключового елемента системи позначилося чітким розподілом ролей між окремими програмами, які працюють на сервері – кожна мережева функція (вебінтерфейс, командне керування сервером, збирання інформації з пристроїв першого рівня) виділена до окремого компонента. У разі, якщо такий підхід використати неможливо (зокрема, об'єднана система агрегації подій), використовувалася парадигма ООП, що максимально абстрагувала окремі аспекти виконання цього завдання.

Висновки

Розробка розподіленої системи базується на архітектурних рішеннях. Саме вони визначають процедуру розробки, а в подальшому і процедуру роботи всієї системи. Незважаючи на відносну обізнаність з методами проектування таких систем, їхня практична реалізація дуже часто пов'язана з обмеженнями, накладеними предметною областю задачі або умовами замовника. У такому разі дуже важливо, щоб архітектор системи міг знайти компроміс між цими вимогами і реалізувати оптимальне рішення. Розглянутий приклад архітектури системи контролю пасажиропотоку Київського метрополітену, що використовується в гетерогенному середовищі, показує основні особливості таких систем і розкриває специфіку їхньої архітектури.

ЛІТЕРАТУРА

1. *Гамма Э.* Приемы объектно-ориентированного проектирования. Паттерны программирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Виллидес. – СПб: Питер, 2007. – 366 с.

2. Ліцензія GNU GPL, що доступна за наступною адресою у мережі Інтернет: http://www.gnu.org/licenses/gpl.html

3. Характеристика мікрокомп'ютера, використаного у системі, на сайті виробника: http://www.embeddedarm.com/products/arm-sbc.php#ts-7200-200mhz-series .

4. *Robert Love* Linux system programming. – O'Reilly Media, – 2007. – 400 с. http://oreilly.com/catalog/9780596009588