Processing Systems I, D.S.Touetzky, Ed. San Mateo, CA: Morgan Kaufman, p. 634-642, 1989. 13. Grossberg S. Non-Linear Neural Networks: Principles, Mechanisms, and Architectures // Neural Networks, vol. 1, pp. 17-61, 1988. 14. Wolfe W.J., Mathis D., Anderson C., Rothman J., Gotler M., Bragy G., Walker R.., Duane G. and Alaghband G. K-Winner networks // IEEE Trans. Neural Networks, vol. 2, pp. 310-315, 1991. 15. Perfetti R. On the robust design of k-winners-take-all networks // IEEE Transactions on Circits and Systems-II: Analog and Digital Signal Processing, vol. 42, № 1, pp. 55-58, 1995. 16. Seiler G. and A.Nossek J. Winner-take-all cellular neural networks // IEEE Transactions on Circits and Systems-II: Analog and Digital Signal Processing, vol. 40, № 3, pp. 184-190, 1993. 17. Yen J.C., Guo J.I. and Chen H.-C. A new k-Winners-take all neural network and its array architecture // IEEE Trans. Neural Networks, vol. 9, pp. 901-912, September 1998. 18. Calvert B.D. and Marinov C.A. Another k-Winner-take-all analog neural network // IEEE Trans. Neural Networks, vol. 11, № 4, pp. 829-838, July 2000. 19. Marinov C.A. and Calvert B.D. Performance analysis for a K-winners-take-all analog neural network: basic theory // IEEE Trans. Neural Networks, vol. 14, № 4, pp. 766-780, July 2003. 20. Hopfield J.J. Neurons with graded response have collective computational properties like those of two-state neurons // in Proceedings of the Natonal Academy of Sciences, №81, pp. 3088-3092, 1984. 21. Lazzaro J., Lyckebusch S., Mahowald M.A. and Mead C.A. Winner-take-all networks of 0(n) complexity // in Advances in Neural Informaton Processing Systems I, D.S.Touetzky, Ed. Los Altos, CA: Morgan Kaufmann, p. 703-711, 1989. 22. Yang J.-F. and Chen C.M. A dynamic K-winners-take-all neural network // IEEE Trans. on Syst., Man. and Cyb.-Part B:Cyb., vol. 27, № 3, pp. 523-526, 1997. 23. Mead C.A. Analog VLSI and Neural Systems // Reading, Addison-Wesley, 1989.

УДК 004.89

Є.В. Буров
Національний університет “Львівська політехніка”,
кафедра інформаційних систем та мереж

ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ ІНФОРМАЦІЙНОЇ МЕРЕЖІ З ВИКОРИСТАННЯМ СЕРВІСНО-ОРІЄНТОВАНОГО ПІДХОДУ ТА МОДЕЛЕЙ ВИКОНАННЯ ЗАПИТІВ

Розглянуто та запропоновано структуру і принципи роботи інтелектуального сервісу, керованого моделями. Формальна специфікація сервісу уможливлює адаптацію параметрів інформаційної системи до зміни вимог бізнес-процесів.

General framework and architecture of intellectual model-driven service is proposed. Formal specification of intellectual service is developed for intellectual information networks design.

Постановка проблеми та аналіз останніх досліджень

Однією з найгостріших проблем галузі інформаційних технологій сьогодні є проблема дедалі більшої складності корпоративних інформаційних систем. З розвитком глобальних інформаційних мереж та встановленням зв”язків між окремими бізнес-структурами завдання керування та проектування таких систем виходить далеко за межі простого керування окремою системою. У міру зростання складності проектувальникам стає все складніше встановлювати, налаштовувати та прогнозувати поведінку систем [1].

Одним із способів вирішення проблеми складності є побудова компонентних систем, які інкапсулюють дані та функції. Це відобразилось у парадигмі об’єктно-орієнтованого програму-

вання, популярних технологіях Java, DCOM тощо. Такі технології дають змогу вирішувати проблему складності систем на рівні компонент програмного коду. Водночас на вищих архітектурних рівнях інформаційної системи, зокрема на рівні компонент системи, які безпосередньо обслуговують бізнес-процеси (БП), створення підходів до проектування складних систем сьогодні на стадії становлення [2, 3]. Активно розвиваються теорія, методи та засоби сервісно-орієнтованої парадигми (SOA – Service Oriented Architecture), як еволюційно наступного після клієнт-серверного підходу до проектування архітектури розподілених інформаційних систем [4, 5].

Згідно з [4] SOA – це різновид архітектури, створений для підтримки сервісно-орієнтованої прикладної логіки. Він складається з сервісів та об'єднань (композицій) сервісів, що відповідають принципам сервісно-орієнтованої парадигми обчислень. SOA визначає архітектуру, метою якої є покращання ефективності, гнучкості та продуктивності підприємства поданням сервісів як головної форми реалізації прикладної (бізнес) логіки.

Сервіси є незалежними програмними модулями. Кожному сервісу відповідає окремий функціональний контекст та набір можливостей, пов'язаний з контекстом. Можливості сервісу, які можуть бути використані зовнішніми програмами, подають опублікованим сервісним контрактом [4].

Сервіси мають такі характерні особливості:

- сервіс визначає інтерфейс, згідно з яким зовнішні користувачі використовують функції сервісу;
- визначення інтерфейсу доступу до сервісу публікується;
- вони незалежні від технологічної платформи своєї реалізації. Деталі реалізації сервісу приховані від його користувачів;
- вони незалежні від структури бізнес-процесів, які вони обслуговують. Один і той самий сервіс може обслуговувати багато БП. Зміна структури БП не впливає на сервіси системи.

SOA не специфікує деталі та конкретну технологію реалізації сервісів. Сьогодні найпопулярнішою платформою для реалізації сервісів є Веб-сервіси, хоча застосовують й інші платформи. Недоліком існуючих систем, реалізованих з використанням SOA, є недостатня гнучкість реалізації сервісів: незважаючи на можливість гнучкого вибору сервісу з набору наявних та комбінування сервісів, сама реалізація сервісу є статичною і не адаптується до зміни середовища. У разі зміни вимог необхідно переписувати реалізацію сервісу.

Перспективним напрямком у проектуванні та керуванні інформаційними системами є проектування з використанням моделей (MDD – Model Driven Design) [6, 7]. На відміну від традиційних підходів, MDD дає змогу створювати системи, для яких постійно змінюються вимоги, або таких, що працюють у змінному середовищі [1].

Сервіси у системі, побудованій з використанням сервісної парадигми, можна зробити гнучкими у реалізації, якщо:

а) в основу реалізації сервісу покласти модель сервісу;

б) додати до реалізації сервісу інтелектуальні функції, що дасть змогу адаптувати поведінку сервісу до зміни середовища.

Запропоновані інтелектуальні сервіси можна використати як основу для побудови інтелектуальних мереж та інформаційних систем. Дослідження вимог та розроблення структури і принципів організації інтелектуальних сервісів стосується ця робота.

Структура та принципи побудови інтелектуального сервісу

У системі проектування сервіси як об'єкт проектування подаються відповідними абстрактними типами (примітивами, шаблонами), на основі яких будують модель сервісу. Доцільно використати сервісну модель для проектування не тільки частин ІС, які заплановано реалізувати з використанням сервісного підходу, але й таких частин, які реалізовані без дотримання вимог сервісної архітектури. Прикладом таких частин можуть бути успадковані системи або системи, для яких з урахуванням існуючих технологій їхньої реалізації застосування сервісного підходу недоцільне.

Для успадкованих систем пропонується реалізувати функціональну оболонку, яка буде надавати сервісні послуги відповідно до визначеного інтерфейсу. Цей підхід, на нашу думку, має такі переваги:

- досягається уніфікований підхід до проектування інформаційних систем, в якому система подається як мережа взаємодіючих сервісів;
- функції успадкованої системи інкапсулюються та подаються для зовнішніх користувачів як набір визначених інтерфейсних звертань. Отже, успадкована система може бути замінена без потреби у перепроектуванні зовнішніх систем;
- система набуває додаткову гнучкість, що дає змогу, наприклад, додати до реалізації сервісу інтелектуальні можливості

Схема головних компонент запропонованої системи-оболонки подана на рис. 1. Зовнішні сервіси звертаються до оболонки через визначений інтерфейс, деталі якого публікуються. Операційна компонента містить програми, що обслуговують зовнішні запити та є посередником між зовнішнім запитом та успадкованою системою. Програми операційної компоненти можуть, наприклад, вимірювати динамічні параметри (такі, як час відгуку) успадкованої системи та зберігати результати разом з статичними параметрами у локальному сховищі цих параметрів. Параметри зі сховища використовуються для обслуговування запитів.

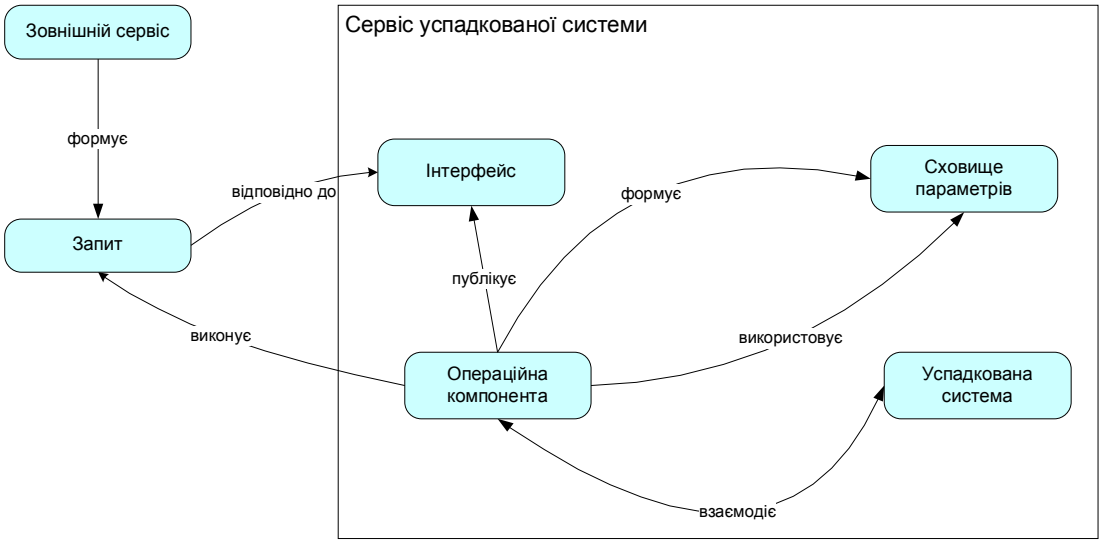

Рис. 1. Складові частини системи-оболонки для сервісу успадкованої системи

Інтелектуальний сервіс, на відміну від звичайного, покращує якість обслуговування запитів, краще адаптується до зміни умов функціонування та надає користувачам нові функції. Структурна схема інтелектуального сервісу подана на рис. 2. Для забезпечення гнучкості реалізації сервісу пропонується застосування модельного підходу, в якому можливі сценарії обслуговування запитів та внутрішні сценарії роботи сервісу зберігаються у Бібліотеці моделей. Внутрішні сценарії сервісу виконують функції, пов'язані не з обслуговуванням запитів, а з підтримкою функціонування сервісу (наприклад, створення архівних копій, оновлення бібліотек моделей та процедур).

Кожна модель бібліотеки створюється з використанням засобів візуального проектування розробником моделей. Модель зберігається у відкритому форматі (наприклад, XML), і містить звертання (з відповідними параметрами) до виконувального коду Бібліотеки процедур. Процедури з Бібліотеки процедур виконують базові функції роботи сервісу з високим ступенем повторного використання різними моделями. Вони незалежні від інтерфейсу сервісу. В результаті виконання процедур змінюються параметри у локальному сховищі параметрів.

Інтерпретатор моделей виконує вибрані моделі згідно з їхнім визначенням. Він використовує значення з локального сховища параметрів.

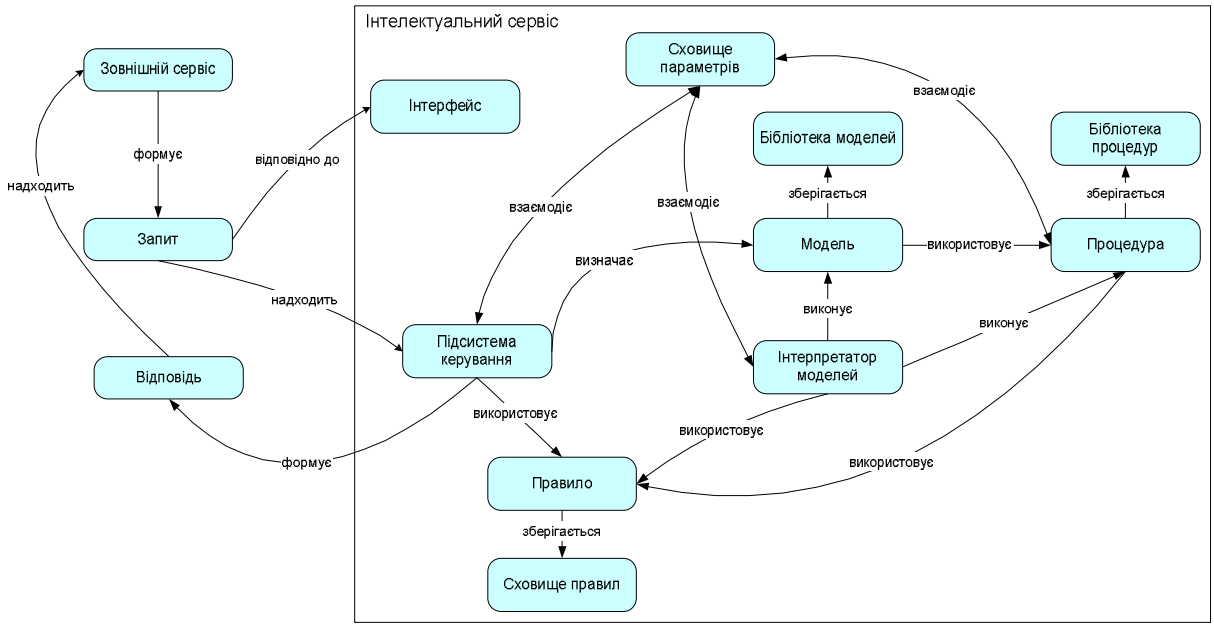

Рис. 2. Концептуальна схема організації інтелектуального сервісу

Підсистема керування виконує такі функції:

- ініціалізує сервіс;
- координує виконання багатьох запитів та моделей одночасно;
- відстежує використання доступних ресурсів;
- підтримує та реалізує загальні правила та політики виконання;
- визначає, яку модель – сценарій виконання вибрати для виконання кожного конкретного запиту та параметри і правила виконання;
- визначає, коли та з якими параметрами виконувати внутрішні сценарії сервісу.

Інтелектуальні функції прийняття рішень пропонується ввести у систему на двох рівнях:

- на рівні підсистеми керування;
- на рівні моделі роботи сервісу – як окрему стадію виконання.

На рівні підсистеми керування інтелектуальна підсистема застосовується для виконання всіх функцій підсистеми керування. Для цього пропонується використовувати сховище правил (шаблонів, patterns). Підсистема керування перед виконанням кожної операції переглядає правила, визначає релевантні правила для поточної ситуації та застосовує їх.

Сховище правил містить правила, що відображають:

- досвід практики виконання запитів;
- методи та рекомендації з раціонального використання ресурсів та планування робіт,
- правила опрацювання збійних та помилкових ситуацій;
- правила підтримки роботи сервісу

На рівні моделі роботи сервісу інтелектуальні функції застосовують як окремі етапи виконання. Також використовуються правила, що стосуються конкретного етапу виконання запиту та відображають досвід та рекомендації з ефективного виконання. Застосування правил змінює параметри виконання або структуру виконання запиту.

Формальна модель системи

У [8] розглянута формальна специфікація інформаційної комп’ютерної мережі, яка використовується для проектування та керування розподіленими інформаційними системами і складається з декількох мережевих специфікацій. Доповнимо цю специфікацію специфікацією мережі сервісів:

$$S = \{NBp, NSe, NPc, NDv\},$$

де NBp – мережа бізнес-процесів, NSe – мережа сервісів, NPc – мережа процесорів, NDv – мережа пристроїв.

Мережа сервісів (Реєстр сервісів, service inventory [4]) – це множина сервісів:

$$NSe = M(Se).$$

На основі базових сервісів будують композиції сервісів [4], які взаємодіють задля досягнення певної мети. Сервіси не залежать від бізнес-процесів, так що кожен окремий сервіс бере участь у багатьох композиціях.

Композиція сервісів $CmSe$ – це підмножина сервісів та зв'язки між ними:

$$CmSe = \{M(Se)', M(LnSe)'\},\ M(Se)' \subset M(Se), M(LnSe)' \subset M(LnSe).$$

Зв'язок двох сервісів (Se_i, Se_j) – це пара $\{Se_i, FnInSe_j\}$, де $FnInSe_j$ – функція, що належить інтерфейсу сервісу Se_j

Сервіс Se визначається специфікаціями його інтерфейсу та реалізації

$$Se = \{InSe, \mathrm{Re}\, Se\}$$

де $InSe$ –інтерфейс сервісу, $\mathrm{Re}\, Se$ – внутрішня реалізація сервісу

Реалізація запропонованого інтелектуального сервісу містить такі частини:

$$\mathrm{Re}\, Se = \{CnSe, ItMoSe, LiPmSe, LiMoSe, LiCdSe, LiRuSe\}$$

де $CnSe$ – підсистема керування, $ItMoSe$ – інтерпретатор моделей, $LiPmSe$ – сховище параметрів, $LiMoSe$ – бібліотека моделей, $LiCdSe$ – бібліотека процедур, $LiRuSe$ – сховище правил.

Підсистему керування опишемо як набір операцій, які вона виконує:

$$CnSe = M(OpCnSe)$$

До найважливіших операцій належить вибір моделі виконання запиту ZpSe ($ZpSe \in InSe$):

$$Op_{\mathrm{mod}} : (ZpSe, LiPmSe, RlRuSe) \longrightarrow MoSe, MoSe \in LiMoSe$$

де $RlRuSe$ – множина релевантних правил зі сховища правил.

Вибір моделі для виконання операцій з підтримки роботи сервісу.

$$Op_{\mathrm{sup}} : (LiPmSe, RlRuSe) \longrightarrow MoSe, MoSe \in LiMoSe,$$

Сховище параметрів $LiPmSe$ можна визначити як множину параметрів $M(PmSe)$, в якій кожен параметр $PmSe = \{IdPmSe, val(PmSe)\}$, де $IdPmSe$ – ідентифікатор (тип) параметра, $val(PmSe)$ – значення параметра.

Визначимо бібліотеку моделей $LiMoSe$ як набір моделей:

$$LiMoSe = M(MoSe).$$

Кожна модель $MoSe$ – це набір окремих елементів виконання:

$$MoSe = M(OpMoSe).$$

Елемент виконання:

$$OpMoSe = \{IdOpMo, PmOpMo, RlRuSe, M(CdOpSe), NxOpSe\},$$

де $IdOpMo$ – ідентифікатор елемента, $PmOpMo$ – специфікація параметрів, $RlRuSe$ – множина релевантних правил, $M(CdOpSe)$ – множина процедур, що виконуються $M(CdOpSe) \subset LiCdSe$, $NxOpSe$ – функція виходу.

Функція $NxOpSe$ визначає наступний елемент виконання або закінчення моделі

$$NxOpSe : (LiPmSe, RlRuSe) \longrightarrow OpMoSe', OpMoSe' \in MoSe \vee OpMoSe' = \varnothing.$$

Сховище правил $LiRuSe$ – це множина правил:

$$LiRuSe = M(RuSe).$$

Правило $RuSe$ містить контекст виконання та дію, яка виконується, якщо контекст відповідає поточному стану

$$RuSe = \{CtRuSe, OpRuSe\}.$$

Контекст виконання – це булева функція, яка істинна, якщо визначені параметри лежать у заданих межах

$$CtRuSe : LiPmSe' \longrightarrow (true,\ false), LiPmSe' \subset LiPmSe$$

$$CtRuSe = \begin{cases} true, \forall PmSe \in LiPmSe'\ val(PmSe) \subseteq sg(PmSe) \\ false, в\ інших\ випадках \end{cases}$$

де $sg(PmSe)$ – визначений правилом діапазон значень параметра $PmSe$.

Бібліотека процедур – це набір виконувальних модулів (коду). Кожен модуль отримує параметри від інтерпретатора моделей та може зчитувати необхідні для виконання дані зі сховища параметрів. Результат виконання модуля повертається інтерпретатору і використовується для прийняття рішення щодо наступного елемента виконання

$$LiCdSe = M(CdSe)$$

$ItMoSe$ – це програма, що зчитує елементи виконання для моделі, виконує їх та визначає наступні елементи або завершує виконання моделі.

Висновки

Запропонована у роботі структура та головні принципи роботи інтелектуального сервісу інформаційної мережі дають змогу створювати системи, які порівняно з існуючими здатні до гнучкої адаптації до змін у зовнішньому середовищі при зменшенні видатків на супровід.

Розроблена специфікація інтелектуального сервісу буде використана в системі автоматизованого проектування інтелектуальних інформаційних систем.

1. Balmelli L, Brown D, Cantor M, Mott M. Model-driven systems development. // IBM Systems Journal, vol 45, Number 3, 2006. 2. Antoni Olivé. Conceptual Modeling of Information Systems. Springer-Verlag Berlin Heidelberg 2007. 3. Eric Evans, DOMAIN-DRIVEN DESIGN, Addison-Wesley, 2004. 4. Thomas Erl. SOA: principles of service design, Prentice Hall, 2007. 5.BPEL Cookbook Best Practices for SOA-based integration and composite applications development. Editors Harish Gaur, Markus Zirn. Packt Publishing Birmingham – Mumbai, 2006. 6. MDA Distilled, Principles of Model Driven Architecture, Stephen Mellor, Kendall Scott, Axel Uhl, Dirk Weise, Addison-Wesley Professional, 2004. 7. Pastor O., Molina J.C. Model-Driven Architecture in Practice. Springer-Verlag Berlin Heidelberg 2007. 8.Буров Є.В. Система формальних специфікацій для проектування розподілених інформаційних систем//.Вісник Держ. ун-ту "Львівська політехніка". – 2000. – № 406.