Inteligentné monitorovanie a modifikovanie výkonu informačných systémov (1)

Príspevok opisuje štruktúru a princípy nového typu architektúry informačných systémov vychádzajúcu zo štandardnej komponentovej architektúry. Uvedená architektúra poskytuje možnosť monitorovania a modifikácie výkonu týkajúceho sa vykonávania operácií poskytovaných komponentmi známymi z modelovacieho jazyka UML. Na to sa vo vykonávacom prostredí pre túto architektúru používajú inteligentné agenty na báze platformy JADE-LEAP.

Úvod

Architektúra informačného systému (AIS) hrá v oblasti softvérového inžinierstva kľúčovú úlohu pri tvorbe informačného systému (IS). Vo všeobecnosti sa na AIS môžeme pozerať ako na organizačnú štruktúru systému vrátane jeho rozkladu na súčasti, jeho prepojiteľnosti, interakcie, mechanizmy a smerné zásady [4]. AIS predstavuje najvyššiu úroveň koncepcie informačného systému v jeho vlastnom prostredí [3] a poskytuje rôzne možnosti, ale súčasne aj obmedzenia pre softvérovú implementáciu.

Význam voľby správnej architektúry informačného systému hrá jednu z kľúčových úloh pri jeho tvorbe. Práve od typu architektúry mnohokrát závisia také vlastnosti, ako sú napr. odolnosť, škálovateľnosť, modifikovateľnosť, flexibilita, adaptovateľnosť na rôzne zmeny prostredia (v ktorom je informačný systém prevádzkovaný), príp. kvalita bezpečnostnej infraštruktúry systému.

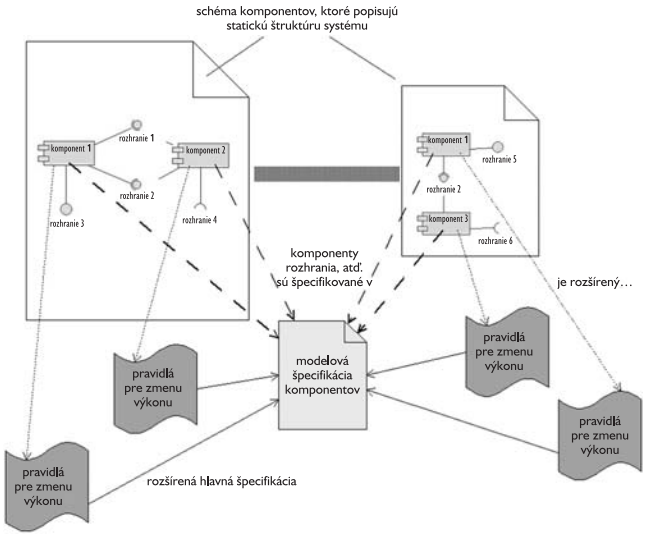
Architektúry založené na vývoji a používaní komponentov sú akceptované v rôznych oblastiach softvérového inžinierstva. Komponentová architektúra [1] (Component-Based Architecture – CBA) je architektúrou obsahujúcou mechanizmy a techniky na vývoj znovupoužiteľných softvérových jednotiek – komponentov. Špecifikácia unifikovaného modelovacieho jazyka (Unified Modeling Language – UML) definuje komponent takto: komponent reprezentuje modulárnu časť systému, ktorá zapuzdruje svoj obsah, a ktorej prejav môže byť nahradený vnútri jeho prostredia [2]. Komponent môže zastupovať čokoľvek, z čoho možno vytvoriť inštanciu počas vykonávania programu [1]. Vnútorná štruktúra komponentu môže obsahovať niekoľko ďalších komponentov. Komponent definuje svoje správanie v kontexte svojich požadovaných a poskytovaných rozhraní [2]. Poskytované rozhrania definujú zoznam operácií, ktoré sú komponentom poskytované, požadované rozhrania definujú zoznam operácií požadovaných komponentom. Komponent môže byť fyzicky reprezentovaný autonómnymi, vykonávateľnými zoskupeniami, napr. dynamickými knižnicami platformy. NET či JAR súbormi známymi z jazyka Java.

Náš návrh

Nami navrhnutá architektúra pre informačné systémy rozširuje štandardnú komponentovú architektúru v nasledujúcich dvoch oblastiach:
- návrh IS,
- vykonávanie IS.

V oblasti návrhu IS rozširuje model IS navrhnutý pomocou modelovacieho jazyka Unified Modeling Language (UML) o dva druhy špecifikácií, pomocou ktorých možno špecifikovať:
- miesto monitorovania výkonu IS,
- pravidlá modifikácie výkonu IS.

Špecifikácie modifikácie výkonu sa vzťahujú na tú časť UML modelu, ktorá opisuje statickú štruktúru IS, týkajúcu sa komponentov. Rozširujú sémantický základ modelu, v ktorom je textovo opísaná sémantika jednotlivých prvkov modelu (obr. 1).



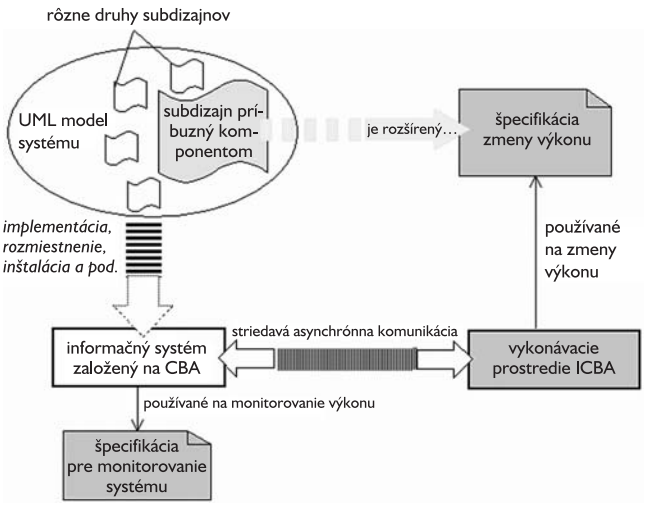
Obr.1 Rozšírenie komponentového UML modelu o špecifikácie na modifikovanie výkonu

Architektúra bola nazvaná ako inteligentná komponentová architektúra (Intelligent Component-Based Architecture – ICBA), nakoľko sa počas vykonávania IS na báze tejto architektúry používajú inteligentné agenty. Monitorovanie a modifikácia výkonu sa týka vykonávania požadovanej operácie, resp. skupiny operácií na jednom alebo viacerých komponentoch.

Monitorovať možno:
- čas vykonania jednej operácie (v [ms]),
- čas vykonania skupiny operácií, resp. určitej jej časti (tiež v [ms]).

Modifikácia výkonu sa týka vykonania:
- jednej operácie určitého komponentu,
- skupiny operácií – tej istej operácie, resp. viacerých operácií na jednom alebo viacerých komponentoch.

Modifikácia výkonu spočíva v urýchľovaní/spomaľovaní vykonávania jednej operácie, resp. skupiny operácií pre konkrétneho používateľa. Treba si totiž uvedomiť, že pri určitých typoch IS, ako sú napr. rôzne digitálne knižnice či informatívne služby, možno očakávať zvýšený nápor používateľov na určité služby IS. Pritom nie každý používateľ pristupuje k službám takýchto IS rovnako často a podobne, nie každý z nich vykonáva tie isté operácie v rámci daných služieb (napr. upload/download) a nie každý z nich zaťažuje IS rovnako. Takto trvá niektorému používateľovi vykonanie požadovanej služby (v CBA najčastejšie realizovanej ako operácia určitého komponentu) krátko, niektorému príliš dlho. Bolo by preto vhodné, keby štandardná CBA obsahovala mechanizmus, pomocou ktorého by bolo možné vykonávanie v prípade konkrétneho používateľa urýchliť, resp. spomaliť – čo práve sčasti rieši ICBA.



Obr.2 Inteligentná komponentová architektúra

Výkon reprezentujúci v kontexte ICBA čas vykonania určitej operácie, resp. skupiny operácií je veľmi dôležitý faktor z hľadiska vývoja softvéru (konkrétne – vykonanie služby pre konkrétneho používateľa), tak aj z hľadiska jeho používania, nakoľko vykonanie jednej, resp. viacerých operácií poskytovaných komponentmi v CBA ústi do konečného reakčného času systému pre jeho používateľov. Na modifikáciu výkonu tiež treba upraviť okrem statickej štruktúry aj tú časť komponentového modelu systému, ktorá opisuje dynamické správanie systému s cieľom komunikácie s vykonávacím prostredím podporujúcim vykonávanie IS na báze ICBA (ICBA Runtime Environment – IRE).

Štruktúra a hlavné časti ICBA

Navrhnutá architektúra (obr. 2) pozostáva z nasledujúcich častí:
- UML model navrhnutého systému – štandardný úplný model IS navrhnutý pomocou jazyka UML, ktorého čiastkové modely týkajúce sa komponentov o. i. opisujú aj miesta monitorovania a modifikácie výkonu;
- špecifikácia pravidiel na modifikovanie výkonu – textová špecifikácia opisujúca spôsob modifikovania výkonu; vzťahuje sa na konkrétne komponenty alebo na konkrétne operácie komponentov;
- IS založený na štandardnej komponentovej architektúre – predstavuje vykonávateľnú formu IS na báze štandardnej CBA, ktorého návrhové modely (vytvorené v UML) týkajúce sa komponentov a dynamického správania systému boli rozšírené o špecifikáciu na modifikovanie výkonu; takýto IS môže obsahovať jednu alebo viacero špecifikácií na monitorovanie výkonu;
- vykonávacie prostredie ICBA – umožňuje samotné monitorovanie a modifikáciu výkonu; s informačným systémom komunikuje pomocou TCP protokolu, pričom je možné využívanie tohto prostredia aj viacerými IS nachádzajúcimi sa na vzdialených počítačových uzloch; toto vykonávacie prostredie používa špecifikáciu na modifikáciu výkonu a zodpovedá za inteligentné agenty, ktoré na základe tejto špecifikácie príslušne upravia vykonávanie požadovanej operácie, resp. skupiny operácií; hoci je vhodné túto špecifikáciu vytvárať ešte v čase návrhu IS, je možná aj jej neskoršia, viacnásobná zmena (podobne je možná aj viacnásobná neskoršia zmena špecifikácie na monitorovanie výkonu);
- špecifikácia miesta/miest na monitorovanie výkonu – v prípade volania jednej operácie určuje, či máme záujem monitorovať časy jej jednotlivých vykonaní; v prípade volania skupiny operácií určuje jednotlivé miesta, v ktorých máme záujem odmerať čas parciálneho vykonania.

Literatúra

(vybrané tituly)

[1] AMIR, R., ZEID, A.: A UML profile for service oriented architectures, Conference on Object Oriented Programming Systems Languages and Applications. Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, Vancouver, BC, CANADA, 2004, ACM Press.

[2] ENDREI, M., ANG, J., ARSANJANI, A., CHUA, S., COMTE, P., KROGDAHL, P., LUO, M., NEWLING, T.: Patterns: Service-Oriented Architecture and Web Services, WebSphere® software, IBM, RedBooks.

[3] Institute of Electrical and Electronics Engineers (IEEE): IEEE Standard 1471 (Recommended Practice for Architectural Description of Software-Intensive Systems), 2000.

[4] RUMBAUGH, J., JACOBSON, I., BOOCH, G.: The Unified Modeling Language Reference Manual. Addison-Wesley Press, 1998.

Ing. Miroslav Beličák

Univerzita Karlova v Praze
Matematicko-fyzikální fakulta
Katedra softwarového inženýrství
Malostranské nám. 25
118 00 Praha 1, ČR
e-mail: Miroslav.Belicak@gmail.com