METODY POROVNÁVÁNÍ VLASTNOSTÍ INFORMAČNÍCH SYSTÉMŮ

METHODS OF COMPARISON OF PROPERTIES OF INFORMATION SYSTEMS

Ivan Vrana a Jan Vrána

Anotace:
Požadovanou funkcionalitu IS při zpracování určitých datových struktur lze zpravidla docílit různými alternativními postupy nebo technologiemi, které se mohou podstatně lišit svými vlastnostmi, např. náklady na vytvoření funkcionality, náklady na rozvoj, náklady na provoz i provozními vlastnostmi výsledného systému. Příspěvek se bude zabývat metodami porovnávání některých typů vlastností informačních systémů v kritických aplikacích.

Klíčová slova:
informační systém, jakost, náklady, funkcionalita, rozvoj, provoz

Abstract:
Required functionality in procesing of certain data structures could be achieved by many alternative solutions which could considerably differ by their overall properties, e.g. costs of initial functionality design, development and operational costs, run-time costs and technical parameters of the resulting information system (e.g. access time). The paper will deal with methods of assessment and comparison of certain types of properties of information systems.

Keywords:

information systém, quality, costs, functonality, development, operation

ÚVOD

Tato práce má za cíl popsat způsob relativního hodnocení kvality počítačových aplikací, které umožní porovnávat různé alternativní technologie a implementační metody pro řešení stejného problému u kritických aplikací. Zaměříme se zejména na:

- nároky na vývoj prvotní funkčnosti aplikace,
- nároky na údržbu a rozšiřování funkčnosti aplikace,
- provozní efektivitu vytvořené aplikace.

Zmiňované tři faktory mají zásadní vliv na (komerční) úspěch vytvářené aplikace. První dva faktory představují přímé ekonomické faktory vývoje a třetí zmiňovaný faktor má principiální vliv na praktickou použitelnost nebo nepoužitelnost vytvářené aplikace.

Budeme uvažovat aplikace kritické z hlediska objemu spravovaných dat a provozních požadavků. Předpokládejme, že vyvíjená aplikace používá relační databázi. Aplikační logika implementuje všechny algoritmy business logiky aplikace a je nezávislá na způsobu uložení dat. Správa dat naproti tomu implementuje algoritmy pro uložení a manipulaci dat.

RELATIVNÍ HODNOCENÍ KVALITY POČÍTAČOVÝCH APLIKACÍ

Nároky na prvotní vývoj funkčnosti aplikace

Nároky na prvotní vývoj funkčnosti aplikace lze chápat jako souhrn nákladů, které je nezbytné vynaložit na vytvoření základní funkčnosti zamýšlené aplikace. Prvotní náklady můžeme dále rozdělit na náklady jednotlivých fází životního cyklu, např. *analýzy* problému, *návrhu* řešení, *implementace* a *testování*. Pro zjednodušení budeme uvažovat, že při prvotním vývoji funkčnosti projde aplikace všemi fázemi pouze jednou.

Složitost *analýzy* aplikační problematiky je závislá hlavně na složitosti problematiky samotné a téměř vůbec nezávisí na způsobu návrhu a implementace aplikace, proto nebudeme dále fázi analýzy uvažovat při srovnávání možných alternativních řešení a jejich implementačních metod.

Složitost *návrhu* výrazně závisí na složitosti samotné problematiky, ale také na použitých implementačních technologiích. Spočívá především v odhalení a návrhu všech možných větví funkčnosti vytvářené aplikace a v návrhu jejich logických algoritmů. Abstraktnost navrhovaných algoritmů a jejich parametrizace je úměrná zamýšlené obecnosti a univerzálnosti. Nejhrubší kvantifikací pracnosti návrhu je množství spotřebovaného času, přestože v tomto měřítku není zohledněn charakter úlohy, zkušenost pracovníka, apod. Charakter úlohy lze zohlednit např. *koeficientem složitosti* dané úlohy.

Pracnost fáze *implementace* lze kvantifikovat například rozsahem (počtem řádků) programového kódu. I zde je ale nezbytné brát v úvahu rozdílnou složitost vytvářeného programového kódu při použití různých implementačních metod. Složitost fáze *ladění* lze pro účely porovnání s jinými alternativami kvantifikovat například počtem větví programu, které musí být prověřeny a otestovány a korigovat např. koeficientem složitosti. *Celkové nároky* na vývoj prvotní funkčnosti jsou souhrnem nároků na výše uvedené činnosti.

Nároky na údržbu a rozšiřování funkčnosti aplikace

Softwarové dílo rozsahem větší než školní úloha nelze vytvořit během jednoho vývojového cyklu. Většina aplikací projde během svého života více než jedním hlavním cyklem vývoje své funkcionality. S tím je nutné počítat už od počátku při přijímání strategických rozhodnutí o použitých technologiích. I v případě náročnosti údržby a hlavně rozšiřování funkčnosti lze celkovou náročnost rozdělit na jednotlivé dílčí fáze, na analýzu problému, návrh řešení, implementaci a testování. Složitost fáze *analýzy* problému opět závisí především na složitosti problému samotného a téměř vůbec nezávisí na použité implementační metodě. Proto ji pro další srovnávání nebudeme uvažovat. Pracnost a náročnost fáze *návrhu* výrazně závisí na zvolené implementační metodě a na charakteru zamýšlených změn a rozšíření funkčnosti. Při malých změnách funkčnosti, které spíše přidávají nové funkčnosti bez změny funkčnosti stávající, nemusí být závislost pracnosti na zvolené implementační metodě výrazná. V případě, že zamýšlené změny funkčnosti mají vliv i na stávající funkčnost, nebo způsobí násobný nárůst složitosti, je náročnost a pracnost návrhu takových změn značně závislá na použité implementační metodě. Jedním z možných kritérií kvantitativního ohodnocení může být např. relativní rozsah oblastí, do nichž je potřeba zasáhnout vůči všem oblastem upravované aplikace (a korigovat faktorem složitosti). Náročnost a složitost *implementace* změn je na použité implementační metodě závislá přímo. Při nevhodně zvolené implementační metodě se každá změna funkčnosti celé aplikace projeví i změnou funkčnosti uložení a správy dat. V případě použití vhodné implementační metody se ale většina pozdějších úprav funkčnosti celé aplikace nemusí do uložení a správy dat vůbec promítnout. Složitost a náročnost *testování a ladění* nové funkčnosti je úměrná rozsahu a

charakteru změn, které nastaly v předchozím kroku. Pro tuto fázi platí stejná charakteristika, jako pro fázi implementace. Hlavní podíl na *celkových nárocích* na údržbu a rozšiřování funkčnosti aplikace mají zejména pozdější fáze životního cyklu.

Provozní efektivita vytvořené aplikace

Předchozí dva oddíly nepřímo pojednávaly o ekonomických hlediscích nebo měřítkách, která je nutno brát v úvahu při rozhodování o technologiích použitých ve vytvářené aplikaci a o tom, jak tato měřítka závisí na použitých implementačních metodách. Nyní kromě ekonomického hlediska bude předmětem zájmu zejména principiální použitelnost nebo nepoužitelnost vytvářené aplikace v reálných provozních podmínkách.

Při posuzování provozní efektivity vytvářené aplikace, resp. uložení a správy jejích dat, je pro každou uvažovanou implementační metodu potřeba zkoumat nároky, které má tato metoda na uložení předpokládaného množství vstupních údajů a principiální složitost nejčastějších operací (selektivní výběry, hromadné výběry, srovnávání, modifikace, atd.) i následky možného nárůstu objemu a složitosti vstupních dat . Podcenění provozní efektivity může zkomplikovat reálné využití vytvářené aplikace.

UVAŽOVANÉ METODY SPRÁVY A ULOŽENÍ DAT

Pro účely základního porovnání vlastností různých alternativních implementačních metod softwarového díla budeme uvažovat následující metody správy a uložení dat:

- Obecnou „objektovou“ metodu
- Pevnou datovou strukturu
- Dynamické relační uložení.

Tyto metody jsou podrobněji popsány v [2] – [4].

Hlavní myšlenkou **obecné objektové metody** je dosáhnout maximální věrnosti uloženého modelu s možností ukládat model předmětného systému se vší obecností v přirozené podobě, jako jednotlivé objekty a vazby mezi nimi. Hlavním cílem tohoto přístupu je:

- uložit logický model předmětného systému se vší obecností,
- umožnit maximální nezávislost aplikací na konkrétní aplikační doméně a
- umožnit maximální flexibilitu aplikací vůči změnám aplikační domény.

Hlavní předností metody obecného objektového uložení dat je její obecnost a univerzálnost. Náročnost vývoje aplikace je úměrná hlavně rozsahu a složitosti aplikační domény. Náročnost všech fází vývoje prudce klesá v pozdějších fázích života aplikace. S relativně malou náročností je také možné provést nasazení vyvinuté aplikace do jiné aplikační domény. Podstatným nedostatkem metody obecného objektového uložení dat je špatná provozní efektivita aplikací založených na této metodě. Dalšími méně závažnými nedostatky jsou nároky na abstraktní myšlení návrhářů a nedostatek prostředků pro vytváření, správu a ladění logického modelu aplikační domény. Metoda obecného uložení dat je vhodná pouze pro správu aplikačních domén malého rozsahu ale s rozmanitou datovou strukturou.

Hlavní myšlenkou **pevné datové struktury** je dosáhnout maximální provozní efektivity vytvářené aplikace. Předmětný systém je datově popsán normalizovaným E-R modelem pomocí entit a jejich vztahů. Entity jsou potom uloženy jako tabulky v hostitelské relační databázi. Funkčně je předmětný systém popsán pomocí množiny algoritmů zpracovávajících jednotlivé vytvořené entity a jejich vztahy. Zpravidla se nepředpokládá nasazení vyvíjené aplikace (informačního systému) do jiné aplikační domény.

Hlavním cílem této metody uložení a správy dat je:

- využít všech možností relační technologie hostitelského databázového stroje,

- umožnit maximální provozní efektivitu aplikací založených na této metodě využívající relačního zpracování uložených dat.

Přednostmi této metody jsou provozní efektivita a malá náročnost vývoje prvotní funkcionality. Zásadním nedostatkem je její nepružnost vůči změnám struktury aplikační domény. Metoda pevné datové struktury je vhodná pro aplikační domény, které mají relativně málo rozmanitou strukturu, která se prakticky nemění. Aplikační doména ale může být velmi rozsáhlá co do objemu dat.

Metoda dynamického relačního uložení dat se snaží o dosažení co možná nejvyšší obecnosti a flexibility vůči změnám aplikační domény, při zachování vysoké provozní efektivity na základě maximálního využití relačních vlastností hostitelského relačního databázového stroje. Transformovaný logický model datové struktury aplikační domény je v obecné podobě uložen ve formě metadat a stává se řídícím elementem chování aplikace. Na základě metapopisu je v hostitelské databázi *dynamicky vygenerována* relační struktura specifických tabulek a vazeb. V každém okamžiku jsou data uložena v pevné „nativní" relační datové struktuře, ve které mohou být velmi efektivně zpracovávána, ale kdykoliv je možné bez zásahu do programového kódu pouze změnou metapopisu změnit strukturu dat a tím i chování celé aplikace.

Obecnost metody dynamického relačního uložení dat umožňuje vytvářet aplikace velmi obecné a nezávislé na konkrétní aplikační doméně, které mají velmi malé nároky na pozdější údržbu a rozvoj. To ji předurčuje k použití v aplikačních doménách, které rychle mění svoji strukturu. Vyvinuté aplikace jsou také schopny relativně snadného nasazení v jiné aplikační doméně. Relační uložení a zpracování dat zase dává aplikacím vynikající vlastnosti z hlediska provozní efektivity a předurčuje tak tuto metodu pro nasazení v datově i strukturně velmi rozsáhlých aplikačních doménách.

Hlavními nevýhodou metody je veliká abstraktnost algoritmů a nutnost vyvíjet a udržovat metadata, což se projeví ve zvýšených nárocích na kvalifikaci a schopnosti vývojového týmu.

Metoda dynamického relačního uložení dat je velmi vhodná pro vytváření aplikací spravujících rozsáhlou a rychle se měnící aplikační doménu. Vyvinutou aplikaci lze navíc s relativně velmi malými náklady dále nasadit na mnohem menší nebo méně dynamickou aplikační doménu.

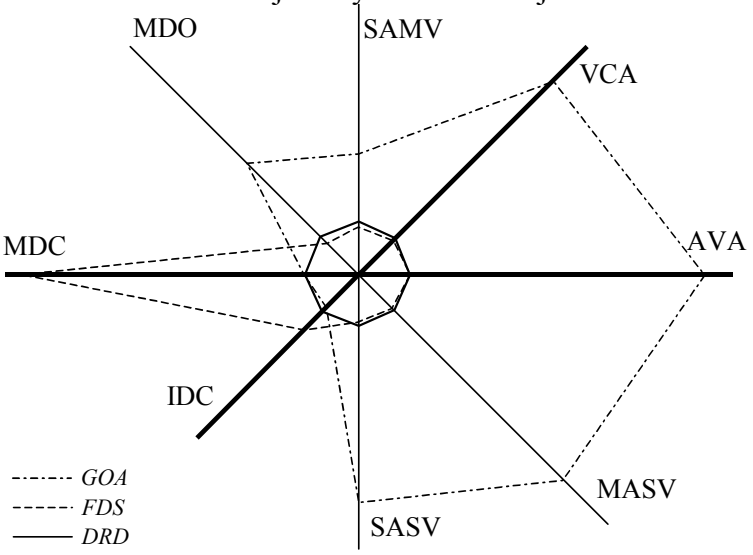
POROVNÁNÍ POPISOVANÝCH METOD

Zde stručně shrneme a porovnáme základní vlastnosti porovnávaných tří metod pro uložení a správu dat. Protože konkrétní vlastnosti jednotlivých metod závisí na charakteru aplikační domény, pro účely tohoto porovnání jsme jako jednotnou aplikační doménu pro všechny uvažované metody zvolili správu rozsáhlé technologické sítě mobilního telefonního operátora. Pro porovnání použijeme následující hlediska:

- Pracnost prvotního vývoje (IDC)
- Pracnost údržby a dalšího vývoje (MDC) funkčnosti aplikace
- Celkovou datovou náročnost (TDC) – součet počtů řádků ve všech tabulkách databáze
- Metadatovou režii (MDO)
- Průměrnou velikost metadatových tabulek (MTS) –vyjadřuje časovou náročnost SQL operací
- Průměrnou velikost datových tabulek (DTS)
- Náročnost výběru (počet SQL dotazů) jedné hodnoty jednoho atributu (SASV)
- Náročnost výběru všech historických hodnot jednoho atributu (SAMV)
- Náročnost výběru aktuálních hodnot všech atributů jednoho řádku (MASV)
- Náročnost výběru všech platných kombinací hodnot atributů jednoho řádku (VCA)

- Náročnost výběru aktuálních hodnot všech atributů dané tabulky parametrů (AVA)

Celkový přehled vlastností jednotlivých metod ve vzájemných souvislostech souhrnně graficky znázorňuje následující normalizovaný pavučinový diagram. Čím menší je hodnota hlediska (čím je hodnota blíže středu), tím je daná metoda v tomto hledisku lepší. Hlediska nejvýznamnější pro celkové hodnocení jsou vyznačena silnější osou .



Z grafu vyplývá, že obecná objektová metoda GOA vykazuje nejlepší vlastnosti v nárocích na vývoj (prvotní funkčnosti a údržby) aplikací. Ve všech hlediscích vyjadřujících provozní efektivitu, především v nejčastějších operacích, vykazuje metoda GOA fatální nedostatky, které prakticky znemožňují její reálné použití v kritických aplikacích.
Metoda pevného uložení dat FDS má naopak hlediscích provozní efektivity vynikající vlastnosti. Naprostou většinu operací je možno realizovat velmi jednoduchým a efektivním způsobem. Cenou za tuto vysokou provozní efektivitu je však vysoká náročnost vývoje, především pak náročnost údržby a rozvoje, která tuto metodu taktéž prakticky diskvalifikuje.
Z grafu je vidět, že metoda dynamického relačního uložení dat DRD je v provozní efektivitě nepatrně horší než metoda FDS, ale zároveň o mnoho řádů lepší než metoda GOA. V náročnosti vývoje je metoda DRD naopak o něco horší než v tomto ohledu nejlepší metoda GOA, ale je mnohem lepší než metoda FDS. Metoda DRD přebírá od obou srovnávaných extrémních metod jejich vynikající vlastnosti a naopak eliminuje jejich velmi nepříznivé vlastnosti, které tyto metody prakticky vylučují z reálného použití pro uvažovaný typ aplikací.

ZÁVĚR

Popsaná metoda hodnocení je vhodná pro posouzení základních kvalitativních vlastností alternativ řešení softwarových aplikací. Pomocí ní lze rozlišit dobré řešení od špatného. Vzhledem k tomu, že řadu parametrů hodnocení je možné pouze kvalifikovaně odhadnout, neposkytuje tato metoda (stejně jako jiné metody hodnocení) přesné kvantitativní vyjádření vlastností jednotlivých řešení. Proto se nehodí pro jemné rozlišení kvality podobných řešení.

Literatura:

[1] J. Vrána: Methods of employing metadata for managing large systems. In DATESO'02 – *Proceedings of Workshop on Databases. Ostrava, ISBN 80-248-0080-2, pp. 44-56, 2002.*

[2] J. Vrána: Dynamic relational data storing method for management of large data systems. *Doctoral dissertation. Czech University of Technology in Prague, 2003.*

[3] J. Vrána and I. Vrana: Effective method for management of large data systems. In: SCI – 8th World Conference on Systemics, Cybernetics and Informatics, Orlando, 2004.

[4] J. Vrána and I. Vrana: Application of the Dynamical Relational Data Storing Method. In: CITSA Int. Conference, Orlando, 2004.

Kontakt:

Prof. Ing. Ivan Vrana, DrSc, - KII PEF ČZU v Praze, vrana@pef.czu.cz

Ing. Jan Vrána, PhD, - Komix s.r.o. Praha, vrana@komix.cz