Automatizace workflow a procesů

Závěrečná zpráva do předmětu Teorie programovacích jazyků
Tomáš Vojta, 12. března 2003

1) Technologie modelování procesů

V této kapitole shrneme výchozí pojmy a přístupy, používané pro modelování procesů. Bude spíše stručná a informativní, neboť jde zpravidla o opakování známých pojmů.

1.1 Metodologie modelování procesů

Metodologie modelování procesů můžeme rozdělit do těchto tří skupin:

- Založené na komunikaci
- Založené na artefaktech
- Založené na aktivitě

Poznamenejme, že každá reprezentace procesu či workflow může využívat všech tří metodologií nebo jejich částí.

1.2 Koncepty modelování procesů

Vyjmenujme pohledy, kterými můžeme nahlížet na techniku modelování procesů:

- Funkcionální (Data flow diagramy)
- Behaviorální (kdy a/nebo za jakých podmínek)
- Strukturní (ER diagramy, hierarchie tříd)

1.3 Analýza modelu

Zmíněné modely slouží pro testování a analýzu procesů. Hlavní úlohou je však odhalit možné chyby a odhadnout výkon workflow systému. Proto je nutné zodpovědět i otázky

- Dosažitelnosti a bezpečnosti
- Uváznutí
- Konfliktů
- Správnosti

Kromě těchto kvalitativních charakteristik sledujeme i charakteristiky kvantitativní. Metody zjišťování obojího druhu charakteristik jsou založeny na *analýze* (vždy využívá formální metody) nebo na *simulaci* (využívá monitorování a animaci).

Mezi formálními metodami mají význačnou pozici Petriho sítě, existují však i jiné. Formální metody není zpravidla nutné použít na celý modelovaný proces. Její použití na klíčové části však může zvýšit důvěru v model.

2) Technologie modelování workflow

Definujme nejprve pojem *workflow*. *Workflow systém* je (neformálně řečeno) systém, který zahrnuje nejen prvky a vzájemné vztahy mezi nimi, ale i časové uspořádání a podmíněnost událostí v systému. Důraz je přitom položen právě na průběh – tok činnosti systému. Pod pojmem *workflow* myslíme jeden průběh procesu – nákup zboží, apod..

Věnujme se v této kapitole správnosti a výkonnosti provádění workflow (WF).

2.1 Paradigmata požadavků na workflow (Workflow Enactment Paradigms)

Tato paradigmata jsou hlavním a nejobecnějším modelem provádění workflow. Většina systémů je založena na plánovači, který zadává entitám úlohy k provedení. Dalším z možných paradigmat je tok dat (data flow), které je vhodnější pro problémově orientované aplikace s méně pevně definovanou strukturou.

Další modely mohou být založeny na vyhledávání informací (information pull) – činnost je založena na spolupracujících inteligentních agentech, kteří pro uživatele získávají informace.

2.2 Záruky provádění workflow systému

Každá prováděcí jednotka může zaručit některé vlastnosti jakéhokoli prováděného WF systému:

- správnost provádění jedné instance WF
- transakční vlastnosti:
  - atomicita selhání (WF je buď proveden nebo ne. V případě selhání nezůstanou žádné částečně provedené akce – např. při převodu peněz mezi dvěma účty odepsání z jednoho a nepřipsání na druhý)
  - konzistence dat (Všechny prováděné workflow mají k dispozici správná a aktuální data. Problémem je v tomto případě souběžný přístup více (mnoha) WF k jednomu údaji – např. stav prostředků na účtu)
- termíny:
  - „tvrdé“ – úloha je provedena v termínu nebo zrušena
  - „měkké“ – cílem je splnit maximum úloh v termínu

Zda budou tyto vlastnosti zaručeny záleží na požadavcích uživatele. Pokud bude WF systém nastaven příliš přísně, sníží se výkon WF, naopak volně definované požadavky na prováděcí jednotky mohou vést k chybám. Poznamenejme, že neexistuje jedno univerzální nastavení parametrů WF systému, které by vyhovovalo všem aplikacím.

2.3 Jazyky pro popis WF

Specifikace WF nezávisí na prováděcím paradigmatu. Jazyků pro popis WF je celá řada – procedurální, deklarativní, založené na pravidlech, toku dat či Petriho sítích, visuální, aj. Dva společné požadavky na jazyky jsou:

- jednoznačnost
- výrazová síla (schopnost specifikovat úlohy a vztahy mezi nimi)

Dále by jazyk měl být schopen vyjádřit očekávané záruky systému, zodpovědnost prováděcích jednotek za úlohy, předpokládané náklady a doby provádění, aj.

Z dalších požadavků na jazyk uveďme modulární strukturu (znuvupoužitelnost kódu) a správa chyb. Rovněž by měl být natolik formální, aby umožňoval analýzu správnosti, jako deadlock, livelock, dosažitelnost, apod..

Uživatelské rozhraní

Uživatelské rozhraní úzce souvisí s výrazovou síla jazyka. Nejobvyklejší je grafické uživatelské rozhraní, které je možno překládat do libovolného specifikačního jazyka.

Snadnost použití rozhraní je nepřímo úměrná jeho modelovací síle. Protože návrhář musí být schopen popsat i složité WF, nebude tuto úlohu schopen provádět každý. Naopak každý by měl být schopen:

- použít pracovní list vyplývající z WF (pracovník)
- začít, řídit a monitorovat provádění WF (pracovník, manažer)

Situace je podobná tradičnímu programování – ne každý umí (a má) programovat, ale každý by měl umět ovládat (dobře navržený) program.

Specifikační spolupráce (specification interoperability)

Tento jev je v současnosti bohužel prakticky vyloučen. Snaha Workflow Management Coalition o standard („Interface 1“ – blíže viz podkapitola 4.2) nestačí – jedním z hlavních problémů je různá výrazová síla modelovacích jazyků různých WF.

2.4 Softwarová architektura služby zajišťující požadavky na workflow (Workflow Enactment Service)

Z mnoha možných architektur systémů správy WF jmenujme základní typy: centralizovaná (úlohy jsou zpracovávány jediným plánovačem), částečně distribuovaná (každé zpracovávané workflow obsahuje vlastní plánovač) či úplně distribuovaná (úlohy se synchronizují na základě vzájemné komunikace samy). Variantou plně distribuované architektury je systém využívající agenty.

Škálování a výkon

Jsou často považovány za nejdůležitější charakteristiky systému. Mají několik aspektů:

- počet úloh zpracovávaných jedním workflow
- počet instancí workflow v systému
- množství zpracovávaných dat

Požadavky na škálovatelnost vycházejí z předpokládaného růstu společnosti – do WF je třeba přidávat role, uživatele, aplikace aj. při zachování (vysokého) výkonu.

K tomu je možno využít pokroků v klient-server zpracování a replikaci databází. Řešení zajišťující škálovatelnost a schopná vyrovnat se s malou šířkou komunikačního pásma však bohužel zpravidla vyžadují homogenní prostředí.

Spolehlivost

Pro většinu uživatelů je pak nejdůležitější vlastností WF schopnost běžet 24 hodin denně 365,25 dní v roce. Hlavní využívanou technikou je replikace a použití záložních serverů. Rovněž formální testování a důkazy správnosti mohou být použity, i když v současnosti nejsou efektivní.

K problémům hardwarových a softwarových selhání se přidávají problémy vyvstanuvší za běhu WF („nedostatek prostředků na účtu“ apod.). I tyto problémy musí WF ošetřit (a vyřešit např. použitím heuristických metod). Zmiňme rovněž rozdíl mezi *spolehlivostí* (souvisí s odolností vůči chybám) a *obnovitelností* (souvisí s integritou dat).

Spolupráce při zpracování (Execution interoperability)

WF různých dodavatelů by měly spolupracovat. Největší úsilí na poli standardizace (přičemž standard je nutnou podmínkou každé spolupráce) vychází z WfMC (Workflow Management Coalition). WfMC definuje pro tento účel referenční model a obecná rozhraní.¨

Dalším rozměrem spolupráce je možnost využívat aplikací třetích stran různými WF systémy. Tyto standardy již existují (COM, CORBA) a umožňují znovupoužití komponent, škálování.

Pružnost (Flexibility)

Systém správy WF by měl být snadno upgradovatelný. Rovněž jeho konfigurace by měla být schopná reflektovat změny struktury organizace. Problém flexibility zahrnuje jak změnu výkonných jednotek WF (měly by být nezávislé na jazyce, v němž je WF specifikován).

Workflow by rovněž měl umožnit změnu instance za běhu – např. vzhledem k náhlé změně okolních podmínek. Tato změna by však měla mít daná omezení neboť může narušit bezpečnost systému či záruky jeho provádění.

2.5 Bezpečnostní hlediska

Hledisko bezpečnosti nemůže být nikdy dost zdůrazněno. Shrňme hlavní otázky bezpečnosti:

- autorizace
- zodpovědnost a delegace zodpovědnosti
- autentizace
- ochrana citlivých dat při přenosu

Někdy je nutné zabezpečení poněkud omezit – v případě nedostupnosti (nemoc, dovolená, úmrtí) klíčového zaměstnance (key escrow). Záznamy o takových událostech je ovšem potřeba spolehlivě zachovat.

2.6 Implementační otázky

Infrastruktura software

Služba zajišťující WF musí využívat jak zaběhnuté standardy, tak nové technologie. Mezi nejpoužívanější komunikační technologie patří:

- TCP/IP
- RPC
- DOM
- CORBA, DCOM, OLE
- Web (HTTP, HTML/XML)

Mezi hlavní nevýhody webově orientovaného přístupu patří:

- bezestavovost HTTP protokolu
- pasivita serveru
- zpracování chyb

Důležitými nástroji pro uložení, přístup a řízení konzistence dat jsou databázové systémy. Proto by měl každý WF systém využívat (distribuovaného/replikovaného) databázového systému. Databáze slouží i k uchování stavu instance WF a specifikace WF. Výzkum na poli distribuovaných objektů probíhá a mohl by ukázat výhodné spojení databázové a komunikační infrastruktury právě v kontextu WF systémů.

Bezpečnost: až na čestné výjimky (Kerberos, HTTPS, SSL, asymetrická kryptografie) nejsou zavedené postupy pro použití WF systémy. A to navzdory významu bezpečnosti v oblasti workflow.

Správa úloh a „wrapper" aplikací

Zahrnutí existujících aplikací do WF systému je velmi důležité, proto je třeba vstupy a výstupy aplikací obalit („wrap") tak, aby komunikovaly se zbytkem WF systému. Wrapper může rovněž zajišťovat funkce jako logování aktivity, řízení přístupu, apod.. Použití wrapperu není výhodné vždy – některé aplikace (např. CASE systémy) vyžadují dvojstrannou komunikaci – tj. využívá jak WF aplikaci, tak aplikace WF systém.

Správce seznamu úkolů je specifickým správcem úloh. Předkládá lidskému uživateli seznam úloh, které má tento vykonat. Poznamenejme, že obecně je to velmi složitý problém. Obvyklé řešení je založeno na jednoduchém rozlišování rolí. Systémy založené na rozlišení rolí na základě zkušenosti či zamezující sdružování rolí (úředník v bance povolí i potvrdí úvěr) nejsou v současnosti implementovány.

Monitorovací nástroje

Monitorování a analýza běhu WF a nástroje pro ně jsou základem efektivity WF. Jmenujme od nejjednodušších aktivity, které nástroje umožňují:

- kontrola průběhu WF
- informace o stavu úloh (úspěšnost ukončení, doba běhu)
- podklady pro audit (pro bezpečnostní účely)
- historie běhu (analýza výkonu, vizualizace)

- predikce běhu úloh

Monitorovací aplikace by měly být aktivní a automaticky reagovat na některé události. Například na nezvyklé podmínky a buď podniknout opatření nebo informovat lidskou obsluhu.

Mnoho produkčních WF systémů umí generovat žádost o manuální zásah (Request for Manual Assistance, RMA). S rostoucí složitostí systémů reálná schopnost obsluhy problém vyřešit klesá a naopak roste potřeba automatických cest řešení problémů.

Mobilita

Mobilní mohou být nejen klienti WF, ale i výkonné jednotky. Typickými problémy mobilních součástí systému jsou častá odpojení, nízká šířka pásma a výkonová omezení. Do praktického využití mobilních WF systémů je nutné pracovat na odstranění vlivu těchto omezení na jejich (WFS) výkonnost a spolehlivost.

3) Transakční vlastnosti workflow systémů

V předcházejících kapitolách jsme viděli, že workflow zahrnují koordinované provádění vícera úloh různými entitami na více místech. I když workflow mohou být specifikovány pomocí různých paradigmat, je žádoucí, aby byly zachovány alespoň některé funkce zabezpečené tradičními transakčními systémy.

3.1 Tradiční transakce

V prvních databázích museli programátoři zachovávat konzistenci dat používáním *metodologie* – takových postupů, aby k zavedení nekonzistence nemohlo dojít. Nástroje, které měli k dispozici ovšem vnesení nekonzistence umožňovaly. Transakce nahradily metodologii *technologií* – programátorem použité nástroje pro manipulaci s daty (transakce) zavedení nekonzistence neumožňují.

Tradiční databázové systémy zaručovaly následující čtyři vlastnosti transakcí.

Vlastnosti ACID

- Atomicita (Atomicity) – operace je buď provedena celá, nebo vůbec. Nemůže dojít k částečnému provedení – např. prostředky z účtu buď převedeny jsou, nebo nejsou, nemohou být jen odepsány, nebo jen připsány.
- Konzistence (Consistency) – po provedení operace zůstane databáze konzistentní. Předpokladem je, že konzistentní byla před započetím operace.
- Oddělenost (Isolation) – současné provádění více transakcí nezpůsobí nekonzistenci databáze. Z pohledu každé transakce je v databázi prováděna pouze tato jediná transakce. Dalším důsledkem je to, že transakce nedává navenek žádné průběžné výsledky své práce.
- Trvalost (Durability) – důsledky provedení transakce jsou v databázi trvalé a přežijí všechna případná následující selhání.

Správné provádění

Správnost provádění sady transakcí je založena na pojmu *serializovatelnosti.* Provádění sady transakcí ne *na pohled serializovatelné (view serializable)*, když existuje jejich sekvenční provedení, které uvede databázi do stejného stavu jako provedení současné. Problém serializovatelnosti na pohled je nespočetný, proto se používá jiné kritérium – *serializovatelnost vzhledem ke konfliktu (conflict serializability).*

Dvě databázové operace jsou v konfliktu, když a jen když:

- jsou vyvolány různými transakcemi
- alespoň jedna z nich je operace zápisu

Současný běh sady transakcí je serializovatelný vzhledem ke konfliktu, jestliže je *konfliktně ekvivalentní* sekvenčnímu provádění, tj. jestli všechny konfliktní operace mají stejné pořadí jako při sekvenčním provádění.

Protože všechny běhy serializovatelné vzhledem ke konfliktu jsou na pohled serializovatelné vzhledem, jsou považovány za správné.

Omezení tradičních transakčních modelů

Tradiční transakce jsou zaměřeny především na bankovní sektor – krátké operace s velkým důrazem na atomicitu a oddělenost. V dalších aplikacích, jako je například správa půjček či pojištění jsou operace složitější a trvají podstatně déle. Právě dlouhé trvání je v rozporu s vlastnostmi ACID (transakce zamkne databázi na dlouhou dobu) a proto je žádoucí vlastnosti atomicity a oddělenosti oželet.

Dalším omezením je nezávislost transakcí – v reálném systému jsou ovšem často operace provázané a proto se objevuje požadavek na spolupráci transakcí.

3.2 Rozšířené transakční modely

Ságy (Sagas)

Sága je sada relativně nezávislých transakcí (T_1 .. T_n), které samy o sobě mají vlastnosti ACID. Pro každou subtrasakci T_k ($1 <= k < n$) je definována *kompenzující subtrasakce* CT_k, která sémanticky „vrátí“ změny vytvořené transakcí T_k. Stav databáze po provedení subtransakcí T_k a CT_k bude stejný, jako by nebyly provedeny.

Pro subtransakci T_n „reverzní“ subtransakce neexistuje – pokud je potvrzena T_n je potvrzena celá sága.

Hlavním přínosem ság je omezení vlastnosti izolace na subtransakce. Proto mohou ságy používat částečné výsledky jiných ság. Je zřejmé, že ságy nevyužívají serializovatelnost jako kritérium správnosti.

Použití ság zvyšuje možný stupeň paralelismu. Zdroje blokované subtransakcí jsou uvolněny okamžitě po jejím ukončení – nečeká se na další komponenty ságy.

Vnořené transakce (Nested transactions)

Tradiční transakční model je *plochý*. V tomto modelu se transakce skládá ze subtrasakcí – výsledkem je hierarchie zvaná *transakční strom.* Větvení uzlů nazýváme tradičně rodič – dcera.

Subtransakce vnořené transakce mohou být potvrzovány i rušeny nezávisle, podle následujících omezení: dceřinná transakce musí začít po začátku rodičovské transakce, rodičovská transakce musí skončit pouze po skončení všech dceřinných transakcí. Jestli je

zrušena rodičovská transakce, musí být zrušeny všechny dceřinné. Jestliže je naopak zrušena dceřinná transakce, může rodičovská pokračovat různými způsoby:

- ignorovat chybu a pokračovat
- znovu spustit subtransakci, která selhala
- provést jinou subtransakci (*kontingenční subtransakci)*
- skončit selháním

Tradiční ACID vlastnosti tento model zachovává, vnořené transakce jsou odděleny jedna od druhé. Subtransakce jedné transakce naproti tomu nejsou trvalé – pouze po potvrzení kořenové transakce.

Hlavními výhodami tohoto modelu jsou:

- zvýšená modularita
- lepší řízení chyb
- vyšší stupeň paralelismu

Otevřené vnořené transakce (Open nested transactions)

Otevřené vnořené transakce nedbají na požadavek izolace subtransakcí a zveřejňují výsledek potvrzených subtransakcí. Pro zachování konzistence je požadováno, aby výsledky subtransakce mohly používat pouze takové, které s danou s. *komutují* (tj. jejich výsledek nezávisí na pořadí provádění). V konvenčních systémech jsou takovými operacemi pouze operace čtení, není to však podmínkou. Například zvyšování hodnoty čítače mohou být komutující operace.

Tento model používá *kompenzace* – sémantické „vrácení“ efektu operace. K zajištění konzistence s dalšími transakcemi je použito konceptu komutace – transakce (T) i její kompenzace (T‘) komutují s jinou t. (S) a databáze je nakonec nezávisle na pořadí provádění transakcí ve stavu (S).

Model otevřených vnořených transakcí přináší modularitu, zvýšení granularity správy chyb a vyšší úroveň intratransakčního paralelismu.

Multidatabázové transakce (Multidatabase transactions)

Tento model byl navržen pro prostředí, kde aplikace využívá vícero databází. Tento model rozšiřuje model *pružných transakcí* (flexible transactions).

Multidatabázová transakce (*globální* transakce) je souborem podtransakcích prováděných lokálními databázovými systémy. Kromě specifikace subtransakcí definuje návrhář prováděcí závislosti, tok dat a požadavky na atomicitu multidatabázových transakcí. Je možné rovněž definovat časové a plánovací závislosti (spouštění v určitý čas, po dokončení/selhání jiné subtransakce).

Komunikace mezi subtransakcemi je zajištěna vstupními a výstupními proměnnými.

Systém musí zajistit ukončení v *přijatelném konečném stavu*, při skončení v jiném, nepřijatelném stavu je narušena atomicita selhání globální transakce.

Polytransakce (Polytransactions)

Základem tohoto modelu jsou vzájemně závislá data (svázaná integritními omezeními) uložená v různých databázích. Polytransakce je metoda správy těchto dat a popisu pořadí vzájemně souvisejících operací změny dat.

Vazby mezi daty jsou zachyceny popisovači datových závislostí (Data Dependency Descriptors, D^3), soubor všech D^3 tvoří mezidatabázové schéma závislostí (Interdatabase Dependency Schema, IDS).

D^3 je pětice <S,U,P,C,A>, kde S je zdrojový objekt, U je upravovaný objekt a P je predikát, který nabývá hodnoty true, je-li splněna podmínka závislosti mezi objekty S a U. C je podmínka určující, kdy musí platit predikát P. Kritéria mohou být založená na specifikaci času (po 22 hodině) nebo stavu dat (méně než 12 kusů ve skladu). Akce A provádí takové změny, aby byla splněna podmínka P. Akce mohou být popsány jakýmkoli formalismem včetně ság, vnořených transakcí apod..

Model S-trasakcí (S-transaction model)

Sémantické transakce (S-transakce) byly původně vyvinuty pro podporu mezinárodního bankovnictví. Model využívá dynamicky generovaný strom transakcí.

S-transakce netrvá na vlastnosti izolace (sub)transakcí, dodržuje však některá omezení: Rollback rodičovské transakce způsobí rollback všech dceřinných transakcí. Efekt potvrzených dceřinných transakcí musí být reverzován provedením kompenzující transakce.

Projekt Carnot (The Carnot project)

Carnot je založen na závislosti mezi úlohami, pro modelování této závislosti je použita CTL logika. Carnot definuje dvě závislosti:

- Implikační – $e_1 \rightarrow e_2$ *:* jestli nastane událost e_1, pak nastane i událost e_2. Události mohou nastat v libovolném pořadí.
- Pořadová – $e_1 < e_2$: jestli nastanou obě události, e_1 nastane před e_2.

Plánovač vynucuje každou závislost odmítnutím, zpožděním či vynucením událostí. Plánovací strategie mohou rovněž zahrnovat podmínky typu hodin reálného času. Závislosti mohou vznikat a zanikat za běhu workflow.

3.3 Metamodely pro rozšířené transakce

Tato sekce popisuje dva rámce pro vývoj modelů aplikací.

ACTA

Transakční meta-model (A Transaction Meta-model, ACTA) je rámcem charakterizujícím celé spektrum interakcí mezi transakcemi, stejně jako efekt transakcí na objekty.

V ACTA může transakce skončit dvěma způsoby – potvrzením a zrušením. Transakce může na jiných transakcích záviset dvěma způsoby:

- závislost na potvrzení (závislá t. může potvrdit až po skončení t. řídící)
- závislost na selhání – zruší-li běh řídící t., musí zrušit i t. závislá.

Vliv transakce na objekty je dán dvěma množinami – množinou potenciálně použitých objektů (ViewSet) a množinou skutečně použitých objektů (AccessSet)

Transakce může delegovat zodpovědnost za dokončení svého účinku na objekt jiné transakci. To je využíváno při předání částečných výsledků či koordinaci činnosti.

TSME

Prostředí pro správu a specifikaci transakcí (A Transaction Specification and Management Environment, TSME) je nástroj pro specifikaci a provádění různých transakčních modelů. Poskytuje implementačně nezávislý jazyk pro specifikaci transakcí jakož i prováděcí prostředí.

Je založeno na DOMS (systém pro správu distribuovaných objektů).

Důležitou charakteristikou je předpoklad ACID vlastností u lokálních systémů.

3.4 Vstříc transakčním workflow

Aby WF systém podporoval transakční workflow, musí splňovat tyto vlastnosti:

- prováděné úlohy mají ACID vlastnosti
- výsledky činnosti splněných úloh mohou být kompenzovány, tj. sémanticky vráceny vykonáním kompenzující úlohy
- workflow může selhat kvůli selhání jedné komponenty. Může ovšem být vykonána jiná úloha, sémanticky ekvivalentní a WF může korektně pokračovat.

Použití „návratu“ označujeme jako zpětné zotavení, provedení ekvivalentní akce jako dopředné zotavení.

Specifikace úlohy

Při modelování úlohy nemusíme brát v úvahu všechny její aspekty. Pro workflow je důležitá tato struktura úlohy:

- množina (z vnějšku) viditelných výpočetních stavů úlohy
- množina (povolených) přechodů mezi těmito stavy a
- podmínky umožňující tyto přechody

Abstraktní model úlohy je stavový stroj (automat) –různým modelům úloh (bez potvrzování, dvojstavové potvrzování, aj.) budou odpovídat různé automaty.

Požadavky na koordinaci úloh

Struktura toku řízení může být dvojího typu. Staticky specifikovaná v závislosti na:

- výpočetním stavu
- výstupních hodnotách
- vnějších proměnných

nebo dynamicky generovaná.

Požadavky na atomicitu selhání

Tradiční přístup vyžaduje, aby po selhání jednotlivé úlohy selhalo celé workflow. To není žádoucí, proto zavádíme povolené koncové stavy, jejichž dosažením WF neporušuje podmínku atomicity selhání. Ostatní stavy jsou nepovolené.

Povolené koncové stavy můžeme rozdělit na potvrzené (WF splnil svou úlohu) a zrušené (WF selhal). Vliv selhání na celkový výsledek WF ovlivňuje použití zpětného zotavení (kompenzace) a dopředného zotavení (vyvolání ekvivalentní úlohy)

Požadavky na atomicitu provádění

Zatímco u transakcí požadujeme aby byla provedena najednou jako celek, u WF můžeme připustit i provádění po částech. Ovšem ani při takovém provádění zpravidla nelze připustit pouze částečné provedení (tj. nikoli všech částí workflow – nedokončený převod prostředků apod.)

4) Stav trhu

4.1 Charakteristika současných produktů

V posledních několika letech bylo uvedeno mnoho workflow nástrojů a systémů pro správu workflow. Odhaduje se, že na trhu je cca. 250 produktů s obratem přes 1 mld USD.

WF produkty můžeme v zásadě rozdělit do tří skupin:

- Ad-hoc – existují pouze krátký čas a jsou určeny k provedení jednoho, zpravidla neopakovaného procesu.
- Administrativní – skládají se z opakovaných a předvídatelných procesů. Odpovídají zaběhnutým procesům jako je např. zpracování objednávky. Tok řízení je znám předem, za běhu jsou úlohy přidělovány jednotlivým výkonným entitám.
- Produkční – obvykle zahrnují více informačních systémů, zpravidla heterogenních a autonomních. Častá je komunikace pomocí API a změny toku řízení za běhu

4.2 Standardy WfMC

V pohledu WfMC se workflow systém skládá ze služby zajišťující WF, která prostřednictvím rozhraní komunikuje s:

- nástroji pro definici procesů
- výkonnými jednotkami WF
- WF klientskými aplikacemi
- Zahrnutými aplikacemi a
- Administrativními a monitorovacími nástroji

Tato rozhraní jsou předmětem standardu WAPI (Workflow API) vydaného koalicí

4.3 Problémy a omezení současných produktů

Shrňme v několika bodech:

- Součinnost – škála činností WF je velmi široká, nalézt proto „společnou řeč“ a používat existující standardy je proto téměř nezbytné. Bohužel praxe je v tomto ohledu spíše opačná
- Výkon a škálovatelnost – jedna výkonná jednotka zpravidla zpracovává více workflow současně. Období špiček je proto pro většinu systémů kritické, zvlášť když

je používána nepřiměřená architektura a přetěžovány vzácné zdroje (roury na UNIXových platformách)

- Transakční vlastnosti
- Bezpečnost
- Absence standardů
- Další – téměř naprosté používání centralizovaného modelu, malá integrace správy změn, nízká integrace s Groupware systémy aj.

5) Literatura

[1] Cichocki, A., et. al.: Workflow and proces automation: Concepts and Technology, Kluwer academic Publisher, Dordrecht, 1998, 117 s.