MŮŽE BÝT MODEL APLIKACÍ?

Vlasta Svatá

Vysoká škola ekonomická v Praze
Katedra systémové analýzy
svata@vse.cz

Principy, na kterých je postavena myšlenka servisně orientované architektury, vyvolaly vlnu nových diskusí týkajících se toho, jak SOA společně s nástroji BPM ovlivní tvorbu aplikací. Podíváme-li se krátce do historie, diskuse, týkající se tvorby aplikací, na poměrně dlouhou dobu ustaly poté, co se přibližně v polovině 90. let rozšířily principy objektově orientovaného návrhu a programování aplikací.

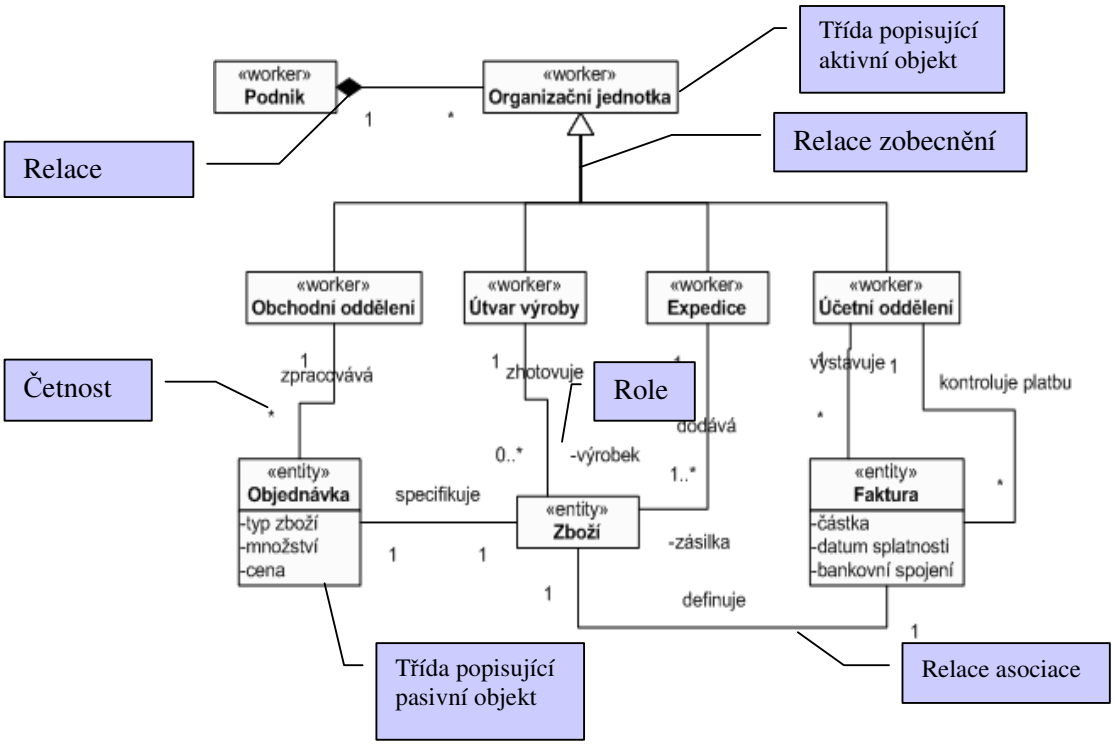
OBJEKTOVĚ ORIENTOVANÉ MODELY

Objektově orientované modely aplikací využívají pro návrh aplikačních programů tzv. objektů a jejich interakcí. Předchozí tradiční modely aplikací představovaly nestrukturovaný „shluk“ úloh zpracovávaných na počítači, které byly do velké míry ovlivněny individuálními znalostmi jednotlivých projektantů a programátorů. Údržba takových programů byla s rostoucí komplexitou modelů neefektivní a často bylo jednodušší vytvořit model nový, který respektoval znalosti nového autora. Objektově orientované modely aplikací byly tak prvním pokusem jak „shluk“ úloh dekomponovat na dílčí diskrétní jednotky – objekty, které se mohou opakovaně využívat ve více různých aplikacích. S objekty byly velmi úzce spojeny akce (operátory). Například datové struktury jsou spojeny s určitými operátory, které jim jsou vlastní. V předchozí etapě modelů byla data a operace, které se s nimi prováděly, striktně odděleny. Rozvoj objektových modelů akceleroval vývoj v oblasti programovacích jazyků, kdy procedurální jazyky byly nahrazeny objektově orientovanými. Moderní objektově orientované metody návrhu aplikací zahrnují vzory návrhu (design patterns – standardní vzory, které mohou být součástí knihoven), pravidla využívání objektů (design by contract – podmínky pro využívání, přínosy, odpovědnost,..) a modelovací jazyky (např. UML).

Přesto, že využívání objektově orientovaných modelů aplikací znamenalo velký skok v kvalitě a efektivnosti tvorby aplikací, nemohly tyto modely překlenout mezeru mezi business procesy a jejich automatizovanou podporou – aplikacemi. Zůstávaly na úrovni aplikací a v rámci jednotlivých aplikací umožňovaly efektivní opakované využívání jednotlivých komponent. Jejich využívání se omezuje na IT specialisty a uživatel tak nemá možnost zasahovat do logiky aplikací, která je předem daná. Pokud tedy dojde ke změně business procesu, je potřeba změnit model procesu například formou BPR a následně promítnout změnu do modelu aplikace. Oba modely (procesu a aplikace) jsou tedy nezávislé a každá změna (jak v modelu aplikace, tak v modelu procesu) musí být zajištěna lidmi, což přináší řadu problémů (jde o diskrétní proces s časovým zpožděním, které závisí na znalostech a dalších aspektech změny (ekonomické, právní, technologické, organizační, apod). Tento postup tvorby aplikací se někdy také nazývá DDA – design driven architecture).

Typickými představiteli objektově orientovaných modelů podnikových aplikací jsou klasické monolitické ERP systémy, které využívají objektově orientovaných modelů vytvářených pro podporu určitých vzorových procesních modelů. Pokud se tyto ERP systémy implementují v podniku, který má odlišné procesy, mohou nastat tři varianty:

- Modely procesů daného podniku se musí v rámci BPR přizpůsobit předpokládaným modelům procesů zabudovaným v modelu aplikace, toto řešení není uživatelský příjemné,
- Modely aplikace (ERP systému) se musejí přizpůsobit procesům daného podniku; v tomto případě jde o tzv. customizaci daného produktu, která ale vede v nejlepším případě k nárůstu nákladů při každém upgradu dané aplikace, v nejhorším případě, kdy se v rámci customizace zasahuje až do obsahu jednotlivých objektů, může dojít k desintegraci celého systému.
- Oba modely (modely procesů daného zákazníka a modely aplikace se musejí vyváženě upravit v rámci nástrojů dostupných v dané aplikaci (správná volba částí – modulů ERP systému – např. u SAP nástroj SAP Composer, správné nastavení volitelných částí programů apod.)



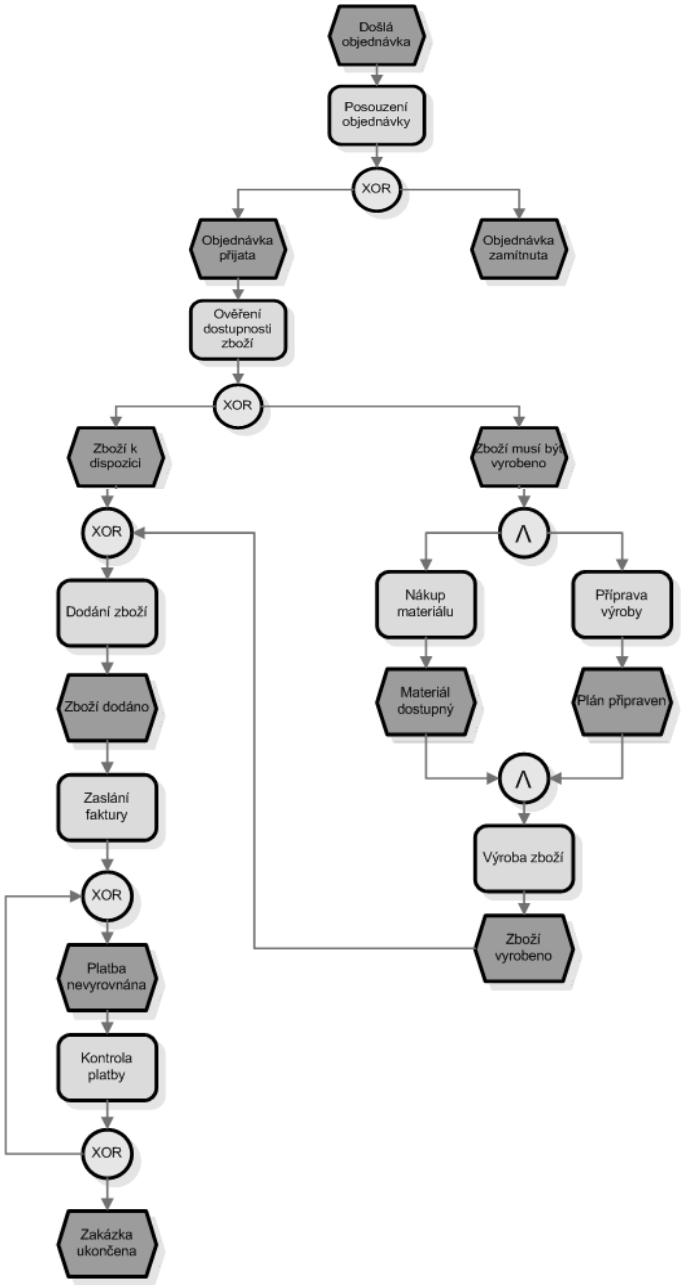
Obrázek 3: Objektový model realizace zakázky (zdroj: [1])

Modely, které jsou zabudované v aplikacích, se v odborné literatuře označují jako plánované nebo dávkové („scheduled“ nebo „batch“). Tyto modely jsou vhodné pro aplikace, kdy se dá předvídat časování prováděných úloh a nepožaduje se rychlá odezva na nové události. Například hlavní plánovač výroby se každé ráno podívá na plánované výrobní objednávky a uvolní je do výroby. Plánování výrobních zdrojů (MRPII) se obvykle dávkově zpracovává přes noc.

SERVISNĚ ORIENTOVANÉ MODELY

S vývojem servisně orientované architektury (SOA) informačních systémů společně s nástroji pro řízení business procesů (BPM) se otevřely kvalitativně nové možnosti a modely tvorby aplikací. Aplikace vznikající v prostředí SOA a jejích nástrojů již nejsou universální

softwarové balíky s inherentní anonymně zabudovanou logikou, ale jde o tzv. kompozitní (komponentové) aplikace, které se „on-line“ vytvářejí na základě modelů business procesů. V tomto případě hovoříme o architektuře aplikací jako o MDD – Model driven design. Vzhledem k tomu, že business procesy se mění na základě událostí, ke kterým dochází v reálném světě a které generují například zákazníci, dodavatelé, konkurenti a další entity uvnitř i v okolí podniků, takto vznikající aplikace se pružně skládají z komponent, které reagují na změněné události. Business proces se tak skládá z událostí, které spouštějí určité funkce, podporované komponentami (business službami) různých aplikací. Hovoříme proto o modelech řízených událostmi (event driven model).

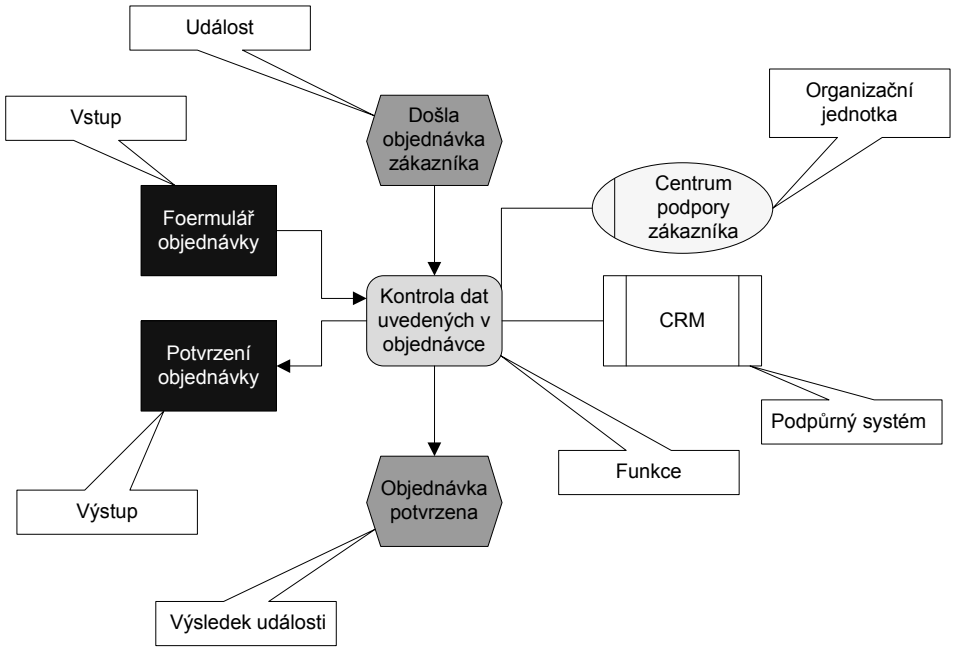


Obrázek 4: EPC model procesu realizace zakázky (zdroj [1])

Business procesy popisované pomocí EPC (Event Process Chain) modelů (diagramů) využívají následující prvky:

- Aktivity (activities) – jsou základními stavebními bloky a určují, co má být v danou chvíli v rámci procesu vykonáno.
- Události (events) – popisují situace před a/nebo po vykonání aktivity. Tvoří propojovací body mezi jednotlivými aktivitami, tj. výstupní událost jedné aktivity může tvořit vstupní událost jiné aktivity.
- Logické spojky (connectors) – slouží ke spojování událostí a aktivit. Tento prvek je nezbytný pro pro popis řídícího tou procesu, kdy jej mohou rozdělovat nebo naopak slučovat.

EPC modely procesů se skládají z prvků procesů, které mají následující doporučenou obecnou strukturu.



Obrázek 5: Prvek procesu v EPC diagramu

Tyto servisně orientované modely přitom nejsou v rozporu s objektově orientovanými modely. Ve skutečnosti objektově orientovaný přístup se často používá při návrhu a implementaci aplikační logiky uvnitř komponent/služeb aplikací. Podle lit. [5] následující body umožňují porovnání obou typů modelů:

- Servisně orientované modely jsou založené na volném provázání jednotlivých entit (služeb). Naopak objektově orientované modely kladou důraz spíše na přesně definované vztahy mezi třídami, což vede k mnohem těsnějším vazbám mezi entitami (objekty).
- Oba modely mají podobný pohled na abstrakci základních entit vytvořenou pro komunikaci mezi těmito entitami. Abstrakci objektově orientovaných modelech vytvářejí přesně definovaná rozhraní objektů. V servisně orientovaných modelech

tuto úlohu většinou plní nějaký druh popisného dokumentu (u webových služeb to je popis služby vytvořený v WSDL)

- Servisně orientované modely předpokládají, že rozsah činností, které daná služba nabízí, se může měnit. Objektově orientované modely definují objekty, které jsou mnohem více specifické v daném oboru působnosti.
- Aktivita služeb je vyvolána až příchodem nějaké zprávy. Zprávy obsahují kromě dat i logiku zpracování a určitým způsobem řídí činnost služeb. V objektově orientovaných modelech jsou data svázána s logikou do objektů. Jednou ze základních vlastností služeb je jejich bezstavovost, naopak zapouzdření logiky a dat do objektů v objektově orientovaných modelech způsobuje jejich stavovou závislost.

V rámci SOA event driven modely zahrnují nejen shora popsaný princip modelování business procesů, ale vzhledem k tomu, že aktivity vyvolávané různými událostmi mohou být zpracovávány i různými aplikacemi řeší formou modelů i způsob, jakým se jednotlivé události budou oznamovat a zpracovávat v rámci procesů. Doposud se rozlišují dvě koncepce – modely:

1. Model EDA (Event-Driven Architecture) je styl aplikační architektury, jehož základem je asynchronní proaktivní (push-based) komunikační model. Takové aplikace umožňují podporu procesů, jejichž cílem je dodávka zboží, služeb nebo informací s minimálním zpožděním. Jejich modifikace jsou rovněž snadnější v porovnání s tradičními architekturami aplikací založených na požadavcích uživatelů.
2. Model CEP (Complex-event Processing). Schopnost reagovat rychle a efektivně na měnící se podmínky je velmi důležitou předností pro podnikání. Model CEP je velmi sofistikovaná forma EDA, která umožňuje vybrat informační hodnotu z různých událostí. CEP systémy najdou vzory v datech o událostech s cílem určit příležitosti a hrozby. Včasné upozornění na takové příležitosti a hrozby se automaticky odesílají vhodným příjemcům často za pomoci Business Activity Monitoring (BAM) systémů. nebo jiných uživatelských informačních kanálů. Výsledkem jsou rychlejší a lepší operativní rozhodování a včasnější odpovědi na důležité situace.

REAKTIVNÍ A PROAKTIVNÍ MODELY

Podle toho, jak se události a následné aktivity/funkce vyvolávají v rámci procesu, je možné u servisně orientovaných modelů (modelů řízených událostmi) rozlišovat mezi tzv. reaktivními modely („request driven“ nebo „pull-based“ modely) a mezi proaktivními modely („push based“ modely). Tento dvojí typ modelů současně pomáhá rozlišovat kvalitu nástrojů pro budování SOA nabízených různými dodavateli.

Reaktivní modely jsou vhodné pro okamžité uspokojení měnících se potřeb uživatelů. Například zákazník banky má potřebu podívat se na stav svého účtu přes internet. Potřebné aktivity se automaticky vyvolají po zadání tohoto požadavku. Většina organizací již převedla svoje procesy a jejich aktivity směřující vně organizace na tento druh modelu komunikace. Jsou tak mnohem pružnější vzhledem k událostem probíhajícím v reálném světě.

On-line pull based model aplikací je zdánlivě řízen událostmi, ale nevyužívá tzv. „event driven“ architekturu aplikací (EDA). Nemůže tudíž uspokojit potřeby a události ve větších business procesech nebo není vhodný pro situace, ve kterých uživatel aplikace (konzument informace) neví kdy a jak požádat o aktuální informaci. Například banka provozující internetové bankovnictví nemůže očekávat, že se vlastník účtu bude každou hodinu dotazovat na stav svého účtu, aby se ujistil, že nedošlo k případným podezřelým transakcím. Zákazník

bude spíše očekávat, že ho banka sama upozorní v případě, že získá informace o nějakých podezřelých pohybech na jeho účtu. Existují tedy úlohy, u kterých je vhodnější model proaktivní („push based“).

Proaktivní modely pro tvorbu aplikací jsou vhodné pro dva rozdílné, avšak související aspekty business procesů.

První z nich byl naznačen v předchozím příkladu. Od aplikací se očekává, aby předvídaly potřeby uživatelů, proaktivně zpracovávaly služby a zasílaly výsledné informace uživatelům. Logika aplikací není tedy pouze reaktivní, ale řídí se událostmi a je schopná se jim přizpůsobit.

Druhá oblast, pro kterou jsou důležité proaktivní modely, se týká vícekrokových business procesů. Pull based model poskytuje zpracování v reálném čase, avšak zpracovává pouze určité úlohy prováděné jednou komponentou a komponentami, na které deleguje podúlohy. Například počáteční krok v procesu objednávky zboží za hotové peníze se realizuje na základě pull based modelu, ale zbývající kroky týkající se vyplnění objednávky, výroby zboží, dodávky a platby se nemohou provádět na základě tohoto modelu, protože jsou v praxi podporovány různými aplikacemi v různých útvarech organizace (možná i v různých společnostech nebo zemích v rámci jedné virtuální organizace).

Komplexní business procesy se nerealizují v jednom kroku, ale jsou dekomponovány do mnoha propojených etap, z nichž je každá prováděna jinou komponentou – například různými lidmi nebo různými IT službami. Pokud je komunikace mezi komponentami podporována kombinací pull based a scheduled based modely, výsledná realizace procesu bude pomalá. Procesy vycházející z modelu push based v kombinaci s EDA architekturou se skládají ze sekvence funkcí, z nichž je každá prováděna samostatnou softwarovou komponentou a každá je spuštěna signálem, který vydává předcházející komponenta v procesu (tento princip se také proto někdy označuje jako „message driven“ model zpracování).

Většina business procesů je vhodná pro kombinovanou aplikaci pull, puch a scheduled modelů. Jen málo z nich je vhodných pro čistě „puch based“ modelování. Na druhé straně je ale zcela zřejmý tlak na poskytování rychlejších a kvalitnějších služeb zákazníkům, což pomalu vede ke stále se více rozšiřujícím modelům podporujícím architekturu EDA.

ARCHITEKTURA EDA

EDA není možné interpretovat pouze jako „push based“ model. Při implementaci EDA je nutné splňovat určité charakteristiky popsané v této kapitole.

Nejdříve se podívejme na definici. Jak již bylo uvedeno v předešlém textu, událost je skutečnost, že se něco stalo (např. bankovní transakce, objednávka zákazníka, koupě domu. Počítače ze své podstaty nemohou zpracovávat události, protože jsou abstraktní. Proto musí událost vytvořit objekt události – elektronický signál nebo zpráva o události. Skutečnost, že pan Novák vyzvednul 100 EUR ze svého bankovního účtu dnes v 10 hodin dopoledne je událost. Počítačový záznam o této transakci (např. ve formě XML zprávy), je v tomto případě objektem události. Počítače jsou schopné zpracovávat objekty událostí, nikoliv události samé. Zpráva obsahující objekt události je oznámení (notification)[60].

Zpracování událostí se definuje jako provádění operací s objekty událostí. Zahrnuje tvorbu, čtení, vyjmutí, transformaci a odpověď na objekty událostí. Systém pro zpracování událostí se

[60] V některých odborných textech se toto nerozlišuje a pod pojmem událost ses myslí přímo oznámení o události.

skládá minimálně ze dvou komponent: snímač, který zaznamenává události a vysílá objekty událostí, a uživatel nebo přijímač, který obdrží objekty událostí a odpovídá na ně. Činnosti spojené se zaznamenáním události a vysláním objektu události jsou oddělené od činností spojených s přijetím objektu události a odpovědí na objekty událostí. EDA aplikace se skládá z komponent, z nichž každá může jak zpracovávat objekty událostí, tak je i vysílat, přičemž objekt události generovaný jednou komponentou je přijat jinými komponentami.

EDA je styl softwarové architektury, který vychází z proaktivního zpracování objektů událostí. Zpracování událostí ale neznamená automaticky EDA. Objekty událostí se mohou zpracovat na základě modelu reaktivního a plánovaného nebo i proaktivního. Objekty událostí mohou být uloženy ve skladištích informací pro pozdější data mining, nebo objekty událostí se mohou být transformovány do balíčků tzv. remote procedure calls (RPC)[61].

Aplikace fungující na základě modelu EDA musí splňovat tři následující podmínky:

1. *Objekty událostí jsou zpracovány na základě proaktivního modelu (push based)*

   objekty událostí se posílají od zdroje události k jejich uživateli v asynchronních zprávách v čase určeném zdrojem události. Toto proaktivní posílání objektů událostí redukuje zpoždění uživatele v reakci na událost.

2. *Komponenty zpracovávají události po příchodu*

   uživatel události odpovídá ihned po obdržení objektu události. Celý proces může odpovídat včas pouze pokud každá jeho část realizuje zpracování ihned po události aniž by čekala na plánovaný čas nebo výzvu.

3. *Objekty událostí neurčují zpracování*

   Objekty událostí neurčují aktivity, které uživatel události musí provést v reakci na konkrétní událost. To snižuje logické propojení mezi zdroji a uživateli objektů událostí. Zdroj jednoduše odesílá zprávu o tom, že k události došlo. Logika toho, co se má provést, je součástí přijímače (nebo přijímačů). Proto vývojáři mají velkou flexibilitu ve změně nebo vytváření nových přijímačů aniž by museli měnit zdroje. Naopak mechanismus Remote Process Calls (RPC) a většina SOA modelů typu požadavek /odpověď služeb zahrnují jméno metody, na jejímž základě ten, kdo požaduje danou službu a poskytovatel služby musí komunikovat. To znamená, že jak žadatel služby, tak její poskytovatel se musí měnit současně (jsou vzájemně propojeni). EDA tak představuje novou schopnost nezávislého zapojování přijímačů i poskytovatelů, což je současně největší výhoda proti modelu požadavek/odpověď.

Existuje celá řada druhů aplikací fungujících na základě modelu událostí. Jeden krok v jednoduchém EDA procesu zpracovává business funkci pokaždé, když obdrží jeden objekt události. Každý objekt události se zpracovává nezávisle na jiném objektu události. Naproti tomu tzv. CEP (Complex Event Processing) aplikace zpracovávají ve stejném čase více objektů událostí.

ZÁVĚR

Pokud bychom se měli vrátit k dotazu uvedeném v titulu tohoto příspěvku, a sice zda: „ Může být model aplikací?“, musíme odpovědět, že s určitým zjednodušením ano. Toto zjednodušení spočívá v tom, že předpokládáme existenci artefaktů aplikací (služeb), které jsou a byly

[61] Remote procedure call (RPC, vzdálené volání procedur) je technologií dovolující programu vykonat proceduru, která může být uložena na jiném místě než je umístěn sám volající program. Příkladem může být výpočet funkce na jiném počítači v síti

vytvářeny na principech objektových modelů a mají v sobě zabudované algoritmy. Tyto služby z různých aplikací je možné skládat pružně do nových celků podle měnících se potřeb business procesů. Pokud tyto celky nazveme aplikacemi, pak může být model procesu současně aplikací. Ve skutečnosti tímto celkem není aplikace tak, jak jsme ji doposud vnímali, ale jde o model komunikace mezi různými aplikacemi. Model procesu pak určuje model komunikace mezi aplikacemi. Tyto modely mohou být v podstatě dvojího druhu: reaktivní a proaktivní.

Modely EDA jsou v praxi stále poměrně málo využívané. Většina programátorů a návrhářů systémů při tvorbě aplikací stále vychází ze zabudovaných modelů (scheduled) nebo reaktivních modelů. Důvodem je velká setrvačnost v myšlení lidí a rovněž zatím malá dostupnost tzv. middlewarových produktů, které umožňují manipulaci se zprávami o událostech (např,. protokoly SOAP[62] nebo MOM[63]).

LITERATURA

[1] FARANA Radim, Informační systémy 2006/07, sylaby k přednáškám http://www.352.vsb.cz/uc_texty/InformacniSystemySyl/InformacniSystemy03_soubory/frame.htm

[2] CHANDY mani K. Ramo Simon, What is Event Driven Architecture (EDA) and Why Does it Matter?, July, 2007

[3] RADECKÝ, Vondrák, Štolfa, Modelování byznys procesů v praxi, FEI, VŠB-TU Ostrava, 2007

[4] SEELEY Rich, Microsoft´s SOA vision: „The model is the application, 2007

[5] WEIS, Rychlý, Architektura orientovaná na služby, návrh orientovaný na služby, webové služby, VUT Brno, 2007

[62] **SOAP** (**Simple Object Access Protocol**) je protokolem pro výměnu zpráv založených na XML přes síť, hlavně pomocí HTTP . Formát SOAP tvoří základní vrstvu komunikace mezi webovými službami a poskytuje prostředí pro tvorbu složitější komunikace.

[63] **MOM (Message Oriented Middleware)** je software, který umožňuje komunikaci mezi softwarovými komponentami nebo softwarovými aplikacemi založený na asynchronní výměně zpráv (na rozdíl od výměny zpráv na principu dotaz–odpověď).