VYUŽITELNOST METODIKY XP PŘI VÝVOJI APLIKACÍ WEBOVÝCH SLUŽEB

Tomáš Brabec
Alena Buchalcevová
Katedra informačních technologií, Fakulta informatiky a statistiky, VŠE Praha

ABSTRAKT
V posledních deseti letech došlo k výraznému zrychlení procesu tvorby nových verzí softwarových produktů. Potřeba pružné a hlavně rychlé reakce na změny zapříčinila vznik nových technologií, architektur i metodik. Jedním z nejvýznamnějších nových konceptů v oblasti IS/ICT jsou služby. Prostředí pro provozování služeb umožňuje definovat architektura orientovaná na služby (Service Oriented Architecture, SOA), webové služby pak představují jednu z technologií pro její realizaci. Omezenost, resp. strnulost stávajících – rigorózních – metodik ztěžuje během implementace služeb přizpůsobování se aktuálním požadavkům. Tento problém řeší nová skupina metodik – agilní metodiky, jejichž zástupcem je mimo jiné Extrémní programování (Extreme Programming, XP). Tento příspěvek se proto snaží posoudit použitelnost XP pro vývoj aplikací na bázi webových služeb.

KLÍČOVÁ SLOVA
Architektura orientovaná na služby, extrémní programování, životní cyklus webových služeb.

1. Úvod

V průběhu uplynulých deseti let došlo k výrazným změnám v přístupu ke tvorbě (nejen) obchodních aplikací. Příčin je mnoho, nejvýznamnějšími jsou však *neustálé změny prostředí* (oblast a cíle podnikání, legislativa, konkurence, zvyklosti, preference a nároky uživatelů, …) a nutnost na ně neprodleně reagovat; snaha maximálně *využívat* již hotové systémy, aplikace a komponenty, stejně tak jako potřeba rychle a snadno *integrovat* nové zákazníky i dodavatele jak do vlastních obchodních procesů, tak do informačních systémů tyto procesy podporujících.

Důsledkem potřeby integrace externích, ale i interních, subjektů do vlastního podnikání je ústup od distribuovaných architektur s těsnou vazbou, jako např. CORBA nebo DCOM, stále větší roli hrají nové otevřené standardy a technologie.

Důsledkem snahy o maximální využití již existujících produktů spolu s požadavky na integraci je prosazení architektury orientované na služby, která využívá volnou vazbu mezi svými jednotlivými prvky. Mnohé aplikace a systémy začaly být s nástupem SOA provozovány formou služeb či alespoň začaly služby využívat. Služby se stávají základními stavebními bloky těchto aplikací a systémů. Webové služby pak představují ideální technologii pro implementaci aplikací na bázi SOA.

Důsledkem neustálých změn a potřeby na ně promptně reagovat je vznik agilních metodik. Dosud užívané metodiky stavějící na rigorózních, přesně dodržovaných postupech, se ukázaly nedostatečně pružné a nepřizpůsobivé novým podmínkám, agilní metodiky se snaží

komplexně řešit požadavky na rychlý a flexibilní vývoj software, průběžnou údržbu a reakci na měnící se podmínky a zadání.

2. Architektura orientovaná na služby

Architektura orientovaná na služby (SOA) definuje použití softwarových služeb spojených volnou vazbou k podpoře obchodních procesů a uživatelů. V této architektuře spolu dva různé programy interagují způsobem, kdy jeden z nich může ve prospěch druhého vykonat určitou jednotku aplikační, obchodní či systémové funkčnosti – vystupuje tedy jako služba. Podobným způsobem může druhý program vykonávat určitou funkčnost (fungovat jako služba) pro třetí program a být tak zároveň poskytovatelem i konzumentem služby.

Interakce mezi službami jsou definovány pomocí jazyka pro popis služeb (WSDL), jímž se popisuje veřejné rozhraní služby, protokol a způsob napojení a formát zpráv pro interakci se službou. Jakákoli interakce je formálně nezávislá na libovolné jiné interakci. SOA se také věnuje způsobu, jímž jsou jednotlivé služby popsány a organizovány, tak aby bylo podporováno automatické vyhledání (typicky v registrech UDDI) a použití vhodných služeb v reálném čase.

Aby bylo možné SOA úspěšně provozovat, musí být splněno několik podmínek [6]:
1) Všechny funkce (obchodní funkce, obchodní transakce složené z funkcí na nižších úrovních, systémové funkce) jsou definovány jako služby.
2) Všechny služby jsou nezávislé a navenek fungují jako „černé skřínky“. Externí komponenty nevědí a ani se nestarají o to, jak služby uvnitř fungují; stačí, když vracejí očekávané výsledky.
3) Rozhraní služeb jsou v nejobecnějším smyslu slova volatelné. Na úrovni architektury nezáleží na tom, zda jsou služby lokální či vzdálené nebo jaké propojovací schéma či protokol byly použity.

2.1. Životní cyklus SOA

SOA má svůj vlastní životní cyklus. High a kol. ve [5] rozlišují čtyři fáze: Model, Assembly, Deploy a Manage. Přes tyto fáze jsou rozloženy kontrolní a řídící procesy fáze Governance & Processes.

Modelování

Primárním cílem fáze Modelování je sestavit model obchodních aktivit a procesů – schéma podnikání (business design). Aktivity budou v architektuře realizovány jako služby. Dokumentace obchodní architektury slouží nejen k naplánování SOA, ale může posloužit též k optimalizaci stávajících obchodních procesů. Během modelování získáme odpověď na otázku jaký druh služeb budeme potřebovat a s jakými daty budou tyto služby pracovat. Model obchodních aktivit může kromě zaznamenání aktuální podoby podnikání sloužit i pro simulaci fungování obchodních procesů.

Sestavení

Ve fázi Sestavení jde o implementaci schématu podnikání. Cílem je poskládat dohromady v rámci implementace schématu podnikání jednotlivé artefakty identifikované v této fázi. Model schématu podnikání se transformuje do množiny definic obchodních procesů a jejich

aktivit. Z těchto definic se odvozují požadované služby. Obchodní procesy jsou poté v architektuře realizovány sestavením identifikovaných služeb – ať už existujících nebo nově implementovaných – do aplikací. Významnou činností této fáze je provedení inventury stávajících systémů a aplikací za účelem nalezení programových komponent, které již splňují (po případných úpravách) požadavky modelu.

Nasazení

Ve fázi Nasazení jsou jednotlivé položky, které dohromady vytvářejí podnikovou SOA, rozmístěny v rámci zabezpečeného a integrovaného prostředí, zajišťuje se dodržení požadavků na výkonnost, dostupnost, integritu a flexibilitu pro budoucí rozšíření.

Správa

Tato fáze se zabývá zejména řešením provozních aspektů aplikací, monitoringem a správou aplikací a jejich provozního prostředí. Každou provozovanou aplikaci je totiž potřebné spravovat a monitorovat její provoz jak z hlediska IT, tak podnikatelského. Získané informace poskytují zpětnou vazbu pro nepřetržitý proces zlepšování jak celkové architektury a jejích jednotlivých prvků, tak vlastního schématu podnikání a obchodních procesů.

Řízení a kontrola

Cílem procesů fáze Řízení a kontrola je prosadit, aby veškeré činnosti prováděné v průběhu životního cyklu byly v souladu se schématem podnikání, a zajistit, že veškeré změny neprobíhají živelně, ale naopak že je jejich průběh kontrolován a řízen příslušnými autoritami.

3. Životní cyklus webových služeb

V životním cyklu aplikací webových služeb můžeme najít stejné fáze jako u životních cyklů jiných typů aplikací. Máme fázi analytickou a návrhovou (Analýza a návrh); fázi realizační, kdy se aplikace implementuje (Realizace); fázi zaváděcí, kdy se aplikace instaluje a uvádí do provozu (Nasazení) a fázi provozu a údržby aplikace (Správa)[1]. Náplň těchto fází může však být velmi specifická, tvořená činnostmi specifickými právě pro aplikace webových služeb. (Aktivity a dodávky jednotlivých fází uvádí Tabulka 1.)

Obdobně jako u životního cyklu celé SOA (viz [5]) potřebujeme i v rámci životního cyklu jednotlivých aplikací fázi řídící a kontrolní (Dohled a řízení). V ní se stanovují mantinely a definují pravidla, směrnice a zásady použitelné v průběhu celého životního cyklu, provádějí se činnosti kontrolní a řídící, rozhoduje se o dalším postupu, řeší se mimořádné situace a změny, jež v průběhu práce nastaly. Jde o úvodní fázi, její činnosti se ale provádějí průběžně po celou dobu trvání životního cyklu aplikace.

3.1. Typy aplikací

Z formálního hlediska rozlišujeme mezi aplikacemi webových služeb tři typy:

[1] Lze namítnout, že v uvedeném výčtu chybí fáze analýzy uživatelských požadavků. Protože webové služby obvykle slouží k realizaci služeb SOA, můžeme předpokládat, že schéma podnikání a model obchodních procesů mohou výstupy této fáze nahradit.

I. Kompozitní aplikace tvořené množinou navzájem spřízněných, integrovaných služeb, které podporují obchodní proces vystavěný na bázi SOA [5].
II. Elementární aplikace tvořené jednou samostatnou a nezávislou službou fungující coby přístupový bod, která k poskytování funkčnosti klientům nevyžaduje spolupráci s jinou službou [4].
III. Adaptéry, což jsou služby umožňující nejen v architektuře SOA využívat funkčnost starších či nekompatibilních systémů a aplikací tak, že jejich původní, pro klienty nevhodné, rozhraní adaptují na takové, se kterým již spolupracující klient umí komunikovat.

Samozřejmě platí, že kompozitní aplikace mohou být složeny také ze služeb, které v našem rozlišení představují jednoduché aplikace a adaptéry.

3.2. Fáze životního cyklu

Všechny tři uvedené typy aplikací se více či méně liší jak v činnostech, které je třeba v rámci jednotlivých fází životního cyklu aplikace vykonat, tak v obsahu (náplni) těchto činností. Tabulka 1 uvádí, jak se činnosti té které fáze u uvedených typů aplikací uplatní.

Tabulka 1 – činnosti a dodávky fází životního cyklu aplikace webových služeb.

Fáze	**Dodávky a činnosti**	**Typ aplikace[2]**		
		I	**II**	**III**
Dohled a řízení	Organizační aspekty - plánování a harmonogram dodávek; složení (role) týmu; definice rozhodovacích procesů a směrnic řešení mimořádných situací; stanovení zásad pro aplikaci změn.	x	x	x
	Řídící aspekty – řízení, dohled a usměrňování průběhu životního cyklu aplikace.	x	x	x
Analýza a návrh	Analýza požadavků nebo realizovaného procesu.	x	x	
	Analýza původního rozhraní připojovaného systému a zprostředkovávané funkčnosti.			x
	Identifikace a kategorizace (hierarchické uspořádání) služeb. [1]	x		
	Specifikace integračních potřeb a vzorů. [3]	x	o	o
	Detailní specifikace služeb a realizačních komponent.	x	x	x
	Návrh rozhraní, struktury a formátu zpráv.	x	x	x
	Model součinnosti služeb v realizovaném procesu (BPEL4WS), choreografie a koordinace služeb (WS-Choreography), transakce (WS-Transaction).	x		
	Model zabezpečení (WS-Security).	x	x	x
Realizace	Implementace nových služeb včetně popisu (WSDL) a sémantiky.	o	o	x
	Využití existujících služeb, popř. vyhledání vhodných služeb – nutné řešit kontrakt s poskytovatelem služby[3], podmínky – SLA, QoS.	o	o	

[2] Znak „x“ ve sloupci „Typ aplikace“ značí, že daná činnost se vykonává, „o“ že její vykonání může být přínosné.

[3] Nemusí se ovšem jednat o psanou smlouvu, za kontrakt lze chápat i souhlas podmínkami poskytovatele o používání služby

		Typ aplikace[2]		
	Přiřazení služeb (a komponent) do celkové architektury SOA podniku (pokud se tak již nestalo).	o	o	o
	Funkční testování komponent a služeb.	x	x	x
	Sestavení aplikace.	x		
	Integrační testování.	x	o	o
Nasazení	Rozmístění aplikace do provozního prostředí, migrace.	x	x	x
	Testování dostupnosti a výkonnosti.	x	x	x
	Zveřejnění (publikování) služeb.	o	o	o
Správa	Monitoring provozu aplikace.	x	x	x
	Vedení záznamů o problémech.	x	x	x
	Detekce, lokalizace a řešení poruch.	x	x	x
	Nastavování provozního prostředí.	x	x	x
	Rutinní údržba.	x	x	x
	Vyhodnocování získaných informací a průběžné vylepšování procesu.	x		

4. Úroveň podpory životního cyklu aplikace webových služeb metodikou XP

SOA a webové služby vznikly především jako reakce na nestálé a proměnlivé obchodní prostředí v posledním desetiletí s cílem umožnit podnikům rychleji a adekvátněji reagovat na změny v podnikání a přitom co nejvíce využívat již existující software. Podobně se agilní metodiky snaží rychle a pružně reagovat na změny. Proto se budeme zabývat posouzením možností, které nabízí metodika Extrémní programování (XP), asi nejznámější zástupce agilních metodik v ČR.

Metodika XP je postavena (viz [2], [7]) na jednoduchosti (nejjednodušší ještě funkční dodávka), malých verzích, metafoře (nahrazuje architekturu), návrhu[4] a jeho každodenním zlepšování, průběžném testování, pravidelných revizích a kontrolách zdrojového textu, párovém programování, společném vlastnictví, nepřetržité integraci, přítomnosti zákazníka na pracovišti, standardním pracovním týdnu a standardech pro psaní zdrojového kódu.

Základními činnostmi jsou: Testování, Psaní zdrojového kódu, Poslouchání a Navrhování. S přihlédnutím k uvedeným vlastnostem a faktu, že XP zastává přístup „Test-Driven Development“, můžeme s trochou nadsázky říct, že každá iterace návrhu a implementace se dělá jen do té míry, abychom byli schopni zprovoznit daný test.

Při posuzování, do jaké úrovně je metodika XP vhodná pro vývoj a provozování aplikací webových služeb, vyjdeme z fází životního cyklu aplikací webových služeb (část 3.2). U jednotlivých činností zhodnotíme, jak metodika XP danou činnost bez dalších úprav řeší či podporuje.

[4] XP neuvažuje oddělené kroky analýzy a návrhu, oboje se průběžně řeší v rámci kroku implementace.

Tabulka 2 – podpora životního cyklu metodikou XP

Činnost	Podpora	Poznámka
Organizační aspekty	Dílčí	Plány verzí[5]; složení týmu; standardy pro psaní kódu.
Řídící aspekty	Dílčí	Společné vlastnictví; fáze Řízení plánovací hry; zákazník na pracovišti; společné hodnocení a rozhodování o změnách
Analýza požadavků či realizovaného procesu	Ano	Průzkum a Závazek plánovací hry; návrh výchozí architektury, její neustálé vylepšování.
Analýza původního systému	Ano	dtto.
Identifikace a kategorizace služeb	Dílčí	V XP tato činnost spadá pod návrh, ovšem není k dispozici žádná konkrétní metoda.
Specifikace integračních potřeb a vzorů	Ne	
Detailní specifikace služeb	Ano	V XP tato činnost spadá pod návrh.
Návrh rozhraní, struktury a formátu zpráv	Ano	dtto.
Model součinnosti služeb	Dílčí	Formálně by se tato činnost řešila v návrhu, částečně souvisí s integrací.
Model zabezpečení	Dílčí	Zabezpečení konkrétních služeb.
Implementace nových služeb včetně popisu a sémantiky	Ano	Vygenerování popisu služeb lze řešit vhodnými nástroji, sémantika je důležitá při případném zveřejnění služby.
Využití, popř. vyhledání, existujících služeb	Dílčí	Využití existujících služeb a jejich zařazení do aplikace (integrace)
Přiřazení služeb do architektury SOA podniku	Ne	Nutné řešit metodikou pro SOA.
Funkční testy	Ano	Testy se píší ještě před vlastní implementací.
Sestavení aplikace	Ano	XP řeší formou nepřetržité integrace.
Integrační testy	Ano	Testy se spouštějí vždy, když se přidává nový přírůstek
Rozmístění aplikace do provozního prostředí	Ano	Při dokončení verze a úspěšné integraci; migrace dat; testy provozní způsobilosti.
Testování dostupnosti a výkonnosti	Ano	Průběžné testování.
Zveřejnění služeb	Ne	
Monitoring provozu	Ano	
Vedení záznamů o problémech	Dílčí	Během implementace se řeší automatizace záznamů.
Detekce, lokalizace a řešení poruch	Ano	Provozování helpdesku apod.
Nastavování provozního prostředí	Ano	„Ladění“ provozního systému.
Rutinní údržba	Ano	
Vyhodnocování získaných informací a průběžné vylepšování procesu	Dílčí	Vztahuje se spíše na průběžné zlepšování vytvářeného software než na realizovaný obchodní proces.

[5] Do první verze se vyberou ta zadání, která nás „donutí“ vytvořit kostru celého systému.

Kromě skutečností, které uvádí Tabulka 2, závisí úspěšnost použití metodiky XP také na národní i firemní kultuře, povaze členů vývojového týmu apod. Obecně musíme také konstatovat (což ostatně činí i autor XP v [2]), že XP není vhodné pro situace, kdy získání zpětné vazby trvá velmi dlouho.

5. Závěr

Metodika Extrémní programování se jako zástupce agilních metodik primárně zaměřuje na co nejrychlejší, relativně malé a časté, dodávky software. Jejím jádrem jsou iterativní vytváření testů, implementace, předložení zákazníkům a využití zpětné vazby pro další iteraci.

Z pohledu vývoje a provozování aplikací webových služeb je metodika vhodná zejména pro dva ze tří typů aplikací popsaných v části 3.1 – adaptéry a jednoduché aplikace. U nich metodika pokrývá takřka celý životní cyklus a nečeká se od ní podpora činností specifických pro kompozitní aplikace webových služeb, jako jsou modely procesů, integrace a součinnost jednotlivých služeb v rámci procesu, přiřazení služeb do architektury SOA podniku či zveřejnění služby. Jinými slovy XP se hodí především pro projekty, u kterých není rozsah prací příliš velký a je možné rychle uvolňovat dílčí verze aplikace. V případě kompozitních aplikací je vhodné Extrémní programování použít jako doplněk komplexnější metodiky zabývající se přímo životním cyklem SOA.

6. Literatura

[1] Arsanjani, A.: *Service-oriented modeling and architecture*. IBM developerWorks, 2004. URL http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/.

[2] Beck, K.: *Extrémní programování*. Praha, Grada 2002. ISBN 80-247-0300-9.

[3] Benatallah, B., Casati, F., Nezhad, H. R. M., Toumani F.: *Developing Adapters for Web Services Integration*. Advanced Information Systems Engineering, 17th International Conference, CAiSE 2005, Porto, June 13-17, 2005. URL http://www.hpl.hp.com/personal/Fabio_Casati/docs/Caise05-adapters.pdf

[4] Benatallah, B., Dumas, M., Sheng, Q. Z.: *Facilitating the Rapid Development and Scalable Orchestration of Composite Web Services*. Distributed and Parallel Databases, 17(1):5-37, January 2005. URL http://www.springerlink.com/content/g35411nk83360242/fulltext.pdf

[5] High, R., Kinder, S., Graham, S.: *IBM's SOA Foundation: An Architectural Introduction and Overview*. IBM, listopad 2005. URL http://www-128.ibm.com/developerworks/webservices/library/ws-soa-whitepaper/

[6] Channabasavaiah, K., Holley, K. and Tuggle, E.: *Migrating to a service-oriented architecture*, Part 1. IBM developerWorks, 2003. URL http://www-128.ibm.com/developerworks/webservices/library/ws-migratesoa/

[7] Kadlec, V.: *Agilní programování*. Brno, Computer Press 2004. ISBN 80-251-0342-0.