ACTA UNIVERSITATIS AGRICULTURAE ET SILVICULTURAE MENDELIANAE BRUNENSIS
SBORNÍK MENDELOVY ZEMĚDĚLSKÉ A LESNICKÉ UNIVERZITY V BRNĚ

Ročník LII 19 Číslo 6, 2004

METODIKA PŘENOSU INFORMACÍ Z PODNIKOVÉ DO SOFTWAROVÉ ARCHITEKTURY

I. Rábová

Došlo: 16. června 2004

Abstract

RÁBOVÁ, I.: *The methodic of information transport from the business architecture to the software architecture*. Acta univ. agric. et silvic. Mendel. Brun., 2004, LII, No. 6, pp. 231-238

The business architecture creation is one of fundamental activities in business process reengineering and also in development of software product that supports the optimised business processes. One part of application development life cycle is modeling of software architecture (functions and data of application). There are two different models, model of business process and model of information system. This article deals with the value of using the same notation (such as UML) to vizualization of the both models and it guidelines the information transfer from business architecture to software architecture.

business architecture, software architecture, function, nonfunction and development characteristics of software architecture, system analysis and design, entity class, boundary class, component, use case, actor, business object, business class

Význam softwarového systému pro podnik dnes už není třeba zdůrazňovat. V současné tržní a konkurenční společnosti však roste jeho komplexnost a složitost. Postupně se stává klíčovým faktorem úspěchu a výrazně se podílí na přidané hodnotě produkce podniku. Softwarový produkt musí být dodán rychle a v kvalitě, která odpovídá nebo přesahuje požadavky zákazníka. Úspěšnost zavádění a využívání nových softwarových produktů je do značné míry ovlivněna hloubkou poznání a formalizací či vizualizací podnikových procesů. Většina firem hledá způsob, jak efektivně zlepšit své podnikové procesy, zvýšit svou kvalitu, snížit náklady a usnadnit si přístup na trh. Ale ani nejlepší metodiky nemohou zajistit, aby jakékoliv zaváděné či renovované technologie napravily nedostatky v podnikových procesech, které tvoří základní náplň činností organizace. Každá aktivita v oblasti inovace nebo nasazení nového aplikačního software musí být provedena na zdravě fungující a efektivně řízený proces. Modelování podnikových procesů a tvorba podnikové architektury může být správným odrazovým můstkem pro vývoj podpůrného software. Ve své disertační práci na téma Metamodel podniku ve vývoji informačních systémů [Rábová, 2002] a v následujících článcích se poměrně podrobně zabývám problematikou modelování podnikové architektury v jazyce UML (Unified Modeling Language). Některé diagramy této podnikové architektury lze s výhodou použít pro tvorbu požadavků na softwarový systém, který podporuje podnikové procesy. Problematiku přenosu informace z modelů podnikové architektury do softwarové architektury detailněji rozpracovávám v předloženém příspěvku.

MATERIÁL A METODIKA

Vlastnosti podnikové a softwarové architektury

Podniková architektura je základem pro popis a pochopení podniku, slouží jako báze znalostí a je strategickou výhodou podniku [Eriksson, Penker, 2000]. Obsahuje seznam částí podniku, jejich strukturu a jejich spolupráci, případně doporučení, jak by se měla vyvíjet v dalších etapách. Podniková architektura umožňuje abstrahovat podnik od různých aspektů nebo pohledů a koncentrovat se jen na jeden aspekt v čase. Dosažení abstrakce, potlačení detailů a irele-

vantních informací je základem pro pochopení komplexních systémů a vztahů v podniku.

Softwarová architektura je struktura systému, která zahrnuje softwarové komponenty a vnější viditelné vlastnosti těchto komponent a vztahy mezi nimi [Eriksson, Penker, 2000]. Softwarová architektura má funkční, nefunkční a vývojové vlastnosti. Stěžejní vlastnosti systému jsou funkční vlastnosti, tedy schopnost systému vykonávat funkce požadované svými uživateli a podporující podnikové procesy. Proto většina funkčních požadavků na software může být a je odvozena z podnikového modelu. Ty, které se nevyskytují přímo v podnikovém modelu, vyplynou často z uvedených funkcí jako jejich podpora nebo zdokonalení. Nefunkční vlastnosti jako výkonnost, bezpečnost a přístupnost nejsou přímo spojené se specifickou činností procesu, ale spíše se týkají softwarového systému jako celku a jejich podpora musí být zahrnuta do celkové softwarové architektury. V některých případech lze tyto nefunkční charakteristiky vyčíst také z podnikových modelů (popis atributů, pravidla připojená k prvkům atd.). Vlastnosti, které se týkají pouze vývoje systému a kvality vyvíjeného systému v termínech modifikovatelnosti, znovu použitelnosti, ceny a integrity, jsou životně důležité pro tvorbu softwarových architektonických rozhodnutí. Lze je však stěží odvodit ze základních diagramů modelu podniku.

Základní cíle pro hledání správného a flexibilního popisu systému, jako je tvorba modelu složeného z několika pohledů, modelování diagramů jednotlivých konceptů, požadavek na správnou volbu úrovní abstrakce, společné prezentace a použití vizuálních modelů UML, jsou pro obě architektury stejné. Domnívám se však, že rozdíly existují.

- První odlišnost vidím v úrovni detailu. Zatímco softwarová architektura bude rozšiřována do velmi detailních artefaktů, například do detailního programovacího kódu za účelem tvorby fungujícího systému, bude testována a iterativně vyvíjena a implementována, vývoj podnikové architektury je často zastaven na dané úrovni abstrakce a další detaily zůstanou nevypracovány (odpovídající úroveň je závislá na aktuálním smyslu modelu).
- Další odlišnost lze vypozorovat ve frekvenci změn. Zatímco nejnižší frekvence změn je na úrovni podnikových procesů, nejvyšší je na úrovni implementace software. Implementace je spojená s vývojem informačních technologií a často podléhá jejich rychlé inovaci a změnám. Úroveň modelování podnikových procesů je nositelem dlouhodobě platných koncepcí. Modely podnikových procesů tak mají nejdelší životnost a z důvodu jejich blízkosti k popisu ekonomických problémů dokumentují i význam informačních technologií.

Využití podnikové architektury pro definování správné softwarové architektury

Přenos informace uložené v modelu podnikové architektury do modelu softwarového systému není jednoduchý nebo automatický proces. Je pokládán za nejobtížnější část analýzy z několika důvodů. Pro tyto modely a diagramy neexistuje mapování jedna ku jedné a neexistuje ani jednoduchý algoritmus pro přeměnu podnikového modelu do softwarového. Jsou to dva odlišné modely s různým smyslem, často tvořené různými týmy odborníků. Podnikový model popisuje celý podnik nebo specifickou část podniku, a ne všechny části podniku se převádějí do softwarových systémů. Ani třídy a objekty v podnikovém modelu nemohou být přímo konvertovány do odpovídajících softwarových tříd a objektů. Pokud se provede konverze jedna ku jedné, vede to často k velmi nepřesné až zmatené softwarové architektuře, kde prvky, které nemohou být nebo neměly by být součástí softwarového systému, se stanou třídami a objekty v systému.

V zásadě mohou nastat dva případy.

- Pokud je vytvořen podnikový model s explicitním a exklusivním smyslem hledání požadavků na softwarový systém a s jeho zpracováním, je správné soustředit se spíše na takové podnikové aspekty, které souvisí se softwarem. Podnikový model může zůstat velmi abstraktní a jen ty procesy, které vyžadují podporu softwaru se dále detailizují. Tento případ je zde rozpracován.
- Pokud však smyslem podnikového modelování je reengineering procesů podniku, neboli cílem je tvorba významného dokumentu podniku jako součásti podnikové strategie nebo zásobníku podnikových znalostí, je potřeba se na podnikový model soustředit více a vypracovat jej detailněji a komplexněji ze všech úhlů pohledu.

VÝSLEDKY A DISKUSE

Jak už bylo uvedeno, použitím stejné notace (v našem případě je to UML, notace objektového přístupu k analýze), lze usnadnit mnohé kroky přechodu podnikového modelu do modelu softwarového. Využití podnikového modelu v softwarovém inženýrství shrnu do následujících pěti možností [Rábová, 2002] a nastíním dále postup při tvorbě specifikačního dokumentu obsahujícího funkční požadavky na softwarový systém.

Klasifikace možných využití

1. Tvorba systémové topologie

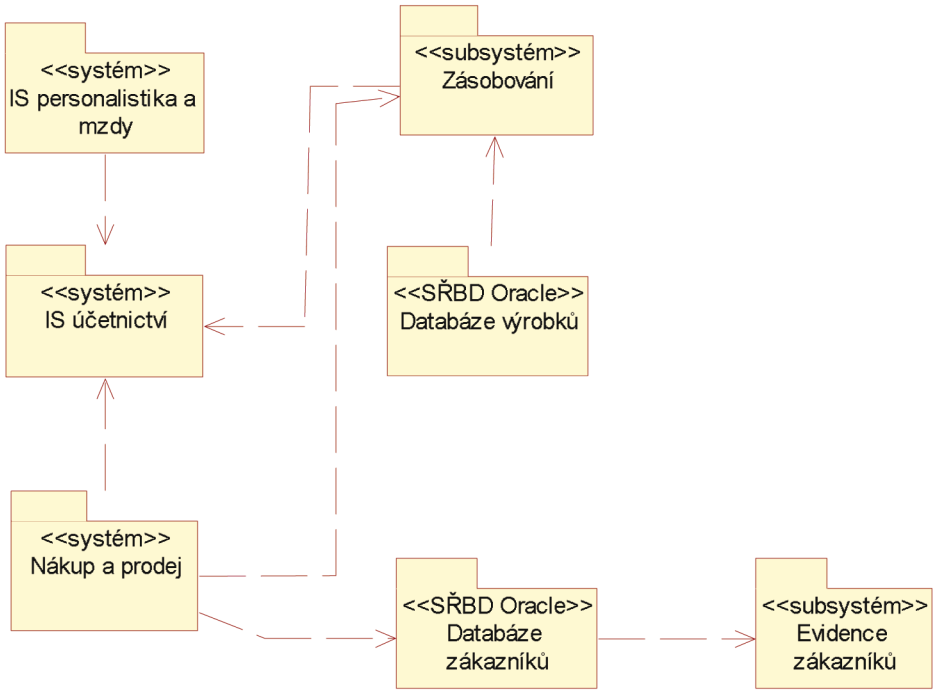
Jedním ze základních použití podnikového modelu je identifikace nejvhodnějších aplikačních systémů a subsystémů pro podnik. Znamená to odpovědět na otázku, které systémy jsou nezbytné a vhodné pro

podporu a fungování optimalizovaných podnikových procesů. Ne všechny systémy v podniku totiž budou znovu vyvíjeny, některé zůstanou v původním tvaru a rozsahu, jiné se rozšíří a inovují a další budou nakoupeny nebo vyvinuty znovu.

Pro vývoj správné topologie systému mohou být využity procesní diagramy, diagramy aktivit nebo assembly line diagramy[1] [Eriksson, Penker, 2000]. Tyto diagramy pomáhají identifikovat aktivity, které vyžadují spolupráci s informačním systémem. Aktivity v podnikových procesech, které indikují použití služeb od informačních systémů, jsou například:

1. Ukládání, třídění a aktualizace informace o jiných objektech v podnikovém procesu, informace o zdrojích nebo podnikových událostech.
2. Zpracování, konverze a prezentace informace.
3. Znalosti a provedení rozhodování, kdy softwarový systém vylepšuje nebo rozšiřuje kvalitu informace.
4. Komunikace mezi různými procesy, zdroji atd.
5. Řízení hardware, kdy všechna zařízení musí být průběžně kontrolována, monitorována a řízena.

Na obrázku 1 je znázorněn výsledný diagram systémové topologie podniku představující základní strukturu systémů, subsystémů a databází, ať už původních, inovovaných nebo nově vyvíjených s jejich vzájemnou strukturou a vztahy. Každý svazek pak sám o sobě může být rozpracován nezávisle na jiných svazcích, avšak při respektování vazeb na ostatní svazky.

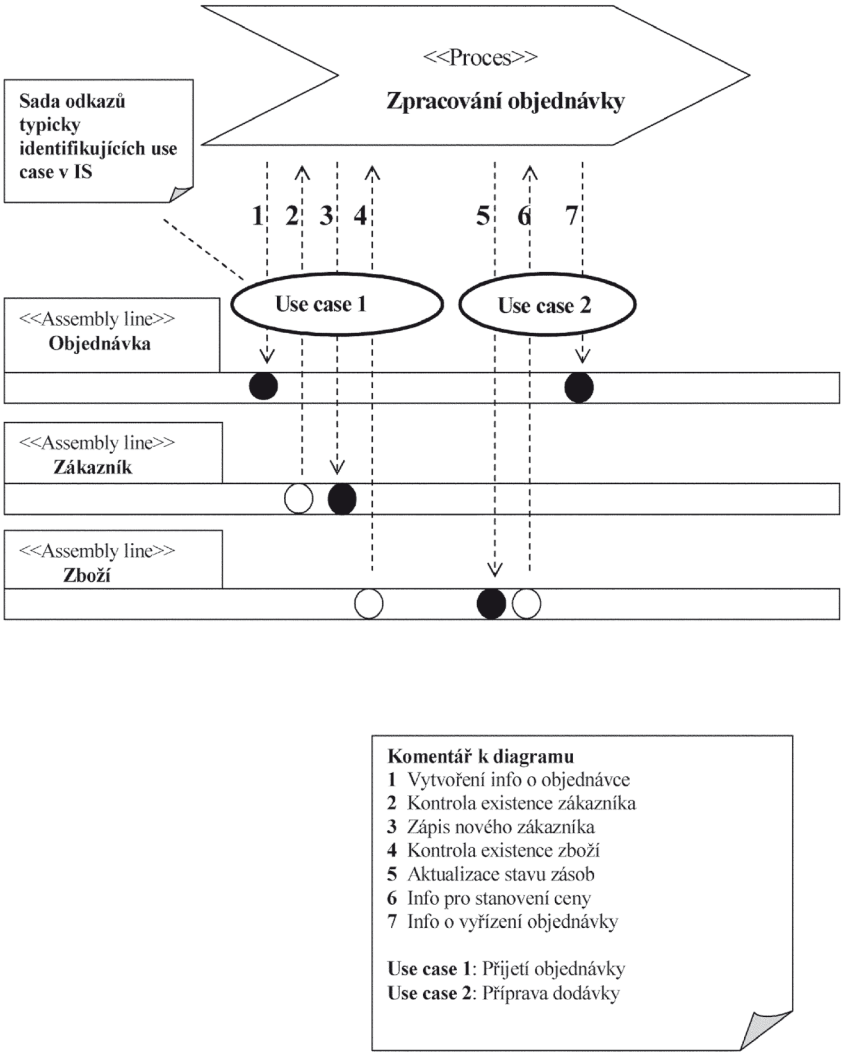


1: *Diagram systémové topologie podniku*

[1] Vzhledem k obtížnému překladu uvádím některé pojmy v angličtině.

2. Hledání funkčních požadavků

Snad nejdůležitější funkcí podnikové architektury v kontextu softwarového inženýrství je možnost jejího využití jako základu pro identifikaci správné sady funkcí nebo use case (případů jednání), kterými by měl softwarový systém nahradit nebo podpořit podnikové procesy. Funkční požadavky na systém jsou popsány v use casech[2], speciálně v textových popisech sekvence interakcí mezi informačním systémem a externím aktorem[3]. Use case pojednává o každém kroku komunikace mezi systémem a jeho okolím.



2: *Assembly line diagram procesu Zpracování objednávky*

[2] Podle [Booch, Jacobson, Rumbaugh; 1999] je use case (v českém překladu nejčastěji případ užití) specifikace posloupností činností, které systém může vykonat prostřednictvím interakce s externími účastníky.

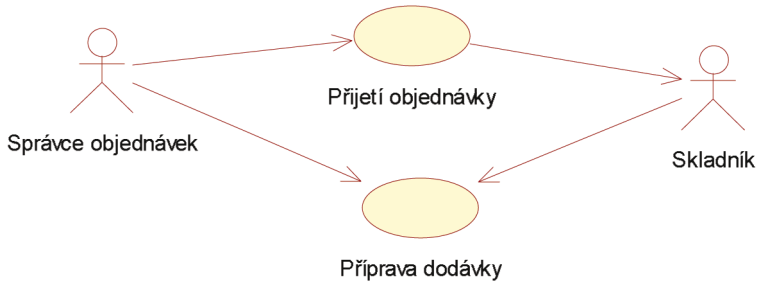
[3] Podle [Booch, Jacobson, Rumbaugh; 1999] je aktor někdo nebo něco, kdo spolupracuje se systémem.

Zde pokládám za nutné připomenout, že textový popis, tedy především scénář use case, specifikuje funkční požadavky, nikoliv samotný use case diagram UML, ten je pouze přehledem aktorů a use case systému. Textový popis use case (scénář) nebo grafický popis use case (sekvenční diagram, diagram aktivit, diagram spolupráce) se sice soustřeďují primárně na funkční požadavky, avšak někdy mohou obsahovat také nefunkční požadavky specifické pro tyto use case (jako je například výkonnost). V podnikovém modelu se use case identifikují z procesních nebo assembly line diagramů [Eriksson, Penker, 2000]. Na tuto druhou možnost identifikace use case z podnikové architektury je zaměřena stěžejní část tohoto článku. Navržený postup pro konkrétní podnikový proces znázorňují diagramy na obrázcích 2 až 4 a obecnější pohled je představen na obrázcích 5 a 6.

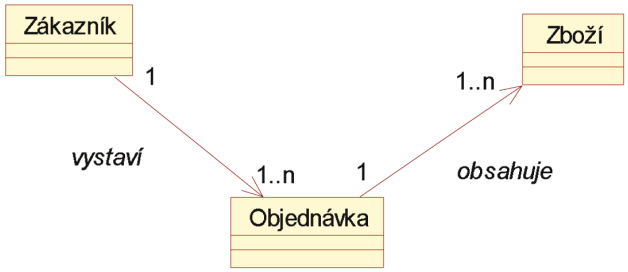
Assembly line diagram na obrázku 2 je unikátní typ diagramů objektové notace UML, který vznikl rozšířením diagramu aktivit UML o informační svazky zvané „assembly line“. V procesním modelování se tento diagram používá výhradně tehdy, je-li smyslem modelování právě tvorba softwarových systémů, které podporují procesy. Je vhodným integračním bodem mezi podnikovým procesem a use case, který představuje ucelenou funkcionalitu v systému. O assembly line diagramech se detailněji zmiňuji v [Rábová, 2002], domnívám se však, že jsou velmi názorné a i bez dalšího výkladu je jejich smysl a význam velmi dobře čitelný. Pro úplnost, šipka do assembly čáry znamená zápis informace procesem a šipka do procesu naopak znamená čtení informace procesem. Jednotlivé svazky jsou zásobníky informací (nejčastěji entity, avšak mohou to být také subsystémy nebo komponenty, podle úrovně abstrakce pohledu na proces) související s analyzovaným procesem. Tato struktura mi připomíná matice CRUD ve strukturované analýze.

Metodický postup pro tvorbu základního use case modelu, který je východiskem pro model softwarové architektury, navrhuji následovně:

- Vytvořte seznam klíčových a podpůrných procesů v podniku. Ke každému procesu namodelujte diagramy procesní a aktivit. Tam, kde předpokládáte softwarovou podporu procesu, vizualizujte situaci pomocí assembly line diagramu.
- Z assembly line diagramu identifikujte, které aktivity indikují použití služeb od informačního systému. Seznam těchto služeb (čar spojujících proces a assembly čáru) umístěte přímo do diagramu k příslušným čarám, nebo pro lepší přehlednost, do



3: *Use case diagram procesu Zpracování objednávky*



4: *Konceptuální model dat procesu Zpracování objednávky*

poznámky umístěné pod assembly line diagramem. Použijte výstižné názvy naznačující funkci v softwarovém systému (vytvoření záznamu o zákazníkovi, aktualizace zboží, kontrola datumu splatnosti atd.).

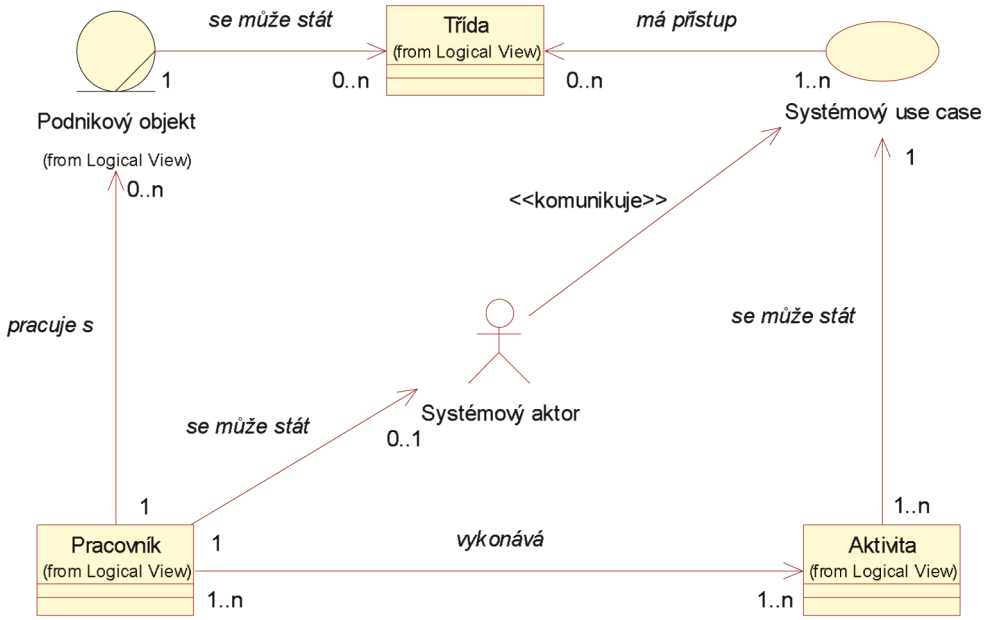
- Spojte assembly čáry do use case, které poskytnou komplexní službu v rámci jedné entitní třídy, subsystému nebo komponenty. Use case jsou stanovené jako specifikace služeb softwarového systému podnikovému procesu. Use case nazvěte ucelenou funkcionalitou, která přináší hodnotu vnějšímu aktorovi. Vytvořte seznam těchto use case.
- Tato use case analýza používá části podnikového modelu k hledání požadovaných služeb, které by mohl softwarový systém poskytnout. Zdroje, jako jsou lidé, stroje nebo jiné systémy v podniku, se stanou aktory pro analyzovaný systém a v use case jsou zachyceny interakce těchto zdrojů v podnikových aktivitách. Identifikujte aktory, které využívají analyzované use case.
- Vytvořte use case diagram a popište jednotlivé jeho prvky. Aktory krátkým slovním komentářem a use case scénářem (textovým nebo grafickým), jehož struktura může být předem dána [Rábová, 2002].
- Assembly line diagram je mocným nástrojem i pro identifikaci svazků objektů v softwarovém systému. Svazky (assembly line), které se nacházejí ve spodní části diagramu, jsou většinou, jak bylo uvedeno výše, zdroje perzistentní informace a reprezentují celý informační systém, subsystém, nebo komponentu, ale také velmi často specifickou entitní třídu prezentující podnikové objekty. Z identifikace těchto tříd v assembly line diagramu můžeme vytvořit konceptuální datový diagram celého systému. To záleží opět na úrovni abstrakce, pro kterou je model vyžadován.

Na obrázku 2 je znázorněn assembly line diagram procesu Zpracování objednávek, který obsahuje jeden proces a tři assembly line svazky. Odkazy z procesu do svazků jsou znázorněné šipkami a popsané v poznámce pod diagramem. Tyto odkazy jsou pak spojeny do dvou use case, které jsou na obrázku 3 spojeny do use case modelu. Ze tří svazků assembly line diagramu je potom na obrázku 4 znázorněn jednoduchý konceptuální diagram dat.

Vizualizací use case modelu a konceptuálního (doménového) modelu je vytvořena dostatečná základna pro postupný vývoj softwarového produktu, který má podpořit podnikový proces [agilealliance.com].

3. Hledání nefunkčních požadavků

Nefunkční požadavky jako výkonnost, přístupnost a bezpečnost nejsou normálně spojené se specifickým use case nebo funkcionalitou. Jsou to obvykle obecné vlastnosti, které ovlivňují funkce systému a musí být udržovány a uvažovány v mnoha use case. Často nejsou nefunkční požadavky ani specifické pro určitý systém v podniku. Tyto požadavky nejsou normálně navrhovány nebo implementovány do specifické komponenty nebo subsystému, ale ovlivňují mnoho nebo všechny komponenty nebo subsystémy. Nefunkční požadavky lze identifikovat z pravidel připojených k podnikovým procesům, zdrojům i cílům



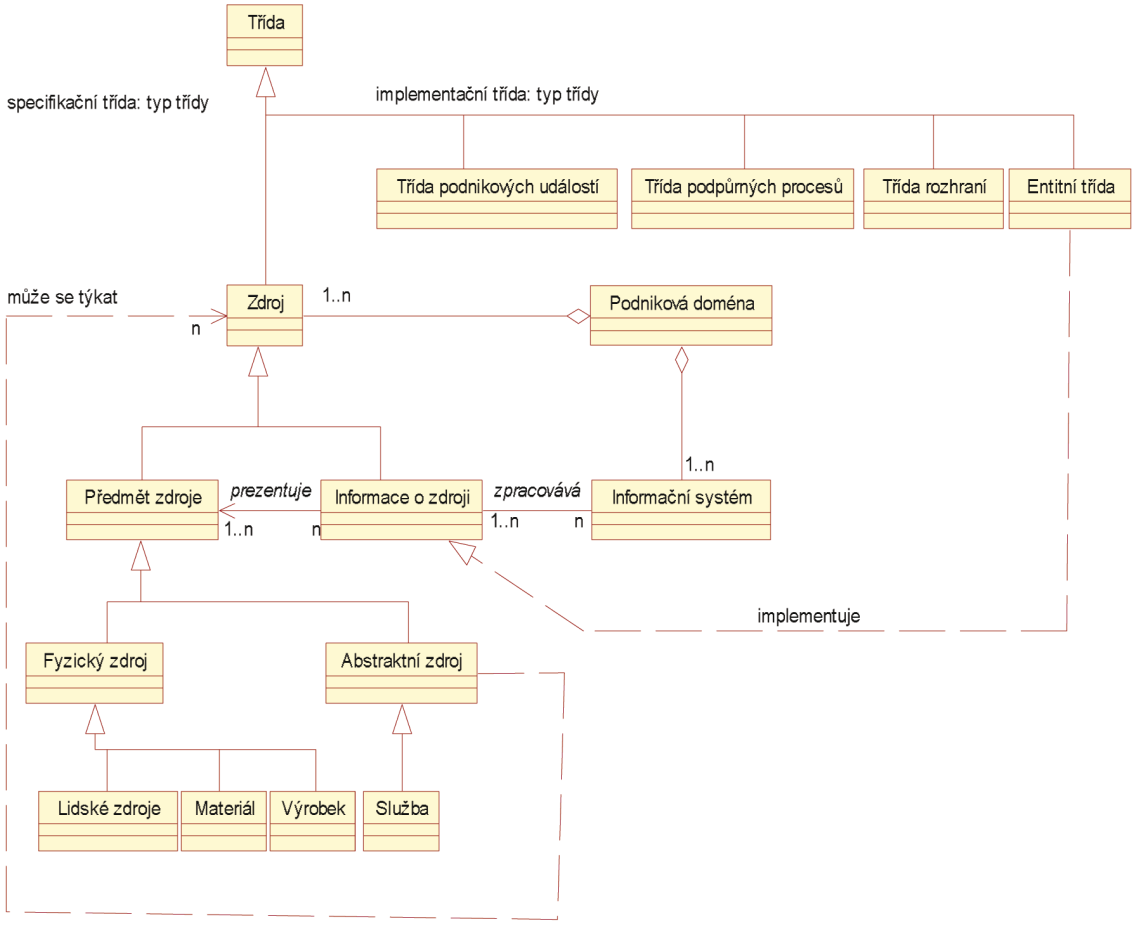
5: *Obecný přenos podnikového modelu do systémového modelu*

v diagramech podnikové architektury. Např. doba nutná k realizaci (doba mezi objednávkou a dodaným produktem), doba odezvy, sledování výkonnosti podnikového procesu, měření kvality, přístupnost, spotřeba zdrojů, bezpečnost nebo spolehlivost.

4. Identifikace tříd analýzy a designu systému

Identifikace tříd, jejich atributů a operací, jejich struktur, vztahů a spoluprací mezi nimi lze s výhodou převést z podnikového modelu do softwarového modelu. A protože budovaný nebo inovovaný softwarový systém se týká podniku a podporuje ho, mnoho tříd (zdrojů) v podnikovém modelu jsou nositeli perzistentních informací a budou implementovány do software. Doporučuji podívat se na podnikový model z pohledu každého systému a identifikovat, které třídy (hovořím o entitních třídách) jsou aktuálně požadovány v každém systému. Některé zdroje zahrnuté v procesu nejsou implementovány do software, ale komunikují s ním přes interface systému, představují objekty třídy rozhraní.

Obrázek 6 je pokusem znázornit vztah mezi třídami podnikové domény (podnikového modelu) a softwarového systému. Zde entitní třídy představují a implementují informační objekty z podnikového modelu. Informační objekt může obsahovat informaci o osobě nebo fyzickém produktu v podniku, ale také o abstraktním zdroji (služba, účet), nebo o jiném informačním objektu. Může také nést informaci o stavu procesu. A je to právě informace o těchto konceptech zdrojů, která je implementována do softwarového systému, nikoliv zdroj sám. Entitní třídy jsou rozpracovávány v etapě analýzy vývojového cyklu softwaru. Třída rozhraní implementuje rozhraní systému, tj. systémové interakce s jinými zdroji v celém procesu. Třídy objektů rozhraní jsou techničtěji orientované a implementují rozhraní v podobě uživatelského rozhraní GUI, rozhraní s jinými systémy nebo rozhraní k hardwarovým zařízením (které automatizují, řídí nebo informují například stroje). Tyto třídy rozhraní jsou částí technického návrhu softwarového systému a tedy nejsou prezentované podnikovými objekty. Je o nich pojednáváno často až v etapě designu.



6: *Vztah mezi třídami podnikového modelu a třídami informačního systému*

5. Identifikace vhodných komponent

Moderní softwarový vývoj používá komponenty jako autonomní svazky funkcionality, které nejsou specifické pro určitý systém, ale mohou být použity v několika systémech. Většina komponentních technologií se soustřeďuje na technické komponenty, v tomto příspěvku jde o definování podnikových komponent, které zahrnují specifickou a znovu použitelnou oblast podniku.

Znovupoužitelnost podnikových komponent založených na jednom z modelů obecné komponenty je klíčem k dosažení neutrality softwarového řešení. Většinu úspěšných podnikových komponent tvoří ty, které mohou být znovu použity v mnoha konfiguracích podle efektivní reakce na podnikové změny.

Domnívám se, že pokud se komponenty stanou spíše podnikově než technicky orientované, bude moci být podnikový model použit pro identifikaci komponent v softwarovém systému. Pro identifikování a specifikování komponent pak je potřebné nalézt korektní sadu služeb pro každou komponentu tak, aby se komponenta stala autonomní a mohla být znovu použita ve více než jednom systému (tj. může být nakonfigurovaná a přizpůsobená v jistém stupni pro každý systém).

Pro tento případ využití informací z podnikového modelu lze opět s výhodou použít již zmíněné assembly line diagramy, s tím rozdílem, že v assembly line nebudou entitní třídy, ale komponenty. Vztahy mezi podnikovým procesem, use case a komponentou mohou být shrnuty takto: *Podnikový proces použije jeden nebo více use case, které dodá systém, a use case mohou být vnitřně implementovány přes obecné komponenty, které dodávají všechny nebo některé use case.*

SOUHRN

Podnikový model a softwarový model jsou často vyvíjeny jako díla a dokumenty dvou různých týmů a modely pak nemají vztah jedna ku jedné. Mnoho prvků v podnikovém modelu nebudou součástí softwarového modelu a ne všechny podnikové procesy budou vyvíjeny nebo podporovány v softwarovém systému. Použití stejného modelovacího jazyka, zde UML, oba modely sbližuje a umožňuje přenos modelovaných informací do požadavků na softwarovou architekturu. Architektura podniku vyjádřená pomocí UML je tedy nejvhodnější architekturou pro tvorbu softwaru. Předložený příspěvek je vkladem k této problematice. Nastínil možnosti přechodu z podnikového do softwarového modelu a upozornil na celkový význam podnikového modelu v oboru softwarového inženýrství.

podniková architektura, softwarová architektura, funkční, nefunkční a vývojové vlastnosti softwarové architektury, systémová analýza a design, entitní třída, třída rozhraní, komponenta, případ použití, aktor, podnikový objekt, podniková třída

LITERATURA

ERIKSSON, H., PENKER, M.: Business Modeling with UML, John Wiley & Sonc, Inc., 2000

RÁBOVÁ, I.: Metamodel podniku ve vývoji informačních systémů, disertační práce, květen 2002

Business Modeling with UML: A Business Process Centred Architecture, The White Paper, http://www. agilealliance.com

BOOCH, G., JACOBSON, I., RUMBAUGH, J.: The Unified Modeling Language User Guide, Addison-Wesley, 1999

Adresa

Ing. Ivana Rábová, Ph.D., Ústav informatiky, Mendelova zemědělská a lesnická univerzita v Brně, Zemědělská 1, 613 00 Brno, Česká republika, rabova@mendelu.cz