AGILNÍ MODELOVÁNÍ A METODA BORM

Robert Pergl, Zdeněk Struska

Česká zemědělská univerzita, Provozně ekonomická fakulta, katedra informačního inženýrství

{pergl, struska}@pef.czu.cz

ABSTRAKT:

Agilní modelování je moderní přístup k efektivnímu využívání potenciálu modelování při tvorbě softwarových systémů. Pro agilní modelování bývá obvykle používáno rozšířené UML. Metoda BORM je komplexní metoda pro analýzu a návrh IS. Tento příspěvek si klade za cíl analyzovat možnosti metody BORM ve spojení s agilním modelováním.

KLÍČOVÁ SLOVA:

modelování, agilní modelování, AMDD, metoda BORM

ÚVOD

Modelování je dnes všeobecně chápáno jako nedílná součást fáze analýzy softwarového projektu a tvorby dokumentace.

Z obecného hlediska lze konstatovat, že modelování je činnost prováděná za účelem:

- tvorby abstrakce konceptu,
- permanentního záznamu informace.

Tyto dva účely je nutno chápat odděleně, neboť podle účelu se liší i charakter modelování. Nepochopení účelu modelování má potom v praxi za následek pouhý „byrokratický“ efekt, kdy modely pro abstrakci konceptu jsou programátory kresleny dodatečně a vydávány za dokumentační modely a na druhou stranu jsou často kresleny modely, které se nikdy nepoužijí a nikdo je nepotřebuje, jen „musí být“. Je nasnadě, že tyto jevy snižují efektivitu tvorby IS. Agilní modelování adresuje především tyto problémy.

Ačkoliv modelování vyžaduje značné zkušenosti a je do určité míry „uměním“, modelování samo je člověku velmi přirozená činnost, kterou na různých úrovních provádí spontánně každý jedinec. Lidský mozek není schopen pracovat s realitou ve své složitosti a pracuje vždy s určitým (zjednodušeným) modelem. Příkladem takového přirozeného modelování je manipulace s čísly, které reprezentují kvantifikační aspekt určitého problému.

Modely dělíme na [13]

- perceptivní,
- kognitivní,
- fyzické.

Perceptivní modely se týkají toho, jak vnímáme věci, kognitivní modely zachycují chápání věcí a souvislostí a fyzické modely hovoří o manipulaci s předměty a koncepty. Kvalitní model musí podchycovat všechny tyto složky.

Z hlediska reprezentace modelu může být model [12]

- slovní
- grafický
- matematický
- fyzický
- aj.

Pro potřeby modelování v softwarovém inženýrství používáme především kombinace slovních a grafických modelů, nicméně i fyzické modely v podobě prototypů jsou velmi vhodné.

AGILNÍ MODELOVÁNÍ

Agilní modelování (AM) [7] je prakticky orientovaná lehká metodika pro efektivní modelování a dokumentaci softwarových systémů. Jedná se o soubor praktik vedených principy a hodnotami, určených pro každodenní použití softwarovými profesionály. Nezavádí žádnou konkrétní notaci, je spíše návodem „jak být efektivním modelářem“. AM může být využita jak u projektů vedených agilně, tak u klasických projektů dle RUP či EUP [2]. Jejím autorem je proslulý Scott W. Ambler. Současná verze nese označní v2 a oproti verzi v1 došlo k některým zjednodušujícím úpravám.

Předpoklady

Aby (jakékoliv) modelování bylo efektivní, zúčastněné osoby

- musí znát (alespoň část) problémové domény,
- být schopni logického myšlení,
- mít co nejlepší komunikační dovednosti,
- smysl pro zdravý nadhled (tj. nesmí mít klapky na očích),
- pro projekt musí mít dostatek časového prostoru,
- musí rozumět své roli v projektu.

Správný agilní model by měl být „jen dostatečně dobrý“. Takový model se vyznačuje těmito rysy:

- plní svůj účel,
- je pochopitelný tím, pro koho je vytvářen,
- je dostatečně přesný,
- je dostatečně konzistentní
- je dostatečně detailní,
- poskytuje pozitivní hodnotu,
- je nejjednodušší možný.

Při modelování obecně narážíme na tzv. paradox pochopitelnosti [13], který spočívá v tom, že účelem modelu je pochopit komplexnost reality, ale abychom model pochopili, musí být co nejjednodušší. Jednoduchý model ovšem nemůže komplexně zachycovat realitu... Pochopení a přesnost jdou tedy proti sobě. „Jen dostatečně dobrý“ model zachycuje nejnutnější komplexitu a snaží se o maximalizaci pochopení a tím i využití modelu.

Jádro AM tvoří hodnoty, základní pricipy, doplňkové principy, základních praktiky a doplňkové praktiky. Tyto pilíře vychází z manifestu agilního vývoje softwaru [8] a jsou analogické ostatním přístupům založeným na agilním paradigmatu.

Hodnoty AM

Hodnoty tvoří filozofický základ, na kterém jsou vystavěny principy a poskytuje primární motivaci pro metodu. Hodnoty jsou

1. **Komunikace**. Je nezbytné, aby probíhala efektivní komunikace jak na úrovni týmu, tak mezi všemi účastníky projektu.
2. **Jednoduchost**. Snaha vyvinout nejjednodušší možné řešení, které splňuje všechny potřeby.
3. **Zpětná vazba**. Získávat zpětnou vazbu často a včas.
4. **Odvaha**. Je třeba odvaha zkoušet nové techniky a postupy.
5. **Pokora.** Pokora je nutná pro připuštění faktu, že nikdo neví všechno a kdokoliv může do projektu přispět něčím hodnotným.

První čtyři hodnoty jsou přejaty z metodiky Extrémního programování Kenta Becka [3].

Základní principy AM

Základní principy tvoří kostru agilního modelování a jsou základem pro konkrétní praktiky.

1. **Modelovat s důvodem**. Musí být jasné pro koho je model vytvářen a za jakým účelem.
2. **Maximalizovat investice**. Vynaložené prostředky (čas, peníze, zařízení) musí být využity co nejefektivnějším způsobem. Investoři jsou ti, kteří rozhodují o způsobu využití prostředků.
3. **Otevřít se změnám**. Je třeba se naučit přijímat fakt, že „změna je život“. V průběhu projektu jsou odhaleny nové požadavky, uživatelé si v průběhu vývoje systému lépe uvědomují své potřeby.
4. **Inkrementální změna**. Všechny změny je třeba provádět postupně, po malých částech.
5. **Vícero modelů**. K dispozici je řada modelovacích prostředků, které je vhodné se postupně učit využívat.
6. **Cestovat nalehko**. Modelovat jen s použitím nejnutnějších nástrojů, technik a vytvářet jen nejnutnější dokumentaci.
7. **Prvotním cílem je software**. Prvotním cílem je kvalitní fungující software splňující požadavky zákazníka.
8. **Druhotným cílem je umožnění dalšího postupu**. Systém musí být navržen dostatečně robustně, aby byly možné budoucí úpravy a rozšiřování.
9. **Kvalitní práce**. Práce by měla být kvalitní, nemusí však být perfektní (viz body 2,6).

10. **Rychlá zpětná vazba.** Rychlá zpětná vazba nutná pro včasné korekce.
11. **Vycházet z jednoduchosti.** Předpokládat, že nejjednodušší řešení je také tím nejlepším řešením.

Doplňkové principy AM

AM obsahuje též doplňkové principy, což jsou podněty, které je vhodné uvážit pro každý projekt a každý tým a pro maximalizaci pozitivního účinku přizpůsobit konkrétní situaci.

1. **Obsah je důležitější než forma.** Pro každý model existuje řada způsobů jeho vyjádření, např. model uživatelského rozhraní lze realizovat nálepkami na tabuli, nákresem na papíře, nákresem v grafickém programu nebo přímo prototypem. Vždy by měl být zvolen nejvhodnější model pro danou situaci.
2. **Otevřená a čestná komunikace.** Lidé by měli mít možnost svobodně nabízet své návrhy a nebát se poskytovat všechny informace.

AM původně obsahovala více doplňkových principů („každý se může učit od každého“, „znalost modelů“, „lokální přizpůsobení“, „pracovat v souladu s lidskými instinkty“), v lednu 2005 se však autor rozhodl je z nové verze metodiky odebrat. Důvodem nebylo, že by tato doporučení byla chybná, ale jsou značně obecná či samozřejmá a tudíž se autor rozhodl je nezačleňovat přímo do metodiky agilního modelování. Detailně se lze o tomto dočíst na http://www.agilemodeling.com/principles.htm.

Základní praktiky AM

Konkrétní praktiky byly formulovány na základě vyjmenovaných principů. Pro optimální využití AM je dobré přijmout všechny principy, nicméně i přijetí jen některých principů přináší výhodu.

1. **Aktivní zapojení všech účastníků.** Úspěch projektu často vyžaduje aktivní zapojení všech úrovní účastníků, od managementu po řadové pracovníky. Důležité je též zapojení osob ze souvisejících projektů budoucích správců systému.
2. **Používat nejjednodušší nástroje.** Autor konstatuje, že většinu agilních modelů je možné nakreslit na tabuli či papír. Používání CASE nástrojů je vhodné, ale musí být v souladu s principy 2,6,7,11.
3. **Modelovat s ostatními.** Pokud je model vytvářen za spolupráce více osob, je výsledek vždy lepší, než kdyby každý modeloval sám. Společnou prací a vzájemnou komunikací vzniká více nápadů, jsou dříve odhaleny chyby a nelogičnosti.
4. **Ověřovat kódem.** Modely by měly být co nejdříve ověřeny skutečným kódem, aby se ukázalo, jestli jsou správné.
5. **Používat správné artefakty.** Tato praktika znamená vytvářet adekvátní modely, které splní úkol. Každý model má své silné a slabé stránky, které je třeba vzít v potaz.
6. **Vytvářet více modelů paralelně.** Tato praktika navazuje na praktiku 5 – je vytvářeno více modelů, které budou vhodně popisovat různé aspekty systému. Tyto modely jsou postupně navzájem vylepšovány.
7. **Iterovat k dalšímu artefaktu.** Tato praktika adresuje nežádoucí situaci, kdy modelář „uvízne“ při modelování nějakého problému a není schopen postoupit kupředu. V takové situaci AM radí pokračovat v práci na jiném artefaktu.
8. **Modelovat v malých přírůstcích.** Toto pravidlo říká, že by neměl být dělán tzv. BDUF – Big Design Up Front, tedy příliš mnoho analýzy a dokumentace před každou

implementací. Nakopak, práce by měly postupovat systémem trochu modelování, trochu kódování, dodání hotového přírůstku.

9. **Kolektivní vlastnictví.** Kdokoliv může pracovat na jakémkoliv artefaktu uzná za vhodné.
10. **Vytvářet jednoduchý obsah.** Obsah všech modelů (požadavků, analýzy, architektury, designu, ...) by měl být udržován co nejjednodušší za současného splnění všech požadavků, které jsou na něj kladeny. Rozšíření modelu by mělo být prováděno jen pokud je k tomu objektivní důvod. Tuto problematiku též diskutuje příspěvek [11].
11. **Znázorňovat modely jednoduše.** Moderní modelovací notace obsahují velké množství prvků a možností. Modelář by si měl zvolit určitou podmnožinu, která mu umožní jednoduše znázornit klíčová fakta.
12. **Publikovat modely.** Tato praktika podporuje princip otevřené a čestné komunikace v týmu. Každý model by měl být snadno a rychle dostupný všem účastníkům projektu. Účastníci by neměli nic skrývat před druhými.
13. **Jeden zdroj informací.** Modely by měly být uloženy pouze na jednom místě a mělo by se předejít opakovanému kreslení stejného konceptu.

Doplňkové praktiky

Obdobně jako u doplňkových principů, i doplňkové praktiky podporují ty základní a je vhodné je nějakým způsobem začlenit do své praxe.

1. **Aplikovat modelovací standardy.** Vývojáři by se měli shodnout na společných modelovacích standardech a používat je.
2. **Aplikovat vzory.** Modeláři by se měli postupně seznámit s nejpoužívanějšími vzory [9],[10] pro analýzu, návrh a design a naučit se je používat.
3. **Zbavovat se dočasných modelů.** Většina vytvářených modelů má charakter pracovních náčrtků a je zbytečné je archivovat poté, co splní svůj účel.
4. **Formalizovat modely kontraktů.** Model kontraktu je třeba v případě, že systém bude navazovat na existující systém či spolupracovat s existujícími systémy.
5. **Aktualizovat jen pokud je to nutné** (v originále doslova „jen pokud to bolí“). Model by se měl aktualizovat jen v případě nejvyšší nutnosti, kdy aktualizace modelu způsobí menší komplikace než její neprovedení. Tato praktika přispívá k udržení jednoduchosti dle praktiky 10.

V rámci zjednodušení metodiky v lednu 2005 byly odstraněny následující doplňkové praktiky: „zvažovat testovatelnost“, „modelovat pro komunikaci“, „modelovat pro pochopení“ a „znovupoužívat existující zdroje“. Důvody odstranění jsou podobné jako u odstraněných doplňkových principů. Zastavme se pouze u praktiky „zvažovat testovatelnost“, která říká, že při vytváření modelu bychom měli zvažovat, jakým způsobem bude možné otestovat, jestli je vpořádku. Jedná se o velmi důležitou praktiku, nicméně AM je doporučováno v praxi kombinovat s vývojem řízeným testováním (Test Driven Development), čímž je toto pravidlo automaticky splněno.

Vývoj řízený agilním modelováním

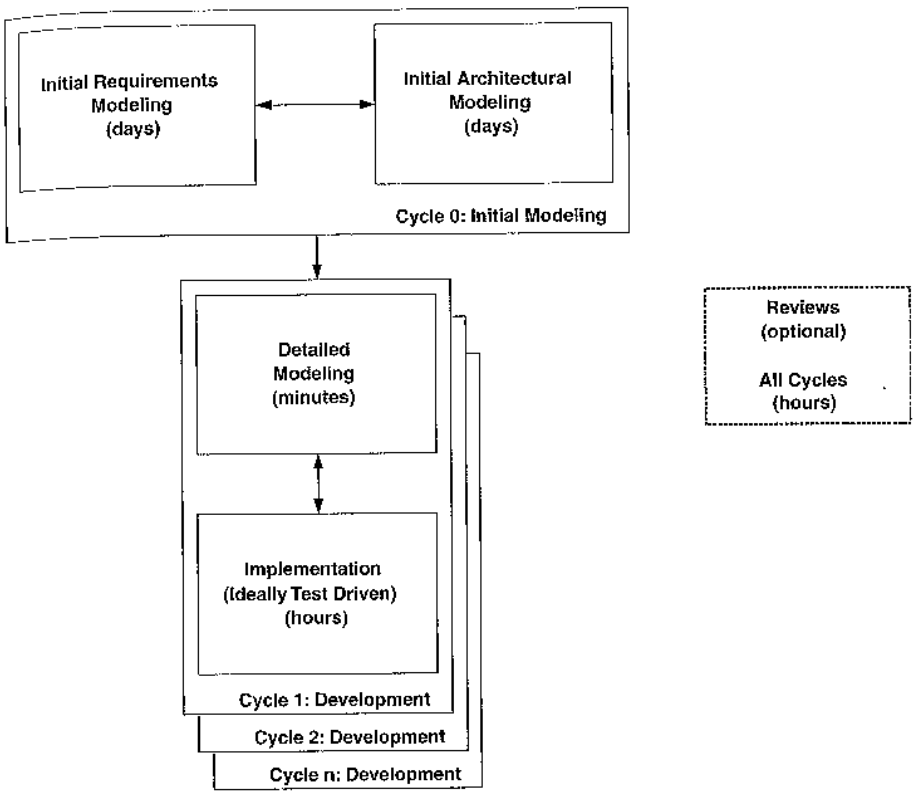
Vývoj řízený agilním modelováním, neboli Agile Model-Driven Development (AMDD) je agilní verzí vývoje řízeného modelováním (MDD). MDD se používá obvykle ve spojení s RUP/EUP a jde robustní cestou vytváření rozsáhlých modelů před psaním zdrojového kódu.

Oproti tomu AMDD vytváří odlehčené agilní modely, které jsou „dostatečně dobré“. AM lze použít ve spojení s prakticky všemi rigorózními i agilními metodikami. I když součástí většiny metodik je nějaká forma modelování, zbývá stále mnoho místa, jak proces vylepšit integrováním AM. Teoreticky se dá říci, že téměř vždy lze používat (alespoň část) AM, jelikož, jak jsme zmínili v části týkající se modelování obecně, modelování je přirozenou lidskou aktivitou. Navíc prvním základním principem AM je „modelovat s důvodem“. Pokud by tedy došlo ke zbytečnému zavedení nějaké formy či postupu modelování do vývojového procesu, bude to proti principům AM. V praxi však můžeme narazit na různé problémy související s organizační strukturou, omezenými možnostmi spolupráce, atd. Tyto aspekty však již přesahují rozsah a zaměření tohoto článku.

Rozdíl mezi AMDD a ostatními technikami založenými na modelování (např. Feature Driven Development) je v tom, že AMDD nespecifikuje, jaké typy modelů mají být vytvářeny. Pouze říká, že má být použit správný artefakt efektivním způsobem.

Hrubé schéma životního cyklu vývoje systému je vyznačeno na obrázku Obrázek 1 (převzato z [1]). Toto schéma velmi připomíná ostatní metodiky z agilní rodiny a říká, že na začátku projektu je nejprve provedena tzv. nultá iterace neboli úvodní modelování, která sestává z navzájem propojených modelování úvodních požadavků a úvodní architektury. V závorkách je uvedena řádová doba, kolik by která činnost měla trvat. V případě modelování úvodních požadavků a architektury se tedy jedná o dny.

Po provedení úvodního modelování přicházejí na řadu jednotlivé iterace 1..n, ve kterých je systém vyvíjen. Každá taková iterace se skládá z několikaminutového modelování funkce a poté její implementace, typicky v řádu hodin. Vývoj se tedy skládá z opakovaných činností, kdy je načrtnut malý model a ten je implementován. Je velmi vhodné, pokud implementace je prováděna též agilně, tj. za použití testování, refaktorizace, párového programování, atd..



Obrázek 1 – AMDD přístup k vývoji

METODA BORM A AGILNÍ MODELOVÁNÍ

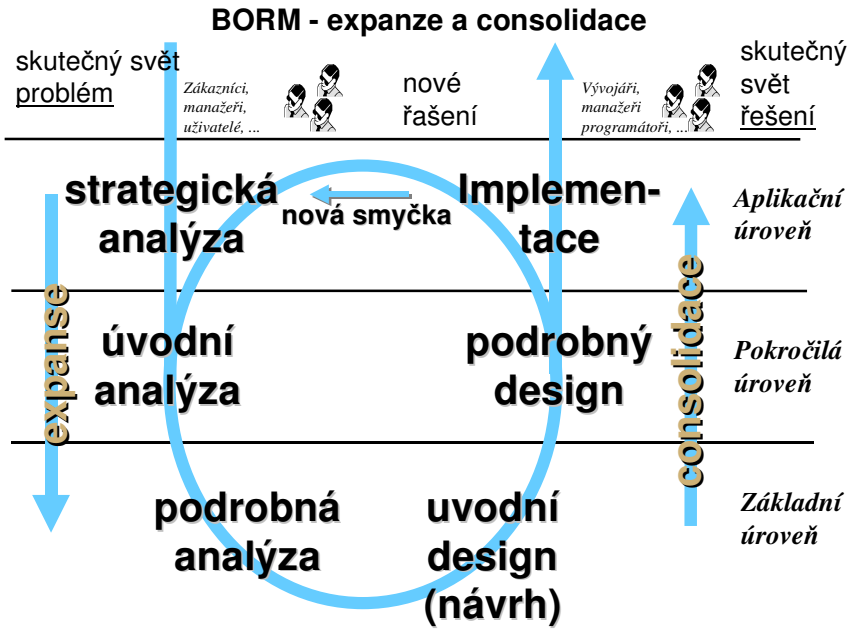
Metoda BORM (Business Objects Relation Modelling) se v posledních letech stala již poměrně známou a uznávanou metodikou a i na této konferenci již byla v minulých ročnících diskutována. Byla vyvíjena postupně od roku 1993 v rámci mezinárodního výzkumného projektu. Od roku 1996 je další vývoj podporován firmou Deloitte&Touche, kde je BORM také prakticky používán. Jeho přínosem je komplexní přístup k úvahám nad informačními systémy. BORM je možné využít nejen ve tvorbě softwaru, ale i k analýze požadavků na projektovaný systém a na modelování business procesů. Právě z procesů a dalších pojmů business modelování se v BORMu postupně vytváří model požadovaného ICT řešení.

Konkrétní použití metodiky BORM pod taktovkou AM záleží na typu projektu. Metodiku BORM lze, podobně jako např. UML, s úspěchem použít jak u agilního přístupu, tak u rigorózního.

BORM rozlišuje 6 fází životního cyklu vývoje systému:

1) **Strategická analýza.** Zde dochází k vymezení samotného problému, je stanoveno jeho rozhraní, jsou rozpoznány základní procesy, které se v systému a také v jeho okolí mají odehrávat.
2) **Úvodní analýza.** Zde dochází k rozpracování samotného problému, jsou mapovány požadované procesy v systému a vlastnosti základních objektů, které se na diskutovaných procesech podílejí.
3) **Podrobná analýza.** Je rozpracování analýzy do detailů jednotlivých typů objektů (sady objektů, třídy objektů) a objektových vazeb (skládání, dědění, závislosti, …).
4) **Úvodní návrh (design).** Je to první fáze, ve které se začínáme snažit systém upravit tak, aby byl schopen softwarové implementace. Proto se zde již nehovoří o analýze, neboť z pohledu zadání by mělo již vše být hotovo a rozpoznáno. Úvodní návrh používá shodné nebo velmi podobné nástroje jako předchozí fáze, ale liší se způsobem práce s nimi.
5) **Podrobný návrh (design).** V této fázi dochází k přeměně prvků již existujícího modelu do takové podoby, která je podřízena cílovému implementačními prostředí. V této fázi se zohledňují vlastnosti konkrétních programovacích jazyků, databází apod.
6) **Implementace** (tvorba, sestavování programu). V této fázi se vytváří (programuje, sestavuje či generuje z CASE nástroje) požadovaný software.

Všechny fáze jsou přehledně zakresleny na obrázku Obrázek 2.



Obrázek 2 – šest fází životního cyklu vývoje systému v BORMu

Při práci v BORMu jsou *ve fázi analýzy* vytvářeny nejčastěji tyto modely:

1. **seznam funkcí**,
2. **tabulka scénářů,** jejíž položky jsou scénáře. Každý scénář má zahájení, aktivitu a ukončení. Podílí se na něm participanti v určitých rolích.
3. **architektura na business úrovni** (vztah funkcí a scénářů),
4. **diagramy ORD** (Object Relation Diagram) modelující procesy, stavy a komunikace mezi participanty. Tyto diagramy jsou detailním rozkreslením scénářů.

Souběžně s vytvářením těchto modelů vznikají navíc **seznamy participantů** a **datových toků**.

Ve fázi *návrhu* jsou vytvářeny modely

1. **diagramy tříd a komponent**,
2. **tabulka podsystémů,**
3. **tabulka balíčků** (packages),
4. **architektura na implementační úrovni** (vztah podsystémů a balíčků)

Pokud budeme chtít BORM použít při AMDD, je třeba vývojové fáze namapovat dle obrázku Obrázek 1, tj. provést úvodní business modelování v řádu dnů a poté najet na krátké iterace obsahující kus modelování a kus implementace. Strategická a úvodní analýza z obrázku Obrázek 2 tedy proběhnou v rámci nulté fáze a podrobná analýza, design a implementace jsou poté obsahem dalších cyklů.

Rozeberme si nyní, jak je možné uplatnit jednotlivé praktiky AM při práci v BORMu:

Tabulka 1 – uplatnění praktik AM při modelování při práci v BORMu

Aktivní zapojení všech účastníků	Jedná se o obecnou praktiku, která se neváže přímo na BORM.
Používat nejjednodušší nástroje	BORMové modely lze snadno kreslit na papír či tabuli. Co se týká digitalizovaných modelů, existují šablonky do Visia a též modelovací nástroj Craft.CASE [5]. Ačkoliv je tento nástroj ještě v rané fázi vývoje a chybí mu doladění na základě zkušeností z praxe, je velmi přehledný a obsahuje šikovné funkce, které umožňují rychle vytvářet a měnit modely. Z tohoto hlediska přímo podporuje AM.
Modelovat s ostatními	Metoda BORM je díky své jednoduchosti a pochopitelnosti i pro neodborníky z tohoto hlediska velmi vhodná. Business diagramy je díky tomu možné vytvářet ve spolupráci se všemi účastníky projektu, omezeními jsou pouze organizační a řídící aspekty.
Ověřovat kódem	K ověření kódem dochází vždy při fázi implementace. Pokud se podaří zkrátit cykly dle obrázku Obrázek 1, je zajištěno brzké ověřování kódem.
Používat správné artefakty	BORM obsahuje vlastní modelovací artefakty, které splňují jak požadavky na modelovací sílu, tak na snadnou práci a pochopitelnost.
Vytvářet více modelů paralelně	V BORMu se začíná vždy seznamem funkcí a pokračuje se tabulkou scénářů, nicméně při další analýze nic nebrání tomu navzájem upravovat tyto modely – např. když odhalíme nový scénář, který nás přivede na funkci, kterou jsme opomněli. Při kreslení ORD diagramů je poté možné mít rozpracován libovolný počet těchto diagramů. Podobně je tomu i ve fázi návrhu.
Iterovat k dalšímu artefaktu	Pokud se například „zasekneme“ při kreslení jednoho scénáře, není v zásadě problém pokračovat jiným.
Modelovat v malých přírůstcích	Práce v metodice BORM je v principu vždy iterativní a je tedy třeba pouze vhodně zmenšit přírůstky.
Kolektivní vlastnictví	Jedná se především o „politické“ rozhodnutí, kdy všechny modely by měly být všem přístupné a kdokoliv by měl mít možnost je měnit. Artefakty v BORMu jsou značně pružné a při vhodně nastavených pravidlech a šikovném využívání CraftCASE je možné
Vytvářet jednoduchý obsah	Jedná se o obecnou praktiku, která se neváže přímo na BORM.
Znázorňovat modely jednoduše	Artefakty v BORMu jsou všechny velmi jednoduché (např. v porovnání s UML).
Publikovat modely	Jedná se o obecnou praktiku, která se neváže přímo na BORM.
Jeden zdroj informací	Jedná se o obecnou praktiku, která se neváže přímo na BORM, nicméně nová verze nástroje CraftCASE bude tento princip podporovat tím, že zavádí funkci slučování projektů.
Aplikovat modelovací	Tato praktika de-facto pouze říká, že je třeba se dobře seznámit

standardy	s metodikou BORM a postupovat dle jejích pravidel.
Aplikovat vzory	Jedná se o obecnou praktiku, která se neváže přímo na BORM.
Zbavovat se dočasných modelů	Jedná se o obecnou praktiku, která se neváže přímo na BORM.
Formalizovat modely kontraktů	Jedná se o obecnou praktiku, která se neváže přímo na BORM.
Aktualizovat jen pokud je to nutné	Jedná se o obecnou praktiku, která se neváže přímo na BORM.

ZÁVĚR

Vytváření modelů je pro lidské myšlení imanentní záležitost. Agilní modelování (AM) a vývoj řízený agilním modelováním je nový přístup, který dělá z modelování efektivní nástroj pro zlepšení softwarového procesu. Vývoj založený na AM – Agile Modelling Driven Development (AMDD) je metoda pro vytváření agilních modelů. Netýká se konkrétních modelů ani metodik a je možné ji s úspěchem použít v běžně používaných metodikách řízení tvorby softwaru, a to jak rigorózních (např. rodina UP), tak agilních.

Metoda BORM je moderní metoda založená na jednoduchých, snadno pochopitelných modelech, které však mají velkou vypovídací schopnost. Metoda zavádí propracovaný transformativní iterativní proces od úvodní business analýzy až po implementaci. Praktiky AM jsou v souladu s praktikami BORMu, lze tedy říci, že metoda BORM je vhodná i pro AMDD. Je třeba ovšem vhodně upravit vývojové cykly, aby byly v souladu s AMDD, což je změna, která může někdy narážet na organizační a jiné problémy.

LITERATURA

[1] Ambler, S. W.: „The Object Primer – Agile Model-driven Development with UML 2.0“, Cambridge University Press 2005, ISBN 978-0-521-54918-6

[2] stránky Scotta Amblera o agilním modelování: http://www.agilemodeling.com

[3] Beck K.: „Extrémní programování“, Grada 2002, ISBN 80-247-0300-9

[4] Merunka V., Polák J. ,Carda A.: Umění systémového návrhu. Praha: Grada Publishing 2003, ISBN 80-247-0424-2

[5] Merunka V.: Modelování podle metody BORM pomocí nástroje Craft.CASE, Sborník konference Objekty 2005, ISBN 80-248-0595-2

[6] Buchalcevová A.: „Agilní metodiky“. Sborník konference Objekty 2002, ISBN 80-21360947-4

[7] Buchalcevová A.: „Agilní modelování“. Sborník konference Objekty 2005

[8] manifest agilního vývoje softwaru: http://www.agilealliance.org

[9] Ambler S.,W.:„Process Patterns : Building Large-Scale Systems Using Object Technology“, Cambridge University Press 1998, ISBN 0-521-64568-9

[10] Ambler S.,W.:„ More Process Patterns : Delivering Large-Scale Systems Using Object Technology“, Cambridge University Press 1999, ISBN 0-521-65262-6

[11] Pergl, R.: “Pár poznámek k metodice BORM z hlediska systémového modelování”. Sborník konference Objekty 2004

[12] Získal J., Havlíček I.: „Ekonomicko matematické metody I“, ČZU PEF 2003, ISBN 80-213-0761-7

[13] stránky autorů projektu DOME týkající se modelování: http://www.htc.honeywell.com/dome/index.htm