

JAK SE POZNÁ DOBRÝ USE CASE MODEL?

RNDr. Ilja Kraval, březen 2010

<http://www.objects.cz>

ÚVOD

Na našem [diskusním fóru](#) se objevil tento zajímavý dotaz:

Honza » 04 březen 2010 15:23

Jak ověřím, že mnou navržený model UseCASE nebo i další je dobře? Spolužák má podobné téma projektu a jeho model UC i tříd je zcela jiný...Jak tedy poznám, že model je kvalitní?

Nejprve bych rád kolegovi poděkoval za příspěvek. Tomuto dotazu se budeme tímto článkem věnovat blíže.

Poznámka: Používejte prosím častěji naše diskusní fórum! Nezdráhejte se zeptat a pošlete své dotazy k návrhu IS pomocí UML!

K ČEMU SLOUŽÍ USE CASE MODEL?

Jak je z příspěvku patrné, dotaz byl evidentně položen studentem, proto se může na první pohled jevit jako „teoreticky složitý“, možná také nesmyslný. Co se má vlastně u Use Case Modelu posuzovat? Jaké vlastnosti dvou Use Case Modelů máme porovnávat? Jak se pozná, který ze dvou modelů téhož řešeného problému je kvalitnější?

Odpověď je následující: Vyhodnocení správnosti Use Case Modelu spočívá v účelu, pro který se tento model tvoří. Z toho je třeba vycházet a z toho nám okamžitě vyplynou požadavky na jeho kvalitu a jeho vlastnosti. ***Tvorba Use Case Modelu je totiž velmi praktická záležitost!***

Jak je uvedeno v knize [Analytické modelování pomocí UML v praxi](#), a jak se také probírá a velmi prakticky procvičuje na našich [školeních](#), Use Case Model spadá do fáze tvorby analytického modelu. Jedná se o tvorbu dokumentace na úrovni abstrakce analytického modelování. Jeden případ užití (tj. Use Case) se chápe přesně ve významu tohoto slovního spojení: Venku se stane událost potřeby použití systému, tj. někdo nebo něco z okolí potřebuje něco od systému, poté přistupuje k systému a provede se (říká se také instanciuje se) „jeden případ užití“. Synonymum pro laika je také „funkcionalita systému“ viděná v kontextu laika budoucího uživatele takto: „něco / někdo venku“ – „potřebuje“ – „použije“, a to, co se v navrhovaném systému vyvolá použitím, je jeden případ užití. Mimochodem z tohoto důvodu má Use Case Model přímou návaznost na tvorbu procesního modelu podniku (tzv. BPM). Diagramy UCM a BPM se tvoří ve fázi analytického modelování jako jedny z prvních diagramů, tj. začíná se jimi. Některé školy modelování jej díky tomu zařazují přímo do fáze „Requirements“, tj. fáze požadavků na systém. Není to úplně přesné, protože modelování pomocí případů užití je již návrhem IS z pohledu funkcionalit, které tento systém nabízí. Předtím však proběhla ještě skutečná fáze sběru požadavků, tj. vznik a správa „haldy požadavků“, jejich uspořádání, schválení apod. To ještě není modelování a není to tedy přímým obrazem (tj. modelem) systému. Z pohledu úrovní abstrakce tvorba Use Case diagramu již spadá do modelování systému na nejvyšší úrovni abstrakce.

Use Case Model patří k nezastupitelným modelům při návrhu IS. Pokud se podíváme na jeho postavení při návrhu IS, pak praktický účel tvorby Use Case Modelu je následující:

1. Transformuje původní (mnohdy dost vágně vyslovené) požadavky zákazníka na funkcionality IS do korektně tvořeného Use Case Modelu v silném modelovacím jazyce UML, který má silnou vnitřní logiku. Proto se dá zhodnotit logická správnost tohoto modelu, dá se zvážit jeho konzistence a přesnost. To při jeho tvorbě vede k dalšímu precizování a doplnění detailů původních požadavků zákazníka (tzv. „brainstormingové kolečka při interview u zákazníka“). Samozřejmě díky této transformaci může nastat přehodnocení původních požadavků zákazníka, například vyloučení logicky protichůdných požadavků, přidání nových apod. Zajímavou otázkou vhodnou k diskusi je také problém, zda mají být rovněž zaevidovány požadavky zákazníka anebo nikoliv. Samozřejmě doporučuji je také zaevidovat, ale nejprve pouze neformálně (v textu bokem apod.), protože „brainstormingová kolečka rozhovorů se zákazníkem“ bývají mnohdy dost hektická. Praxe ukazuje, že je velmi naivní představa, že od zákazníka získáme na první pokus úplné a relevantní funkcionální požadavky. Teprve až tvorba modelu případů užití prokáže jejich korektnost a úplnost. Doporučuji pro takovýto první neformální zápis požadavků od

zákazníka použít velmi praktickou záležitost, kterou je nutná tvorba zápisu výsledku rozhovorů se zákazníkem (například v projektech, kde jsem působil jako vedoucí projektu, se nazývaly tyto dokumenty jako „zápisy z interview se zákazníkem“). Teprve v dalším již formalizovaném kolečku se tzv. Requirements zapisují do CASE nástroje (například EA, který tuto evidenci přímo podporuje). Další zajímavou otázkou je tvorba systémů, které nejsou tak zvané „na klíč“ a které jsou připravovány obecně na trh, a proto při tvorbě Use Case Modelu neexistuje relevantní zákazník žádající konkrétní IS. V tom případě se v SW firmě musí imitovat zákaznickovy představy nejlépe formou „brainstormingového týmu“, do kterého jsou pozváni všichni zainteresovaní včetně externích konzultantů. Z toho týmu vycházejí nové požadavky, které poté hlavní analytik spolu s obchodním oddělením reviduje.

2. Současně se model případů užití stává srozumitelným a čitelným zadáním pro programátora. Znamená to, že jeho předání do nižší úrovně abstrakce je dobrým a korektním zadáním práce pro programátora.

POŽADAVKY NA KVALITU MODELU PŘÍPADŮ UŽITÍ

Z předešlých bodů přímo vyplývají požadavky na kvalitu Use Case Modelu.

1. Výstup dokumentace z Use Case Modelu pro zákazníka (poznámka: po drobných úpravách a dodaný spolu s BPM modelem!) musí být pro zákazníka velmi dobře srozumitelný, tj. musí být čitelný pro jakéhokoliv laika, který rozumí dané problematice a je pouze lehce obeznámen s principy Use Case Modelu.
2. Model případů užití musí splňovat představy zákazníka z hlediska úplnosti a správnosti požadavků funkcionalit na něj.
3. Model případu užití musí být konzistentní ve své logice. Jedná se zejména o kontrolu obdoby „zákona zachování informace“. Jestli nějaký případ užití pracuje někde s nějakým evidovaným prvkem v nějakém stavu, pak tento prvek musel být jednak někde vytvořen a také někde nastaven do tohoto stavu. Pokud tomu tak není, chybí někde případy užití. Například takovou chybou je, když se ve scénáři nějakého případu užití dočteme „Obsluze se zobrazí seznam osob...“, ale nikde v žádném případě užití se osoby do IS nezakládají.
4. Na druhé straně model případů užití nesmí být zbytečně složitý a nesmí mít nadbytečné části. Znamená to, že máte-li k dispozici dvě varianty řešení (které splňují

všechny vyjmenované předešlé body), tak lepším řešením je to, které je jednodušší a stojí méně peněz.

5. Model případu užití splňuje požadavky z pohledu programátora. Chybou by bylo, kdyby programátor dostal spolu s modelem případu užití do ruky problém „co má vlastně programovat“. Naopak zadání do programování musí být jasné a zřejmé do všech detailů. Samozřejmě opět zde platí princip „kolečko zpětné vazby“, tj. bylo by naivní se domnívat, že první odevzdání modelu případů užití do programování bude stoprocentní a proto programátoři k němu nebudou mít žádné výhrady. Při realizaci budou v modelu případu užití určitě narážet na nejasnosti, protimluvy, neúplnosti apod., a proto budou zpětně vznášet požadavky na změny a na doplnění tohoto analytického modelu.

ODPOVĚĎ NA DOTAZ

Z předešlých uvedených bodů lze formulovat také odpověď na dotaz z diskusního fóra.

1. Prvním posouzením kvality modelu je otázka srozumitelnosti modelu a textu pro laika, který by měl výstupy dokumentace z daného modelu číst. Pokud je nesrozumitelný laikovi, model je špatně. Proveďte tedy výstup dokumentace stejně jako pro zákazníka (případně doplňte) a vysvětlete tuto dokumentaci někomu „mimo programátorský obor“, ale přitom někomu, kdo zná danou problémovou oblast. Pokud vám nebude rozumět, není model dobře vytvořen.
2. Jako druhé posouzení kvality modelu případů užití je odpověď na otázku, do jaké míry model splňuje požadavky zákazníka. Pokud je tvůrcem modelu student, tak zde může vzniknout určitý problém, protože samotné jednorázové zadání od lektora například jako zápočtová práce nemusí být jako jediné úplné a stoprocentní. Navíc student nemusí mít možnost ujasnit si nějaké detaily pomocí „interview se zákazníkem“. Tento fakt by mohl být důvodem, proč by dva studenti mohli vypracovat z hlediska téhož zadání dva vcelku rozdílné modely a přitom by oba dodrželi „úvodní zatím blíže nespecifikované požadavky od zákazníka, zde lektora“.
3. Další otázkou je úplnost modelu z hlediska logiky. Zkontrolujte v modelu všechny scénáře, zda se někde nepracuje s instancemi, které nikde nebyly založeny resp. pracuje se s jejich stavy, které nikde nebyly zadány. Pokud ano, model není v pořádku.

4. Pokud je model podle předešlého bodu logicky v pořádku, ověřte, zda model neobsahuje nadbytečné funkcionality, které zákazník nechtěl anebo zda neobsahuje příliš složité postupy, které lze zjednodušit. Pokud ano, model zjednodušte.
5. Jako student dejte daný model přečíst nějakému příteli - programátorovi a zeptejte se ho, zda by byl podle něj ochoten napsat program. Pokud se bude programátor ošívat, tak model je špatně. Naopak, pokud programátor odpoví, že by to rád naprogramoval a bude se chtít hned hrnout k práci (☺), model jste jako zadání napsal velmi dobře.

Nakonec mám ještě jeden návrh: Můžete mi poslat [e-mailem](#) další podrobnosti k dané problematice, pokud by nebyl nějaký problém se zveřejněním, mohli bychom Vaše téma rozvinout dále.

Konec článku

- [Využijte praktický a levný „Workshop UML“ s revizí postupů návrhu IS ve vaší firmě](#)
- [Pobytový kurz Návrh IS pomocí OOP a UML v Bílých Karpatech jaro 2010](#)
- [Pobytový kurz Návrh IS pomocí OOP a UML v Praze jaro 2010](#)
- [Prezenční kurz Návrh IS pomocí OOP a UML v Praze jaro 2010 \(pro účastníky z Prahy a okolí\)](#)