

7 DŮVODŮ, PROČ JE TŘEBA PŘI NÁVRHU INFORMAČNÍHO SYSTÉMU VYŽADOVAT TVORBU USE CASE DIAGRAMŮ VČETNĚ PROCESNÍHO MODELOVÁNÍ

Část 3

(článek je pokračováním předešlého)

RNDr. Ilja Kraval, únor 2009

<http://www.objects.cz>

ÚVOD

Jedno z doporučení návrhu informačních systémů se dá vystihnout slovy: „Dbejte na to, aby projekt byl tzv. *USE CASE DRIVEN*, tj. aby byl řízen pomocí diagramů případů užití“.

Jedná se o doporučení nikoliv pro samotné vývojáře, ale pro vedoucího projektu. O tom, proč je toto doporučení opravdu platné, pojednává tento článek. V předešlých článcích jsme si uvedli tyto důvody (i když jich může být teoreticky i více):

1. Velmi vhodné zadání algoritmů chodu aplikace již z analýzy až do programování
2. Výrazné zamezení efektu bobtnání projektu
3. Možné odhady pracnosti na projektu a jeho řízení
4. Efektivnější a snadnější tvorba uživatelské dokumentace
5. Vynikající podklady pro funkcionální testování
6. Podklady pro marketingové materiály a obchodní prezentace
7. Efektivní tvorba dokumentu funkční specifikace produktu jako přílohy smlouvy mezi odběratelem a dodavatelem SW

V předešlém článku byly popsány body 1, 2 a částečně bod 3, kdy se odhadují pracnosti úplně na začátku projektu ve fázi tzv. strategického modelování. V této kapitole se zmíníme o

odhadech pracnosti na projektu již přímo nad vyhotovenými případy užití, tj. když je k dispozici nejenom jejich výčet, ale i jejich „vnitřní popis“ například pomocí scénářů.

ODHADY PRACNOSTI VE FÁZI ANALYTICKÉHO MODELOVÁNÍ

Existuje spousta různých metod vyhodnocení pracnosti projektu vycházejících z UC modelu. I přes svou různorodost jsou všechny postaveny na jedné základní myšlence: Provádí se „ohodnocení případů užití“ nějakou veličinou a poté se vypočte nějakým vzorcem „součtování“ hodnot přiřazených k případům užití. Samozřejmě výsledek je vždy hrubým odhadem a nebude nikdy přesný (bohužel zejména na začátku projektu, kdy je odhad nejžádanější).

METODA UCP

Jednou z neznámějších metod je metoda nazývaná „USE CASE POINTS“ (UCP). Musím zde poděkovat jednomu z diskutujících na našem fóru za velmi pěkné odkazy, které umístil do příspěvku na našem fóru [zde](#), hlavně článek v pdf formátu popisující tuto metodu je opravdu velmi kvalitní!

Postup UCP je poměrně dost složitý (jako příklad textu cituji) : „Nejprve musíte zjistit počet aktorů a počet prvků USE CASE. Prvek USE CASE zařadíte do kategorie složitosti (jednoduchý, střední, složitý) podle počtu transakcí (počet transakcí může být odvozen z počtu kroků popsaných ve scénáři). Někteří autoři doporučují vynechat USE CASE typu extends a includes. Podle kategorie složitosti přiřadíte jednotlivým prvkům USE CASE váhu. Dále musíte ohodnotit dílčí technické faktory a faktory prostředí stupněm 0 (nemá vliv) až 5 (silný vliv). Tyto faktory se týkají konkrétního projektu. Dosazením do Karnerových vzorců získáte počet USE CASE POINTS (UCP)...“

Jak vidět, tak myšlenky této metodiky a výpočty jsou opravdu poměrně dost složité.

Mimochodem tato metodika UCP je zavedena přímo v nástroji Enterprise Architect (anebo alespoň se jeví jako velmi podobná), blíže viz například help tohoto nástroje [zde](#).

METODA PŘÍMÉHO ODHADU A ÚSKALÍ OBJEKTIVÉHO PARADIGMATU

Rád bych se však zmínil o jednom zajímavém efektu, který se může projevit při odhadu pracnosti. Je to důsledek objektového paradigmatu, který byl vysvětlován v předešlých článcích.

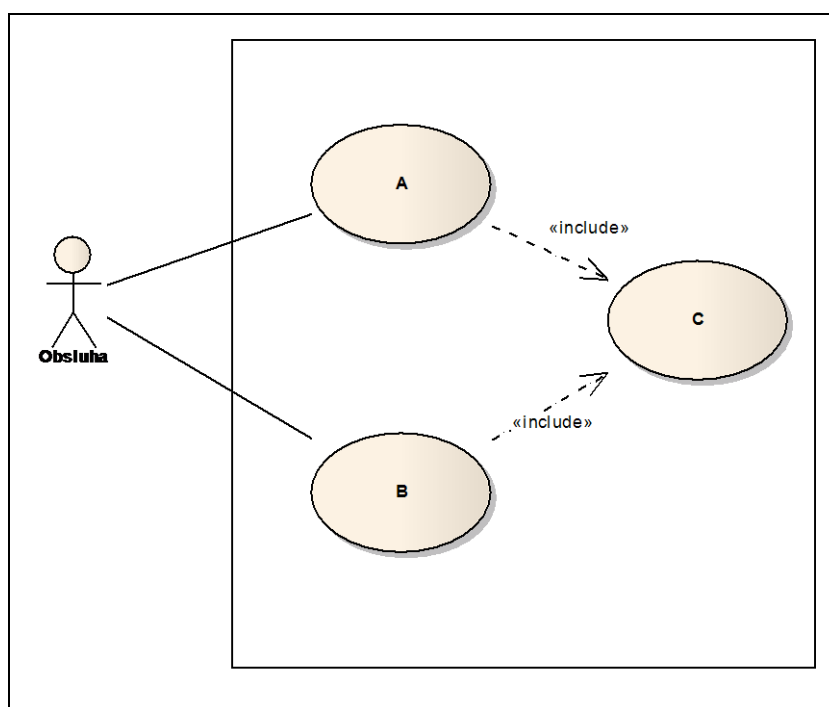
Pro vysvětlení tohoto efektu uvedu v té souvislosti jinou metodu výpočtu, kterou jsem měl možnost používat v praxi a která je opravdu velmi jednoduchá až „primitivní“. Ze zkušenosti mohu potvrdit, že má vcelku dobré výsledky.

Po vyhotovení případů užití a jejich scénářů včetně interakcí, tj. <<include>>, <<extend>> a generalizace, vedoucí projektu ve spolupráci s hlavním analytikem projektu, spolu s hlavním technologem projektu a s vedoucím programátorů projektu, provedou takřkajíc „napřímo“ odhad pracnosti případů užití a to tak, že se ohodnotí každý daný případ užití přímo dvěma hodnotami: „spodní hranice minimum“ odhadu pracnosti a „horní hranice maximum“ odhadu pracnosti. Samozřejmě je to odhad hodně subjektivní, ale i předešlá metoda má podobný charakter. Rozdíl je v tom, že tato „primitivní přímá“ metoda v podstatě neprovádí výpočet pomocí kategorizací případů užití (resp. jiných dalších veličin), ale všechny parametry této kategorizace dávají uvedení pracovníci svým subjektivním odhadem rovnou jako „přímý výsledek“. Každý případ užití tak získává dvě hodnoty: Jedna je „spodní“ hranice odhadu pracnosti (tj. optimistický odhad) a druhá je „horní“ hranice odhadu pracnosti (tj. pesimistický odhad). Odhad pracnosti je pak dán pochopitelně pomocí součtování hodnot.

Poznámka: Pro zájemce mohu (po lehké úpravě) tuto utilitu poslat mailem. Pokud chcete, napište mi na objects@objects.cz

Rád bych zde upozornil na jeden efekt, který souvisí s objektovým paradigmatem.

Představme si, že jsme našli v UC modelu v následující situaci, kdy případy užití A a B „inkludují“ v sobě případ užití C:

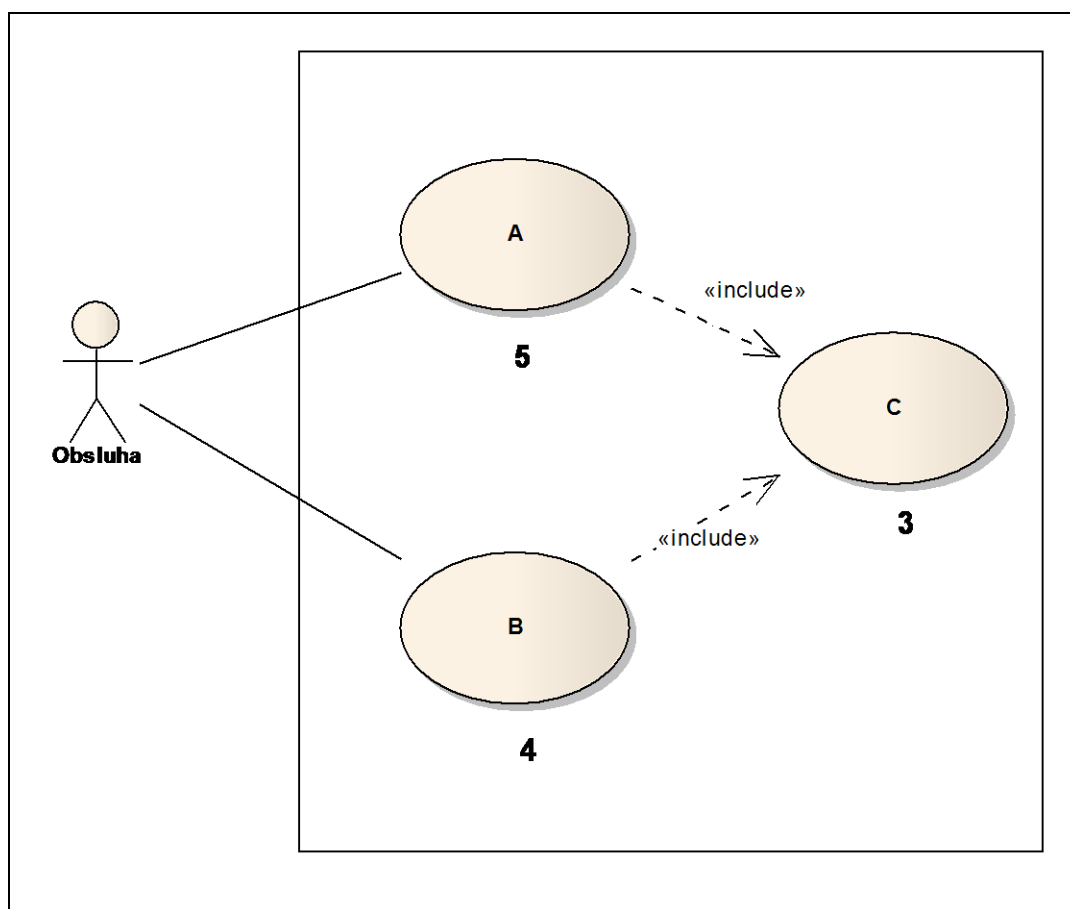


Obrázek 1 Příklad užití C je použit v interakci <<include>> z A i z B

Zaměřme se zde na důsledky efektu opětovné použitelnosti. Příklad užití A v určitém svém scénáři „volá“ případ užití C a stejně tak případ užití B také v určitém bodě scénáře „volá“ tentýž případ užití C. Je zřejmé, že případ užití C by měl být realizován pouze jednou. Ve scénáři je navíc velmi žádoucí, aby pokud píšeme scénáře korektně a podle přesných pravidel (velmi podrobně se tato pravidla probírají [na tomto školení](#)) tak, aby se toto volání objevilo přímo v přesném bodě scénáře například ve formulaci typu „...provede se případ užití C“.

Všimněme si, jak zde zafungovala znovupoužitelnost (tj. re-use): Pokud tedy ohodnocujeme případy užití nějakou součtovou veličinou pracnosti, měli bychom ohodnotit daný scénář pouze bez vytknutí, tj. hodnotíme pouze čistý scénář bez onoho volání, aby se nám hodnota C neopakovala v součtu. Je zřejmé, že pokud technologové dodrží opětovnou použitelnost z analýzy až do technologie (například vkládáním nové instance téže části obrazovky *frame* a za ní připojenými objekty), potom se pracnost díky re-use snižuje.

Pokud si tedy do předešlého obrázku přimalujeme ony odhady, tak by obrázek vypadal například takto:



Obrázek 2 Zobrazené pracnosti na obrázku se sčítají, pokud každá z nich reprezentuje pouze vnitřní odhad vnitřní implementace bez C

V tomto příkladu je ohodnocen scénář případu užití A číslem 5 (např. člověkodny), ale pozor: V tomto hodnocení nesmí být zahrnuto hodnocení scénáře případu užití C, tj. 3! Stejně tak scénář B je ohodnocen číslem 4, ale v tomto hodnocení není zahrnuto hodnocení případu užití C, tj. opět 3!

Pokud začínáme tvořit systém a ptáme se na pracnost A, pak existují dva pohledy na danou pracnost přesně tak, jak nám velí objektové paradigma:

1. Pohled zně na A, tedy pohled na A jako celek včetně C. Celá pracnost celého A při pohledu zně je 8 (celý scénář včetně „inkludovaných“).
2. Pohled zevnitř, tedy na vnitřek A, tedy pouze scénář A, který volá C. Samotné A (tedy bez C) má pracnost 5.

Podobně bychom zjistili totéž u B: při pohledu zně má pracnost 7, zevnitř ve své struktuře jako samotný scénář B bez C má pracnost hodnotu 4.

Tento rozdíl pohledů je třeba zohlednit, protože je zřejmé, že pracnost realizace všech tří případů užití by neměla být dána součtem vnějších pohledů, ale všech pohledů vnitřních na scénáře bez interakce. V našem případě tedy 5 + 3 + 4. V důsledku to znamená, že pokud provedeme vytknutí přes <<include>>, měli bychom pracnost přehodnotit, protože je dána pouze vnitřním scénářem bez odskoků volání.

Navíc dá ještě doporučit, aby se zohlednila i režie realizace samotného <<include>>, tj. aby se zohlednila pracnost daná „rozchozením“ tohoto volání. Jedná se například o pracnost vložení již hotového „kusiska“ formuláře do jiného „kusiska formuláře“ a nutnost prací pro rozchození tohoto volání. Zde by měla stačit jedna dopředu zvolená konstanta (například 0.25 člověkodnů na jedno <<include>>), kterou se vynásobí počet nalezených <<include>>.

Vůči této primitivní metodě „ohodnoť přímo“ by se dalo namítnout, že je velmi subjektivní. S tím se dá souhlasit, ale není bez zajímavosti, že u trochu zkušenějších pracovníků dochází k zajímavé shodě s malými odchylkami a přitom dochází navíc ke kýženě shodě s následnou realitou s malou odchylkou. Zjistil jsem, že nejlepší odhad mají sami programátoři a mezi nimi jsou vynikající vedoucí programátorů. Je také vhodné výsledky nechat „ošlehat“ brainstormingovou diskuzí přímo mezi programátory, zejména při rozdělování prací je vhodné toto provádět v týmu.

ZÁVĚREČNÝ PŘÍKLAD

Můžeme využít tento článek pro zajímavou statistickou „hru“, jak se liší odhady pracovníků na jeden a tentýž scénář.

Zadání příkladu:

Napíši zde jeden scénář případu užití a k tomu dodám model tříd. Zkuste odhadnout pracnost spodní a horní hranice v člověkodnech. Technologicky se jedná o webovskou aplikaci se střední vrstvou (JAVA nebo .NET), v pozadí relační databáze (ORACLE, MS SQL, MYSQL, POSTGRESQL). Pokud máte zájem, napište mi svůj odhad na mail objects@objects.cz

Odpovědi poté vyhodnotíme.

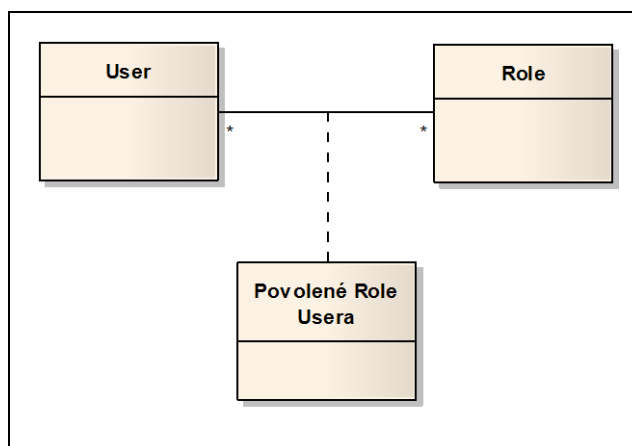
Případ užití přihlášení a výběr role

Obsluha zadá Username a Password (Password zahvězdičkovat). Podle Username se nalezne User v seznamu Userů. Pokud nenalezen, pak viz ERROR1. Zadaný Password se zahašuje MD5 a zverifikuje s Passwordem u nalezeného Usera. Pokud nesouhlasí, pak viz ERROR1.

Obsluze se zobrazí seznam povolených Rolí pro daného Usera. Obsluha vybere jednu Roli. Nalezený User i vybraná Role se drží po celou dobu další práce obsluhy až po odhlášení z Role resp. do odhlášení ze systému.

ERROR1: Obě zadávaná pole se vyčistí a objeví se hlášení „špatně zadané heslo nebo neexistující uživatel“

Nalezený analytický model tříd pro tento scénář:



Obrázek 3 Analytický pojmový model tříd

Těším se na vaše odhady na mailu objects@objects.cz !

V příštím článku si popíšeme další postupy ve využití procesního modelování a tvorby případů užití.

Diskuse k článku viz [Fórum Serveru objektových technologií zde](#)

<http://www.objects.cz>

strana 8

Pokračování příště!