Specifika procesního řízení a modelování v rámci agilních metodik

Specifics of process management and modeling in area of agile methodologies

Josef Myslín

Abstrakt: Příspěvek se snaží zodpovědět otázku, zda je možné použít běžné definice a přístupy procesního modelování pro agilní metodiky. A ačkoliv odpověď zní ano, objevují se zde aspekty agilního vývoje, které je nutné respektovat a brát v úvahu. Proto je dalším cílem příspěvku přinést základní odpovědi na to, jakým směrem se může ubírat vývoj procesního modelování, které bude skutečně odpovídat potřebám a filosofii agilního vývoje software včetně ukázky řešení konkrétního problému. Nejedná se v žádném případě o vyčerpávající práci, ale spíše o nástin budoucího výzkumu, který autor plánuje do dalšího období.

Klíčová slova: vývoj SW, agilní metodika, procesní modelování, UML, EPC, BPMN

Abstract: The paper tries to answer the question whether it is possible to use common definitions and approaches for process modeling for agile methodology. And although the answer is yes, there appear aspects of agile development, which must be respected and taken into account. Therefore another aim of this paper is to provide basic answers to what direction it might take the development of the process modeling so that it will truly respond to the needs and philosophy of agile software development including presentation of solution of concrete problem. This is by no means an exhaustive work, but rather an outline for future research that is planned by the author.

Keywords: SW development, agile methodology, process modeling, UML, EPC, BPMN

JEL Classification: L86

Úvod

Agilní metodiky patří mezi moderní způsoby přístupu k vývoji software. V mnoha projektech, pakliže jsou používány správně a rozumně, mohou výrazně zvýšit kvalitu a komfort vývoje, a to jak pro zákazníka, tak i pro samotný vývojový tým. Jedním ze zásadních omylů, které panují ve spojitosti s agilními metodikami vývoje SW, je představa, že agilní metodiky nelze procesně řídit, neboť v nich žádné pevné procesy neexistují. Takové pojetí je ovšem zásadně chybné a vede postupně v řetězci úvah a aktivit až k selháním projektů, ze kterých je následně viněna metodika samotná. Agilní metodiky se snaží o zjednodušení procesů, o odbourání zbytečné byrokracie, o urychlení rozhodovacích i výkonných procesů. To je bezesporu faktem.

Zjednodušení a zrychlení však nelze chápat fatalisticky. Abychom se z prostředí jasné a logické metodiky nedostali zpět do anarchie a neřízeného projektu, můžeme zjednodušovat a zrychlovat jen do takové míry, abychom neodbourali jasné procesní vymezení. Tedy agilní metodika se v žádném případě nesnaží zrušit definované procesy, ale pouze je zjednodušit. I nadále jsme schopni, pokud o to máme zájem, procesy definovat a (především) procesy modelovat tak, aby

byly „viditelné“ a snadno pochopitelné pro všechny členy vývojového týmu i další stakeholdery.

Je ovšem zcela zřejmé, že některé přístupy a metody nebudou fungovat či budou fungovat nedostatečně. A to právě proto, že odlišný přístup agilního vývoje software vyžaduje také odlišný přístup procesního řízení a modelování. Tento příspěvek se snaží definovat odlišnosti a nastínit základní aspekty „nového“ přístupu k procesům v prostředí agilního vývoje.

Vymezení agilních metodik vývoje software

Jako základ celého agilního přístupu je Agilní manifest (Agile Manifesto) [17]. Nejedná o žádný oficiální dokument, zákon, předpis. Jedná se o jakési prohlášení, které sepsala skupina mužů věnujících se vývoji software a kteří byli přesvědčeni, že tuto činnost lze provádět lépe než doposud. Tento přístup odpovídá agilnímu principu, kdy se snažíme minimalizovat byrokracii a další činnosti, které se ukazují jako ne zcela potřebné pro vývoj software.

Například tradiční metodiky se snaží o vyčerpávající dokumentaci v podstatě jakékoliv aktivity. Změna je strašákem – snaha o změnu je většinou bagatelizována, odmítána, a pokud se ukáže jako nevyhnutelná, je postoupena do zničujícího dlouhodobého martyria schvalování. A takto vlastně fungují všechny procesy – trvají dlouho, vyžadují značnou byrokracii, aktivizují často množství lidí.

Agilní přístup se tedy snaží o eliminaci toho, co je výše popsáno. O eliminace zbytečné byrokracie, zbytečného dokumentování každé sebeméně významné události či aktivity, o zjednodušení procesů změny [19]. Ale nikoliv eliminaci živelnou, eliminaci za každou cenu. To je velmi častý omyl mnoha příznivců agilního programování, zejména takových, kteří si přečetli několik statí a mají pocit, že nyní budou vyvíjet v podstatě v prostředí anarchie.

Nic takového, agilně řízené projekty mohou vykazovat (a pro správné a korektní výsledky by měly vykazovat) všechny znaky toho, že vámi využívaný softwarový proces je minimálně na třetí, lépe však na vyšší úrovni CMM. CMM neboli Capability Maturity Model je široce uznávaný a využívaný model měření úrovně vyspělosti jakéhokoliv procesu [20]. Není tedy důvod, proč jej nevyužívat při měření vyspělosti agilního procesu vývoje SW. CMM měří strukturovanost projektu, měří míru pořádku v procesu, nezabývá se interním uspořádáním procesů. Agilní přístup neznamená nepořádek, znamená stejně pořádek jako přístup tradiční. Jen znamená jiný přístup k pořádku, jiný přístup k rozumně chápaným procesům, jiný přístup k efektivitě.

Proces vývoje SW a specifika agilního vývoje

Procesem vývoje SW rozumíme všechny činnosti životního cyklu vývoje SW. Obecná shoda panuje ve vymezení počátku takového procesu. Počátkem procesu vývoje je okamžik, kdy dochází k zahájení spolupráce mezi vývojářem a zákazníkem, kdy dochází k jednoznačné shodně na tom, že software bude vyvíjen a kdy dochází ke tvorbě prvních vývojových artefaktů (zcela úmyslně neuvádím jakých, protože právě to je otázkou zvolené metodiky a přístupu). Shoda není dokonalá v případě ukončení procesu vývoje SW, nicméně je zřejmé, že tento konec procesu nenastává ukončením samotného primárního vývoje a odevzdáním díla zákazníkovi, neboť v dnešní době si nelze představit softwarový projekt, který by zároveň nedefinoval mnohaletou technickou podporu. Osobně tedy za konec daného procesu považuji

a) **U SW díla vyvíjeného na míru** – okamžik, kdy dané softwarové dílo je s konečnou platností vyřazeno z provozu a z užívání. Teprve v této chvíli můžeme říci, že práce na daném softwarovém díle jsou ukončeny.

b) **U SW díla vyvíjeného jako krabicový produkt** – okamžik, kdy je rozhodnuto o definitivním ukončení technické a jiné podpory daného softwarového díla. Teprve v tuto chvíli můžeme říci, že práce na daném díle jsou ukončeny.

V různých učebnicích a publikacích nalézáme na tento problém odlišné odpovědi, nicméně velmi často se ukazuje, že není rozpor v základním pochopení myšlenky, nýbrž v odlišném přístupu. Takto je tomu například u Vondráka [1], kde sice SW proces v publikaci končí předáním SW díla, nicméně tento fakt je způsoben pedagogickým záměrem vysvětlit pouze tuto část. Je nepochybné, že přístupy ve fázi vývojové a údržbové jsou diametrálně odlišné. V zásadním pojetí však rozpor nenastává.

Specifické pojetí procesu vývoje SW u agilních metodik

Pakliže se zaměříme na agilní metodiky vývoje SW, pak je na místě základní otázka – jakým způsobem se mění tradiční definice softwarového procesu, resp. procesu jako takového. Vondrák [2] definuje business proces jako „*po částech uspořádanou množina procedur a aktivit, které společně realizují podnikatelský nebo strategický cíl, obvykle v kontextu organizační struktury.*“

Tato definice je poměrně jednoduchá (přesto však výstižná) a lze oprávněně očekávat, že nebude zásadních rozporů mezi různými autory ve vztahu k takové definici. Otázka, která byla položena výše, se však týká skutečnosti, zda ve světě agilního vývoje lze takovou definici použít vzhledem k odlišnému pojetí. Ze zkušenosti docházím k přesvědčení, že na samotné definici není nutno nic měnit, pouze je nutné akceptovat jistou vágnost termínů v této definici se vyskytujících a přizpůsobit tomu způsob uvažování o procesech, potažmo pak ovšem také způsoby modelování. Jako vágní lze v agilním vývoji chápat zejména pojmy:

a) **(Podnikatelský nebo strategický) cíl** – agilní metodiky se vyznačují tím, že se výborně uplatňují právě v projektech, které nemají na začátku přesně definovaný cíl a které tento cíl definují průběžně, často metodou pokus omyl. Není tedy neobvyklé, že se již vytvořený produkt mnohokrát přepracovává a že se často zahazují i velké programové celky z toho důvodu, že se neukázaly jako šťastné řešení zadání zákazníka.

b) **Organizační struktura** – zatímco u tradičních metodik vývoje je organizační struktura jasně vymezená a z větší části pevně daná po celou dobu projektu, u agilních metodik tomu tak není. Je definováno mnohem méně rolí, činnosti rolí se překrývají, nejsou definovány jednoznačné manažerské kompetence, v mnohém splývá postavení vývojáře a zákazníka, neboť ten je často velmi pevně zapojen do vývojového týmu.

c) **Procedura / aktivita** – tradiční metodiky se snaží velmi precizně odpovídat na otázku – kdo a kdy má vykonat co a jak to má vykonat. Agilní metodiky toho nejsou schopné, snaží se pouze naznačit, že by ta či ona aktivita měla být provedena, protože je to zkrátka pro vývoj software nezbytné. Velmi často u definice aktivity splývají role, není definován jasný časový rámec.

Ukazuje se tedy, že budeme muset změnit některé aspekty přístupu k takovému procesu, ale také že budeme muset změnit některé aspekty modelování procesu samotného. Budeme muset využít přístupy, které budou respektovat odlišnosti. Jako dobrou zprávu můžeme ovšem uvést fakt, že základní pojetí procesu nám zůstává zachováno. Ostatně, jak o tom hovoří Wiegers [4], cílem vývoje musí být vždy uspokojení požadavků zákazníka. To a nic jiného je imperativem a měřítkem úspěšnosti projektu.K tomu, jak k požadavkům přistupujeme, se můžeme lišit, ale nemůžeme se lišit v principiálním cíli. Tento článek se snaží dát představu, jakým směrem se můžeme ubírat v úvahách o modifikacích tradičních metod procesního modelování, které se týká procesu vývoje SW, jde-li nám o agilní metodiky vývoje.

Specifika procesního modelování pro agilní metodiky

Procesní modelování jakožto disciplína vědy i praxe pochopitelně není omezena jen na oblast vývoje SW. Ukazuje se, že procesní modelování může být velmi užitečné v podstatě ve všech sférách lidské činnosti, neboť nám poskytuje možnost rychlého a přehledného náhledu na jednotlivé procesy. Takto jsme mohli popsat nepochybné přínosy procesního modelování v mnoha oblastech – jen namátkou lze vybrat regionální plánování [11] či zemědělství [12]. Bezpochyby lze význam procesního modelování vidět pro jakékoliv řídící úkoly, neboť jen dobré poznání procesů dá řídícím pracovníkům možnost kvalifikovaného rozhodnutí založeného na faktech, nikoliv pouze na domněnkách. Dobře vytvořené procesní modely dokonce umožňují poloautomatické či zcela automatické simulace, které umožňují v definovaných procesech nacházet problematická kritická místa a umožňují tak reengineering procesů [8]. Umění procesního modelování tak může sloužit v mnoha zdánlivě nesouvisejících oblastech.

Jestliže se pro potřeby tohoto příspěvku omezíme pouze na oblast vývoje software, pak můžeme hovořit o dvou možných cestách, jak využít procesní modelování. Jednak pro vytvoření tzv. modelu metaprocesu, tedy modelu samotného procesu vývoje software, což je naznačeno např. v [7], tak při samotném vývoji, kdy modelujeme konkrétní procesy našeho zákazníka, abychom následně byli schopni vyvinout software, který bude zákazníkovy procesy podporovat. To totiž není vůbec běžné – velmi často dochází k vývoji SW způsobem, který procesní modelování opomíjí. Důsledkem poté je systém, který může fungovat dobře, ale který není dobrý [3]. Není dobrý v tom smyslu, že nerespektuje proces zákazníka. Zákazník tak v důsledku musí změnit své procesy ne proto, že by se ukázaly jako špatné, ale proto, že software v jeho firmě „uvažuje jinak“. To je nejhorší možné řešení, kdy se člověk stává otrokem software.

V oblasti modelování procesů se v průběhu času vytvořilo několik metod, které mají svá specifika a své oblasti použití. Pochopitelně neexistuje jediná dokonalá metoda, jak modelovat procesy. Vždy záleží na možnostech a zkušenostech procesního analytika, na dostupných nástrojích, na komplexitě procesů, ne oblasti, v níž procesy modelujeme.

Tradiční metody procesního modelování

Není cílem tohoto příspěvku vyčerpávající přehled metod procesního modelování, který naleznete v publikacích, které byly využity i při zpracování tohoto příspěvku [1], [3], [5], [10].. Cílem je pouze nástin možností, které procesní analytik může využít. Ukazuje se, že pro potřeby agilního vývoje lze využívat kombinace jednotlivých metodik podle toho, které aspekty dané metodiky se ukážou jako prospěšné.

Patrně každému vývojáři je znám jazyk (či spíše grafická notace) UML – Unified Modeling Language. Tato grafická notace se stala standardem v oblasti modelování softwarových systémů. Přes veškeré rozšíření, oblíbenost a možnosti, které nám UML nabízí, se nejedná o metodiku vývoje, ale pouze a jen o grafický nástroj, který slouží k modelování softwarového systému. To ve výsledku vedlo k tomu, že se jazyk UML stal opravdu univerzálním nástrojem využívaným ve většině metodik vývoje SW, bez ohledu na to, zda náleží k metodikám tradičním či agilním. O samotném agilním vývoji s využitím notace UML pojednává například [6]. Nicméně u modelování procesů samotných jsme v poněkud odlišné situaci. Jazyk UML vznikal v době, kdy procesní modelování nebylo ještě běžnou součástí procesu vývoje SW, a tudíž není pro tuto oblast přizpůsoben. To, že je mnohdy využíván, je spíše důsledkem jeho široké rozšířenosti a objektivních a subjektivních překážek pro použití sofistikovanějších metodik modelování procesů. K tomu, aby UML bylo dobrou volbou, chybí několik podstatných prvků – chybí například časoví i jiné triggery, je problematická hierarchizace procesů a mnoho dalšího. UML se však vzhledem k rozšířenosti a oblibě stalo základem pro další, tentokrát již sofistikovanější metody [16] – ať už se jedná o vizuální podobu či přístup

k modelování procesů. Navíc se ukazuje, že pro modelování procesů v rámci agilního vývoje může být UML rovněž dobrým základem, který je nutno pouze doplnit o některé prvky známé z jiných metodik, případně o některé zcela nové přístupy.

Jednou z takových metod je metoda EPC (Event-driven process chain), případně eEPC (rozšíření původní metody, dnes pod pojmem EPC většinou rozumíme eEPC). Tato metoda využívá jako svou základní filosofii fakt, že i v běžném životě dochází ke sledu událostí a reakcí na tyto události. Jedná se o poměrně snadno pochopitelnou a snadno použitelnou metodu, navíc dobře podporovanou běžně dostupnými softwarovými nástroji – jako příklad lze uvést například Microsoft Visio. Problematická může být skutečnost, že EPC není příliš formálně ukotveno, a proto se může snadno stát, že náš model bude obsahovat deadlocky a další problematické partie. Práce Mendlinga [15] však ukazuje možnou cestu, jak problémy vyřešit a metodu EPC s úspěchem používat.

Patrně nejpoužívanější metodou současnosti je BPMN – Business Process Modeling Notation. Velkou výhodou této metody je podrobná, precizní, aktualizovaná a snadno dostupná dokumentace v podobě BPMN standardu [18]. Metoda BPMN vychází rovněž ze standardu UML, nicméně jej výrazně doplňuje právě o ty prvky, principy a přístupy, které v UML chybí. Objevují se dokonce snahy zapojit BPMN přímo do vývojového procesu, konkrétně pak jako nástroj vizualizace use case [14] – jedná se tedy o přímou ambici nasadit BPMN jako nástroj sloužící místo tradičních aktivitních diagramu – ty byly ostatně také přímým základem BPMN. Jako rozumně navržený a aktuální nástroj hodnotí metodu BPMN také Allweyer [13]. Důležitým aspektem BPMN je také existence jazyka BPEL (Business Process Execution Language) – při dodržení některých pravidel je možné procesy přepsat do jazyka pro jejich přímé vykonání v systémech [9].

Existují pochopitelně mnohé další metody procesního modelování, nicméně není účelné tyto v tomto příspěvku uvádět. Pouze by přinášely zvýšený rozsah, nikoliv ovšem zvýšenou informační hodnotu příspěvku. Základní charakteristiky popsaných metod naleznete v tabulce.

Tabulka 1 - Srovnání metod modelování [10]

Metoda	Použití	Nevýhody
UML diagramy	Strukturální diagramy, diagramy aktivit, diagramy interakcí	Komplexita modelovaných systémů neumožňuje vyčerpávající zachycení všech detailů. Je zde výrazná potřeba kompromisu, volba vhodné úrovně abstrakce. V UML chybí notace pro popis například hierarchie subprocesů, transakcí, výjimek, časových triggerů
BPMN	Obecně použitelná metoda pro popis sekvencí aktivit a dalších aspektů procesu, rovněž využitelná pro rozsáhlé procesy.	Neformálnost a z ní vyplývající nemožnost algoritmické verifikace modelovaného procesu – to je ovšem problém všech neformálních metod.
EPC	Obecně použitelná metoda pro popis sekvencí aktivit a dalších aspektů procesu, rovněž využitelná pro rozsáhlé procesy.	Notace není formálně definovaná, v procesu může být snadno namodelován deadlock a tudíž nebude možné dosáhnout konce procesu.

Specifika procesního modelu agilního vývoje

Jestliže hovoříme o agilním vývoji software a o agilním manifestu jako jeho stěžejním dokumentu, pak je nutné chápat jeho výklad. Samotný text je běžně dostupný na síti internet, a to dokonce i v českém znění. Již to, že byl proveden překlad do českého jazyka, svědčí o tom, že se jedná o přístup rozšířený a oblíbený. Ale výklad, tedy pochopení toho, co je vlastně cílem agilního přístupu. Pokusím se tedy o vlastní výklad toho, jak lze chápat agilní manifest, byť jsem si dobře vědom, že tento výklad nemůže být brán jako jediný možný.

1. **Jednotlivci a interakce před procesy a nástroji** – procesy a nástroje jsou důležité, nicméně pro agilní metodiky jsou důležitější jednotlivci, jejich potřeby, jejich vklad do projektu. Stejně tak interakce. Ve skutečnosti nástroje a procesy pouze slouží k tomu, aby ve výsledku uspokojily potřeby jednotlivců a aby se uskutečnily jednotlivé plánované interakce. To je první významný rozdíl oproti tradičnímu přístupu, který lpí na procesech často bez ohledu na to, zda lidé v projektu jsou spokojeni (to ovšem není správně). Procesy agilního přístupu jsou celkově plánovány jako jednodušší a minimalistické, ale zároveň ovšem plnící svůj primární účel.
2. **Fungující software před vyčerpávající dokumentací –** fungující software je primárním cílem, vyčerpávající dokumentace je spíše nástrojem, který k danému cíli provedu. A v agilních metodikách preferujeme fungující software. Pozor – to není důvod k tomu, abychom nedokumentovali. Agilní metodiky pouze preferují dokumentaci přirozenou, založenou na tom, že vývojář průběžně dokumentuje svou činnost současně s touto činností. Nikoliv to, aby tento vývojář často seděl hodiny nad dokumentací, která navíc musí mít předepsaný formát, využívat předepsané jazykové prostředky a být psána pomocí standardních šablon a formulářů.
3. **Spolupráce se zákazníkem před vyjednáváním o smlouvě** – různá smluvní jednání bývají často velmi komplikovaná, účastní se jich vedoucí pracovníci, účastní se jich právníci, musí se doladit nejmenší detaily. Často je pak výsledkem jen to, že sice máme precizní smlouvu, nicméně nemáme software. Proto agilní metodiky preferují rozumnou spolupráci před cizelováním dokonalých smluv, ústní komunikace je brána jako komunikace standardní. Opět tím není dotčena nutnost smluvních ujednání, jen nejsou tak protežovány a je preferována normální lidská komunikace, která umožní velmi rychle zahájit práce na samotném projektu.
4. **Reagování na změny před dodržováním plánu** – tradiční metodiky obvykle nadekretují plán a vyžadují jeho striktní dodržování. Nedodržení plánu pak znamená velký problém. Je však nutno brát v potaz často nízké povědomí zákazníka o oboru informačních technologií - často si vůbec neuvědomuje, co je možné a co nikoliv, případně to, jak bude vypadat výsledek. Změna je prostě nutná, pakliže má výsledek skutečně zákazníkovi sloužit. Jenže mnohdy tradiční přístup změny blokuje. Zcela jim zabránit nelze, nicméně vhodným nastavením složitého byrokratického postupu lze změny výrazně eliminovat.

Není však cílem příspěvku rozebrat agilní manifest, což bylo ostatně učiněno již mnohokrát. Co tedy z tohoto výkladu vyplývá pro oblast procesního řízení a v důsledku také procesního modelování? Základním specifikem procesního modelu agilního vývoje je fakt, že proces (a jeho model) stojí v pořadí důležitosti za lidmi v projektu. To ovšem neznamená, že proces není důležitý, což ostatně potvrzuje i Abrahamsson v [21]. Proces musí sloužit jednoznačně jako nástroj, který má lidem v projektu být nápomocen. Tomu musí odpovídat i příslušné modely, které musí popisovat reálný proces a které musí být snadno změnitelné v případě, že se proces ukáže jako nevhodný. Velké, složité a komplikované modely jsou proto nevhodné, už proto, že samotný proces by měl být co možná nejjednodušší.

Budeme tedy vyžadovat jednoduchou, snadno použitelnou metodiku. Snadno použitelná metodika pak bude taková, která umožní procesní modely snadno konstruovat, snadno

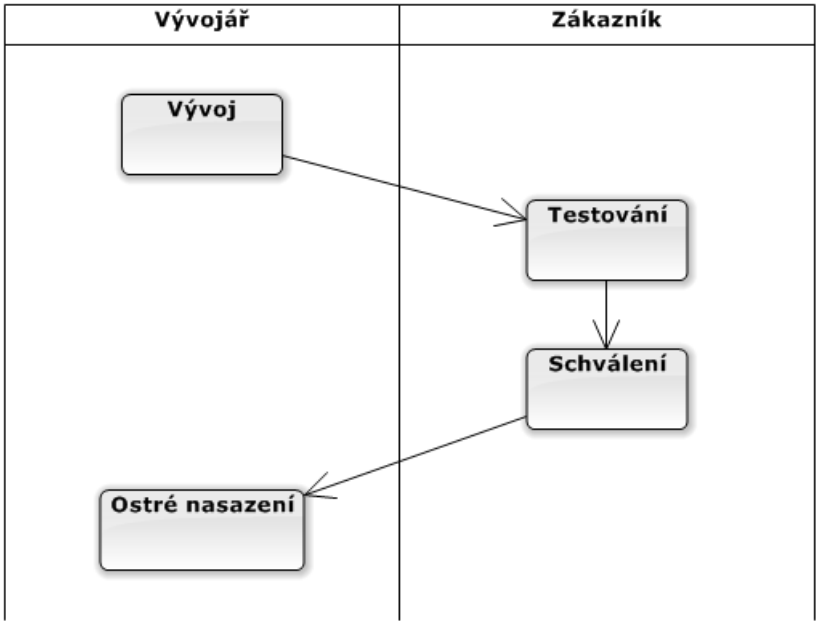
modifikovat, snadno číst, která umožní vytváření a editaci modelů v jednoduchým a běžně používaných nástrojích. Je nutné pochopit, že v mnoha ohledech můžeme jít proti standardnímu proudu procesního modelování, které zavádí rigorózní a pokud možno dobře formalizované metody, které využívá sofistikované nástroje, které mají sice mnoho funkcí, ale které vyžadují často velké počáteční investice finanční i lidské. Lze konstatovat, že agilní metodiky zůstávají agilními i v oblasti procesního modelování (a obecně chápáno v oblasti procesního řízení jako takového).

Modelování kolaborace v prostředí agilních metodik

Jako ukázku přístupu, který můžeme využívat při modelování procesů agilního vývoje (hovořím o popisu samotného procesu vývoje SW, ne o procesech zákazníka – tam pro takové rozhodování musíte vědět, zda zákazníkovy vlastní procesy jsou řízeny agilně či nikoliv) jsem zvolil modelování **kolaborace**, čímž rozumím aktivity, které jsou vykonávány v součinnosti více rolí. Případně jsou vykonávány vágně definovanými rolemi. To je totiž v agilním vývoji naprosto běžné a je velmi obtížné popsat to s použitím tradičních paradigmat procesního modelování. Není přitom problém v symbolech a notacích, ale ve specifikacích použití. Ty totiž počítají s tím, aktivita či procedura je provedena tou či onou rolí. A takto v agilním vývoji často nelze uvažovat, protože agilní vývoj předpokládá daleko užší spolupráci jednotlivých rolí, případě běžně chápané role v tradičním pojetí vývoje jsou v agilním vývoji koncentrovány do jedné role fyzické.

Pro ukázku principu volím zcela záměrně UML. Záměrně proto, že tato metoda modelování je velmi jednoduchá a to, co pro reálné modelování se ukazuje jako velmi nešťastné, tedy absence složitějších konstrukcí, se pro pedagogické účely tohoto příspěvku ukazuje naopak jako velmi přínosné. V praxi bychom použili nejspíše metodu EPC či ještě lépe BPMN (zde je možné výstupy této úvahy přenést téměř doslovně), nicméně pro iniciální úvahu o **modelování kolaborativních aktivit** bude diagram UML zcela postačující.

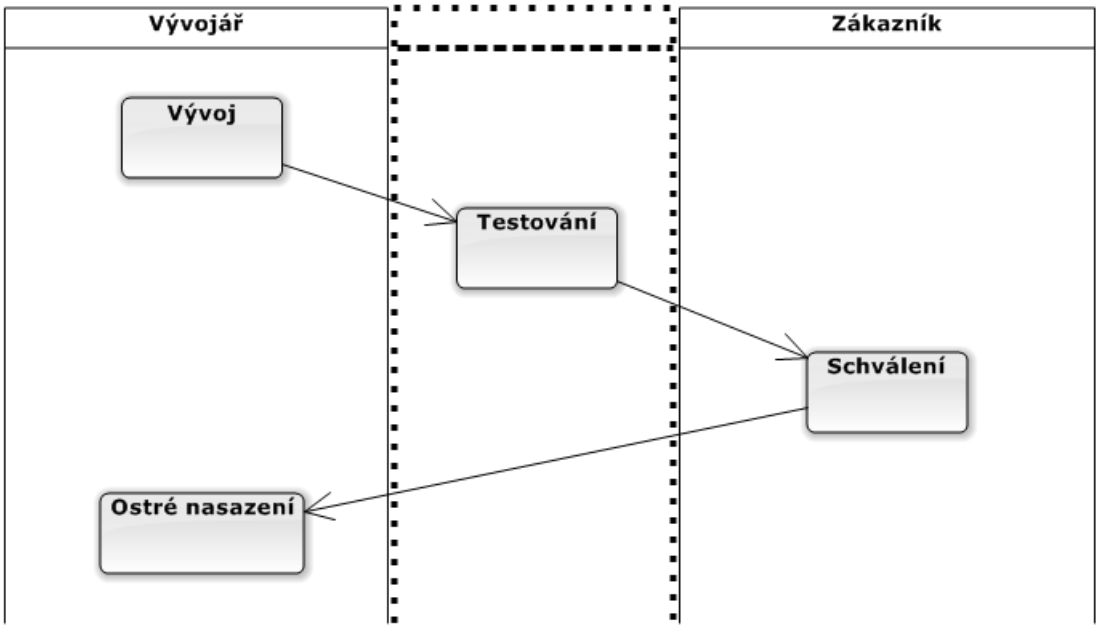
Jako typickou ukázku si můžeme představit jednoduchý model procesu, který je zobrazen na obr. 1. Z pochopitelných pedagogických důvodů je model maximálně zjednodušen, aby optimálně vynikl řešený problém. V tradičním pojetí je jasné, kdo provádí jednotlivé aktivity. V tomto případě je určeno, že aktivitu *Vývoj* provede osoba v roli *Vývojář*, aby následně byl výsledek předán k aktivitě *Testování* a následně k aktivitě *Schválení*. Tyto dvě aktivity má na starosti osoba v roli *Zákazník*. Po schválení je produkt předán opět osobě v roli *Vývojář* za účelem provedení aktivity *Ostré nasazení*. Jak již bylo uvedeno, proces byl zjednodušen, proto neuvažujeme možnost neschválení či jiných nelinearit procesu. Hlavním cílem je ukázat jednoznačnost **přiřazení role – aktivita**. Jednotlivé role mají vymezenou svou **swimlane** a u jednotlivých aktivit je v procesním modelu jednoznačně zřejmé, ke které roli náleží – to je definováno umístěním aktivity v příslušné swimlane.



Obrázek 1 - Běžný procesní model u ostře vymezených rolí a kompetencí tradiční metodiky. Zdroj: vlastní zpracování

Jestliže se nyní přesuneme do jiného procesu vývoje, tentokrát řízeného agilním způsobem, může nám tento model připadat nevhodný a nepopisující realitu. V agilním vývoji software se totiž tiše předpokládá úzká **kolaborace**, tzn. spolupráce mezi zákazníkem a vývojářem, u mnoha metodik dokonce tak úzké, že je zcela smazána hranice mezi vývojářem a zákazníkem, neboť zákazník se stává členem vývojového týmu (např. metodika Extrémní programování). V jiných metodikách není spolupráce tak úzká, nicméně i zde se předpokládá, že zákazník bude alespoň občas úzce spolupracovat s vývojovým týmem - ať už na pracovišti vývojářů nebo na pracovišti zákazníka (např. metodika SCRUM).

Pro ilustraci předpokládáme, že aktivita *Testování* je prováděna ve spolupráci osoby v roli *Vývojář* a osoby v roli *Zákazník*. Nyní se tedy objevuje otázka, jak tuto skutečnost v diagramu vyznačit. Původní model znázorněný diagramem na obrázku 1 je v takovém případě evidentně nesprávný. Řešením není ani přesunutí aktivity *Testování* do swimlane osoby v roli *Vývojář*. Takový přesun by byl stejně mylný. Jako další řešení by se nabízelo přidat aktivitu *Testování* k oběma rolím a vložit ekvivalent AND spojky. V takovém případě bychom v procesu pokračovali až poté, co by aktivitu *Testování* dokončily osoby v obou rolích. Ani takové řešení nevystihuje situaci – v agilních metodikách neprovádí samostatnou aktivitu zákazník a vývojář, testování se provádí jako jedna aktivita, na které se podílí obě role. Je tedy nutné, aby model tuto skutečnost respektovat a aby se v něm nacházela jediná aktivita *Testování*, a to takovým způsobem, aby bylo zřejmé, že se na ní podílí jak osoba v roli *Vývojář*, tak osoba v roli *Zákazník*. To bohužel není ve standardním pojetí procesního modelování možné, bez ohledu na to, zda použijeme metodu BPMN či UML.



Obrázek 2 – Procesní model s kolaborativní swimlane pro potřeby agilní metodiky. Zdroj: vlastní zpracování

Potřebujeme možnost definovat, že určitá činnost nesouvisí jen s jednou konkrétní rolí, ale že tato činnost spojena s více rolemi, v tomto případě se dvěma rolemi. Zvolím tedy řešení spočívající ve vytvoření DSML – domain specific modeling language, tedy doplním standardní specifikaci metody procesního modelování o další prvek. Tímto prvkem je speciální swimlane, kterou nazvu **kolaborativní swimlane**. Tato swimlane, kterou vidíte na obr. 2, nedefinuje další roli, ale pouze (a vzhledem ke grafickému odlišení je snadno rozpoznatelná) vymezuje v diagramu prostor pro činnosti, které jsou společné pro role definované ve dvou sousedních swimlane.

Kolaborativní swimlane je tedy velmi jednoduchý způsob, jak vyřešit problém kolaborativních aktivit, tedy aktivit, na kterých se podílí více než jedna role. Přitom se jedná o způsob, který je snadno použitelný i v rámci stávajícího software pro modelování procesů – fakticky se jedná o běžnou swimlane, kterou pouze graficky odlišíme, což pro většinu používaných programů není žádný problém. Je k diskusi, zda označit kolaborativní swimlane právě tak, jak jsem učinil v modelovém případě. Existuje pochopitelně více možností – například odlišným podbarvením. Kolaborativní swimlane lze rovněž využít při kolaboraci více rolí, v takovém případě již nebude kolaborativní swimlane mezi dotyčnými swimlane jednotlivých rolí, ale bude umístěna jiným způsobem a bude také náležitě označena v popisku.

Závěr

Tento článek je skromným příspěvkem do diskuse na téma specifik procesního přístupu a procesního modelování v rámci agilního řízení softwarových projektů. Pokusil jsem se předložit drobnou rešerši stavu poznání v této oblasti a definovat příslušná specifika platná pro agilní metodiky řízení softwarových projektů. A zároveň nastínit řešení jednoho konkrétního drobného problému. V každém případě kolaborativní swimlane představuje drobný krok k vytvoření metodiky vhodné pro agilní metodiky. Taková metodika pochopitelně bude založena na metodikách již existujících a bude spíše doplněna a rozšířena o principy potřebné právě u agilních metodik. Jedná se pochopitelně o potřebnou oblast a jedná se o výzvu pro další výzkum.

Reference

Vondrák I., Úvod do softwarového inženýrství, skripta VŠB-TU Ostrava, 2002

Vondrák I., Metody byznys modelování, skripta VŠB-TU Ostrava, 2004

Myslín Josef: Business modelování, skripta pro výuku stejnojmenného předmětu na VŠMIEP, 2012, ISBN: 978-80-86847-61-0

Wiegers, K.: Požadavky na software

Myslín, J.: Requirements Engineering, In: Doktorandské dny 2011. Praha: Česká technika - nakladatelství ČVUT, 2011, s. 153-162. ISBN 978-80-01-04907-5.

Ambler, S: The Object Primer 3rd Edition, Agile Model Driven Development with UML 2.0, Cambridge University Press, 2004

Merunka V.: Objektové modelování, Praha: Alfa Nakladatelství, s.r.o, 2008, ISBN: 978-80-87197-04-2

Gunasekaran, A., Kobu, B.: Modelling and analysis of business process reengineering. International Journal of Production Research 40(11), 2521-2546 (2010)

Ouyang, C., Dumas, M., van der Aalst, W. M. P., Hofstede, A. H. M.: From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way. ACM Transactions on Software Engineering and Methodology 19(1) (2009)

Rývová, A., Myslín, J.: Requirements Engineering, In: Objekty 2011. Žilina: Žilinská univerzita v Žiline, Fakulta riadenia a informatiky, 2011, vol. 1, p. 137-146. ISBN 978-80-554-0432-5.

Merunka, V., Myslín, J., Merunková, I.: Regional management - modeling and simulation approach, In: 5th International Conference on Information and Communication Technologies in Agriculture, Food and Environment. Thessaloniki: HAICTA (Hellenic Association of Information and Communication Technology in Agriculture, Food and Environment), 2011, p. 531-548. ISBN 978-960-89024-3-5.

Merunka, V., Myslín, J. - Unal, B.: Formal Techniques of Data Structure Design in Modern Database Systems, In: ITAFFE 2010, Samsun. Samsun: Ondokuz University, 2010, vol. 1, p. 15-27. ISBN 978-960-8029-45-3.

Allweyer, T.: Unternehmen als Process Engine? Möglichkeiten und Grenzen mit BPMN. In : BPMN, Potsdam, 2010

Lübke, D., Schneider, K., Weidlich, M.: Visualizing Use Case Sets as BPMN Processes. In Society, I., ed. : Proceedings of the Third International Workshop on Requirements Engineering Visualization, Barcelona, 2006

Mendling, J.: Detection and Prediction of Errors in EPC Business Process Models Disertation edn. Institute of Information Systems and New Media Vienna University of Economics and Business Administration, Vienna (2007)

Loos, P., Allweyer, T.: Process Orientation and Object-Orientation - An Approach for Integrating UML and Event-Driven Process Chains. Institut für Wirtschaftsinformatik University of Saarland, Saarbrücken (1998)

Agilní manifest [online], http://agilemanifesto.org/iso/cs/, citováno 11.12.2015

Referenční příručka BPMN online, http://www.omg.org/spec/BPMN/2.0/, převzato dne 11. 11. 2015

Sommerville I.: Software Engineering, Pearson; 10 edition (April 3, 2015), ISBN-13: 978-0133943030

Paulk M., Curtis B., Chrissis M.B., Weber C.V.: Capability Maturity Model for Software, Software Engineering Institute, Carnegie Mellon University, 1993

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile Software Development Methods: Review and Analysis. Oulu, Finland, VTT Publications 478, 2002., ISBN: 951-38-6010-8

Ing. Josef Myslín, Katedra aplikované informatiky Vysoké školy podnikání a práva, Vltavská 585/14, 150 00 Praha 5,e-mail: josef.myslin@vspp.cz