Semestrální práce ke kurzu
4IT421 Zlepšování procesů budování IS

Téma: Pair Programming Is No Panacea

Semestr	ZS 2016/2017
Autoři	Michaela Mia Machová (xmacm90), Matěj Kučera (xkucm46), Miroslav Novotný (xnovm174)
Datum odevzdání	18. 12. 2016

Abstrakt

Cílem této semestrální práce je seznámit čtenáře s přístupem zvaným párové programování, které je jednou ze základních praktik agilní metodiky - extrémního programování. Práce si v prvé řadě klade za cíl definovat základní pojmy a principy, které se vyskytují v souvislosti s párovým programováním a dále přiblížit čtenáři kontext a souvislosti, s nimiž je tato problematika spjata. Dalším cílem je přiblížení výhod a zejména pak úskalí, které mohou souviset s programováním v páru.

Klíčová slova

Extrémní programování, XP, párové programování, párování, programování v páru, výhody párového programování, úskalí párového programování.

Obsah

1 Úvod

Párové programování je termín, se kterým se mnoho programátorů setká pouze v rámci otázek k nějaké univerzitní zkoušce. Jedná se však o metodiku, která je velice dobře popsaná mnoha zdroji. Nicméně tyto zdroje velice často uvádějí pouze velice lákavá pozitiva, která může párové programování přinášet. Proč tedy není párové programování hojně rozšířené, např. i do školní výuky nebo mezi startupové týmy? Odpověď na tuto otázku chceme přinést právě v této semestrální práci.

Cílem této semestrální práce tedy je, jak již bylo výše naznačeno, seznámit případného čtenáře s pojmem párového programování a se souvislostmi, ve kterých se tento pojem, resp. metodika samotná, objevuje, a dále provést rešerši zdrojového článku *Pair programming is no panacea* (HIGBEE, 2016), který hovoří o zmíněné metodice zejména v negativních slovech. Tuto rešerši pak doplníme o informace z dalších zdrojů a rozšíříme tak kontext, ve kterém se může čtenář rozhodnout, zda je párové programování dobrá metodika pro jeho práci.

Samotná práce je členěna do několika kapitol, přičemž první tři obsahové kapitoly seznámí čtenáře s prostředím, ve kterém se pohybujeme (pojmy, historie a principy párového programování) a následující dvě představí jeho výhody a následně i úskalí, na které je třeba brát zřetel.

Vzhledem k tomu, že ve velkém množství textů, pojednávajících o párovém programování, je tato metodologie spíše doporučována a vyzdvihována jako univerzálně vhodná metoda, zaměříme se spíše na její negativa. Ke splnění cíle práce se pokusíme přispět také vlastními zkušenostmi, které jsme nasbírali v praxi - porovnáme tedy subjektivní pocity z využití párového programování.

Na konci práce je uveden seznam informačních zdrojů, ze kterých čerpáme, přičemž hlavním zdrojem je zmiňovaný článek *Pair programming is no panacea* (HIGBEE, 2016).

2 Seznámení s pojmy

Pro porozumění tomu, co znamená párové programování, je nejprve nutné zařadit tuto metodiku do kontextu a definovat základní pojmy, které jsou v souvislosti s problematikou párového programování běžně využívány.

2.1 Extrémní programování

Extrémní programování (rovněž známé pod zkratkou XP), je jednou z mnoha agilních metodik vývoje softwaru. XP je založeno na 4 základních hodnotách - jednoduchosti (je programováno pouze to, co je nezbytně nutné ke splnění aktuálních požadavků a co nejjednodušším možným způsobem), odvaze pouštět se do změn (a zahodit tak například celodenní práci nebo stovky řádků zdrojového kódu), komunikaci se všemi zainteresovanými osobami a zpětné vazbě (nejčastěji ve formě testování) (C2 WIKI, 2014).

Svůj název získala tato metodika díky skutečnosti, že tradiční a běžně osvědčené činnosti a principy dovádí do extrémů. Když se například osvědčí revidování zdrojového kódu, dle extrémního programování budeme neustále revidovat. Jestliže se osvědčuje testování, budou všichni a všechno neustále testovat. Pokud se osvědčí jednoduchost, integrace, krátké iterace nebo případně návrh budeme neustále zjednodušovat, integrovat, zkracovat iterace, navrhovat… (KADLEC, 2003)

Výsledkem této metodiky, je díky společné práci celého týmu, jednoduchým postupům a všudy přítomné zpětné vazbě, mnohdy stabilní, produktivní a velmi rychlé řešení, které se dokáže flexibilně přizpůsobovat nejasnému nebo rychle se měnícímu zadání (WELLS, ©2013)

Extrémní programování zavádí několik základních praktik (JEFFRIES, ©2016). Konkrétními příklady těchto praktik jsou mimo jiné:

1) Psaní testů před samotným kódováním.
2) Refaktorizace kódu - úprava zdrojových kódů, bez vlivu na funkčnost softwaru, ale s dopadem na efektivitu a rychlost výsledné aplikace (například odstranění duplicit v kódu)
3) Kontinuální integrace - systém je po celou dobu plně integrován. Sestavení může probíhat až několikrát denně.
4) Dodržování programovacích standardů - celý tým dodržuje společný standard kódování, což má za následek srozumitelnost kódu pro celý tým a efektivnost práce. Navíc je podporován princip společného vlastnictví kódu - napsaný kód je všech, kdokoliv může měnit cokoliv.
5) Párové programování.

2.2 Párové programování

Párové programování (také známé jako párování nebo programování v páru) je, jak již vyplývá z předchozího odstavce, jednou ze základních praktik extrémního programování. Podstata párového programování spočívá v tom, že veškerý zdrojový kód, je vždy psán společně, právě dvěma programátory (WELLS, ©2013; AGILE ALLIANCE, ©2015). Tento postup zajišťuje, že veškerý kód je revidován a sledován dalším programátorem (na rozdíl od individuálního kódování). Oba programátoři tak spolu mohou neustále komunikovat, sdělovat si své postřehy, záměry, sdílet nápady a znalosti, čímž získávají vzájemnou a velmi rychlou zpětnou vazbu. Díky rozdílným znalostem a dovednostem navíc vytvářejí prostor pro sdílení těchto vědomostí (JEFFRIES, ©2016).

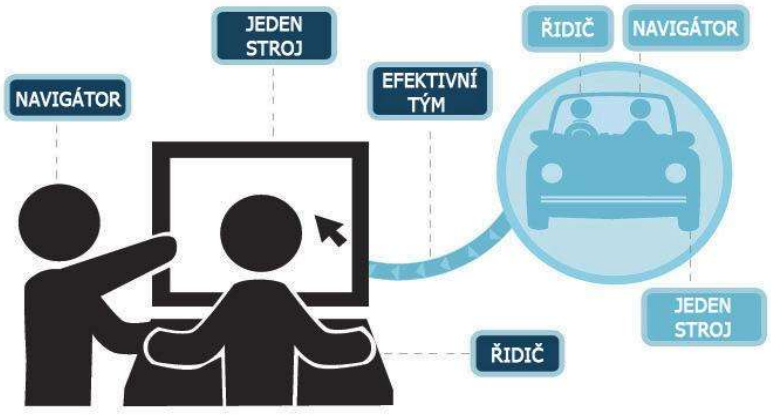
Párové programování typicky probíhá tak, že dva programátoři sedí bok po boku u jedné počítačové stanice, sdílejí jeden monitor, jeden zdrojový kód a jeden úkol. Myšlenkou programování v páru je spolupráce dvou rozdílných lidí, s odlišnými znalostmi, na stejném úkolu, se stejným cílem (WELLS, ©2013; AGILE ALLIANCE, ©2015).

Při párování je možné definovat různé druhy párů v závislosti jak na znalostech programátorů, tak i v závislosti na typu jejich osobnosti. Příkladem rozdílnosti párů jsou například: expert-expert, expert-nováček, nováček-nováček, extrovert-extrovert, introvert-introvert a podobně (WILLIAMS, a další, 2002). Párové programování také nabízí, vedle tradičního přístupu zvaného řidič-navigátor, několik dalších možných technik a přístupů, jako je ping-pong párování, pár typu implementátor-architekt, hlava-ruce a další (KNESL, 2014).

2.2.1 Řidič a navigátor

Základní pojmy, které se v souvislosti s metodikou párového programování běžně používají, jsou řidič a navigátor. Oběma programátorům, kteří tvoří pár, je přiřazena jedna role - buď role řidiče anebo navigátora, jejich úlohu velmi zjednodušeně a názorně zobrazuje Obrázek 1. Za řidiče je označován ten, který má na starost ovládání (v tomto případě myš a klávesnici) a je tedy zodpovědný za psaní kódu. Na druhou stranu navigátor přemýšlí nad psaným kódem, sleduje optimálnost a efektivnost řešení.

Každý pár má tedy vlastní vnitřní strukturu. Programátora - řidiče, který v daném momentě pracuje s klávesnicí, řeší konkrétní implementaci, přemýšlí pouze nad nejlepším způsobem implementace jedné konkrétního části, která je zrovna programována. Na druhou stranu jeho kolega - navigátor, sleduje práci řidiče, přemýšlí nad souvislostmi a optimálností aktuálního řešení, hledá potenciální problémy, chyby a zároveň se neustále snaží vymýšlet lepší řešení. Navigátor tedy na rozdíl od řidiče uvažuje v širším kontextu a s větším odstupem, zároveň přemýšlí s ohledem na celkovou globální strategii a pokládá si otázky typu: bude řešení ve výsledku fungovat? Není navrhované/implementované řešení příliš a zbytečně složité? Je řešení efektivní? Jak by to šlo udělat lépe? (AGILE ALLIANCE, ©2015; RAYGUN.COM, 2015)



Obrázek 1 - Role řidiče a navigátora (Zdroj: autorka podle (BAIN, ©2015))

Role řidiče a programátora se tedy kromě rozdílných činností, které mají na starost, liší i v rozdílnosti jejich pohledů na programovaný úkol. Zatímco řidič má na zadaný úkol mikro pohled, navigátor sleduje problém z makro pohledu (BAIN, ©2015).

3 Historie párového programování

Lidé využívají párového programování v různých formách, již po několik desetiletí. Za nejdůležitější historické milníky, jsou dle knihy Pair Programming Illuminated (WILLIAMS, a další, 2002) a podle portálu Agile Alliance (AGILE ALLIANCE, ©2015) považovány především následující události.

Vůbec nejstarší zaznamenaná zmínka o párovém programování pochází z 50. let 20. století, kdy dva programátoři Bill Wright a Fred Brooks, vyzkoušeli programovat v páru, přičemž tímto způsobem napsali zhruba 1500 řádků zdrojového kódu, který na první spuštění fungoval.

Dalším milníkem je událost, která se stala na počátku 70. let 20. století, kdy zaznamenává a využívá párového programování Dick Gabriel, který je mimo jiné známý vymyšlením Common Lisp[1] a představením pojetí vzorů a jazykových vzorů v softwarové komunitě.

Začátkem 80. let 20. století, použil Larry Constantine termín - dynamické duo (dynamic duo). Larry Constantine, který je autorem více než 150 odborných článků a 16 knih, poznamenal, že ve společnosti Whitesmiths, Ltd. pozoruje rychlejší a zároveň bezchybnější produkci zdrojových kódů. Toto zvýšení produkce a efektivity je dle něj způsobeno tím, že na kódu pracují dvě chytré hlavy zároveň, které spolu mohou neustále vést dialog. Ve svých pracích navíc došel k závěru, že práce dvou programátorů v týmu vede k větší účinnosti a lepší kvalitě.

V roce 1993 byla vydána studie “Výhody kolaborace/spolupráce studentů programátorů”, která je jednou z prvotních studií, jenž poukazuje na výhody párování za účelem programování specifických úloh.

[1] Rozšíření funkcionálního programovacího jazyku zvaného Lisp

Na základě výzkumu, který byl proveden v rámci Pasterova projektu v roce 1995, uveřejnil James Coplien, který je jedním z nejvlivnějších lidí v problematice softwarových vzorů, publikaci s názvem Vývoj v párech (Developing in Pairs). Autor v díle uvádí, že lidé někdy potřebují poradit s řešením problémů a že některé problémy jsou větší než jednotlivec. Řešením takovýchto problému je párování dvou vzájemně kompatibilních lidí, kteří budou pracovat spolu a tím budou schopni společně vyprodukovat mnohem více, než by byli schopni vytvořit samostatně. James Coplien zároveň uvádí, že díky párování je menší pravděpodobnost programátorské slepoty než je tomu u vývojáře, pracujícího samostatně.

Největší a zároveň nejznámější skupina párových programátorů je součástí skupiny programátorů, kteří následují metodiky Extrémní programování. Za vývojem metodiky XP stojí Kent Beck, který se skupinou dalších lidí pracoval na jejím vytvoření počátkem 90. let. V rámci této metodiky byly představeny možné výhody párového programování.

V roce 2000 byla vydána kniha “Pair Programming Illuminated”. Jejími autory jsou Laurie Williams a Robert Kessler. Jedná se o první knihu, která je výhradně věnována teorii, diskusím, praxi a různým studiím.

Kolem roku 2000 byly při vysvětlování párového programování představeny definice rolí řidiče a navigátora, tyto role však vyvolávaly debaty a spekulace, které lze pozorovat například v reakci Sallyanna Bryanta, který na tuto skutečnost poukázal článkem nesoucím název “Párové programování a tajemná role navigátora”.

V roce 2003, napsal anonymní uživatel článek (na portál Wiki C2), pojednávající o Ping-Pong programování, jakožto populární variantě, která slučuje principy párového programování a vývoje řízeného testy.

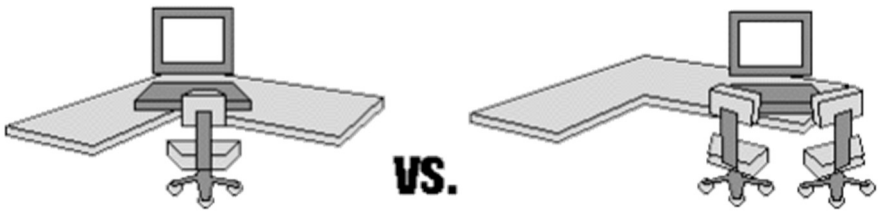
S párovým programováním jsou z historického hlediska dále spjaty tato jména: John Von Neummann, Fred Brooks, Jerry Weinberg, Richar Gabriel nebo například Edsger Dijkstra, kteří párové programování propagovali, vyzdvihovali a vštěpovali komunitě.

4 Principy párového programování

Jak již bylo uvedeno výše, základní myšlenkou párového programování je, že všechen zdrojový kód je psán ve dvojici. V rámci této dvojice má každý z programátorů přiřazenou svoji roli, kterou si však frekventovaně prohazují. Kromě měnění rolí v rámci týmu se během dne obměňuje i složení těchto dvojic. Dva lidé tak spolu obvykle nepárují dlouhodobě - maximální doba jednoho párování trvá cca 5 hodin. Týmy mohou být měněny i několikrát denně. Párování by rovněž nemělo trvat příliš dlouho (obvykle se uvádí, že by nemělo trvat déle než ¾ pracovního dne), jakmile tedy začnou být programátoři unavení, mělo by se v párování pokračovat následující pracovní den (KNESL, 2014).

Je důležité si uvědomit, že párové programování není instruktáží. Mezi programátory v týmu tedy není vztah učitel-student, ale jedná se o dva sobě si rovný kolegy, kteří pracují na společném úkolu (i když jeden z nich má více zkušeností nebo znalostí než druhý). Ačkoli se nejedná a nemá jednat o mentoring, v praxi jsou běžně sestavovány páry junior-senior vývojář a mentoring se stává víceméně vedlejším efektem této kooperace. Párové programování je sociální dovedností, kterou je nutné se postupně, časem, naučit (WELLS, ©2013).

Dalším důležitým faktorem je pohodlí pracoviště, které by mělo poskytovat dostatečný komfort oběma programátorům a nemělo by klást zbytečné technické překážky. Prostředí, ve kterém programátoři, tvořící jeden pár, pracují, by mělo být pohodlné, oba by měli mít kolem sebe dostatek místa, počítač by měl být umístěný tak, aby na něj oba jednoduše a dobře viděli, ale i dosáhli a to vše bez nutnosti pohybu židlemi a podobně. Místnost, ve které probíhá párové programování, musí mít tedy přizpůsobený jak nábytek, tak je nutné přizpůsobit i počítačové prostředí - například velikost fontu. Rozdílnost v rozmístění kanceláře velmi zjednodušeně zobrazuje Obrázek 2, kde vlevo je zobrazeno rozložení pracovního místa programátora, který pracuje individuálně a vpravo je rozmístění, které je upraveno pro párové programování (C2 WIKI, 2014; BAIN, ©2015; AGILE ALLIANCE, ©2015).



Obrázek 2 - Příklad umístění počítače, na kterém probíhá párové programování. (Zdroj: (C2 WIKI, 2014))

5 Výhody párového programování

V této kapitole jsou představeny hlavní výhody párového programování. Jsou uváděné napříč všemi příručkami a články o extrémním programování, a pokud se obvykle firma rozhodne párové programování využívat, vedou ji k tomu následující benefity. Tato kapitola pramení z mnoha zdrojů (jak literárních, tak i našich dosavadních zkušeností), v nichž se zde uvedené informace opakují mnoha různými interpretacemi (BAIN, 2015; BRECK-MCKYE, 2016).

5.1 Kvalitnější návrh architektury a kódu

Hlavním principem a výhodou párového programování je to, že výsledný návrh aplikace a i samotný kód (jeho čitelnost, optimalizace a použitelnost) je obvykle kvalitnější než u aplikace vytvářené jedním programátorem. Byť je možné stejné kvality dosáhnout i individuální prací programátora (např. refactoringem nebo podrobnějším návrhem vyžadujícím časově náročnou analýzu), párové programování přináší tuto kvalitu už v momentě psaní kódu.

Toto je žádané například v případě, kdy je potřeba výsledný kód spustit na první běh (např. SQL procedury spuštěné nad živou databází) nebo v případě, kdy se jedná o zcela klíčovou část aplikace, která musí být navržena s velkou přesností a opatrností, neboť na ni bude navazovat vyšší množství dalších částí aplikace a jejíž změna by v budoucnu představovala problém.

Tím, že u jednoho počítače (resp. jedné klávesnice) sedí dva lidé a sdílejí společný cíl, doplňují se vzájemně svými dovednostmi, zkušenostmi a aktuální postřehy a dosahují tím výše zmíněných výhod párového programování.

Vedle lepší “funkčnosti” kódu, kvalitnějšího návrhu, je výsledkem párového programování i kód, který více odpovídá programátorským standardům a dobrým zvykům. Pokud programuje sólový programátor, nemusí mu tolik vadit nevhodně pojmenovaná proměnná, protože “přece ví, co je zač”. Pokud programuje ve dvojici a jeho kolega se ho otáže např. na význam nějakého pojmenování, je to signál k tomu, že pojmenování není vhodné a nejspíše zvolí odlišné, které je více popisné. Totéž platí i o dodržování konvencí, kdy je díky neustálému dohledu od svého kolegy, řidič nucen dodržovat patřičné programátorské zvyky.

5.2 Okamžitá zpětná vazba

Každý programátor potřebuje, stejně jako většina ostatních profesí, zpětnou vazbu ke svému dílu - tedy programovému kódu. Dělá ten kód to, co má? Je správně ošetřený? Nemůže vést k neočekávanému výsledku? Je čitelný, dobře srozumitelný pro ostatní kolegy? Na všechny tyto otázky obvykle programátor dostane odpověď pouze tehdy, pokud se o svůj kód podělí s ostatními kolegy.

Pokud tak učiní jen při odevzdání projektu na nějakém týmovém úložišti, může být na zpětnou vazbu pozdě (na jeho řešení již navázaly jiní programátoři a změna může být složitá) anebo nemusí být tak přesná a důkladná, pokud jeho kód někdo jen “proletí očima”.

Během párového programování je tato zpětná vazba takřka okamžitá. Navigátor hledí svému kolegovi přes rameno a vidí, že píše něco, co je špatně čitelné, nesrozumitelné, nevalidní nebo chybné. Nerozumí tomu, zeptá se na to, nechá si to vysvětlit a doporučí řidičovi např. přepsání nebo alespoň vhodné okomentování. Stejně tak vidí navigátor program v širších souvislostech a může přemýšlet nad tím, jestli správně navazuje na práci ostatních kolegů. Zda nedojde k nějakému konfliktu funkcionality apod.

5.3 Rychlejší řešení záseků

Říká se, že programování je jen řešením menších či větších záseků. Velkou část času programátor tráví tím (pokud zrovna nepíše dokumentaci), že uvažuje, jak by vyřešil nějaký problém. Ať už přemýšlí vlastní hlavou, vyhledává informace na Googlu nebo se radí s kolegy, uvažování nad řešeným problémem zabírá více času, než samotné psaní kódu.

Řešení každého problému jde vždy lépe ve dvou a totéž platí i u párového programování. Pokud jsou na vymýšlení, uvažování a implementaci řešení dva lidé (přičemž oba mohou přispět různými zkušenostmi, které během své kariéry nasbírali), bude vyřešení záseku nejspíše rychlejší a efektivnější. Zatímco jeden píše, druhý může vyhledávat.

Často uváděným benefitem párového programování je i to, že zásekům lze předcházet. K záseku se programátor snadno dostane špatným návrhem (nikoliv nutně návrhem celé aplikace, ale třeba jen několika-řádkového algoritmu). V případě, že druhý programátor poskytuje okamžitou zpětnou vazbu, poradí svému kolegovi, že se ubírá směrem, ve kterém bude muset řešit závažnější problém.

5.4 Snadnější udržení “flow”

Během práce působí na programátora mnoho externích vlivů, které ho mohou rozptýlit od práce. Mezi nejčastější patří třeba elektronická pošta (přijde email, na který je potřeba odpovědět) nebo zpráva na sociálních sítích (opět je třeba odpovědět). Pokud člověk programuje sám, není většinou problém na chvíli vývojové prostředí minimalizovat, přečíst si email, odpovědět, otevřít Facebook apod.

Pokud programátor pracuje ve dvojici během párového programování, většinou se na práci soustředí vyšší měrou, než kdyby programoval sám. Není to ani z toho důvodu, že by bylo člověku “trapné” přečíst si před tím druhým zprávu na sociální síti, ale spíše tím, že to nemá programátor zapotřebí. V momentě, kdy se jeho myšlenky odpoutají od zdrojového kódu, je povětšinou živou diskusí přitažen zpět a programování si opět získá jeho plnou pozornost.

S tímto se pojí fakt, že párové programování je mnohem více psychicky náročné než sólové programování, neboť právě nefungují tato častá polevení v pozornosti. Ta je téměř stoprocentně upřena na cílový úkol a programovat se tak dá jen několik hodin, rozhodně nikoliv standardní pracovní dobu, tedy 8 hodin. Podle některých zdrojů tak pracovníci párují např. jen polovinu pracovní doby, někde i méně a např. jen na specifických kratších úkolech.
K tématu flow je třeba doplnit, že někteří autoři uvádějí, že schopnost udržení flow během párového programování může být naopak i nižší, než u běžného programování a to zejména v případě, kdy je jeden z programátorů výrazně zkušenější než druhý (programování je často přerušováno vysvětlováním principů, které jednomu z dvojice připadají samozřejmé, druhému nikoliv) (ROBINSON, 2014).

6 Úskalí párového programování

Tato kapitola je věnována negativním aspektům, na které je třeba brát zřetel v prostředí párového programování. Při vypracovávání byl jako základ použit článek *"Pair programming is no panacea"* (HIGBEE, 2016). Článek ovšem, dle na našeho názoru, neobsahuje dostatečně praktické informace o nevýhodách užití metodiky. Z tohoto důvodu jsou v kapitole zahrnuty i jiné zdroje. Velkou část tvoří zpovědi jednotlivých programátorů, kteří zažili aplikaci metodiky párového programování v praxi a určitou dobu podle ní pracovali. Dále jsou zde použity výsledky studií, zabývajících se rozdílem mezi individuální a skupinovou prací.

6.1 Nevyváženost zkušeností

Jak již bylo zmíněno v kapitole 4 Principy párového programování výše**,** tato metodika jasně stanovuje, že mezi programátory v týmu není vztah učitel-student, ale jedná se o dva sobě si rovné kolegy.

V praxi ovšem bývá tento předpoklad často porušován. Příkladem může být tzv. fenomén "*Mladej, sleduj, jak se to dělá*". V tomto případě v týmu spolupracuje juniorní a seniorní programátor. Výsledkem je, že zkušenější z dvojice programuje většinu času, zatímco z juniora se stává pouhý pozorovatel. Tento stav může být užitečný v případě mentoringu nebo představování systému nově příchozímu programátorovi. To ovšem není účelem párového programování (HIGBEE, 2016; AGILE ALLIANCE, 2015).

Aby tomuto jevu mohlo býti předcházeno, metodika párového programování nabízí techniku zvanou "Ping-Pong párování". V podstatě se jedná o velmi časté střídání rolí navigátora a řidiče (v řádech minut).

Pro ilustraci uvádíme příklad střídání rolí v prostředí Test-Driven-Developmentu:

Programátor A - Napíše test, který neprojde.
Programátor B - Naprogramuje funkcionalitu, a test projde.
Programátor A - Provede refaktorizaci.
Programátor B - Napíše test, který neprojde.
Programátor A - Naprogramuje funkcionalitu, a test projde.
Programátor B - Provede refaktorizaci.
…atd.

Tímto způsobem dochází k rozdělování práce rovnoměrně mezi oba programátory.

6.2 Špatné párování osobností

Dalším základním prvkem párového programování je správná komunikace v týmu. Myšlenka existence rolí řidiče a navigátora může fungovat pouze v případě, že pokud jeden mluví, druhý naslouchá. S tímto se automaticky pojí problém párování vhodných osobností v týmu.

V praxi je možné narazit na tvrdohlavé programátory, kteří se drží vlastních postupů a nejsou schopni přijmout nový pohled na problém. Skrze své ego pak zabíjejí potenciál nového/jiného/možná i efektivnějšího řešení, které by bylo možné uplatnit (HIGBEE, 2016).
Na druhou stranu existují i jedinci příliš bázliví, s nízkým sebevědomím. Ti jsou schopni efektivně pracovat samostatně. Nicméně při práci v týmu mohou mít strach prosazovat své návrhy či poukazovat na chyby ostatních.

Výsledkem párování těchto dvou osobností je podobná situace, jako byla zmiňovaná v podkapitole výše. Jeden z programátorů působí pouze jako pozorovatel, bez možnosti zasáhnout do tvůrčího procesu. Programuje výhradně ten druhý, který si v mnoha případech také automaticky dělá revizi kódu. Tento stav představuje monumentální plýtvání zdroji.

Někteří lidé nemohou nebo jen prostě nechtějí přijmout prostředí párového programování. Mají špatné pracovní návyky, špatnou hygienu, jsou hlasití, nedokáží uznat jiný názor než ten svůj, nebo celou řadu dalších atributů. Ty nemusejí být překážkou při individuální práci, ovšem pro efektivní průběh párového programování mohou být rozhodující (BRECK-MCKYE, 2016).

6.3 Dvě hlavy ví více než jedna

Častým argumentem používaným pro prosazovaní párového programování, bývá kvalitnější návrh architektury a kódu – jak již zmiňuje předešlá podkapitola věnovaná výhodám.

Dle výsledků studie "*The Costs and Benefits of Pair Programming*", dochází při párovém programování ke snížení počtu řádků kódu, při zachování stejné funkcionality oproti individuálnímu vývoji:

"The pairs not only completed their programs with superior quality, but they consistently implemented the same functionality as the individuals with fewer lines of code… We believe this is an indication that the pairs had better designs." (COCKBURN, a další, 2015; WILLIAMS, 2015)

Nižší kvantita řádků ovšem v tomhle případě nemusí automaticky znamenat vyšší kvalitu celého kódu. Kvalita návrhu se odvíjí dle mnohem sofistikovanějších atributů než počtu řádků (HIGBEE, 2016).

Jako příklad je možné uvést:

- Znovupoužitelnost komponent
- Robustnost řešení
- Výpočetní složitost
- Provázanost komponent
- Čitelnost kódu
- … a další.

Představte si situaci, kdy v týmu pracují programátoři s naprosto odlišnými přístupy k návrhu:

- Pro prvního je základem abstrakce - co nejméně konkrétních případů užití, velké množství definovaných rozhraní, minimum provázanosti mezi komponentami a maximální znovu použitelnost řešení.
- Druhý je opakem prvního - veškerý návrh podřizuje výpočetní složitosti.

Základem úspěchu programování tohoto páru je v prvotní fázi ujasnění principu, jak bude dodávaná funkcionalita navržena. V opačném případě bude výsledkem nejasný návrh, který bude založen na kompromisech a skutečný potenciál nebude využit – kód nebude ani dostatečně abstraktně založený, ani obzvláště výpočetně efektivní (BRECK-MCKYE, 2016).

6.4 Chybí druhá perspektiva

Autor článku "*Pair Programming Is No Panacea"* zmiňuje jako nevýhodu párového programování problém spojený se sdílenou perspektivou.

Představme si modelový příklad pracovního týmu, jehož členové spolupracují každý den již 5 let. Pokud pracovníky rozdělíme a před každého postavíme tentýž problém, zjistíme, že ve většině případů dojdou k podobným řešením.

Čím déle lidé pracují společně, tím je vyšší šance, že budou sdílet stejnou perspektivu, což může být velkou nevýhodou v případech, kdy je nutné přijít s kreativním řešením problému.

Kreativita vychází z lidí, kteří se dokáží podívat na problém z různých úhlů, ne pouze z jednoho (HIGBEE, 2016).

6.5 Zabíjí kreativitu

Jak již bylo zmíněno výše, sdílená perspektiva s sebou může přinášet také úbytek kreativity.

Dle článku "*The Rise of the New Groupthink*" v The New York Times a výzkumu amerických psychologů Mihaly Csikszentmihalyi a Gregory Feist jsou lidé kreativnější, pokud je jim pro práci ponecháno soukromí a nejsou vyrušováni (CAIN, 2012).

Ve článku se také nachází zajímavá citace Steva Wozniaka (pozn.: vizionář a spoluzakladatel společnosti Apple):

"Most inventors and engineers I've met are like me… they live in their heads. They're almost like artists. In fact, the very best of them are artists. And artists work best alone…. I'm going to give you some advice that might be hard to take. That advice is: Work alone… Not on a committee. Not on a team." (CAIN, 2012).

Studie také uveřejňuje seznam faktů, které odlišují běžné programátory od těch pracujících v top firmách. Překvapivě není na prvním místě výše platu ani větší zkušenosti. Jedná se o množství soukromí, velikost osobního pracovního prostoru a nízká míra vyrušování (CAIN, 2012).

Párové programování bohužel neumožňuje programátorovi vytvářet nezávislé kreativní řešení.

Představte si situaci, kdy pracuje programátor sám:

- Je předloženo zadání nové funkcionality. Pro její úspěšné naprogramování existuje velké množství přístupů. Programátor dostane inovativní nápad, který by mohl představovat elegantní řešení problému. Je ovšem na pochybách, zda je tato varianta aplikovatelná. V této situaci mu ale nic nebrání věnovat několik minut na vytvoření jednoduchého prototypu, a pokud se mu to nepodaří, ničemu to nevadí, nikdo o tom totiž neví a může zkusit jinou variantu (BRECK-MCKYE, 2016).

Nyní si přestavte stejnou situaci v oblasti párového programování:

- Je předloženo zadání nové funkcionality. Pro její úspěšné naprogramování existuje velké množství přístupů. Programátor dostane inovativní nápad, který by mohl představovat elegantní řešení problému. Programátor je ovšem na pochybách, zda je tato varianta aplikovatelná. Nyní je ale nutné přednést myšlenku kolegovi. Projít s ním návrh implementace, ideálně krok po kroku. V poslední řadě doufat, že nebude moc kritický v případě neúspěchu.

Tento druh pracovního prostředí není přívětivý pro vytváření nových nápadů (BRECK-MCKYE, 2016).

6.6 Nutná synchronizace

Další nevýhodou párového programování může být fakt, že tato metodika vyžaduje velmi dobrou synchronizaci mezi oběma programátory. Dokonce i v těch nejjednodušších věcech.

Oba by měli přicházet i odcházet z práce ve stejnou dobu. Stejně tak je nutné, aby měli přestávku ve stejný čas. I dovolená by měla probíhat ve stejném termínu.

A zde může vznikat problém.

Jak bude možné pokračovat, když jeden z programátorů nebude moci pracovat? Pokud se jedná pouze o pár pracovních dnů, pravděpodobně nezbyde nic jiného než pracovat individuálně. Nicméně je tu také možnost vybrat nového programátora a do páru ho přiřadit. V této variantě je ovšem nutné počítat s časem na seznámení nového člena s úkolem přizpůsobení se práci s novým spolupracovníkem. Co když ale neexistuje náhrada za chybějícího člena a úkol vyžaduje alokaci dvou programátorů?

V těchto případech se může párové programování stát značně neúčinnou metodikou vývoje (GALANOPOULOS, 2014).

6.7 Může snižovat sebevědomí

Při aplikování metodiky párového programování je nutné mít na paměti, že týmová práce je výsledkem spolupráce.

Pokud budete jako manažer vyžadovat, aby programování probíhalo pouze v páru, vystavujete se riziku vytvoření pracovního prostředí, které bude znehodnocovat individuální práci. Tento fakt může postihovat i prostředí, kdy je párové programování využíváno pouze u psaní kódu, nikoliv například u návrhu.

Jednotliví programátoři pak mohou vnímat tento stav jako nedůvěru v jejich schopnosti ze strany vedení. U starších programátoru to může mít fatální následky spojené s odchodem ze zaměstnání. V případě juniornějších pracovníku to může vést k horšímu sebevědomí v budoucích projektech.

Dalším negativem, plynoucím z nařízeného párového programování může být fakt, že se pracovníci mohou po určité době stát závislí jeden na druhém. Zjednodušeně se dá říci, že ztratili sebevědomí ve své individuální práci (HIGBEE, 2016).

7 Závěr

Cílem této práce bylo poskytnout čtenáři dostatek informací o párovém programování, zvážit pozitiva i negativa, které tato metodika přináší a nabídnout dostatek podnětů k přemýšlení o tom, zda je párové programování metodika, která stojí za vyzkoušení. Dle našeho názoru se nám podařilo tento cíl zcela naplnit.

Párové programování, podobně jako kterákoliv jiná metodika, může programátorům přinést do jejich práce mnoho dobrého, nicméně je potřeba neprosazovat párové programování bez přihlédnutí k jeho nevýhodám. Dle našeho názoru podpořeného výše uvedenými argumenty se jedná o velice specifickou metodiku vhodnou jen pro určité týmy, určitá prostředí a projekty. Jistě stojí za to, se o této metodice něco dozvědět a zvážit její použití, avšak je nesmysl prosazovat ji za každou cenu. Její použití je vhodné tam, kde je potřeba zajistit kvalitní kód, návrh, architekturu a funkčnost v co nejrychlejším čase anebo při programování klíčových částí systému, na které navazuje mnoho dalších komponent a jejichž návrh a zpracování je pro výsledný produkt zcela zásadní.

V návaznosti na tuto práci můžeme doporučit k přečtení některé publikace a články, ze kterých tato práce vychází, zejména zdrojový článek Pair programming is no panacea (HIGBEE, 2016), který velice dobře shrnuje, proč není párové programování všespásnou metodikou. Pro komplexnější vzhled do této metodiky je k dispozici kniha Pair programming illuminated (WILLIAMS, a další, 2002), jenž se zabývá tímto tématem poměrně podrobně.

Seznam zdrojů

AGILE ALLIANCE. ©2015. Pair Programming. *Agile Alliance*. [Online]. ©2015 [Citace: 9. 10. 2016]. Dostupné z: https://www.agilealliance.org/glossary/pairing/.

BAIN, Lucy. ©2015. Try pair programming. *Atlassian Developers*. [Online]. 21. 5. 2015 [Citace: 9. 12. 2016]. Dostupné z: https://developer.atlassian.com/blog/2015/05/try-pair-programming/.

BRECK-MCKYE, Jimmy. ©2016. *Zpověď programátorů na téma: Jaké jsou možné nevýhody párového programování?* [Online]. ©2016 [Citace: 14. 12. 2016]. Dostupné z: http://softwareengineering.stackexchange.com/

C2 WIKI. 2014. Extreme Programming. *C2 Wiki [Online].* 9. 11. 2014 [Citace: 14. 12. 2016]. Dostupné z: http://wiki.c2.com/?ExtremeProgramming.

-. 2007. Pair Programming Facilities. *C2 Wiki [Online].* 31. 1. 2007 [Citace: 14. 12. 2016]. Dostupné z: http://wiki.c2.com/?PairProgrammingFacilities.

CAIN, Susan. 2012. The Rise of the New Groupthink. *The New York Times* [Online]. 15. 1. 2012 [Citace: 14. 12. 2016]**.** Dostupné z: http://www.nytimes.com/2012/01/15/opinion/sunday/the-rise-of-the-new-groupthink.html

CHONG, Jan, Robert PLUMMER, Larry LEIFER, Scott R KLEMMER, Ozgur ERIS a George TOYE. 2005. Pair Programming: When and Why it Works. *Psychology of Programming Interest Group Workshop [Online].* 2005 [Citace: 14. 10. 2016]. Dostupné z: http://hci.stanford.edu/publications/2005/pairs/PairProgramming-WhenWhy.pdf

COCKBURN, Alistair a Laurie WILLIAMS. 2015. The Costs and Benefits of Pair Programming. *Collaboration @ CSC - NCSU [Online]*. 2015 [Citace. 14. 12. 2016]. Dostupné z: http://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF

GALANOPOULOS, Ilias. 2014. Response to "Why Pair Programming Is The Best Development Practice?". *SAPM: Course Blog [Online]*. 2014 [Citace: 14. 12. 2016]. Dostupné z: http://blog.inf.ed.ac.uk/sapm/2014/03/05/response-to-why-pair-programming-is-the-best-development-practice/

HIGBEE, Wes. 2016. Pair Programming Is No Panacea. *InfoQ.com.* [Online]. 29. 6. 2016 [Citace: 9. 10. 2016]. Dostupné z: *https://www.infoq.com/articles/pair-programming-no-panacea?utm_campaign=rightbar_v2&utm_source=infoq&utm_medium=articles_link&utm_content=link_text*

JEFFRIES, Ron. ©2016. What is extreme programming?. *RonJeffries.com* [Online]. ©2016 [Citace 14. 12. 2016]. Dostupné z: http://ronjeffries.com/xprog/what-is-extreme-programming/

KADLEC, Václav. 2003. Extremismus nemusí být na škodu: extrémní programování. *Živě* [Online]. 13. 5. 2003 [Citace 14. 12. 2016]. Dostupné z: http://www.zive.cz/clanky/extremismus-nemusi-byt-na-skodu-extremni-programovani/sc-3-a-111714/default.aspx

KNESL, Jiří. 2014. Jak funguje párové programování. *Jiří Knesl* [Online]. 18. 3. 2014 [Citace 14. 12. 2016]. Dostupné z: http://www.knesl.com/jak-funguje-parove-programovani

RAYGUN.COM. 2015. So how good is pair programming, really?. *Raygun.com* [Online]. 2. 11. 2015 [Citace: 9. 10. 2016]. Dostupné z: https://raygun.com/blog/2015/11/how-good-is-pair-programming-really/

ROBINSON, Brian a kolektiv. 2014. Pair programming and flow. *C2 Wiki [Online]*. 2014 [Citace: 18. 12. 12. 2016]. Dostupné z: http://wiki.c2.com/?PairProgrammingAndFlow

WELLS, Don. ©2013. Extreme Programming: A gentle introduction. *Extreme programming* [Online]. ©2013 [Citace: 9. 10. 2016]. Dostupné z: http://www.extremeprogramming.org/

-. ©2013. Pair Programming. *Extreme programming* [Online]. ©2013 [Citace: 9. 10. 2016]. Dostupné z: http://www.extremeprogramming.org/rules/pair.html

WILLIAMS, Laurie a Robert KESSLER. 2002. *Pair programming illuminated*. Boston, MA: Addison-Wesley, 2002. ISBN 9780201745764.