Extrémne programovanie a iné agilné metodológie

Zimný semester 2007/08

Ing. František Gyárfáš, PhD.

Katedra aplikovanej informatiky

gyarfas@ii.fmph.uniba.sk

http://www.ii.fmph.uniba.sk/~gyarfas/

Manifest agilného programovanie

11.-13. Februára 2001 sa zišlo v lyžiarskom stredisku v Utah 17 špičkových programátorov lyžovať a diskutovať o princípoch programovania.

Manifest agilného programovanie

Druhá úloha

Zamyslite sa nad princípmi programovania, ktoré sú podstatné **pre Vás osobne** a spíšte ich vo forme manifestu.

Manifest môjho programovania

Termín odoslania: **piatok 19.10.2007 – 24.00**

Situácia

Agilné metodológie reagujú na nedostatky štandardných metód

Problém	Náprava
Vývojári nerozumejú zadaniu	Postupný vývoj úplných verzií
Nefunguje spolupráca so zákazníkom	Zákazník sa zúčastňuje vývoja
Izolovanosť programátorov	Veľmi častá integrácia
Neistota kvality	Neustále vytváranie a zbiehanie testov
Formálne postupy	Zodpovednosť programátorov
Závislosť na špecialistoch	Spoločne zdieľaný kód
Veľký tresk	Postupné odovzdávanie
Navršovanie zlých rozhodnutí	Odvaha k zmene

Základné piliere agilného programovania

Umožniť a **prijať zmenu je omnoho efektívnejšie, ako sa snažiť zmene zabrániť**.

Je potrebné byť pripravený reagovať na nepredvídateľné udalosti, pretože tie určite nastanú. **Jedinou istotou je zmena.**

Zákon agilnej architektúry

Agilné metódy nie sú v opozícii k architektúre programov, iba sa domnievajú, že návrhy, schémy a diagramy sú menej dôležité ako samotní architekti.

Základné princípy

Fungujúci softvér je hlavný ukazovateľ postupu.

Odovzdávajte funkčný softvér často (od niekoľkých týždňov či mesiacov). Čím častejšie, tým lepšie.

Zákazníci a vývojári spolupracujú denne na projekte.

Vítajte zmeny požiadaviek hoci aj v priebehu vývoja, pretože zmena môže byť pre zákazníka konkurenčnou výhodou.

Najúčinnejším spôsobom výmeny informácií je konverzácia.

Základné princípy II.

Maximalizujte množstvo práce, ktorú netreba urobiť.

Obmedzte dĺžku pracovnej doby. Agilné procesy predpokladajú udržateľný vývoj programov donekonečna. Nadčasy to neumožňujú.

Vývojárom poskytnite podporu, ponechajte zodpovednosť a nechajte ich pracovať.

V pravidelných intervaloch sa tímy zaoberajú úvahami, ako byť efektívnejší a potom tomu prispôsobia svoje správanie.

Najlepšia architektúry, špecifikácie a návrhy vznikajú v samo sa organizujúcich týmoch.

Najznámejší reprezentanti agilného programovania

- **Extrémne programovanie**
- **Metodika SCRUM Development Process**
- **Lean Development**
- **Feature Driven Development**
- **Test Driven Development**

Extrémne programovanie (XP)

Extrémne programovanie je metodika pre malé až stredne veľké programátorské tímy (2-10), ktorí sa vyvíjajú softvérové projekty so zadaním, čo sa môžu často a rýchlo meniť.

Niektorým ľuďom sa javí XP ako triezve a praktické uvažovanie. Väčšine sa ako také nejaví.

V zásade je možné povedať, že XP používa určité známe postupy, ktoré dovádza do extrémov.

XP sľubuje programátorom

XP sľubuje programátorom, že:

- budú neustále pracovať na veciach, ktoré sú skutočne podstatné.
- s nepriaznivými situáciami sa nebudú vyrovnávať sami.
- budú rozhodovať o veciach, ktorým rozumejú a nebudú musieť robiť veci, na ktoré nemajú.

XP sľubuje zákazníkom

XP sľubuje zákazníkom, že:

- z každého týždňa programovania vyťažia maximálnu možnú hodnotu.
- v intervale niekoľkých týždňov uvidia pokrok, ktorý ich zaujíma.
- majú možnosť zmeniť smer projektu uprostred vývoja bez toho, aby to spôsobilo abnormálne vysoké náklady.

Ako to XP chce dosiahnuť

- Programový kód píšu **dvojice programátorov** za jedným počítačom
- **Najprv sa píšu testy**, potom k nim zdrojový kód.
- Napísaný kód sa **integruje ihneď** po odladení.
- Pracuje sa **max 40 hodín** týždenne.

Ako to XP chce dosiahnuť II.

- Programy sú neustále pripravované na odovzdanie zákazníkovi.
- Čo je rozumné preprogramovať (refaktorizácia), urobí sa ihneď, ako je na to čas.
- Programuje sa iba to, čo je nevyhnutne nutné.
- Vývoj programu riadia neustále **iterácie**.
- Všetko je tak jednoduché, ako je to iba možné.

Charakteristika tímu XP

- malé skupiny programátorov (do 10 ľudí)
- spoločná znalosť kódu
- všetci programujú všetko
- spoločné vlastníctvo kódu
- spoločné zásady písania programov
- zástupca zákazníka je člen vývojového tímu

Pravidlá všemocnosti peňazí

Medzi manažérmi softvérových projektov je často rozšírená ilúzia, že peniaze dokážu všetko.

Pravidlo paralelizmu pri programovaní

Deväť žien nevynosí dieťa za mesiac.

Sebavedomý dodatok k pravidlo paralelizmu

Ani osemnásť žien nevynosí dieťa za mesiac.

Štyri premenné

Vývoj programového systému riadia štyri premenné:

- **náklady**
- **čas**
- **kvalita**
- **šírka zadania**

Vlastnosti premenných

Tieto premenné majú svoje špecifické vlastnosti:

Náklady: sú najviac obmedzenou premennou. Na začiatku by mali byť náklady malé a postupne by mali pribúdať. Priveľa peňazí nedokáže adekvátne zrýchliť projekt. Primálo ho ale ohrozí.

Čas: je väčšinou v rukách zákazníka. Je najmenej pružnou veličinou. Priveľa času projektu škodí, pretože projekt potrebuje spätnú väzbu v reálnej prevádzke. Ak však je na projekt málo času, utrpí kvalita projektu.

Kvalita: nie je dobrá riadiaca premenná projektu. Delí sa na **vonkajšiu a vnútornú**. Vonkajšiu kvalitu hodnotia zákazníci. Obetovanie vnútornej kvality kvôli skráteniu času je lákavé. Je síce možné krátkodobo dosiahnuť zisk, ale výsledné náklady bývajú obrovské. Paradoxom je, že **vyžadovanie lepšej kvality skracuje dĺžku projektu**.

Šírka zadania: pre vývoj softvéru je **kľúčovou premennou**. Je to veľmi premenlivá veličina. Menšia šírka zadania, ktorá ešte rieši potreby zákazníka, umožňuje dosiahnuť lepšiu kvalitu, rýchlejšie a lacnejšie.

Zákaznícke priority

Zadávatelia si často myslia, že môžu určovať hodnoty všetkých štyroch premenných.

„*Všetky* ***(šírka zadania)*** *naše požiadavky musia byť urobené do konca mesiaca* ***(čas)***. ***Kvalita*** *je absolútna priorita. Veď sme si to zaplatili.* ***(náklady)*** *“*

Výsledok takéhoto zadania často býva, že na **kvalitu** nikto nebude prihliadať, stres spôsobí, že sa **čas** vymkne kontrole a mnohé funkcie chýbajú. Výsledkom bude neúplný, nekvalitný softvér s veľkým oneskorením.

Riziká softvérových projektov

Softvérové projekty poznajú celý rad rizík, ktoré sa notoricky a cyklicky opakujú celé dejiny softvérového inžinierstva.

Oneskorený harmonogram

Krachujúci poruchový systém

Miera poruchovosti

Nepochopenie zadania

Zmeny zadania

Prebytok funkcií

Fluktuácia pracovníkov

Oneskorený harmonogram

XP požaduje:

prednostné implementovanie funkcií s najvyššou prioritou.

veľmi krátke cykly na uvoľňovanie nových verzií (niekoľko mesiacov).

v rámci jednej verzie jednotýždňové až štvortýždňové iterácie pre uvoľňovanie funkcií, požadovaných zákazníkom.

v rámci iterácie sa plánujú úlohy jedno až trojdňové.

Krachujúci poruchový systém

XP vytvára a udržuje počas celého vývoja kompletnú sadu testov, ktoré sa spúšťajú pred aj po každej integrácii (niekoľko ráz denne).

Chybám nie je dovolené, aby prežívali a hromadili sa.

Miera poruchovosti

XP testuje systém ako z hľadiska programátorov, tak aj z hľadiska funkčnosti.

Testy funkčnosti píšu samotní zákazníci.

Nepochopenie zadania

XP požaduje od zákazníka, aby bol integrálnou súčasťou vývojového týmu.

Špecifikácia projektu sa počas vývoja neustále upresňuje.

Zákazník má možnosť aj počas vývoja projektu pracovať s hotovými časťami systému.

Zmeny zadania

XP skracuje cyklus uvoľňovania nových verzií, čo umožňuje pri ďalšej verzii rozšíriť či zmeniť zadanie.

Pri voľnosti plánovania si riešitelia ani nemusia všimnúť, či ide o nové zadanie, alebo iba ďalšiu časť pôvodného.

Prebytok funkcií

XP trvá na tom, aby sa riešili iba úlohy s najvyššou prioritou.

Fluktuácia pracovníkov

XP žiada od programátorov, aby prijali zodpovednosť za odhadovanie termínov a dokončenie vlastnej práce. Ľudia s vlastnou zodpovednosťou ťažšie nechávajú nedokončenú robotu.

XP neumožňuje ľuďom sa izolovať, čo býva často zdrojom frustrácie.

XP uplatňuje kolektívne vlastníctvo celého programového kódu, čo zabraňuje tomu, že niektorým častiam programu rozumie iba jeden človek.

Každú časť programu vytvárali aspoň dvaja programátori.

Sociálne hodnoty

Krátkodobé ciele jednotlivcov sú často v spore s dlhodobými sociálnymi cieľmi.

Spoločnosť sa naučila riešiť tento problém tak, že si vyvinuli zdieľané sady hodnôt, ktoré podporila mýtami, rituálmi, poverami, zákonmi, odmenami a trestami.

Ak tieto spoločné hodnoty chýbajú, majú ľudia tendenciu vracať sa k svojim krátkodobým osobným záujmom.

Sociálne hodnoty v XP

XP má štyri svoje hodnoty:

komunikáciu

jednoduchosť

spätnú väzbu

odvahu

Komunikácia

XP sa zameriava na to, aby ľudia považovali komunikáciu na všetkých úrovniach za **prirodzenú a nevyhnutnú súčasť** svojej práce.

Používa postupy, ktoré sa **bez komunikácie nedajú realizovať**.

- **Párové programovanie**
- **Testovanie jednotiek**
- **Odhadovanie úloh**

Výsledok je, že spolu počas celého projektu komunikujú programátori, zákazníci aj manažéri

Čo neznamená, že komunikácia niekedy nelezie na nervy, nezdržuje a nevyrušuje.

Jednoduchosť

Základná otázka: **čo je najjednoduchšie riešenie, ktoré ešte môže fungovať?**

Jednoduchosť nie je jednoduchá.

Jedna z najťažších vecí na svete je neuvažovať nad vecami, ktoré bude potrebné implementovať zajtra, za týždeň, za mesiac. Nutkavé myšlienky na budúcnosť sú motivované snahou zabrániť budúcim nákladom na zmenu.

XP hrá lotériu. Vsádza na to, že je lepšie urobiť jednoduchú vec dnes a zaplatiť zajtra trochu viac za prípadnú zmenu, ako urobiť zložitejšiu vec dnes, aj keď nemusí byť potrebná nikdy.

Spätná väzba

Programátori trpia chorobou z povolania: **optimizmom, že program bude fungovať.**

Táto choroba sa lieči spätnou väzbou.

Kouč v XP hovorí: **Nepýtaj sa mňa, spýtaj sa programu.**
A dodáva: **Už si napísal test*?***

Spätná väzba funguje s neustálym testovaním všetkých zložiek systému. Testujú sa všetky časti programu od najjednoduchších metód po zložitú integráciu, testujú sa zadania, testuje sa funkcionalita. To všetko sa testuje počas celého vývoja systému od návrhu až po odovzdanie.

Odvaha

V 25 týždni projektu (z 30) objavil niektorý z testov zásadnú chybu v architektúre systému. Vývoj funkcionality sa spomalil a temer zastavil.

Čo má tím urobiť? Upchať chybu záplatou, alebo zastaviť ďalší vývoj, vrátiť sa na začiatok a zmeniť architektúru?

Na rozdiel od väčšinovej reálnej praxe XP sa jednoznačne stavia za návrat k základom a opravenie chyby. Toto rozhodnutie si vyžaduje veľkú odvahu.

Veľkú odvahu si vyžaduje vyhodiť napísaný kód.

Za všetkými štyrmi hodnotami sa ukrýva ďalšia, ktorá im dáva zmysel: **rešpekt.**

Druhá úloha

Zamyslite sa nad princípmi programovania, ktoré sú najpodstatnejšie **konkrétne pre Vás osobne** a spíšte ich vo forme manifestu.

Manifest môjho programovania

Termín odoslania: **piatok 19.10.2007 – 24.00**

Cyklus vývoja

Dvojice programátorov pracujú vždy spoločne.

Vývoj programov je riadený pomocou testov. Najprv testujeme a potom programujeme. Až keď všetky testy fungujú a nevieme vymyslieť žiadne ďalšie testy, je proces programovania funkcie ukončený.

Dvojice nezabezpečujú iba to, aby testy fungovali. Rozvíjajú aj návrh systému, jeho analýzu a implementáciu.

Integrácia včítane integračného testovania nasleduje ihneď po ukončení vývoja.