OBLIKOVANJE OKOLJA ZA EKSTREMNO PROGRAMIRANJE

Matevž Rostaher, Ivan Slamek, Andrej Kline
OdaTeam d.o.o., Meljska cesta 36, SI-2000 Maribor
e-mail: info@odateam.com, URL: http://www.odateam.com

Izvleček

Ekstremno programiranje je zamišljeno kot lahka metodologija razvoja programske opreme, namenjena manjšim projektnim skupinam, ki programsko podpirajo poslovne procese v okolju, kjer potrebe niso vnaprej natančno določene in se nenehno spreminjajo. Lahko rečemo, da je tovrstnih razvojnih okolij v našem prostoru kar nekaj, zato je disciplina, ki bo predstavljena v pričujočem prispevku, namenjena prav ljudem, soudeležencem razvoja v teh okoljih. Avtorji nimamo namena vsiliti lastnih pogledov in načina dela nikomur od vas. Predstavili vam bomo ključne elemente ekstremnega programiranja in vas seznanili z načinom, kako smo ga mi sprejeli v naše razvojno okolje, kako nas je njegovo spoznavanje spremenilo v obdobju enega leta od vpeljave in kakšne so naše izkušnje ob uporabi novega načina razvoja.

Abstract

Extreme programming (XP) is a lightweight methodology for developing software with main target in smaller project groups that are obliged to develop in an environment where requirements are not fully specified in advance and/or are changing constantly. Nowadays, most software projects have to deal with changing requirements, therefore the following paper is meant for everybody involved in such a developing process. The authors do not want to enforce their own views of working style to anybody. They just present the basic elements of XP and explain the way of adopting it into the development environment; how they have changed their thinking and the way they live since beginning to implement the discipline two years ago.

■ ■ ■

1. Ekstremno programiranje (XP) kot disciplina

Okrajšava XP in problem prevoda.

Najprej glede poimenovanja. Ekstremno programiranje je dobeseden prevod izvornega pojma "Extreme programming", ki v angleškem govornem področju označuje metodologijo, predstavljeno v nadaljevanju. Slovenski prevod ni enostaven, vsaj ne takšen, ki bi bil izražen z enim samim pojmom ali besedno frazo. Ob našem srečanju z ekstremnim programiranjem smo se oprijeli izraza skupinsko programiranje (skupinski proces razvoja, ipd.), ki pa ni najbolj posrečen niti ustrezen, saj je jasno dejstvo, da se programska oprema razvija znotraj skupine in da je doba t.i. «one man band« programerjev že davno za nami. Ekstremno programiranje obsega mnogo več kot le skupinsko programiranje. Zato imenujmo ekstremno programiranje preprosto XP in pod tem izrazom razumimo disciplino z lastnostmi, opisanimi v nadaljevanju.

Kaj je in kaj ni ekstremno programiranje?

XP je nizko-obremenjen, učinkovit, nizko-rizičen, predvidljiv, znanstven in svež način razvoja programske opreme, ki predlaga (ne narekuje!) nekoliko drugačne smernice v vseh fazah razvoja. Ne uvaja nobenih revolucionarnih novosti, vendar pa obravnava vsa sredstva, potrebna v razvojnih projektih (delovni prostor, ljudi, metodologija razvoja, način vodenja in upravljanja) na svojstven, precej drugačen, za nekatere celo čuden, ekstremen, način. Z obrazložitvijo, da XP skuša vsakega od svojih sestavnih elementov privesti do ekstrema, pojasnimo tudi uporabo besede « extreme » v izvornem izrazu. XP skuša do skrajnosti poenostaviti načrtovanje, razvoj, komunikacijo, testiranje in kodiranje, ampak na način, da se vse naštete dejavnosti čimbolj dopolnjujejo in medsebojno sodelujejo. Vse dejavnosti bomo opisali v nadaljevanju, kjer bo tudi postalo jasno, zakaj jih XP res udejanji na (ekstremno) enostaven način, pa četudi se na prvi pogled zdi, da XP vnaša v njih dodaten napor.

Najprej poglejmo, v katerih elementih se metodologija XP razlikuje od ostalih metodologij :

- zgodnji, konkreten in stalen odziv s pomočjo kratkih razvojnih ciklov
- inkrementalni pristop k načrtovanju, ki kaj hitro privede do celotnega načrta, pričakovanega v življenjski dobi projekta

- prilagodljivo načrtovanje implementiranja funkcionalnosti, ki prožno reagira na spremembe poslovnih potreb
- avtomatski testi, ki jih pišejo programerji in stranke za opazovanje napredka v razvoju, omogočajo sistemu, da se razvija in da zgodaj prestreže napake
- zanaša se na ustno komunikacijo, teste in na programsko kodo, ki komunicira (pojasnjuje tudi drugim programerjem) strukturo in namen
- zanaša se na evolucijski proces načrtovanja, ki traja tako dolgo, kot živi sistem
- zanaša se na tesno sodelovanje programerjev, ki imajo zgolj običajne sposobnosti
- upošteva tako kratkoročni instinkt programerjev kot dolgoročni interes projekta (drugače povedano, popolnoma podpira posebnost in inovativnost posameznih programerjev, ampak le v skladu z dobronamernostjo za razvoj celotnega projekta!).

Discipliniranost XP

se kaže v nekaterih, sicer redkih principih, od katerih metodologija ne odstopa. Npr. testiranje. Kdor ne piše testov, ali bolje, kdor ne piše avtomatskih testov, ne udejanja XP! Polovičarstva ni. XP ima pravilo: Ali si zraven ali nisi.

Ali je potrebna še ena nova disciplina?

XP v osnovi ne vnaša nobenih novih prvin. Tehnike programiranja, ki jih uporablja, so znane že desetletja. Načini vodenja ljudi, ki jih postavlja v ospredje, pa celo že stoletja. Kaj je torej novega pri tej disciplini?

Razvoj programske opreme je povezan z riziki, kot so: zamik planiranih datumov izdaje programja; ukinitev projekta; stroški popravkov izdanega sistema so tako veliki, da je sistem potrebno zamenjati; odstotek pojavljanja napak v sistemu je zelo visok; programska oprema ne ustreza poslovnim zahtevam; poslovne zahteve se med implementacijo spremenijo; izdaja programske opreme, ki uporabniku ne prinaša vrednosti; programerji, ki se naveličajo projekta v agoniji, zavrnejo sodelovanje. XP se dotika teh problemov na vseh nivojih razvoja: teži h kratkim iteracijam projekta, h kratkim ciklom izdaje; uporabnike se vpraša za čim manjšo smiselno izdajo, ki bodisi postopno gradi sistem ali pa povzroči malo škode, če je implementacija neustrezna; kreira in vzdržuje obširno zbirko testov, najbolje za testiranje celotnega sistema, ki se izvajajo ob vsaki spremembi, tudi večkrat na dan; testiranje vseh implementiranih enot opravljata oba programerja v paru, testiranje funkcionalnosti pa le za to določeni programerji; najbolj nujna funkcionalnost se implementira najprej, s čimer se rešimo nevarnosti po implementiranju neuporabnega sistema; stranke so sestavni del integracije sistema, s čimer se specifikacije sistema redefinirajo vzporedno z novimi zahtevami uporabnikov.

Vidimo torej, da je XP zelo pozitivno naravnana disciplina, ki v osnovi teži k uspešnosti projektov, ki vztrajno dodajajo poslu vrednost in ki se kontrolirano razvijajo!

2. Ekonomski vidik vpeljave XP razmišljanja v razvoj programske opreme

Štiri spremenljivke.

XP kontrolira potek projektov s pomočjo štirih spremenljivk:

- stroškov,
- časa,
- kakovosti in
- obsega.

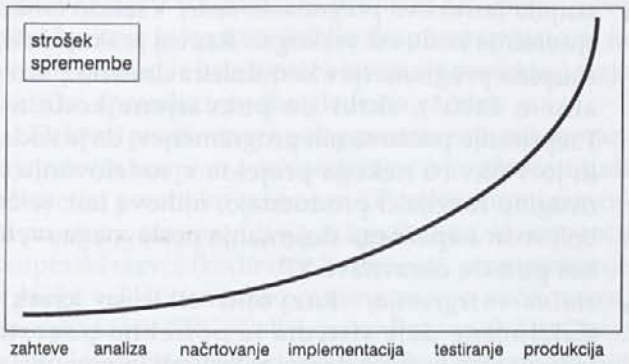
Igra razvoja programske opreme se igra po pravilu, da zunanji vplivi projekta (poslovne stranke, uprava) izberejo tri od štirih spremenljivk. Razvojna skupina pa kot rezultat izbere četrto spremenljivko.

Ponavadi uprava in poslovne stranke določijo prve tri spremenljivke; s kakšnimi vložki želimo v določenem časovnem obdobju doseči kakšno kakovost sistema. Nikakor ne smejo uprava ali stranke določati obsega sistema (kode). To je izključno v domeni razvijalcev. Nasploh XP strogo ločuje poslovne odločitve od odločitev tehnične narave. Prve naj sprejema poslovni svet (poslovne stranke), slednje pa razvojna skupina.

Poleg tega kakovost sistema praviloma ni spremenljivka, vsaj na dolgi rok ne. V zgodnjih iteracijah sicer XP svetuje načelo "naredi najenostavnejšo možno stvar, ki lahko deluje", ki ne zagotavlja popolne funkcionalnosti sistema. Ta bo dodana z naslednjimi iteracijami, ob nenehnem testiranju, ob posvetovanjih s poslovnimi strankami, kontrolirano! Kakovost sistema pa je konstantna, torej edina spremenljivka, ki je XP ne daje na izbiro. Kakovost naj bo skozi vse faze razvoja sistema največja možna.

Strošek spremembe.

Ena splošnih ugotovitev in celo predvidevanj programskega inženirstva je, da stroški sprememb v programu naraščajo eksponentno s časom (Slika 1). V zadnjih desetletjih si razvoj programske opreme, gledano kot skupnost ali kot tendenca, prizadeva zmanjšati ta strošek – z jeziki nove generacije, z boljšimi podatkovnimi bazami, z boljšo navado programiranja, z boljšimi razvojnimi okolji in orodji, z novimi notacijami. Vpeljava navedenih novosti je vsekakor dobrodošla, že za boljše počutje in ugodnejše delo razvijalcev, vendar izpostavljenega problema ne reši. Zanimivo, vendar žal resnično.



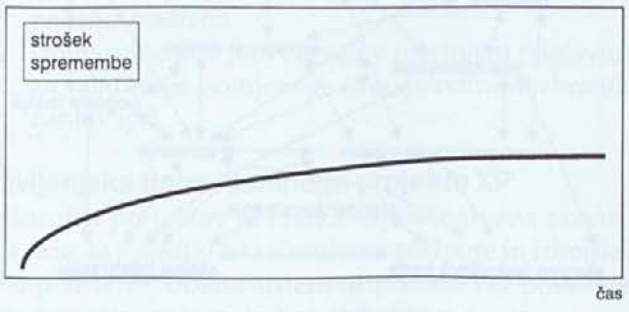
Slika 1: Strošek spremembe raste eksponentno s časom

Ena od tehničnih premis XP pa je ravno zmanjšanje rasti tega stroška, ki naj se s časom približa neki asimptotični vrednosti (Slika 2). Tako nam že sama predpostavka, da stroški sprememb v sistemu s časom ne postanejo enormni, omogoča razmišljati drugače. Velike odločitve o nadaljnji usmeritvi sistema (tudi projekta) lahko prenesemo v kasnejši čas, ko bodo zahteve popolnejše in jasnejše. Prav tako je pri slednji rasti možno načrtovati strošek spremembe, ki bo ob morebitni spremembi potreben v prihodnosti.

Tako kot XP teži k vzpostavitvi okolja, v katerem je strošek sprememb s časom relativno majhen, nikakor pa ne eksponentno velik, tako si tudi obratno discipline XP brez predpostavke, da so spremembe poceni, ne moremo zamisliti. Kako bi sicer razvijali pod načelom "naredi najenostavnejšo stvar, ki lahko deluje" zdaj, nadaljnja funkcionalnost pa bo dodana kasneje, ko bo to neznansko drago!?

Zadrževanja nizke vrednosti stroška sprememb seveda ni možno doseči kar tako. Potreben je skupek premišljenih tehnologij in razvojnih navad, ki jih prav XP skuša zbrati pod okriljem svoje discipline.

Na tehnološki strani so objekti gotovo ključna tehnologija. S sporočili, ki jih izmenjujejo in metodami, ki se izvajajo v ozadju, so pripravljeni na majhne, obvladljive spremembe, ki nimajo učinka na celoten sistem. Nadalje lahko sklepamo na tehnologijo podatkovnih baz, kjer so tiste objektno zasnovane v prednosti glede sprememb, saj je podatkom pripeta tudi koda.



Slika 2: Strošek spremembe s časom ne zraste dramatično

Nikakor pa ne trdimo, da je objektna tehnologija, katere privrženci smo, pogoj za izvajanje XP. XP se v osnovi naslanja na naslednje elemente:

- preprosto načrtovanje, brez kakršnihkoli idej o poslovnih potrebah, ki še niso uporabljane, ampak bi se utegnile pokazati kot koristne v prihodnosti
- avtomatizirani testi, ki nam vlijejo zaupanje pri spreminjanju sistema; da ne naredimo neke spremembe sistema v strahu, kaj nam bo povzročila na netestiranih delih sistema
- ogromno prakse pri spreminjanju in načrtovanju; da se ne bojimo spreminjati sistema.

3. Okolje, v katerem lahko XP zaživi

Spoznali smo že namen in ključne elemente discipline. Zdaj lahko strnemo spoznanja v glavnih vrednotah in principih, ki jih zagovarja XP.

Štiri vrednote:

- komunikacija,
- preprostost,
- odzivnost in
- pogum.

Pri komunikaciji so zajete vse udeležene strani: razvijalec-razvijalec, razvijalec-stranka, razvijalec-vodstvo... Preprostost smo že večkrat omenili. Je ena najtežje dosegljivih vrednot za razvijalca XP. Namreč pri razvoju ni enostavno odmisliti vseh zahtev, ki so trenutno res nepotrebne. Odzivnost je sposobnost strank seznaniti se z implementiranim delom sistema in reagirati nanj. XP teži k predajanju manjših enot sistema čimprej v uporabo, da bodo tudi morebitne napake čimprej ugotovljene. Pogum pa XP obravnava kot vrednoto z največjo težo, saj le osebje, ki si drzne začeti in spreminjati sistem, lahko živi in razvija v skladu s filozofijo discipline.

Vrednote se ponovno prepletajo, npr. pogum razvijalca je tem večji čim boljša je komunikacija z drugimi razvijalci, pa tudi z vodstvom. Razvijalec, ki je močno kaznovan za vsako manjšo napako, se sprememb boji, pa tudi nove stvari razvija z občutkom strahu pred grajo.

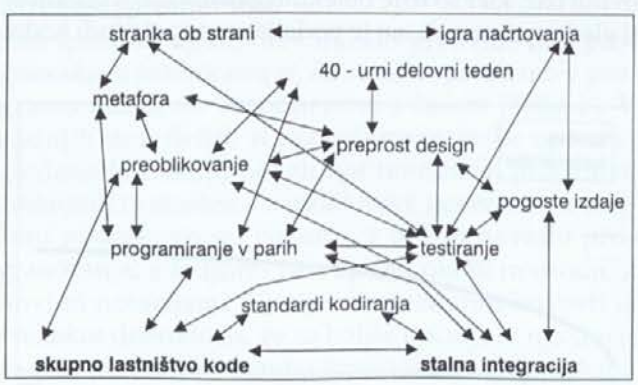
Principi XP:

Kot vsaka druga metodologija, tudi XP v končni fazi sloni na skupku principov. Teh je kar nekaj: igra načrtovanja, majhne in pogoste izdaje, metafora, preprost dizajn, testiranje, preoblikovanje, programiranje v parih, skupno lastništvo kode, stalna integracija, 40-urni delovni teden, stranka zmeraj prisotna in standardi kodiranja. Seveda se medsebojno povezujejo (Slika3), večinoma samostojno sploh ne morejo nastopati. Vsekakor pa, še preden na sliki dobite vtis, kako kompleksen in zamršen je XP, pojasnimo pomen principov in njihovo medsebojno sodelovanje.

- *igra načrtovanja*; Razvoj programske opreme je zmeraj razvijajoči se dialog obeh strani (poslovne in tehnične) glede tega, kaj je možno in kaj zaželeno. Poslovni ljudje odločajo o obsegu razvoja sistema, o prioriteti razvoja posameznih delov, o komponiranju izdaj (pod)sistema ter o datumih teh izdaj. Seveda poslovne odločitve ne morejo biti sprejete brez ustreznih tehničnih posvetovanj in odločitev glede: ocenjenega časa, potrebnega za implementacijo posameznih delov sistema; posledic, ki jih prinašajo poslovne odločitve (npr. izbira baze podatkov); načina organiziranosti razvojne skupine; podroben urnik razvoja.
- *majhne in pogoste izdaje*; Vsaka izdaja naj bo tako majhna, kot je le možno, da še zmeraj predstavlja zaključeno celoto. V poslovni svet naj prinese implementacijo najbolj vrednih zahtev (tistih z najvišjo prioriteto). Pogostost izdaj naj bo vsakih nekaj tednov.
- *metafora;* Metafore služijo za skupno poimenovanje tehničnih elementov sistema in povezav med njimi, v smislu nedvoumne komunikacije med vsemi osebami, vpletenimi v razvoj sistema. Npr. pojem zavarovalna pogodba nedvoumno določa abstrahiran element zavarovalniškega poznavanja, pa čeprav je to za uporabnike sistema viden kot en vnos v brskalniku, na katerega lahko klikajo in dobijo ustrezne povezave, za razvijalca pa predstavlja točno določen tip objekta z atributi... Metafore v XP nadomeščajo t.i. arhitekturo sistema, največkrat poznano kot "velike škatle s povezavami".
- *preprost design*; Pravilen design v vsakem trenutku razvoja ustreza naslednjim aktivnostim:
  1. zagon vseh testov
  2. preverjanje podvojenosti logike
  3. označevanje vseh namer ali idej, pomembnih za programerje
  4. uporaba čim manj razredov in metod.
- *testiranje*; XP prakticira funkcijske teste in teste enote. Oba načina testiranja sta bila podrobno predstavljena v prispevku [12]. Seveda je med razvojem ključnega pomena oblikovanje in vzdrževanje avtomatskih testov, katerih uporabo smo že omenili. Na tržišču so prisotna odlična testna ogrodja, tako za Javo kot Smalltalk: JUnit, SUnit.
- *preoblikovanje*; Vedno, ko programer naleti na situacijo, da mu preoblikovanje obstoječe kode olajša nadaljnji razvoj novih zahtev, naj uporabi tehnike preoblikovanja. Tudi v ta namen so na tržišču orodja za podporo, npr. "Refactoring browser" za Smalltalk. Podrobneje bo ta princip opisan v prispevku [11].
- *programiranje v parih*; Princip, po katerem vsak zlog nove kode ustvarjata oba programerja, združena v razvojni par. Tudi testiranje poteka v dvoje. Podrobneje je bil ta princip opisan v prispevku [13].
- *skupno lastništvo programske kode*; Vsakdo lahko spreminja kodo od vsakogar. Razvoj je skupinski, skupina programerjev komunicira dnevno ("v realnem času"), skrbi da pokvarjene kode ni. Prepričanje posameznih programerjev, da je koda, ki jo v okviru nekega projekta v sodelovanju z drugimi razvijalci producirajo, njihova last, je že bolj stvar napačnega dojemanja poslovnega sveta kot pa naše obravnave XP.
- *stalna integracija;* Razvojni cikli so kratki. Zaključene dele sistema je potrebno pogosto (včasih tudi večkrat dnevno) testirati in povezati v delujočo celoto (še preden se jo preda uporabniku kot naslednjo izdajo).
- *40-urni delovni teden*; Projektno zasnovano delo zahteva seveda velike napore vseh sodelujočih. Zato XP že ob načrtovanju časa razvoja kot svoje spremenljivke računa z 8 urami dela dnevno. Na kratki rok se sicer da delati tudi več ur dnevno, vendar sodelujoči v projektu že po nekaj tednih postanejo demotivirani za delo.
- *stranka ob strani*; Prisotnost akterjev iz poslovnega sveta, za katerega se sistem razvija, smo že omenili kot nenadomestljivo pridobitev v smislu izboljševanja zahtev, hitrega odziva, zmanjšanja rizika napačno razvitega sistema ipd.
- *standardi kodiranja;* Pravila morajo biti jasna vsem programerjem, da lahko ti komunicirajo kar z izvorno kodo, ne pa s tekstovnimi opisi. Zelo pomembno je poznavanje vzorcev in njihova dosledna uporaba.

Razvojno okolje = delovni prostor + razvojna skupina + razvojno orodje

Disciplina XP je namenjena skupinskemu razvoju do 10 razvijalcev. Sodelujočih poslovnih strank je lahko seveda več, vendar ne hkrati pri enem projektu več kot deset, šteto skupaj z razvijalci. Z večanjem števila razvijalcev principi XP izgubljajo svojo veljavo. Naj še enkrat poudarimo, da v sklopu XP vse osebe sodelujejo pri načrtovanju, kodiranju in testiranju (tehnično osebje in



Slika 3: Principi XP podpirajo drug drugega

poslovni svet). V kadrovski strukturi XP vpeljuje dve novi funkciji: inštruktor (vodja skupine razvijalcev) in sledilec (oseba, ki beleži potek izvajanja projekta; izvaja metrike, spremlja izvedbo glede na načrte).

Delovni prostor mora poleg posameznih zahtev razvijalcev (in nasploh vseh sodelujočih) zadoščati tudi nekaterim posebnim zahtevam XP: velik skupni razvojni prostor z razporeditvijo računalnikov, ki omogoča skupinski razvoj (kodiranje, testiranje), programiranje v parih; velika namizna površina za igro načrtovanja, uporabo kartic CRC (Collaborator Responsibility Class) – [13]; ločen integracijski računalnik. Vedno mora biti na razpolago dovolj hrane in pijače, ugodna pa je tudi možnost počitka ali celo spanja.

Razvojno orodje naj bo izbrano v skladu z metodologijo in prepričanjem razvijalcev. Vsekakor je pomembno, da nobenega segmenta orodja (CASE orodij, podatkovne baze, testnih ogrodij, ...) vodstvo ne vsili razvojni skupini, ampak da skupina argumentira vodstvu izbiro orodij in mu svetuje pri tovrstnih odločitvah.

4. Psihološki vidik vpeljave XP

Uvajanje tovrstne discipline vsekakor spremeni obnašanje in mišljenje marsikaterega poslovnega subjekta, ki sodeluje v razvoju. Poglejmo si še s te plati zahteve XP.

Ločitev tehnične in poslovne odgovornosti.

Tehnično osebje mora biti osredotočeno na probleme tehnične narave. Projekt mora biti voden s poslovnimi odločitvami, katerih osnova pa morajo biti tehnične odločitve glede ocene stroškov in rizika.

Poslovni svet naj izbere:
- obseg ali čas posameznih izdaj
- relativno prioriteto predlaganih potreb
- natančen obseg predlaganih potreb.

Razvojni oddelek pa mora prispevati:
- ocenjen čas, potreben za implementacijo različnih funkcionalnosti
- oceno posledic uporabe različnih tehničnih disciplin
- izbiro razvojnega procesa, ki kar najbolje ustreza osebnemu prepričanju razvijalcev in je v skladu s politiko podjetja
- izbiro principov (navedenih v prejšnjem poglavju) za validiranje ocenjenega časa, pravilnosti zbranih zahtev ipd.

Življenjska doba idealnega projekta XP

Filozofija projektov je kratka vzpostavitvena razvojna faza, ki ji sledijo leta simultane podpore in izboljševanja sistema, dokler sistem ne prinaša več poslovne vrednosti in ga je najbolje upokojiti.

Naj na tem mestu omenimo še vlogo vodstva kot izvrševalca tudi ukinitve projekta, če se ta izkaže za zgrešenega (kljub uporabi XP!).

Pravilo 20-80,

ki ga razvijalci programske opreme razumejo po načelu "80% koristi prinese 20% razvoja", XP obrne malo drugače, in pravi: producirajmo najprej 20% najbolj vredne (vredne za stranke) funkcionalnosti in jo dajmo takoj v uporabo; ostalih 80% funkcionalnosti ima za stranke nižjo prioriteto in jih zato lahko implementirano skozi kasnejše izboljševanje sistema.

Kaj bremeni XP?

Vsekakor je možnih več dejavnikov. Najbolj banalen je odpor do same ideje. Ampak pustimo ta primer ob strani.

Kot glavno breme lahko izpostavimo čustva, zlasti strah. Če se nekdo počuti pod pritiskom, ker mora povezovati vse spoznane principe XP ali pa mu je bilo dodeljenih preveč opravil (znotraj posamezne izdaje), potem dela pod pritiskom. V tem primeru se osebno počuti slabo in dela napake. Takšno delo mu je vse prej kot v veselje.

Kot nadaljnje težave lahko navedemo še zmožnost razvijanja kompleksnih sistemov na enostaven način, kot ga zagovarja XP. Težko je nekaterim posameznikom priznati neznanje ali nezadovoljivo znanje na določenem področju razvoja. Prav tako je težko podreti čustvene ovire med razvijalci ali celo znotraj njih.

Skoraj največjo nevarnost pri taki organiziranosti dela pa predstavlja dejstvo, da ima lahko majhen problem velike posledice.

Potrebne so izkušnje in pogum!!!

Kdo naj ne udejanja XP?

Disciplina nima nekih ostrih mej uporabe. Vendar pa lahko najdemo čiste primere, ko je bolje ne poskusiti z njenim udejanjanjem:
- velike razvojne skupine
- nezaupljive stranke
- uporaba tehnologije, ki ne podpira majhnih in pogostih sprememb sistema (v takšnih sistemih je vsaka pozna sprememba namreč strahotno draga)
- okolja, kjer je povratna informacija od strank počasna (reda nekaj mesecev)
- delovno okolje ni primerno; za neprimernost zadostuje že razdelitev razvoja v 2 etaži.

5. Odateam xp-erti pri delu

V preostalih poglavjih vam bomo predstavili način, na katerega smo privzeli disciplino XP v našem podjetju. Kako smo implementirali posamezne principe, kako se je s tem spremenilo naše delo in kaj smo s tem pridobili.

Delovno okolje.

S prehodom na način življenja XP smo si pred dobrim letom dni najprej poiskali nove poslovne prostore. Naš cilj je bil velik skupen prostor, kjer ustvarjajo vsi člani razvojne skupine ves čas skupaj. Delovne računalnike smo po nekajkratnem poizkušanju razporedili tako, da je delo razvojne skupine najbolj komunikativno. Našo začetno postavitev prikazuje Slika 4a, ugotovljeno optimalno razporeditev pa Slika 4b. Delo je podrejeno programiranju in testiranju v spreminjajočih se parih.

Povsem smo ločili privatno sfero «dela» od poslovnih aktivnosti. Delovni računalniki so namenjeni izključno razvoju in komunikaciji s poslovnimi strankami. Za lastno uporabo ima vsak od razvijalcev privatni, prenosni računalnik, ki mu omogoča opravljanje vseh privatnih aktivnosti bodisi doma bodisi v poslovnih prostorih, AMPAK izven prostora razvoja.

V poslovnih prostorih je še velika miza za modeliranje z uporabo kartic CRC, za posvetovanja pri načrtovanju in za pogovore s poslovnimi strankami. Spet so privatne mize ločene od te skupne, « razvojne delovne ploskve ». Stene imamo polepljene s t.i. nalogami (orig. « task ») in pravili, ki se jih pri delu držimo.

Naj na koncu opisa prostora še omenimo, da je nujno potrebna seveda tudi hrana in pijača! To je edina stvar, ki nastopa tako na privatnih kot na razvojnih mizah.

Delo razvojne skupine.

Poglavitno vlogo pri načrtovanju novih funkcij sistema ali pri preoblikovanju obstoječih delov sistema imajo naše poslovne stranke, konkretno osebe z natančnim poznavanjem zavarovalništva. Poslovne in tehnične odločitve so vidno ločene, v skladu z opisom, podanim v poglavju 4. Vodstvo postavlja grobe časovne načrte, razvojna skupina pa jih – po končanem načrtovanju in spoznavanju zahtev – precizira in postavi naloge razvoja.

Design

Za modeliranje uporabljamo kartice CRC (« Collaborator Responsibility Class »). Sodelujejo vsi razvijalci, najpogosteje skupaj s predstavnikom poslovnega sveta. Ugotovitve (ustvarjene odgovornosti in sodelovanja) se prenesejo v implementacijo. Z uporabo kartic vsem razvijalcem postane sfera, ki jo abstrahirajo, popolnoma jasna. V primeru, da kartice ne zadostujejo, se razvijalci lotimo testov. Ti zagotovo pojasnijo še vsako morebitno nejasnost, saj če znaš napisati test za del sistema, tudi razumeš, kakšne so njegove zahteve in kaj producira.

Ob igranju kartic si postavimo tudi naloge, za katere je posamezni razvijalec odgovoren (v 4. poglavju omenjena « accepted responsibility »), sestavimo tudi skupen seznam potreb po oblikovanju ali preoblikovanju (t.i. « ToDo List »). Ocenimo število idealnih programerskih dni, potrebnih za izpeljavo projekta ter z upoštevanjem obremenilnega faktorja (ki zajema vse dejavnike, ki nas pri razvoju motijo: vzdrževanje, študij, ...) ocenimo dejanski čas izpeljave projekta.

Testiranje

Tako kot programiranje tudi testiranje poteka v parih. Testiranje obsega teste enote (« Unit tests »), ki jih izvajamo razvijalci za vsako razvito enoto sistema – pogosto pa tudi pred samo implementacijo – ter funkcijske teste (« Functional tests »), ki jih napišejo sodelujoči iz poslovnega sveta, da validiramo razviti sistem.

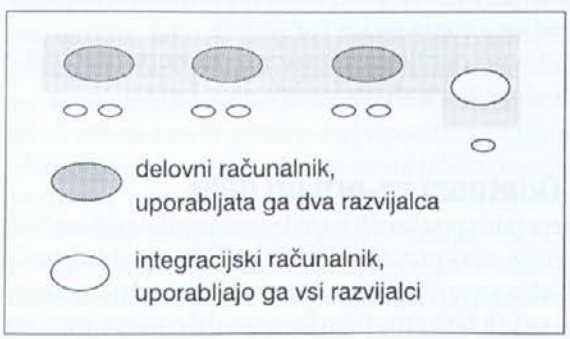
Pri testiranju si pomagamo z orodji SUnit in JUnit, predstavljenimi v lanskoletni predstavitvi [12].

Kodiranje

Kodiranje poteka zmeraj v parih, dva programerja za enim računalnikom. Dosledno udejanjamo načela, opisana v tretjem poglavju : uporaba vzorcev, metafor, pravil kodiranja. Koda mora biti tista, ki komunicira.

Proces kodiranja zahteva spremljanje izvajanja zastavljenih nalog, spoznanih bodisi v času načrtovanja bodisi v predhodnih fazah kodiranja. V ta namen imamo osebo, imenovano sledilec (tracker).

Integracija kode poteka pogosto, tudi večkrat dnevno.



Slika 4a: Prvotna zamisel razporeditve delovnih računalnikov

Slika 4b: Optimalna razporeditev delovnih računalnikov

Vzdrževanje
je dejansko lažje, če sistem pokaže v svojem delovanju manj napak. Logično. Če stremimo k zagotavljanju kakovosti sistema skozi ves razvoj in udejanjamo položno krivuljo strošek/čas, je vzdrževanja manj in je cenejše. Pa smo spet pri enem od "ekstremov".

6. Sadovi uporabe XP

Po dobrem letu dni udejanjanja discipline imamo v celotnem poslovanju (ne samo znotraj razvojne skupine) zelo pozitivne izkušnje.

Resda smo se morali privaditi na precej specifičen način dela. Morda je še največ preglavic razvijalcem povzročal princip ločitve privatnih aktivnosti od poslovnega dela. Naš način dela, izhajajoč iz objektne usmerjenosti, se ni kaj dosti spremenil. Principi XP so nam že dalj časa znani. Morda se jih nismo toliko zavedali ali jih tako dosledno uporabljali.

Pridobitve so naslednje :

- Homogena razvojna skupina, ki tesno sodeluje z vodstvom in s strankami.
- Prijateljski odnosi krepijo spoštovanje in odnos pri delu; odnos do sodelavcev, do strank.
- Uvajanje novih sodelavcev je enostavno, saj se skoraj takoj lahko vključijo v delo, seveda kot par izkušenejšemu razvijalcu.
- Razvoj je hitrejši, saj je komunikacija med razvijalci ter komunikacija razvijalec-stranka ena poglavitnih vrednot discipline.
- Avtomatski testi omogočajo razvoj z večjo samozavestjo, ni bojazni pred spreminjanjem sistema.
- Poslovne stranke so bolj zadovoljne, saj aktivno sodelujejo pri načrtovanju in nam s tem posredujejo svoje zahteve in želje po izboljšavah.
- Razvojni cikli so kratki, sistem se razvija v manjših, kontroliranih etapah.
- Naše vodstvo in razvojna skupina skupaj popolnoma zaupamo disciplini. Privadili smo se na novi način dela. Rezultati nas spodbujajo k nadaljnjemu razvijanju tega načina dela, izpopolnjevanju posameznih načel, k prilagajanju načel za naše potrebe in tudi k prilagajanju nas samih tem načelom.

Slabosti:
Težave so seveda – kot pri drugih panogah – psihološke narave. Če se razvijalci ne strinjajo z načeli discipline, je bolje, da je ne udejanjajo.

Zdaj, ko ste prebrali ta prispevek, imate dve možnosti; XP lahko vzljubite ali pa se vam bo vse skupaj zdelo nesmiselno. Če stavite na prvo možnost, vam v referencah podajamo izvrstno izhodišče za globlje spoznavanje discipline. Če se vam zdi vaš dosedanji način dela prikladnejši, pa ste dobili svež pogled na razvoj programske opreme.

REFERENCE

1. BECK Kent, "Extreme programming explained", Addison Wesley, 1999
2. BECK Kent, "Guide to Better Smalltalk", Cambridge University Press, 1999
3. BECK Kent, "Smalltalk Best Practice Patterns", Prentice Hall, 1997
4. GABRIEL Richard P., "Patterns of Software", Oxford University Press, 1996
5. GAMMA Erich et.al., "Design Patterns", Addison-Wesley, 1995
6. WIRFS-BROCK Rebecca et.al., "Designing Object-Oriented Software", Prentice-Hall, 1990
7. BROWN William J. et.al., "Anti Patterns", John Wiley & Sons, 1998
8. ROBERTS Don et.al., "Why Every Smalltalker Should Use the Refactoring Browser", The Smalltalk Report, letnik 6, številka 10, september 1997
9. WILKINSON Nancy M, "Using CRC Cards", SIGS Books, 1995
10. C3 Team, Chrysler Corp., "Case Study, Chrysler Goes to Extremes", Distributed Computing, oktober 1998
11. DeMARCO Tom, "Peopleware", Dorset House 1999
12. GRAJFONER Uroš, "Ponovna uporaba (Refactoring)", Zbornik srečanja Objektna tehnologija v Sloveniji 2000, junij 2000
13. GRAJFONER Uroš, REPINC Peter, "Testiranje OO sistemov", Zbornik srečanja Objektna tehnologija v Sloveniji 99, junij 1999
14. ROSTAHER Matevž, KLINE Andrej "Proces skupinskega razvoja programske opreme (Extreme programming)", Zbornik srečanja Objektna tehnologija v Sloveniji 99, junij 1999
15. http://www.c2.com/cgi/wiki?ExtremeProgramming, Extreme Programming Discussion
16. http://www.xprogramming.com, XP Magazine
17. Mailing lista: *extremeprogramming@egroups.com*
18. http://www.extremeprogramming.org

Matevž Rostaher je študiral telematiko na Tehnični univerzi v Grazu. Študent računalništva in informatike na Univerzi v Mariboru in od leta 1992 zunanji sodelavec podjetja OdaTeam d.o.o., kjer se je ukvarjal z analizo, načrtovanjem in s programiranjem informacijskih sistemov z uporabo objektne metodologije in razvojnega orodja Visual Smalltalk in Java. Od leta 2000 projektni vodja v podjetju FJA OdaTeam d.o.o. Sodeloval je v pri uvedbi ekstremnega programiranja v proces razvoja informacijskih sistemov.

Ivan Slamek, študent računalništva in informatike na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru. Dela kot razvijalec informacijskih sistemov v podjetju FJA OdaTeam d.o.o. Sodeloval je pri razvojnih projektih v Smalltalku in Javi ter pri uvajanju prvin ekstremnega programiranja v proces razvoja informacijskih sistemov.

Andrej Kline je diplomiral na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru, kjer je zagovarjal diplomsko nalogo »Uporaba Jave pri zagotovitvi varnosti oddaljene povezave«. Zdaj je študent podiplomskega študija na tej fakulteti in razvijalec informacijskih sistemov v podjetju FJA OdaTeam d.o.o. Sodeloval je pri razvojnih projektih v Smalltalku in Javi in pri uvajanju prvin ekstremnega programiranja v proces razvoja informacijskih sistemov.