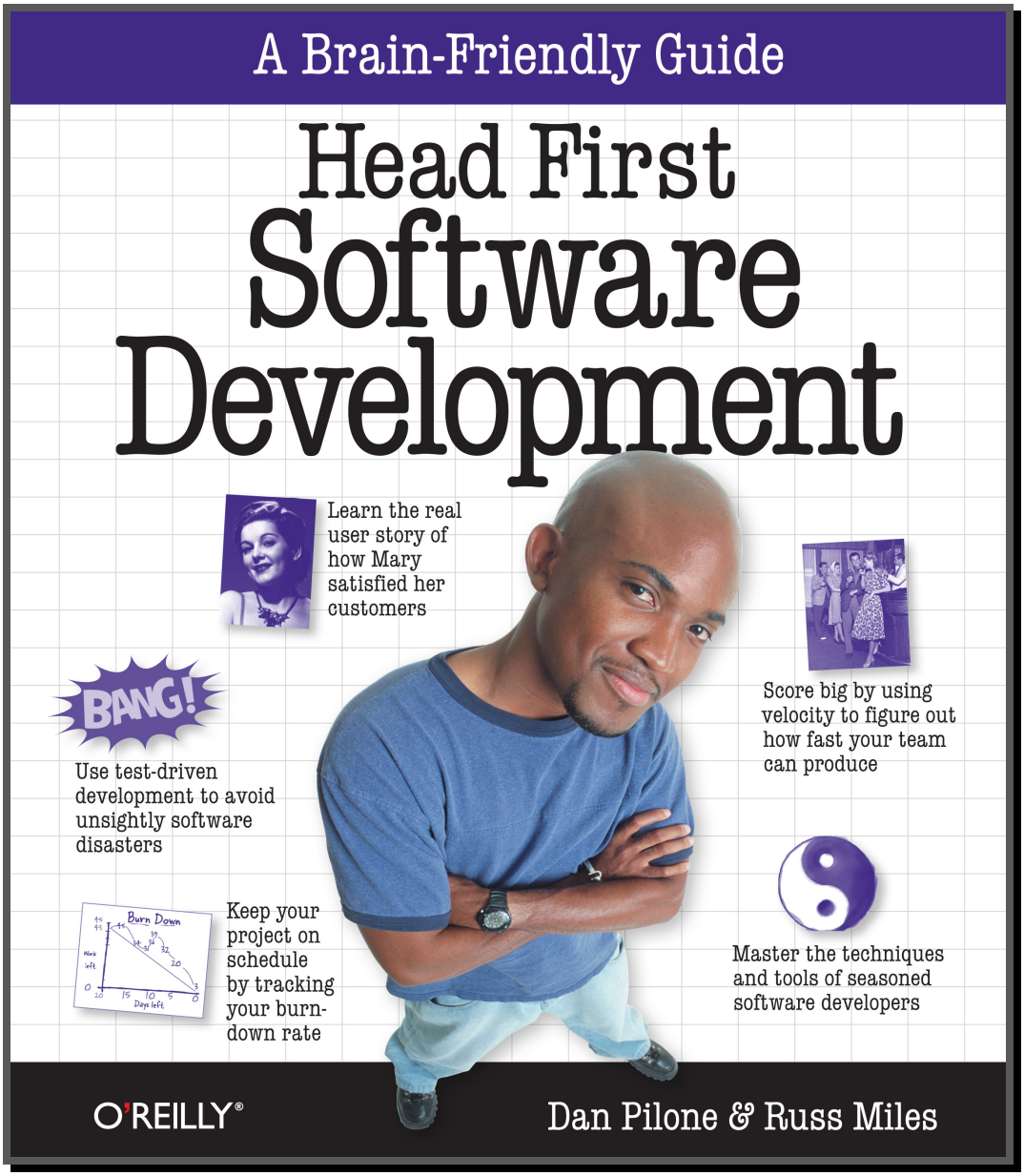
Zhvillimi i Softuerit

konceptet kryesore



përkthyer dhe përshtatur nga Ridvan Bunjaku

Prishtinë, dhjetor 2009

Përmbajtja

Hyrje

Këtu janë pikat kryesore nga libri “Head First Software Development”, një udhëzues në procesin e zhvillimit të softuerit në grup, që është miqësor për trurin. Megjithëse libri është voluminoz (me afër 500 faqe), dendësia e tij është e vogël, me faqe plot figura, vizatime e komente qesharake, me shembuj atraktivë, dhe krejt kjo nën një themel të fuqishëm profesional mbi lëminë. Libri është i dobishëm për të gjithë zhvilluesit softuerikë dhe sidomos për menaxherët e projekteve softuerike.

Stili i librave të serisë Head First është atraktiv, i këndshëm dhe e bën të lehtë absorbimin e lëndës që në fakt nuk është e lehtë.

Ridvan Bunjaku

18 dhjetor 2009

1. Zhvillimi i softuerit të shkëlqyeshëm - Kënaqja e klientit tuaj

- **Feedbacku** që vjen nga secili iteracion është vegla më e mirë për me u siguru që softueri juaj i plotëson nevojat e klientëve tuaj.
- **Iteracioni** është një projekt i plotë në **miniaturë**.
- Softueri i suksesshëm nuk zhvillohet në vakum. Ai ka nevojë për **feedback të vazhdueshëm** nga klienti juaj duke i përdorë iteracionet.
- Zhvillimi i mirë i softuerit dorëzon softuer të shkëlqyeshëm, **në kohë** dhe **brenda buxhetit**.
- Është gjithmonë më mirë me i dorëzu **disa** nga veçoritë që **punojnë perfekt** sesa të gjitha veçoritë që nuk punojnë qysh duhet.
- Zhvilluesit e mirë zhvillojnë softuer; **zhvilluesit e shkëlqyeshëm** e dorëzojnë si duhet softuerin.

Teknikat e zhvillimit

- ❑ Iteracioni ju ndihmon me qëndru në drejtimin e duhur
- ❑ Planifikoni dhe balansoni iteracionet tuaja atëherë kur (jo nëse) ndodh ndryshimi
- ❑ Çdo iteracion rezulton në softuer që punon dhe merr feedback nga klienti juaj në çdo hap të rrugës

Principet e Zhvillimit

- ❑ Dorëzoni softuer që nevojitet
- ❑ Dorëzojeni softuerin me kohë
- ❑ Dorëzojeni softuerin brenda buxhetit

2. Mbledhja e kërkesave - Të ditunit se çka don klienti

- **Qiellkaltrimi** (blueskying) e bën klientin tuaj me mendu në mënyrë të madhe (think big) kur ua jep kërkesat e veta.
- **Tregimi i përdoruesit** (user story) e përthekon një ndërveprim me softuerin nga perspektiva e përdoruesit.
- Tregimet e përdoruesve duhet me qenë **të shkurta**, rreth tri fjali të gjata.
- Tregimi i shkurtë i përdoruesit është **tregim i matshëm i përdoruesit**.
- Tregimi i përdoruesit nuk duhet me ia marrë një zhvilluesi më shumë se 15 ditë për me dorëzu.
- **Zhvilloni iterativisht** kërkesat tuaja me klientin për me mbajtë atë në qark në çdo hap të procesit.

Teknikat e Zhvillimit

- ❑ Qiellkaltrimi, Vëzhgimi dhe Luajtja e roleve
- ❑ Tregimet e përdoruesve
- ❑ Pokeri i planifikimit për vlerësim

Principet e Zhvillimit

- ❑ Klienti e di se çka don, por nganjëherë duheni me i ndihmu me definu qartë këtë
- ❑ Bëjini kërkesat me qenë të orientuara kah klientët
- ❑ Zhvilloni dhe përpunoni kërkesat tuaja iterativisht me klientin

3. Planifikimi i projektit – Planifikimi për sukses

- Hapi i parë në planifikimin se çka do të zhvilloni është me kërku nga klienti **me i prioritizu kërkesat e veta**.
- **Guri Kilometrik (Milestone) 1.0** duhet me u dorëzu **sa më herët** që mundeni.
- Gjatë Gurit Kilometrik 1.0 provoni **me iteru rreth një herë në muaj** për me mbajtë në vijë punën tuaj të zhvillimit.
- Kur nuk keni mjaft kohë me ndërtu çdo gjë, kërkoni nga **klienti** me **riprioritizu**.
- Planifikoni iteracionet tuaja duke e futë në llogari **shpejtësinë** e ekipit tuaj që nga **fillimi**.
- Nëse vërtet nuk mundeni me bë atë që nevojitet në kohën e lejuar, **jini të sinqertë** dhe **shpjegojani pse-në** klientit.
- Posa të jeni pajtu për një grup të arritshëm të tregimeve të përdoruesve për Gurin Kilometrik 1.0, është koha me konstuktu **tabelën tuaj të kontrollit** (dashboard-in) dhe me fillu me zhvillu!
- **Klienti juaj e prioritizon** se çka ka brenda dhe çka ka jashtë Gurit Kilometrik 1.0.
- Ndërtoni **iteracione të shkurta** rreth 1 muaj kalendarik, **20 ditë pune kalendarike**.
- Nëpër një iteracion softueri juaj duhet me qenë **i ndërtueshëm** (buildable) dhe **i ekzekutueshëm**.
- Aplikojeni **shpejtësinë** e ekipit tuaj në vlerësimet tuaja për me i ra në fije se sa punë saktësisht mundeni **me menaxhu realisht** në iteracionin tuaj të parë.
- Mbajini klientët tuaj të lumtur duke dalë me një Gur Kilometrik 1.0 që **mundeni me arritë** ashtu që mundeni me qenë konfidentë për me dorëzu dhe me u pagu. Pastaj nëse dorëzoni më shumë, ata do të jenë edhe më të lumtur.

Teknikat e Zhvillimit

- Iteracionet idealisht duhet me qenë jo më të gjata se një muaj. Kjo domethënë se i keni 20 ditë pune për iteracion.
- Aplikimi i shpejtësisë në planin tuaj ju lejon me u ndi më konfidentë në aftësinë tuaj me i mbajtë premtimet e zhvillimit ndaj klientit tuaj.
- Përdoreni një tabelë të madhe (në kuptimin e vërtetë) në murin tuaj për me planifiku dhe me monitoru punën e iteracionit tuaj aktual.
- Inkuadrojeni klienin kur zgjedhni se çfarë tregime të përdoruesve munden me u kry për Gurin Kilometrik 1.0, dhe kur zgjedhni se në cilin interacion do të ndërtohet një tregim i përdoruesit.

Principet e Zhvillimit

- Mbajini iteracionet të shkurta dhe të menaxhueshme
- Në fund të fundit, klienti vendos se çka ka brenda dhe çka ka jashtë Gurit Kilometrik 1.0
- Premto, dhe dorëzo
- GJITHMONË ji i ndershëm me klientin

4. Tregimet e përdoruesve dhe detyrat - Fillimi i punës së vërtetë

- Organizoni **takime ditore në këmbë** (daily standup meetings) për me u siguru që i trajtoni çështjet herët.
- Mbani takimet në këmbë për **më pak se 15 minuta**.
- Takimi në këmbë ka të bëjë me **progresin**, **çështjet problematike**, dhe **freskimin e tabelës suaj**.
- Provoni me i planifiku takimet tuaja në këmbë për në **mëngjes** ashtu që secili e din se ku qëndron në **fillim të ditës së punës**.

(5. Dizajni mjaft i mirë – Kryerja me dizajn të shkëlqyeshëm)

6. Kontrolli i versioneve – Zhvillimi mbrojtës

- Insektet (bugs) në versionet e publikuara janë zakonisht prioritet më i lartë për klientin se sa implementimi i veçorive të reja.
- Rregullimet tuaja të insekteve duhet me afektu softuerin e publikuar dhe me u implementu edhe në versionet në-progres të softuerit tuaj.
- Rregullimi efektiv i insekteve varet nga aftësia me i lokalizu versionet specifike të softuerit tuaj dhe me i bë ndryshimet në ato versione pa e afektu zhvillimin aktual.
- **Valixhja** (trunk) është vendi ku duhet me shku zhvillimi aktiv; ajo duhet me prezentu gjithmonë versionin e fundit të softuerit tuaj.
- **Tagu** (etiketa) është emër që i bashkangjitet një rishikimi specifik të elementeve në depon tuaj të kodit ashtu që të mundeni me iu kthy atij rishikimi më vonë.
- Nganjëherë mundet me ju dashtë **me i apliku ndryshimet e njëjta në degë (branch) dhe në valixhe** nëse ndryshimi aplikohet te të dyjat.
- **Degët** janë kopje të kodit tuaj në të cilat mundeni me bë ndryshime pa e afektu kodin në valixhe. Degët zakonisht fillojnë nga një version i taguar (i etiketuar) i kodit.
- **Tagat janë statikë** - ju nuk zbatoni ndryshime në ta. **Degët** janë për **ndryshimet që nuk i doni në valixhe** (apo për me mbajtë kodin larg ndryshimeve që bëhen në valixhe).
- **Bëjeni backup** depon tuaj të kontrollit të versioneve! Ajo duhet me pasë krejt kodin tuaj dhe historinë e ndryshimeve në të.
- Gjithmonë përdorni **mesazh të mirë të zbatimit** kur e zbatoni (commit) kodin tuaj - ju dhe ekipi juaj do ta çmoni këtë më vonë.
- **Përdoni tagat lirshëm.** Nëse ka ndonjë çështje lidhur me nevojën me ditë se qysh ka qenë kodi para një ndryshimi, etiketojeni (tagojeni) atë version të kodit tuaj.
- **Zbatoni** (commit) **shpesh** në depo të kodit, mirëpo kini kujdes mos me prishë kodin e njerëzve tjerë. Sa më shumë kohë kaloni ndërmjet zbatimeve, aq më të vështira do të jenë bashkimet.
- Ka shumë **vegla GUI** për sistemet e kontrollimit të versionit. Ato ndihmojnë shumë me bashkimet dhe me trajtimet e konflikteve.

Teknikat e zhvillimit

- ❑ Përdorni vegël të kontrollimit të versionit për me i përcjellë dhe me i shpërnda ndryshimet në softuerin tuaj - te ekipi juaj.
- ❑ Përdorni tagat (etiketat) për me i përcjellë gurët e mëdhenj kilometrik në projektin tuaj (përfundimet e iteracioneve, publikimet, rregullimet e insekteve, etj.)
- ❑ Përdorni degët për me mirëmbajtë kopje të veçantë të kodit tuaj, por degëzoni vetëm kur është absolutisht e nevojshme.

Principet e zhvillimit

- ❑ Gjithmonë dijeni se ku duhet (dhe ku nuk duhet) me shku ndryshimet
- ❑ Dijeni se çfarë kodi ka shku në një publikim të caktuar - dhe jini të aftë me u kthy te ai përsëri
- ❑ Kontrollojeni ndryshimin dhe shpërndarjen e kodit

6 ½ Ndërtimi i kodit tuaj – Fute tabin a në slotin b...

- Vegla e ndërtimit (build) është thjesht **vegël**. Ajo duhet me bë ndërtimin e projektit tuaj **më të lehtë**, jo më të vështirë.
- Shumica e veglave të ndërtimit përdorin **skriptë të ndërtimit**, ku mundeni me specifiku çka me ndërtu, disa grupe të ndryshme të instruksioneve, dhe lokacionet e fajllave dhe resurseve të jashtme.
- Sigurohuni që me kriju mënyrë për **me i pastru** të gjithë fajllat që i krijon skripta juaj.
- Skripta juaj e ndërtimit është **kod** dhe duhet me u verzionu dhe me u regjistru (check-in) në depon tuaj të kodit.
- **Veglat e ndërtimit janë për ekipin tuaj**, jo vetëm për ju. Zgjedhni vegël të ndërtimit që funksionon për secilin në ekipin tuaj.

- Të gjitha projektet, pos të voglat, kanë **proces jotrivial të ndërtimit**.
- Ju doni me **regjistru** dhe me **automatizu** njohurinë se **qysh me ndërtu sistemin tuaj** - idealisht me një komandë të vetme.
- **Ant** është vegël për ndërtim për **projektet Java** dhe i regjistron informatat e ndërtimit në një fajll XML të quajtur build.xml.
- Sa më shumë i përdorni përparësitë e **konventave të zakonshme**, aq më **familjar** do t'i duket projekti juaj dikuj tjetër, dhe aq më lehtë do të jetë me integru projektin me vegla të jashtme.
- **Skripta juaj e ndërtimit** është po aq pjesë e projektit tuaj si edhe **cilado pjesë tjetër e kodit**. Ajo duhet me u regjistru në kontrollin e versioneve me çdo gjë tjetër.

Teknikat e zhvillimit

- Përdorni vegël të ndërtimit për me skriptu ndërtimin, paketimin, testimin, dhe instalimin e sistemit tuaj.
- Shumica e IDE-ve përdorin tashmë vegël të ndërtimit. Bëhuni familjarë me atë vegël, dhe mundeni me ndërtu mbi atë që IDE e ka të gatshme.
- Trajtojeni skriptën tuaj të ndërtimit sikurse kodin dhe regjistrojeni në kontroll të versioneve.

Principet e zhvillimit

- Ndërtimi i projektit duhet me qenë i përsëritshëm dhe i automatizuar
- Skriptat e ndërtimit e rregullojnë skenën për veglat tjera të automatizimit
- Skriptat e ndërtimit shkojnë përtej automatizimit vetëm hap-pas-hapi dhe munden me pasë edhe vendime të kompajlimit dhe të logjikës së instalimit

7. Testimi dhe integrimi i vazhdueshëm – Gjërat thehen

- Përdorimi i veglave të **Integrimit të Vazhdueshëm** domethënë se diçka është gjithmonë duke e shiku kualitetin e kodit në depo.
- **Testimi i automatizuar** mundet me qenë adiktiv (me kriju varësi). Ende ju bjen me shkru kod, prandaj është argëtuese. Dhe nganjëherë i prishni gjërat. Edhe kjo është argëtuese.
- Bëjini **publike për ekipin** rezultatet e ndërtimeve të vazhdueshme të integrimit dhe të raporteve të mbulimit - ekipi është pronar i projektit dhe duhet me u ndi përgjegjës.
- Bëjeni veglën tuaj të integrimit të vazhdueshëm që **ta dështojë një ndërtim** nëse dështon një test i automatizuar. Pastaj bëjeni që **t'i dërgojë email zbatuesit** (committer) derisa ta rregullojë atë.
- Testimi i **funksionalitetit të tërësishëm** është kritik për me deklaru një projekt si projekt që funksionon.

Teknikat e zhvillimit

- ❑ Ka pamje të ndryshme të sistemit tuaj, dhe duheni me i testu të gjitha ato
- ❑ Testimi duhet me i marrë parasysh rastet e suksesshme si dhe rastet e dështimit
- ❑ Automatizojeni testimin kurdo që është e mundur
- ❑ Përdorni vegël të integrimit të vazhdueshëm për me automatizu ndërtimin dhe testimin e kodit tuaj në secilin zbatim (commit)

Principet e zhvillimit

- ❑ Testimi është vegël që jua bën me dije se ku është projekti juaj në çdo kohë
- ❑ Integrimi i vazhdueshëm jua jep konfidencën që kodi në depon tuaj është korrekt dhe ndërtohet (builds) si duhet
- ❑ Mbulimi i kodit (code coverage) është metrikë shumë më e mirë e efektivitetit të testimit se sa numërimi i testeve

8. Zhvillimi i drejtuar nga testet - Mbajtja e kodit tuaj të llogaritshëm (accountable)

- TDD (Test-driven development) domethënë se do të **rifaktoroni kod** shumë. Keni prishë diçka shumë keq? Thjesht përdoreni veglën tuaj të kontrollit të versioneve për me u rrokullisë prapa (roll back) aty ku keni qenë më herët dhe provoni përsëri.
- Nganjëherë testimi do të **ndikojë në dizajnin tuaj** - jini të vetëdijshëm për vendimet (trade-off-at) dhe zgjedhni me kujdes a ia vlen **testueshmëria e rritur**.
- Përdoreni **paternin strategji** me **injektimin e varësive** për me ndihmu në **ndarjen e klasave** (shçiftimin, decoupling).
- Ruani testet tuaja në **strukturë paralele** me kodin tuaj, siç është follderi `testet/`. Shumica e veglave të ndërtimit dhe të testimit të automatizuar funksionojnë mirë me këtë konfiguracion.
- **Përpiquni ta keni të shkurtë kohën e ndërtimit dhe të ekzekutimit të testit** ashtu që ekzekutimi i suitës së plotë të testeve të mos jua ndalë shpejtësinë e zhvillimit.

Teknikat e zhvillimit

- ❑ Shkruani së pari testet, pastaj kodin që i kalon ato teste
- ❑ Testet tuaja duhet me dështu fillmisht; pastaj pasi t'i kaloni ato mundeni me rifaktoru
- ❑ Përdorni objekte të rrejshme (mock objects) për me ofru variacione për objektet që ju nevojiten për testim

Principet e zhvillimit

- ❑ TDD ju detyron me u përqendru në funksionalitet
- ❑ Testet e automatizuara e bëjnë rifaktorimin më të sigurtë; do ta dini menjëherë nëse keni prishë diçka
- ❑ Mbulimi i mirë i kodit është shumë më i arritshëm në qasje TDD

9. Përfundimi i një iteracioni – Po vijnë krejt bashkë...

- Nëse keni hapësirë në fund të iteracionit, kjo është kohë e mirë për **me diskutu për ide** (brainstorm) **për tregimet e reja** që kanë mundë me ardhë. Edhe ato duhet me u prioritizu me gjithçka tjetër, por është mirë me i regjistru.
- **Rezistoni joshjes** me harru për krejt shprehitë tuaja të mira në ditën e fundit apo të parafundit të iteracionit. Mos e "përvjedhni" atë veçori të shpejtë që ka prioritet të vogël veç sepse e keni një ditë, apo me e bë atë rifaktorim të vogël për të cilin jeni "të sigurtë që nuk ka me prishë asgjë." Keni punu vërtet fort për me kry një ditë a dy më herët, mos e prishni.
- Punoni fort për me pasë **marrëdhënie të shëndoshë** me **ekipin tuaj të testimit**. Dy ekipet munden me bë njëri-tjetrin mizerje (të trishtuar) nëse komunikimi shkon keq.
- Regjistrimi i kohës aktuale të kaluar në një detyrë ndaj kohës së vlerësuar për një detyrë nuk është i nevojshëm meqë shpejtësia juaj do t'i llogarisë edhe gabimet e vlerësimit. Por, nëse e dini se diçka ka shku vërtet keq, **ia vlen me diskutu në rishikimin e iteracionit**.

Teknikat e zhvillimit

- Kushtojini kujdes ritmit tuaj të djegies (burn-down rate) - sidomos pasi të kryhet iteracioni
- Ecja e iteracionit është e rëndësishme - hiqni tregime nëse ju duhet me vazhdu iteracionin
- Mos i ndëshkoni njerëzit që e kanë kry punën herët - nëse gjërat e tyre punojnë, lejini me përdorë kohën shtesë për me vazhdu tutje apo me mësu diçka të re

Principet e zhvillimit

- Iteracionet janë mënyrë për me i imponu afatet e ndërmjetme - përmbajuni atyre
- Gjithmonë llogariteni ditën ideale për anëtarin mesatar të ekipit
- Mbajeni në mendje pikturën e madhe kur i planifikoni iteracionet - dhe kjo mundet me përfshi edhe testimin e jashtëm të sistemit
- Përmirësojeni procesin tuaj iterativisht nëpër rishikimet e iteracioneve

10. Iteracioni tjetër – Nëse nuk je thy... prapë më mirë rregulloje

- Kur kaloni në iteracionin tjetër, gjithmonë **kontrolloni me klienten** për me u siguru që puna që jeni duke e planifiku është puna që ajo don me u kry.
- **Shpejtësia** juaj dhe e ekipit tuaj **rillogaritet** në fund të secilit iteracion.
- Lejojeni klientin tuaj **me i riprioritizu tregimet e përdoruesve** për iteracionin e ri, duke u bazu në ditët e punës që i keni në dispozicion për atë iteracion.
- Pavarësisht a jeni duke shkru kod të ri apo duke e përdorë kod të dikujt tjetër **ende i tëri është vetëm softuer** dhe procesi juaj mbetet i njëjtë.
- Çdo pjesë e kodit në softuerin tënd, pavarësisht a është kodi juaj apo i një pale të tretë si Ushqimet Merkur, duhet me u përfaqësu nga të paktën një tregim i përdoruesit.
- **Kurrë mos supozoni asgjë** për kodin që jeni duke e ripërdorë.
- Një interfejs i shkëlqyeshëm ndaj një librarie të kodit nuk është garancion se kodi punon. **Mos i besoni asgjëje** derisa nuk e keni pa vetë duke punu.
- **Kodi shkruhet një herë por lexohet (nga tjerët) shumë herë.** Trajtojeni kodin tuaj ashtu siç do ta trajtonit çfarëdo pjesë tjetër të punës që ia prezentoni njerëzve tjerë. Duhet me qenë i lexueshëm, i besueshëm dhe i lehtë me u kuptu.

11. Insektet – Ndrydhja e insekteve si profesionalist

- Çdo gjë rrotullohet rreth **funksionalitetit të orientuar kah klienti**.
- Ju e shkruani dhe e rregulloni kodin për m'i **kënaqë tregimet e përdoruesve**.
- Ju **rregulloni vetëm çka është e prishur**, dhe e dini se çka është e prishur sepse keni **teste që dështojnë**.
- **Testet janë rrjeta juaj e sigurisë.** Ju i përdorni testet për me u siguru që nuk keni prishë asgjë dhe për me ditë se kur keni rregullu diçka.
- Nëse **nuk ka test** për një pjesë të funksionalitetit, atëherë është njëjtë si me thënë se funksionaliteti është i prishur.
- Derisa kodi i bukur është gjë e shkëlqyeshme, ***kodi funksional triumfon ndaj kodit të bukur secilën herë***. Kjo nuk domethënë me i lënë gjërat me mbetë të leckosura, por gjithmonë mbajeni në mend se pse po punoni në këtë kod në radhë të parë: **për klientin**.

- Para se me ndryshu edhe një rresht të vetëm të kodit, **merreni pronësinë** ndaj tij duke e shtu atë në procesin tuaj të ndërtimit dhe duke e vendosë nën menaxhment të kodit burimor.
- **Merreni përgjegjësinë** për krejt kodin në softuerin tuaj. Nëse shihni problem, atëherë mos klithni "është kodi i dikujt tjetër"; shkruajeni një test, pastaj **rregullojeni**.
- Për asnjë rresht të vetëm të **kodit mos supozoni** se **funksionon** derisa nuk ka një **test** që e vërteton këtë.
- **Softueri që funksionon** vjen i pari; **kodi i bukur** është i dyti.
- Përdoreni **testin e krenarisë**. Nëse do të ishit të lumtur që dikush tjetër e lexon kodin tuaj dhe do të mbështetej në softuerin tuaj, atëherë me gjasë kodi është në formë të mirë.

Teknikat e zhvillimit

- ❑ Para se ta ndryshoni edhe një rresht të vetëm të kodit, sigurohuni që është i kontrolluar dhe i ndërtueshëm
- ❑ Kur insektet e godasin kodin që nuk e njihni, përdoreni një test të shpejtë (spike test) për me vlerësu se sa kohë do të marrë për me i rregullu ata
- ❑ Futeni në llogari edhe konfidencën e ekipit tuaj kur e vlerësoni punën e mbetur për me i rregullu insektet
- ❑ Përdorni testet për me ju tregu se kur është rregullu një insekt

Principet e zhvillimit

- ❑ Jini të ndershëm me klientin tuaj, sidomos kur lajmet janë të këqija
- ❑ Softueri që funksionon është prioriteti juaj kryesor
- ❑ Kodi i lexueshëm dhe i kuptueshëm vjen si prioritet i dytë i afërt
- ❑ Nëse nuk e keni testu një pjesë të kodit, supozoni se nuk punon
- ❑ Rregullojeni funksionalitetin
- ❑ Jini krenar për kodin tuaj
- ❑ Krejt kodi në softuerin tuaj, edhe bitat që nuk i keni shkruar ju, janë përgjegjësi e juaja

12. Bota reale - Të pasunit e një procesi në jetë

- **Merreni mendimin e ekipit tuaj** sa herë që do të bëni ndryshime në proces; edhe ata duhet me jetu me ndryshimet tuaja.
- Çdo ndryshim i procesit duhet me u shfaqë **dy herë**: një herë **me vendosë me bë** dhe një herë **me vlerësu a ka funksionu a jo**.
- Shmangiuni **më shumë se një vendi për me i ruajtë kërkesat**. Është gjithmonë ankth i mirëmbajtjes.
- Jini **skeptikë** për proceset magjike, **nga-kutia**. Secili projekt ka diçka **unike**, dhe procesi juaj duhet me qenë fleksibil.

Teknikat e zhvillimit

- ❑ Vlerësoni në mënyrë kritike të gjitha ndryshimet në procesin tuaj, me metrika reale
- ❑ Formalizoni dokumentet tuaja nëse duhet, mirëpo gjithmonë dijeni se si po jep vlerë kjo gjë
- ❑ Mundohuni fuqishëm që procesin me ndryshu vetëm ndërmjet iteracioneve

Principet e zhvillimit

- ❑ Zhvilluesit e mirë zhvillojnë - zhvilluesit e shkëlqyeshëm dorëzojnë
- ❑ Zhvilluesit e mirë zakonisht munden me tejkalu një proces të keq
- ❑ Proces i mirë është ai që e lejon ekipin tuaj me qenë i suksesshëm