AGILE SANASTO

Sanasto kettärän kehittämisen tärkeimmistä termeistä

Contribyte

Acceptance Test Driven Development
Hyväksyntätestiohjattu kehittäminen

Tapa kehittää ohjelmistoja, missä kehittäjät ja asiakkaan edustajat (usein tuoteomistaja) kirjoittavat hyväksyntätestit ennen varsinaista ohjelmistokehittämisen aloittamista. Yhteinen hyväksyntätestien kirjoittaminen auttaa kehittäjiä ymmärtämään kehitettävältä toiminnolta vaaditut ominaisuudet.

Acceptance criteria
Hyväksyntäkriteeri

Hyväksyntäkriteerit määrittelevät, mitä kehitettävän asian tulisi tehdä ja millä tavalla sen tulisi toimia. Hyväksyntäkriteerien tulisi olla kirjoitettu "maallikkokielellä", niin että myös ei-ohjelmistokehittäjät tai testaajat ymmärtävät ne. Hyväksyntäkriteerit voivat sisältää myös määrittelyä, miten kehitettävä asia toimii vikatilanteissa.

Acceptance tests
Hyväksyntätestit

Hyväksyntätestit ovat seurausta hyväksyntäkriteereistä. Ne on kirjoitettu testin muotoon, kun hyväksyntäkriteerit ovat enemmän vapaamuotoinen määritelmä tehtävälle asialle.

Backlog
Kehitysjono

Ainoa henkilö, joka saa priorisoida asioita on tuoteomistaja. Kehitystiimi aloittaa jonon kärjessä olevia asioita. Jonon kärjessä olevien asioiden pitäminen "aloitusvalmiina" eli hyvin määriteltyinä, keskusteltuina ja tarpeeksi pieninä on tuoteomistajan vastuulla.

Backlog Grooming
Kehitysjonon jalostaminen

Kehitysjonon laatua parantava, yleensä säännöllisesti toistuva kokous. Tässä kokouksessa tuoteomistaja ja kehitystiimi käyvät läpi kehitysjonolla olevia asioita, ja keskustelevat ja parantavat asioiden määritystä ja hyväksyntäkriteerejä.

Backlog Item
Kehitysjonolla oleva asia

Yksittäinen asia kehitysjonolla. Tämä voi olla EPIC, Käyttäjätarina (user story), tehtävä (task), korjausta vaativa virhe (bugi) tai muu asia.

Burndown -kaavio
Jäljellä oleva työ -kaavio

Kaaviossa on aika X-akselilla ja jäljellä oleva työ Y-akselilla. Voidaan käyttää tuotteelle, releaselle tai Sprintille.

Burnup -kaavio
Valmiiksi saatu työ -kaavio

Kaaviossa on aika X-akselilla ja valmiiksi saatu työ ja suunniteltu työ Y-akselilla. Voidaan käyttää tuotteelle, releaselle tai Sprintille.

Branch
Haara

Ohjelmiston haara. Piste, jossa eri versiot ohjelmistossa lähtevät eroamaan toisistaan. Mitä enemmän haaroja ohjelmistotuotteesta on, sitä hankalammaksi käy tuotteen ylläpito. Mitä lähempänä pääkehityshaara on todistettua tuotannon laatua, sitä helpommin tiimi pystyy toimittamaan tuotantoon versioita. Täydellisesti automatisoitu testaus voi mahdollistaa myös jatkuvan tuotantoon siirron kehityshaarasta.

Continuous Delivery
Jatkuva valmius tuotantoon siirtoon

Jokainen koodimuutos testataan automaattisesti ja toimitetaan automaattisesti tuotannonkaltaiseen "staging" ympäristöön, ja on siirrettävissä tuotantoon erikseen päätettäessä "napinpainalluksella".

Continuous Deployment
Jatkuva tuotantoon siirto

Jokainen koodimuutos testataan automaattisesti ja toimitetaan tuotantoon automaattisesti.

Continuous Integration
Jatkuva integraatio

Jokainen koodimuutos integroidaan, käännetään ja yksikkötestataan kaikkien muiden koodimuutosten kanssa. Kehittäjät laittavat muutoksia sisään versionhallintaan tiheästi. Jatkuva integraatio pakottaa kehitystiimin tekemään asioita pienissä palasissa, ylläpitämään yhteensopivuutta muihin ratkaisun osiin ja ylläpitämään tuotannon kaltaista laatua myös kehityshaarassa.

Daily Scrum (tai pelkkä Daily)
Päivittäinen kokous

Lyhyt, maksimissaan 15 minuutin päivittäinen kokous, jossa kehitystiimi koordinoi Sprintin aikaista tekemistä

Definition of Done
Valmiin määritelmä

Lista asioita, jotka tiimi tai organisaatio on määritellyt sellaisiksi, mitkä pitää olla tehtynä, jotta asia voidaan katsoa olevan valmis. Samaa tuotetta tekevien ihmisten ja tiimien on syytä käyttää samanlaista valmiin määritelmää.

Definition of Ready
Työ, mikä on valmis aloitettavaksi

Tiimi ei saisi aloittaa ihan millaista tarinaa tahansa. Tarina, joka aloitetaan ilman kunnon määrittelyä, venyy ja vanuu helposti "kun on ihan kiva lisätä siihen toimintoja" ja "ehkä se tarvitsee vielä tämän ominaisuuden". Puutteellisesti määritellyt tarinat on mahdotonta sanoa milloin ne ovat valmiita! Tämä on eräs suurimpia tiimin hukkatyön syitä. Hyvät tiimit määrittelevät DoR:n jossa sanotaan esimerkiksi että asioilla mitä otetaan työn alle, on kunnon määritelmä, asiakastarve, hyväksyntäkriteerit ja tarvittaessa käyttöliittymäluonnos.

Development Team
Kehitystiimi

Scrumin kehitystiimi sisältää tiimin jäsenet ja Scrum Masterin. Product Owner ei kuulu kehitystiimiin, mutta saa norkoilla lähettyvillä.

EPIC
Iso kehitysjonolla oleva asia

Suurta kehitysjonolla olevaa asiaa voidaan kutsua EPICiksi. EPICit pitää pilkkoa pienempiin, Sprinteissä toteutettaviin palasiin ennen kuin työtä voidaan aloittaa.

Estimate
Arvio

Kehitysjonolla olevan asian toteuttamiseen arvioitu kuluva työmäärä. Arvio tehdään yleisimmin tarinapisteinä, mutta voidaan tehdä myös muilla tavoin (esimerkiksi ideaalitunteina tai -päivinä).

Extreme Programming

90-luvun puolivälissä kehittynyt ohjelmistokehityksen metodi, joka koostuu useista eri käytännöistä. Moni XP:n käytännöistä on edelleen validi ja käytössä useissa Scrum-tiimeissä. Scrum ja XP ovat usein toisiaan täydentäviä metodeja.

Impediment
Este

Yleensä liittyen tiimin Sprintille hyväksymään kehitysjonon asiaan. Tiimin jäsenet ovat törmänneet ongelmaan, joka estää tai merkittävästi hidastaa edistymistä, ja ovat yrityksistä huolimatta epäonnistuneet ratkaisemaan ongelmaa. Tällöin on kyseessä este. Scrum Masterin vastuulla on pyrkiä löytämään keinoja ratkaista este.

Information Radiator

Tiimin "ilmoitustaulu" joka voi olla käsin piirretty, tulostettu, tai näyttö. Ilmoitustaulu sisältää hyödyllistä tietoa kaikille, jotka sen ohi kävelevät. Tieto voi olla esimerkiksi käännöksen tila, automaattitestien kattavuus, määrä ja tulokset, kehitysjonon tila, Sprintin tai releasen edistymiskäyrä...

Iteration
Iteraatio

Voidaan myös käyttää nimitystä Sprintti. Tarkasti määritellyn mittainen ajanjakso (timebox – aikalaatikko, yleensä 1-3 viikkoa), joka toistuu ilman taukoja kunnes tarvittavat ominaisuudet tuotteeseen on saatu valmiiksi.

Kanban

Kanban on alunperin Toyotan tehtailleen kehittämä toiminnanohjausjärjestelmä, jossa työn kulku tiimin prosessin läpi visualisoidaan "kanban taululle". Työn visualisointi auttaa tiimiä huomaamaan, mitkä asiat ovat työn nopeamman virtauksen tiellä. Kanban auttaa erityisesti tilanteissa, joissa tiimille tulee eteen hankalasti ennustettavia ongelmia. Tiimit, jotka toimivat asiakastuessa tai lähellä sitä, voivat haluta käyttää Kanbania ennemmin kuin Scrumia. Myös Scrumissa käytetään yleensä kanban-taulua Sprinttiin valitun työn tekemisen visualisointiin.

Minimum Viable Product

Vanhempi määritys MVP:lle tarkoitti pienintä mahdollista kokoelmaa ominaisuuksia, minkä voi viedä markkinoille. Modernimpi, Lean Startupin mukainen määritys MVP:lle tarkoittaa pienintä mahdollista kokoelmaa ominaisuuksia, minkä kehitystiimi voi tehdä tarkoituksenaan oppia lisää asiakkaan tarpeista ja tuotteen vaatimuksista. Jälkimmäinen, modernimpi määritys korostaa sen merkitystä, että tiimi tekee yleensä aina oletuksia, ja yleensä on hyvä idea rakentaa MVP versio tuotteesta validoimaan niitä oletuksia.

Pair Programming
Pariohjelmointi

Osa Extreme Programmingin käytäntöjä. Pariohjelmoinnissa kaksi kehittäjää tekee työtä saman monitorin ääressä. Toinen kehittäjistä on enemmän havainnoimassa ja toinen enemmän tuottamassa koodia. Pariohjelmointi on nopea tapa tuottaa korkealaatuista koodia. Todellista pariohjelmointia näkee harmillisen harvoin.

Persona
Persoona

Persona on fiktiivinen käyttäjä, jonka motivaatiot, tarpeet ja käyttäjätavat auttavat kehitystiimiä asettumaan todellisen käyttäjän saappaisiin. Jos personat eivät ole käytössä, kehitystiimin on aika helppo alkaa suunnitella ohjelmistoa itselleen, joka yleensä johtaa siihen että lopputuote ei ole kovin helppokäyttöinen muille käyttäjille. Personat pitäisi tehdä yhdessä kehitystiimin ja organisaatiossa mahdollisesti olevien user experience -asiantuntijoiden kanssa, ja niiden olisi hyvä olla tiimihuoneessa kaikkien näkyvillä.

Planning poker
Suunnittelupokeri

Suunnittelupokerissa käytetään apuna pelikortteja, joissa on eri työmääräarvioita. Jokainen tiimin jäsen saa oman korttipakan. Kun aletaan arvioimaan jonkin asian työmäärää, ensin asian kuvailee tai kertoo tiimille asiakkaan edustaja (usein tuoteomistaja) tai joku muu henkilö. Sitten jokainen tiimin jäsen valitsee sopivan työmääräarviokortin ja näyttää sen yhtäaikaa muille. Jos arvioissa on eroja, tiimi keskustelee erityisesti suurimman ja pienimmän työmääräarvion esittäneiden ihmisten näkemyksistä. Tämä käytäntö auttaa paljastamaan kehitettävässä asiassa mahdollisesti olevia "piilossa" olevia töitä, ja saa aikaan konsensuksen siitä miten isoksi työmäärä lopulta arvioidaan. Käytäntö auttaa saamaan asiat sprinteissä valmiiksi, ja parantamaan kehitysjonolla olevien asioiden laatua

Product backlog
Tuotteen kehitysjono

Katso "Backlog" yllä. Koko tuotteen kehitysjono, joka yleensä sisältää useamman releasen verran ominaisuuksia. Tuotteen kehitysjono ei kuitenkaan saisi olla liian pitkä. Liian pitkä kehitysjono johtaa tehottomaan kehitysjonon jalostamiseen, ja siihen että prioriteetissa alimpana olevia asioita ei saada koskaan tehdyksi. Tuoteomistajan vastuulla on ylläpitää tuotteen kehitysjonoa. Tärkein asia on sanoa joillekin pyynnöille "EI", eli pitää kontrolli siitä mitä tuotteen kehitysjonolle päästetään.

Product owner
Tuoteomistaja

Tuoteomistaja on yksi Scrumin perusrooleista. Hänen vastuullaan on kehitysjonon ylläpito, sidosryhmien ja asiakkaiden kanssa kommunikointi, ja tuotteita kehittävien tiimien tulosten hyödyn maksimointi valitsemalla oikeat asiat kehitettäväksi. Suuri osa tuoteomistajan työtä on kommunikointia, määrittelyä ja priorisointia, mutta tuoteomistaja ei kuitenkaan saa unohtaa myös tiimin hyvinvointia ja oppimista, koska sillä on suuri vaikutus työtehoon ja tuloksiin.

Refactoring
Refaktorointi

Refaktoroinnilla tarkoitetaan koodin uudelleenkirjoittamista tai rakentamista, ilman että koodin toteuttamat toiminnot muuttuvat. Refaktoroinin tavoitteena on koodin luettavuuden ja laadun parantaminen. Refaktorointia kannattaa tehdä jatkuvasti, aina kun koodiin kosketaan. Jos refaktorointia ei tehdä ja sen tarvetta harkita, johtaa se pitkässä juoksussa kehitysvauhdin hidastumiseen "sammakko kuumassa kattilassa" tapaan, ja tuote tai organisaatio kuolee pois.

Release
Julkaisu

Ketterässä toimintatavassa tiimi(t) pyrkivät vähintään joka sprintin lopussa saamaan valmiiksi julkaisukelpoisen tuotteen. Hyvät tiimit pystyvät ylläpitämään julkaisukelpoista tilannetta myös sprintin aikana. Tuoteomistaja päättää, milloin tehdään varsinainen julkaisu, eli annetaan tuote jonkin sidos- tai käyttäjäryhmän käyttöön. Erilaisia julkaisun ilmenenmismuotoja voi olla vaikkapa early access, sisäinen julkaisu tai alfa tai beta julkaisu. Julkaisut tyypillisesti numeroidaan major ja minor julkaisunumeroin, esimerkiksi 3.1.1.Jos organisaatiossa on tapana julkistaa tuote jatkuvasti tuotantoon, julkaisun käsite voidaan ymmärtää myös käsittämään tietyllä ajanjaksolla toteutettavat toiminnot. Tällöin esimerkiksi vuoden 2020 ensimmäisen kvartaalin toteutetut toiminnot voisivat kuulua releaseen 20.1, vaikka ne julkaistaisiinkin juoksevasti koko ajan kvartaalin aikana. Tällöin julkaisu toimii enemmän suunnittelun apuna eikä niinkään varsinaisesti viittaa siihen, milloin käyttäjät saavat uudet toiminnot käyttöönsä.

Retrospective
Retroperspektiivit

Säännöllisesti toistuva tiimin kokous, jossa mietitään miten toimintaa voisi parantaa. Retrospektiivi kannattaa fasilitoida, yleensä Scrum masterin toimesta. Retrometodeja on paljon, ja käytettyä tapaa kannattaa vaihdella, jotta kokous ei muodostuisi tylsäksi pakkopullaksi.

Scope
Määritelty laajuus

Usein puhutaan "Release scopesta" kun halutaan kuvata sitä osuutta tuotteen kehitysjonosta, joka on tavoitteena releasoida. Tiimin kehitysnopeutta (velocity) voidaan sitten verrata releasen määriteltyyn laajuuteen, ja tästä saadaan varsin tarkka arvio valmistumiseen tarvittavasta ajasta. Jos nopeus ei ole riittävä tavoiteaikatauluun, voidaan määriteltyä laajuutta pienentää. Mitä aikaisemmin tällaiset muutokset tehdään, sitä parempi.

Scrum

Suosituin ketterä kehitystoimintamalli. Scrum ei ole pelkästään ohjelmistokehitykseen soveltuva, vaan sopii kaikkeen toimintaan missä tulevat työtehtävät voidaan kuvata kehitysjonolle, ja missä voidaan antaa tiimille aikalaatikon (time box) mittainen kehitysrauha.

Scrum Master

Scrumin rooli. Scrum Masterin tehtäviä ovat muun muassa tiimin valmennus parempaan Scrumiin, esteiden poisto, tiimin seremonioiden fasilitointi ja tuoteomistajan apurina toimiminen.

Scrum of Scrums

Usean tiimin yhteinen Scrum, tyypillisesti viikottain. Osa useita erilaisia "skaalattuja" ketteriä metodeja.

Scrum Team

Scrum tiimiin kuuluu tuoteomistaja, Scrum Master ja kehitystiimi

Sprint

Määrämittainen ajanjakso, yleensä 1-3 viikkoa, missä tiimi lupaa toimittaa valitut asiat tuotantolaatuisina, loppuun asti tehtyinä. Sprintin voi nähdä "miniprojektina" jossa asioita saadaan valmiiksi. Pienissä palasissa tehty toiminnallisuus paljastaa tiimin todellisen nopeuden, ja auttaa suunnittelemaan julkaisun aikataulua ja sisältöä, ja välttämään projektin lopun pitkää kypsyttelyvaihetta.

Sprint Backlog

Katso yllä "Backlog". Sprint backlog tarkoittaa niitä kehitysjonon asioita, jotka kehitystiimi lupasi toimittaa sprintin aikana tuotantolaatuisina. Sprintin kuluessa tiimi seuraa backlogin edistystä "todo" tilasta "done" tilaan, esimerkiksi sprint burndown-kaaviolla.

Sprint Planning

Sprintin suunnittelukokous, tyypillisesti pidetään sprintin ensimmäisenä päivänä. Kokous kestää noin 2-3 tuntia. Tuoteomistaja tuo kokoukseen ehdotuksen tärkeimmistä kehitysjonon asioista, ja tiimi keskustelee niistä ja päättää miten suuri osa niistä voidaan ottaa mukaan sprinttiin. Päätös perustuu historiatietoon (velocity) ja suunniteltuun kapasiteettiin (joka ottaa huomioon koulutukset, lomat yms). Suunnittelukokouksen tuloksena pitäisi syntyä Sprintin toteutussuunnitelma, joka saattaa sisältää yksittäisien kehitysjonon asioiden tekemiseen tarvittavat tehtävät (tasks) ja myös prioriteettijärjestyksen missä työt aloitetaan.

Sprint Review

Sprintin katselmus, jossa käydään läpi mitä saatiin valmiiksi, mitä ei, mitä opittiin. Sprint review sisältää yleensä demon, jossa Sprintissä valmiiksi saatuja asioita demotaan tiimin sisällä ja sidosryhmille. Demo saattaa joskus olla myös erillinen kokous. Sprinttikatselmuksessa pitäisi keskustella myös siitä, ollaanko huomattu että backlogilta puuttuu jotain tai prioriteettijärjestys on muuttunut. Periaattessa sprinttikatselmus on myös paikka tehdä suuria suunnanmuutoksia ("pivot").

Sprint Goal
Sprintin tavoite

Tuoteomistaja määrittelee sprintin tavoitteen. Tavoite auttaa tiimiä priorisoimaan asioita sprintin sisällä.

Story Mapping
Käyttäjätarinakartta

Keino visualisoida ja järjestää tuotteen ominaisuuksia hierarkisesti. Käyttäjätarinakartta voi myös auttaa release suunnittelussa ja tarinoiden pilkkomisessa ja sen huomaamisessa, onko kaikki tarinat listattu kehitysjonoon.

Story Point
Tarinapiste

Tapa arvioida kehitysjonolla olevien asioiden työmäärää. Yleensä ohjeena on, että tarinapistettä ei sekoiteta kalenteriaikaan, mutta ei se nyt niin vakavaa ole jos niin tehdään. Ideana kuitenkin on että pienet tarinat ovat 0.5 tai 1 pistettä, ja sitten isompia tarinoita verrataan niihin. Tiimin kannattaisi asettaa "yläraja" sille miten suuren arvion saaneita tarinoita sallitaan aloittaa. Suurempien tarinoiden määrittely on paljon vaikeampaa, ja saattaa johtaa siihen että niitä ei saada valmiiksi ja ne vuotavat seuraavaan Sprinttiin. Story pointin sijaan tiimi saattaa haluta arvioida työmääriä myös "ideaaliajassa" esim ideaalitunneissa tai ideaalipäivissä. Tämä on aivan sallittua. Eri tiimien arvioita ei voi verrata toisiinsa

Story Splitting
Tarinoiden jakaminen

Tarinoiden jakaminen pienempiin, kunnes ne ovat tarpeeksi pieniä toteutettavaksi. Tarinoita voidaan jakaa milloin vain, mutta yleensä se tehdään Backlog Grooming/refinement kokouksessa, Sprintin suunnittelukokouksessa, tai hankkeen alussa kun tuotteen backlogia rakennetaan esimerkiksi Story mappingin avulla. Tarinoita voidaan jakaa monella eri tavalla, mutta pääasia olisi se, että jakaminen tehdään niin että jokainen tarina tuo pystysuunnassa toimintoja loppukäyttäjälle. Väärin jaettu on, jos tehdään yhdessä tarinassa käyttöliittymä, toisessa backend, ja kolmannessa testataan. Hyvä tarinoiden jakotapoja kuvaava juliste löytyy linkeistä.

Super-seremoniat

Contribyten termi, joka viittaa parhaisiin mahdollisiin käytäntöihin eri agile-seremonioissa

Technical Debt
Tekninen velka

Tekninen velka kuvaa tilannetta, missä tiimi "menee helpoimman kautta" eli tekee toimintoja harkitsematta refaktorointitarvetta, koodin luettavuutta, testattavuutta tai robustisuutta. Jos teknisen velan annetaan jatkuvasti kasvaa, se pitkällä tähtäimellä hidastaa tiimin nopeutta, koska asiat muuttuvat vain koko ajan hankalammiksi. Purkkaviritysten teko yhden kerran voi olla perusteltua, mutta koko talon rakentaminen purukumista on itsemurhaa. Vastuu teknisen velan kasvun välttämisestä on kehitystiimillä, ei kenelläkään muulla. Jos kehitystiimi ei ota tätä vastuuta itselleen, vastuu valmentaa tiimiä ottamaan se on Scrum Masterilla. Tuoteomistajan kannattaa olla varuillaan tiimin kanssa, joka on liian innokas toimittamaan featureita, mutta ei tunnu välittävän pitkän aikavälin koodin terveydestä.

Test Driven Development
Testiohjattu kehittäminen

Läheisesti sidoksissa "hyväksyntätestiohjattuun kehittämiseen" (ATDD). TDD ja ATDD erot ovat lähinnä siinä että ATDD keskittyy business ja asiakasymmärryksen lisäämiseen ja keskusteluun kehitystiimin ja asiakasta ymmärtävän tiimin välillä, sellaisella kielellä että molemmat ymmärtävät mistä puhutaan. TDD keskittyy tilanteisiin missä ongelma on hyvin ymmärretty, ja kehittäjä lähtee liikkeelle koodaamalla yksikkötestit ensin, ja sitten alkaa lisäämään toteutusta kunnes yksikkötestit menevät kaikki läpi. Sinällään kummatkin ovat tarpeellisia – on hyvä aloittaa ATDD-tyyppisellä keskustelulla ja sitten edetä TDD-tyyppiseen toteutukseen.

Timebox
Aikalaatikko

Aikalaatikko kuulostaa Hollywood-elokuvalta, mutta joo, näin se suomennetaan. Tämä tarkoittaa ketterässä kehittämisessä sekä iteraatioiden pitämistä saman mittaisena (ja sitä ettei iteraatioiden välissä pidetä taukoja, vaan seuraava alkaa välittömästi edellisen loputtua), että myös sitä että seremoniat on timeboxattu tietyn mittaisiksi, eli että kokoukset ei veny.

Unit Testing
Yksikkötestaus

Yksikkötestauksessa koodi, mikä lopulta toteuttaa halutut loppukäyttäjälle tarjotut toiminnot, testataan pienissä osissa. Tyypillisesti yksikkötestaus tarkoittaa, että koodin jokaiselle funktiolle kirjoitetaan omia testejä. Testit kirjoittaa yleensä ohjelmistokehittäjä. Testit kirjoitetaan vaatimusten ja hyväksyntäkriteerien pohjalta. Yksikkötestit ajetaan automaattisesti jatkuvan integraation käännöksien jälkeen, ja tällöin paljastuu jos muutokset ovat rikkoneet mitään vanhoja toimintoja.

Usability Testing
Käytettävyystestaus

Käytettävyystestaus on metodi, missä testaavat henkilöt pyrkivät arvioimaan onko kehitetty toiminnallisuus tarpeeksi helppo käyttää. Tässä auttaa personoiden käyttäminen. Käytettävyystestaukseen on myös olemassa spesialisteja, mutta käytettävyystestausta voi tehdä myös kehitystiimin ja tuoteomistajan voimin. Käytettävyystestausta voi tehdä joko havainnoimalla oikeita käyttäjiä, tai sitten niin että testausta tekevä henkilö kuvittelee itsensä oikean käyttäjän housuihin.

User Story
Käyttäjätarina

Käyttäjätarinan perusjuttu on että se kuvaa toteutettavan toiminnon käyttäjän näkökulmasta. Hyvässä käyttäjätarinassa kuvataan käyttäjärooli, asia mitä käyttäjä haluaa, ja sitten se motivaatio MIKSI käyttäjä sen asian haluaa. Tämä ohjaa tiimiä määrittelemään asian oikein ja priorisoimaan asian oikein. Tiimi jatkaa vielä tämän jälkeen tarkentamalla käyttäjätarinan määrittelyä hyväksyntäkriteereillä. Näin kerrottuna tarina on paremmin käsiteltävissä ja arvioitavissa.

Velocity
Tiimin nopeus

Tiimin nopeus eli velocity on paras tiedonlähde sen ennustamiseen, milloin tietty määritelty laajuus (scope) saataisiin nykytiedon valossa valmiiksi. Velocity mitataan sen perusteella kuinka paljon kehitysjonon asioita ollaan saatu todellisesti valmiiksi tietyssä aikayksikössä. Scrumia käyttävät tiimit mittaavat velocityä per sprintti. Kanbania käyttävät mittaavat esimerkiksi per viikko. Avainasiana on, että on myös määritelty DoD eli valmiin määritelmä. Muuten velocitystä ei ole paljoakaan hyötyä.

Version Control
Versiohallinta

Järjestelmä, mihin kehittäjät laittavat koodin, ja mikä seuraa jokaisen tiedoston versioita. Nykyään ohjelmistokehitys ilman versiohallintaa on käytännössä mahdotonta (ja turhaa – sen verran helppoja ja halpoja työkalut ovat). Versiohallintaohjelmistoja on paljon, suosituimpia ovat versionone, TFS ja git. Gitistä on myös maksullisia versioita jotka toimivat muiden työkalujen kanssa saumattomasti kuten Atlassianin Bitbucket.

Contribyte

Tulevaisuuden voittavien tuoteorganisaatioiden ja tiimien valmentaja

TOIMINNAN KEHITTÄMINEN | VALMENTAMINEN | KOULUTUKSET | TUOTEKEHITYKSEN TYÖKALUT