Article original a: Tutsplus - SCRUM: The Story of an Agile Team

Scrum és una de les tècniques àgils més usades. No va de programar; sinó que s'enfoca en l'organització i la gestió del projecte. Si tens una estona, deixa'm que t'expliqui com funcionem al nostre equip i com hem adoptat Scrum.

Una breu història
Les arrels de Scrum s'extenen més enllà de l'Era Àgil. La primera menció a aquesta tècnica és del 1986 (Takeuchi i Nonaka) pel desenvolupament de productes comercials. El primer paper on es descriu oficialment Scrum, escrit pel Jeff Sutherland i en Ken Schwaber és del 1995. Poc després (2001) van aparèixer l'Agile Manifesto i el llibre Agile Software Development with Scrum (Schwaber i Beedle).

AIXÍ ÉS COM VAM COMENÇAR

Tot depen de la dedicació i de la voluntat. Si vols que el teu equip sigui eficient, productiu i lliuri a temps la feina, hauràs d'adoptar d'alguna forma les tècniques àgils. Scrum pot ser ideal o no per vosaltres, però és probablement una de les millors maneres de començar.

L'equip de desenvolupament (l'equip) on treballo ja coneixia força d'Àgil. Vam canviar del desenvolupament en Cascada a processos més àgils i alliberàvem amb freqüència. Vam conseguir treure versions periòdicament entre els tres i sis mesos, amb un número de bugs decentment baix. Tot i així estàvem lluny del que hem conseguit. Trobàvem a faltar algun procés, o regles, que ens forcéssin a canviar la nostra perspectiva sobre el desenvolupament. Aquest era el moment quan el nostre coordinador d'equip ens va introduir Scrum. Aquesta persona va prendre el rol de Scrum Master.

El Scrum Master

El **Scrum Master** és un dels rols més importants. És responsable de crear un pont entre el **Product Owner** i l'**Equip**. Aquesta persona té habitualment un fort coneixement tècnic i participa activament dins del desenvolupament. És comunica amb el **Product Owner** sobre **Històries d'usuari** (històries) i com s'organitza el **Backlog**.

El Scrum Master coordina l'equip de desenvolupament però no "microgestiona" (l'equip s'autoorganitza). Al principi, però, el Scrum Master pot "controlar" a una part de l'equip per millorar la seva interacció i autoorganització.

INTRODUÏNT EL CONCEPTE DE "SPRINT"

Abans de conèixer el concepte Sprint no podiem imaginar alliberar versions entre tres i sis mesos, pensàvem que era massa ràpid i que no ens donava temps per integrar i polir els errors dels nostres productes.

Sembla contradictori, com podem tenir el producte estable cada setmana? Però realment és possible. En primer lloc vam escurçar els notres cicles de producció de tres mesos a sis setmanes i finalment a quatre. Això es va fer sense canviar el nostre estil, simplement

introduïnt regles per fer Sprints en menys d'un mes. No pot haver màgia, han de ser regles que beneficiïn el vostre projecte. El primer canvi va ser aplicar el concepte de **Sprint Planning**.

Sprint Planning

Aquest és una de les reunions que proposa Scrum. Abans de començar cada Sprint es reuneixen l'equip, el product owner i el scrum master per planificar el Sprint. La durada de la reunió pot allargar-se d'un parell d'hores a un dia complet. El que fem es revisar el product backlog i decidir quines històries s'inclouran al Sprint.

The Product Owner

És la persona responsable de definir les històries i mantenir el product backlog. També és un enllaç entre l'equip i la direcció. El product backlog avalua les peticions dels stakeholders (persones rellevants), la direcció, els usuaris i d'altres peticions (informes de bug, enquestes als usuaris, etc.)

Aquesta persona prioritza el backlog, proporcionant el màxim valor als stakeholders en el temps més curt. Aconsegueix això assignant la feina del màxim valor als sprints de l'equip. Sembla sofisticat i... ho és! De fet, donar la direcció al desenvolupament del producte endevinant quines funcionalitats són les que aporten més valor és la part més difícil del procés. De vegades és molt senzill, quan molts usuaris demanen una funcionalitat concreta.

Si es proporciona una funcionalitat molt demanada, s'augmenta la lleialtat dels usuaris que la demanen. Però això pot ser difícil perquè implementar una altra funcionalitat pot fer el sistema útil a molts més usuaris. Quina és la elecció correcta llavors? Depen de la direcció que porti el negoci i el Product Owner ha de prendre la decisió d'on vol portar el producte. A la nostra empresa aquestes decissions es comuniquen amb l'equip, no és un requeriment de Scrum però és molt útil per orientar als nous equips. Compartir informació ajuda a comprendre perquè es prenen certes decisions i perquè algunes funcionalitats que semblen òbvies s'endarrereixen o "cauen".

EL TAULER DE PLANIFICACIÓ

Recordo el matí que va passar: vaig arribar a l'oficina i em vaig trobar el nostre Scrum Master preparant un tauler de planificació de tasques amb un full A4 i un "celo". No tenia ni idea de que estava fent. Com cada matí vaig anar a fer-me un café i em vaig quedar mirant que feia. Quan va acabar una pissarra blanca estava enganxada a la pared. Tenia vàries columnes farcides de "post-it" (notes) de colors. D'això fa dos anys.

La pisarra ara dona suport al Procés de Desenvolupament Àgil (*Lean Development Process*) que usem avui. Recordeu, Agile va de canviar i adaptar-se al canvi. **Mai seguiu regles cegament sense saber perquè**.

Així doncs, que hi ha al tauler?

Columnes pel procés de desenvolupament

Tenim quatre columnes principals

- **Backlog de propera versió** – És on hi ha totes les històries de la "propera versió" del producte. Sí, el producte està preparat per lliurar al final de cada sprint, però això no vol dir que el lliurem normalment. Nosaltres tenim habitualment sprints setmanals.
- **Backlog del Sprint** – Quan planifiquem, negociem amb el product owner que vol que fem al sprint. Com sabem el que ens dona temps a fer? Estimant la dificultat de cada història (veure més avall "estimant històries"). El resultat final d'aquesta planificació és moure les notes de les històries que farem al sprint setmanal.
- **Treballant** – Aquesta és simple. Quan un membre de l'equip agafa una història, la passa a aquesta columna per mostrar a tothom que ell se n'està encarregant. Això no és per controlar a ningú sinó per la comunicació de l'equip. Tothom sap en que estan treballant els seus companys. A la foto de dalt, les enganxines de colors mostren qui està treballant a la història.
- **Fet!** – Aquí és on deixem les històries fetes. És molt important definir amb precisió "qué està fet". Quan acaba el sprint totes les històries i "bugs" del sprint backlog han d'estar a aquesta columna.

Altres equips divideixen la columna de “treballant” en d’altres com disseny, programació i test d’acceptació. Cada equip pot triar quines li van bé!

La definició de “fet”

Que està “fet”? Quan podem confiar que una història està realment completa? Com ho sabem? “Fet” s’ha de definir de manera clara i precisa per que tothom sapigui quan una història està completa. Aquesta definició és pròpia de cada equip i també pot variar entre projectes del mateix equip. No hi ha cap regla exacta, i és una idea que s’ha de comentar en les reunions i decidir en grup que vol dir una “història feta”. Alguns aspectes que poden formar part són:

- Crear un disseny minimalista.
- Crear una maqueta de la interfície.
- Usar TDD o asegurar-se que estan fets els tests unitaris.
- Escriure el codi.
- Fer les proves d’acceptació creuades entre membres de l’equip.
- El sistema complet es pot compilar i muntar amb el nou codi.
- Els tests d’acceptació se superen i el codi s’ha integrat al sistema.

Hi ha múltiples idees del que es pot incloure en la definició de fet. Pren la part que consideris necessària pel teu equip, fes-la servir a fons i fes-la evolucionar amb el temps. Si noteu que la definició s’està quedant desfasada, podeu eliminar o afegir idees útils. A la foto de dalt, els notes verds descriuen el que considerem que s’ha fet, per cada part del procés.

OMPLINT EL TAULER

Com ho farem, ens preguntàvem? Fins llavors no feiem servir els notes per planificar. Feiem servir un software per gestionar les històries d’usuari i els bugs, però res més. Després de dinar el scrum master es va presentar amb una muntanya de notes de colors. Ens va explicar que eren mentre en omplia una dotzena.

Les històries d’usuari

Una història d’usuari és una frase breu que defineix una característica o funcionalitat.

Aquestes representen les principals funcionalitats que volem implementar. Tenen el nom de històries d’usuari perquè representen la perspectiva d’un usuari. Naturalment amb

usuari ens referim a una persona que fa servir l'aplicació que representa un grup o rol (p.e. administrador del sistema, cap, usuari restringit, etc.
Un exemple d'història és algo com "Com un usuari, vull compartir una carpeta amb els meus companys d'equip".
Fins llavors no teniem un product owner definit i era el scrum master qui s'inventava les històries. Això és acceptable al principi, però **recomanem fortament** que se separin els dos rols. D'altra manera, com pot negociar el scrum master el backlog del sprint amb el product owner?
Podrieu pensar "Per què negociar? No és millor que hi hagi una única persona que decideixi que fer i quan?". No és així. Aquests dos rols quedarien influits punt de vista d'una única persona. Dues persones, per contrari, tenen més facilitat per proporcionar un camí més objectiu per l'equip i aconseguir que s'arriba a la meta final: software amb més valor.
Les històries d'usuari representen tot el que s'ha de fer, estan representades per notes grocs al tauler de la foto.

Errors, errors, errors

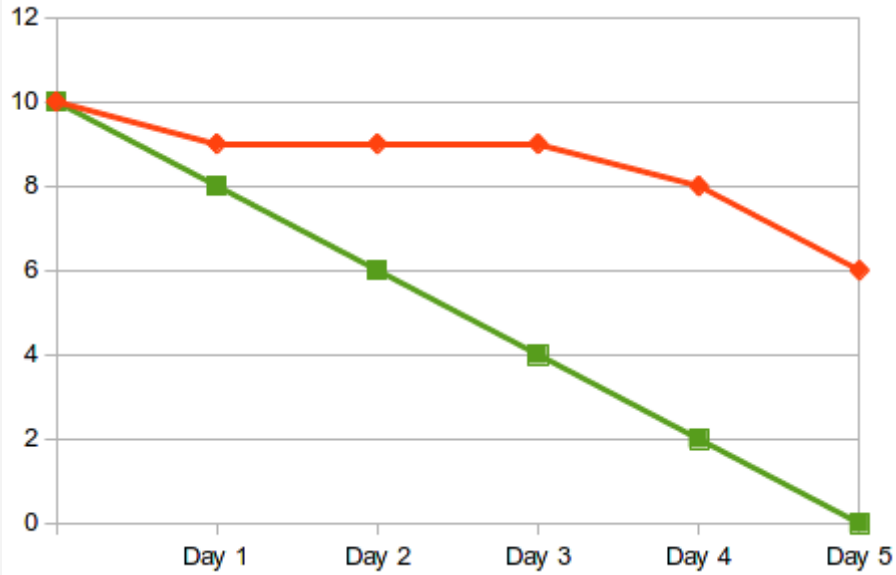
Fer un bon seguiment dels errors (bugs) és increïblement important.
Nosaltres també afegim els errors al tauler. Veus aquests notes vermells? Són els errors que cal que arreglem per la propera "versió".
Cada equip tracta els errors a la seva manera. El nostre equip els barreja amb la realització d'històries, però sempre comencem el Sprint resolent els errors assignats. Conec altres equips que acumulen els errors per un període de tres sprints i dediquen el quart sprint a resoldre'ls. D'altres divideixen els sprints entre 25% pels errors i el 75% per les històries, i d'altres treballen a les històries als matins i es posen amb els bugs a la tarda. Cada empresa ho fa a la seva manera.
Depen de tu trobar la millor solució pel teu equip, i per suposat donar seguiment als vostres errors.Escriviu els bugs a notess per donar seguiment als errors del vostre sistema i les tasques de solució dels errors. Donar seguiment als errors és increïblement important.

I know of other teams who pile up bugs for a period of three sprints, and spend every fourth sprint fixing them. Others split sprints into 25/75 for bugs/stories. Further, other teams may work on stories in the morning, and bugs after lunch; it simply depends on the company. It's up to you to find the best solution for your team, and of course, keep track of your bugs. Write them on sticky notes so that you can track your system's issues and the fixes for those issues. Tracking your bugs is incredibly important.

Tasques i sub-històries

Les tasques s'escriuen com frases senzilles des del punt de vista del desenvolupador
Idealment, cada història hauria de ser prou curta per completar-se amb certa facilitat, però la divisió d'històries en d'altres més petites pot ser difícil. Alguns projectes simplement no poden permetre's aquesta possibilitat. Tot i així trobareu blocs grans de feina en els que

poden treballar més d'un membre de l'equip. És important dividir la feina en parts més petites i fàcils de gestionar.
Un enfocament és dividir històries grans en "tasques", frases simples des del punt de vista dels desenvolupadors. Per exemple, la història anterior sobre compartir una carpeta es podria dividir en vàries tasques com ara: "Crear la UI per compartir", "Implementar el mecanisme de compartir en públic", "Implementar la funcionalitat de control d'accés", "Afegir checks de control d'accés a la UI", i d'altres. La idea és que es pensa més en la història cada vegada que es descomposa en tasques més petites i així el teu equip tindrà més control sobre la història.
Nosaltres fem servir notes blau per les tasques al tauler. Mireu la darrera columna ("Fet!") i veureu com agrupem les tasques a sota de les històries d'usuari. En l'exemple anterior vam dividir la història en quatre tasques.



Tasques tècniques

Algunes activitats s'han de completar per finalitzar una tasca, història o el sprint complet. Aquestes tasques són usualment relacionades amb la infraestructura, però us podeu trobar amb que les tasques requereixen canvis als istema. Aquest procés pot ser part o no de la història. Per exemple, pot ser que calgui instal·lar una aplicació externa per implementar la funcionalitat de compartició de carpeta. Això es considera part de la història? Podem fer-ho o no. Vam determinar que aquestes tasques s'haurien de separar de la història i això ens va ajudar a fer un millor seguiment d'aquestes tasques addicionals. Al tauler podreu trobar aquestes tasques extres representades amb notes verdes. N'hi ha una al sprint backlog i tres al testing.

El backlog tècnic

Per a un equip jove amb poca experiència amb Àgil i Scrum, és útil ressaltar aquestes tasques amb un mini-backlog.

Aquest backlog és per tasques de infrastructura, com ara actualitzar les proves automatitzades del sistema, configurar un nou servidor i d'altres coses que fan més fàcil el desenvolupament del dia a dia. Aquesta feina s'ha de completar en algun moment però no va lligada als sprints.
No cal que separeu aquests ítems a un backlog separat, de fet conec equips que els tenen al backlog comú. Nosaltres vam abandonar el backlog tècnic separat ja que considerem que aquests ítems són tan importants com la resta de tasques d'històries. Tot i així per un equip nou és útil separar-les. Proveu si us funciona, sinó barregeu-les al mateix tauler amb notes d'un altre color.

EL GRAN DESAFIAMENT: LES ESTIMACIONS

Durant una reunió de planificació, decidim quines histpories d'usuari i errors del product backlog (o el backlog de versió al nostre cas) per incloure al següent sprint. Això pot sonar simple però realment és bastant complicat.
El product owner es presenta amb un grup d'històries per treballar-hi. Aquesta llista conté habitualment més feina de la que es pot fer al sprint. El scrum master, juntament amb l'equip, ha de convèncer al product owner del que es pot fer realment al sprint. Amb el temps aquest procés es fa més senzill, tal i com el product owner enten la velocitat aproximada de l'equip. Llavors, l'equip pot anar fent-se més productiu amb cada sprint i així permetent completar més històries.El truc està a tenir un equip que realment vol sobrepassar les expectatives!
Podem calcular les històries presentades pel product owner de vàries maneres:

Punts història

Els Punts Història són un dels mètodes més comuns per estimar històries, errors o tasques. No són necessàriament la millor opció, però són una bona manera de començar. Així doncs qué és un punt història? Al principi del procés, l'equip busca al tauler les històries més simples. No importa com siguin de difícils o el que es trigui en completar-les. Aquestes històries es fixen com referència amb el valor d'un punt. En alguns projectes aquesta història pot ser tan simple com arreglar elements de la UI en deu minuts i en d'altres pot equivaler a tres persones treballant durant dos hores. Ara que tenim una referència, podem estimar d'altres històries i errors i assignar els punts corresponents per comparació.
Això pot ser més difícil del que sembla i hi han diverses escales que ajuden a estimar les històries. Les més conegudes són:

- La sèrie de Fibonacci, els números 1,2,3,5,8 i potser el 13, tot i que ens sembla passa gran.
- Les potències de 2, els números 1,2,4,8 i també el 16, però també ens sembla passa gran.

Podeu triar la que us faci sentir més còmoda, sigueu àgils! Potser us funciona millor fer servir una escala amb increments de dos punts. Bravo per vosaltres!

Semàfors

Els números són genials i molta gent tècnica els adora. Però us podeu trobar que l'equip, en algun moment, comença a associar els punts amb el temps. **Això està malament!** Les estimacions van de mal cap a pitjor quan això pasa. Els punts d'història mai s'haurien de relacionar amb el temps.
Usar colors de semàfors pot alleugerir el problema. El nostre equip va fer el canvi fa uns mesos i ens va ajudar a canviar el nostre punt de vista.
Cada color té un significat:

- Verd vol dir que una tasca es pot fer en un sprint
- Groc es refereix a una tasca més complicada, que requereix més esforç de diversos membres de l'equip però té una alta probabilitat de poder-se fer al proper sprint.
- Vermell s'assigna a històries molt difícils que completar durant el sprint. Això és més probable amb sprints curts i equips petits.

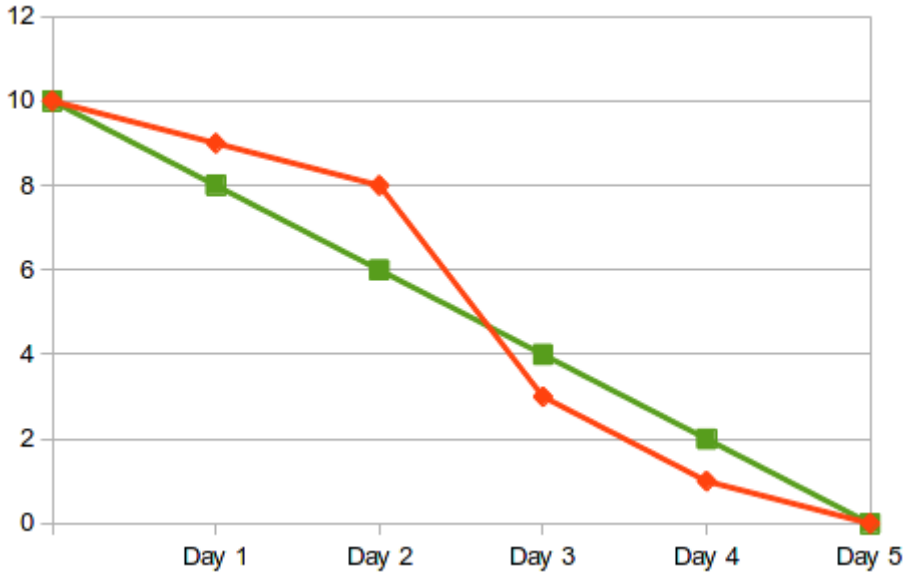
Talles de samarreta

Si no us agraden els números i els colors us semblan massa artístics, és hora per les talles de samarreta! Aquesta tècnica suggereix abandonar la idea de comparar les complexitats de les històries amb el seu temps de realització. Simplement es posa, en comptes de números, talles de samarreta com S, M, L, XL i XXL per les vostres històries.
A mi mai m'han acabat d'agradar personalment aquests tipus d'estimacions, però ei! Alguns pensen que és la millor opció. Proveu-ho si us sentiu còmodes amb la idea.

El product owner, els interessats i els directors han de saber que poden esperar al final d'un sprint. Ells han de poder decidir si les històries en les que s'ha treballat s'haurien de publicar i cal que sapiguin quan les funcionalitats estan preparades. No és una bona solucio esperar publicar totes les funcionalitats noves juntes a les "grans vesions" del producte. Publicar les funcionalitats amb més valor de manera més freqüent funciona molt millor. Per aconseguir-ho, ells han de saber que estarà disponible en el curt termini i aquesta informació l'ha de donar l'equip. Així doncs, l'equip ha d'estimar el millor possible que pot fer durant el sprint.

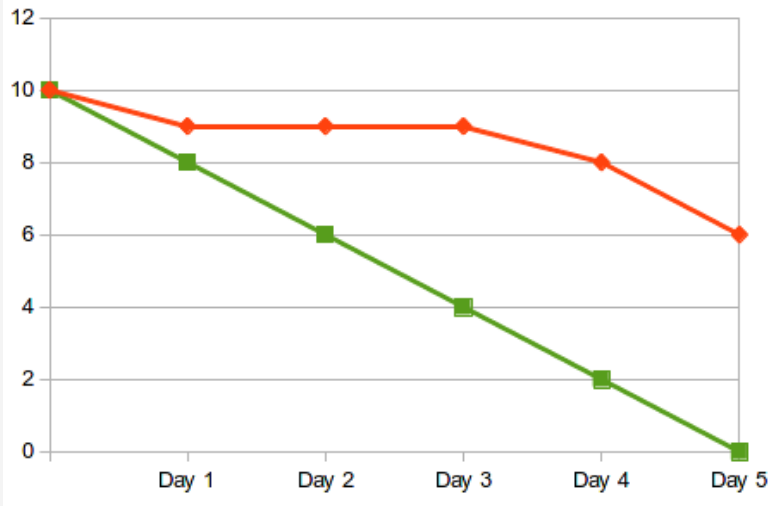
MESURANT LA VELOCITAT DEL DESENVOLUPAMENT

Així que voleu veure com de be ho hem fet durant el sprint actual? Un mètode freqüent és la recta de burndown:

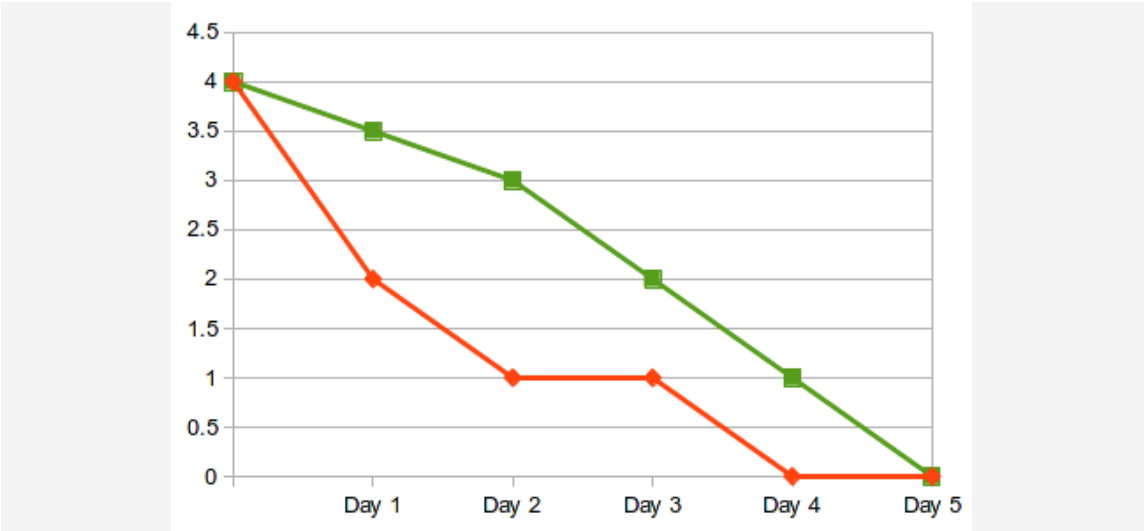


En aquesta recta tenim un sprint de 5 dies i asumim que podrem fer deu punts per sprint. Cada valor representa els punts que queden al finalitzar el dia. La linea verda marca el camí ideal: completar dos punts per dia. La linea vermella marca el nostre rendiment real o la verdadera velocitat del desenvolupament.
El meu equip tenia un full DIN A4 enganxat a sobre del tauler de planificació, tant del sprint actual com dels anteriors. Al finalitzar el dia, un dels membres de l'equip era responsable de calcular els punts completats durant el dia. És simple, els punts fets corresponen a les històries mogudes a la columna "Fet!".
No hi ha històries mig-fetes. Si una història està feta, està feta. Sinó està completa, no es compta al tauler de burndown.

Per suposat, fallareu (com no!!!) a l'estimar! Això passa molt sovint al principi i malauradament no hi ha manera d'evitar-ho. No hi ha manera de conèixer exactametn quants punts es podran fer al sprint. La vostra primera taula pot tenir aquest aspecte:



La nostra primera recta segurament s'assemblava a això. Penso que no vam completar ni la meitat del que ens vam comprometre. Per què? Bé, perquè estimar és difícil. Tant dona el que feu o com de bons sigueu, quan algú us pregunti com de difícil és fer algo que no heu fet abans passareu una estona complicada fent una estimació acurada. No entreu en pànic! Feu el millor que podeu. Amb el temps, es fa més fàcil. Podeu arribar a estimar amb un 70% de precisió en sprints curts. Si els sprints són més llargs la vostra precisió probablement caurà, perquè hi ha més històries per estimar i més variables que poden anar malament.
Quan això pasa, ajusteu. Pel següent sprint, preneu quatre punts, com a aquesta recta:



Això torna a estar malament. Heu sigut massa conservadors i heu acabat aviat. És una reacció natural que l'equip ajusti en excés.
Quan treballeu amb rectes de burndown, es recomana fer una mitja dels punts dels 3 a 5 sprints anteriors per anticipar els punts disponibles per el proper sprint. Per suposat al principi no tindreu aquesta informació disponible, de manera que probablement no tindreu la precisió posterior. Ja està bé.
Després d'un temps les vostres rectes es començaran a semblar més i més al primer exemple, tindreu una velocitat sustinguda i completareu la majoria de les vegades les històries previstes.

Velocitat?

Aquest terme es refereix a la velocitat del desenvolupament. Informa de quanta feina podeu fer en un sprint. Un dels conceptes més importants en Agile és tenir una velocitat consistent. Fer que l'equi produeixi a un ritme sostenible. Amb la gestió tradicional dels projectes, la velocitat cau tal i com avança el projecte. La complexitat i la rigidesa fan baixar la velocitat del desenvolupament.
Les metodologies i tècniques àgils pretenen que el desenvolupament es faci a un ritme constant. Lliurar el producte ràpid ara, i encara més ràpid al futur. Com ho fan? Scrum és un dels elements del puzzle. D'altres peces importants són les tècniques que fan servir els programadors per fer el seu codi i el procés de desenvolupament millors. Per exemple XP

(eXtreme Programming), Pair Programming, TDD (Test Driven Development), etc. Totes aquestes juntes poden fer l'equip realment potent.
Així doncs mesurem la velocitat, però que en fem del resultat? Mesurarem la velocitat per fer millors prediccions, no per jutjar l'equip o els seus membres.

SEMPRE MIREU ENRRERE I MILLOREU

Despres dels nostres primers sprints setmanals, el nostre scrum master ens va demanar una reunió. Va començar preguntant-nos sobre les coses bones i dolentes que van passar al darrer sprint. Això pot ser incòmode al principi, però és increïblement important. Descriure el que va anar malament al passat ajuda a ser conscient. I per suposat, també és molt útil ressaltar el que va anar bé!
Aquestes reunions s'anomenen habitualment "retrospectives". Aquí teniu alguns exempes de les nostres retrospectives:

Coses dolentes

- Els membres de l'equip es barallaven excesivament.
- El membre de l'equip X no col·laborava massa quan feia programació en parelles
- Hem perdut massa temps amb coses petites, com X o Y
- No hem fet la programació en parelles de manera consistent
- No vam fer els tests unitaris pel mòdul M

Quan discutiu els problemes, proveu d'aparcar els vostres sentiments personals, parleu del que us preocupa. Aquesta és l'única manera de que l'equip resolgui el problema. La cosa més important és proposar una solució per cada problema. No escribiu simplement les coses a la llista i deixant-les olvidades, penseu bé quina solució els hi podeu donar i com la implementareu al proper sprint.

Coses bones

- Vam acabar a temps
- Vam ser capaços de parlar sense discutir
- Alguns de nosaltres vam ser més receptius a rebre idees per millorar
- Vam escriure tot el codi usant la TDD

De nou, destaqueu totes les coses positives possibles, especialment al principi i amb els programadors més joves. Per exemple, aconseguir escriure tot el codi amb TDD pot ser una gran consecució per un equip jove. Feu que aquestes coses els facin sentir realment bé i que ho vulgui fer més i millor. Pasa el mateix per un equip sènior, ells tenen potser altres coses que destacar (el TDD es fa per reflex).

VULL VEURE QUE HEU FET A AQUEST SPRINT

La demo és per ensenyar als interessats al projecte (incloent el product owner) l'avanç del projecte.

El títol d'aquesta secció és la frase exacta que ens va dir el scrum master. En aquell moment també feia de product owner. Abans d'acabar el sprint ens demanava que li ensenyéssim el que haviem pogut fer. Vam preparar una demo en un entorn controlat. Scrum proposa fer aquestes demos al final del sprint, just abans de la retrospectiva, la reunió interna de l'equip. L'equip hauria de preparar un entorn especial per mostrar les funcionalitats del producte que s'han afegit durant el sprint.

Us podreu preguntar a vosaltres mateixos perquè he parlat d'entorn controlat si diem que el producte ha d'estar preparat al màxim per ser usat a "producció". Això no vol dir que la funcionalitat en si hagi d'estar totalment preparada. Sovint hi haurà funcionalitats que són massa grans per encabir-se en un únic sprint. El producte romandrà estable però la funcionalitat no està completament preparada. Quan els interessats al projecte veuen la demo, volen veure la funcionalitat i que pot fer. En molts casos, per demostrar funcionalitats no acabades, caldra preparar algun entorn especial.

Addicionalment, basat en aquestes demos, el product owner pot determinar que una funcionalitat gran ja està "prou bé" i que val la pena publicar una versió del producte als usuaris. La majoria de les vegades, fins i tot si una funcionalitat no està prou completa, enviar una versió al client pot ajudar a guanyar un feedback molt valuós i orientar la finalització de la funcionalitat de la manera que més satisfaci al client.

Això sona força simple. Soy un equip àgil, teniu tots els tests superats i el vostre producte es estable. Com de difícil pot ser preparar una demo ràpida? És més difícil del que sembla! El nostre equip va necessitar, si recordo correctament, més de cinc intents abans de preparar correctament la demo. Per sort, els interessats no estaven presents a les demos de preparació!

ENCARA NECESSITEM MÉS GUIA

En aquestes reunions, cada membre de l'equip ha de contestar tres preguntes.

El nostre scrum master ens va proposar fer una reunió diària. Si, diària! Cada matí al mateix lloc!

Trobo que això és una cosa molt bona pels nous equips, que no es senten encara còmodes entre ells o amb el projecte. A aquestes reunions diàries, anomenats Daily Scrums (scrums diaris) hi va la gent de l'equip a una hora concreta del dia. Haurien de ser aviat per fer-se abans de començar a treballar al projecte. El nostre equip les feia a les 10h del matí, no era fàcil reunir-nos a aquesta hora però va ser la decisió correcta.

El scrum diari és una reunió curta i senzilla (no més de quinze minuts). El seu objectiu és que els membres de l'equip sapiguin qui està fent que i determinar quin són els colls d'ampolla i problemes que té l'equip.
Com voliem assegurar-nos que aquestes reuniones no s'allargaven, les feiem de peu. La gent es cansa habitualment després d'estar quinze minuts aixecats i això és perfecte! Si trobeu que la gent de l'equip estan buscant algun lloc per seure i descansar és que la reunió s'està allargant massa.

Durant aquestes reunions tots els membres de l'equip havien de contestar tres preguntes:

- **Que vas fer ahir?** S'espera una explicació curta, de 2 a 3 frases.
- **Que tens planificat fer avui?** Una resposta amb el mateix nivell de detall, coses com "treballaré en aquesta història avui".
- **Trobes algun problema amb el procés?** En cas positiu, quin? Es poden solucionar ràpidament? Això hauria de ser una resposta ressaltant el problema i les solucions, si es coneixen. No s'haurien de fer discusions detallades durant aquesta reunió. El scrum master hauria de prendre nota del problema i treballar amb l'equip per solucionar-ho, una vegada ha acabat la reunió.

Solucionar els problemes i els impediments dels desenvolupadors hauria de ser prioritari per l'equip perquè aquests puguin continuar amb el desenvolupament tant aviat com sigui possible. Sovint la persona que es troba el problema el pot solucionar de manera ràpida, d'altres requereix l'ajuda d'un membre de l'equip i, en alguns moments, el problema és tan important que l'equip ha de parar el desenvolupament i concentrar-se exclusivament en la cosa que els impedeix continuar la seva feina.

Recordo que el meu equip es va trobar aquests bloquejos en diverses ocasions. Hi havia tasques i històries que semblaven ser prou òbvies al principi però que després que algun programador comenci a treballar es tornen més complexes i confuses. Vam descobrir vàries vegades que pensàvem una llibreria externa ens podia facilitar la funcionalitat que ens calia i vam acabar concentrant tots els nostres esforços en trobar una altra amb més capacitats, o fins i tot implementant una solució pròpia.

La majoria dels nostres projectes es fan en PHP. En algun moment vam necessitar fer una interfície amb VMWare i vam veure que les llibreries oficials de VMWare estaven disponibles per Java i Perl. També hi havia una opció no oficial per Ruby. Estavem segur que podriem usar una d'aquiestes i simplement fer unes crides "exec()" des de PHP per capturar la sortida com una cadena. Va resultar que fer això era quasi impossible. Cap de les API va funcionar tal i com esperàvem. Algunes estaven abandonades o eren incompletes, i era gairabé impossible "parsejar" les sortides. Finalment ens vam veure obligats a fer algo que no havia fet ningú abans: implementar una API en PHP per VMWare. Malauradament no hi havia cap altra opció raonable per fer-ho.
Aquest problema va ser massiu, va endarrerir els plans inicials vàries setmanes! Per suposat, ho vam notificar al product owner de manera immediata i d'acord amb ell vam

planificar un nou calendari i desenvolupar noves històries que incloien la creació d'aquesta llibreria. Els vostres problemes seran sovint més senzills. Alguna persona es pot quedar encallada en alguna lògica complexa, però sovint el matí següent poden sorgir idees i solucions. D'altres vegades els programadors interpretaran malament el requeriment o la solució, però amb l'ajuda d'un company podran rectificar. Això són els típics problemes.

CONCLUSIÓ

Ja em arribat a la conclusió. Com a mínim ja heu vist com el meu equip es va iniciar amb Scrum. Algunes regles ens van ser molt útils, d'altres no tant. També algunes regles les vam seguir només al principi i d'altres s'han incorporat a l'equip i es respecten de manera religiosa.
Només puc recomanar que us passeu a un procés àgil, proveu Scrum i preneu les vostres pròpies conclussions. Estic segur que trobaréu problemes i dubtes per adoptar-lo al vostre equip. Sigueu àgils i adapteu-lo al vostre estil de treball, als vostres projectes i personalitats, i no tingueu por d'afegir les vostres pròpies regles. Després de tot, l'agilitat es refereix a la capacitat d'adaptar-se i no seguir de manera cega unes regles establertes!