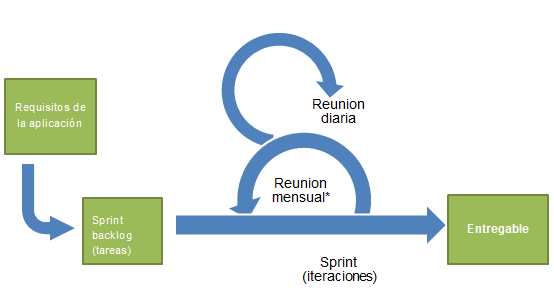
Enginyeria del Software – Tema 1

1. La productivitat ve relacionada amb el software, les dades y la anàlisis
2. Enginyeria del software té mes a veure amb humans que amb maquines:
   1. Els teus companys de treball
   2. El teu manager
   3. El becari sota la teva responsabilitat
   4. Qualsevol persona C-level per sobre teu
      1. CEO: Chair Executive Officer
      2. CTO: Chief Technology Officer
      3. COO: Chief Operating Officer
      4. CFO: Chief Financial Officer
      5. EVP: Executive Vice President
3. Agile Software Development (abans, Cascada)
   1. Mètodes iteratius i incrementals, solucions i requisits evolucionen en el temps segons l’evolució del projecte.
   2. Equips auto-organitzats i multi-disciplinats
   3. Presa de decisions a curt termini
4. Scrum
   1. Framework, no una implementació final
   2. Compatible amb altres metodologies com peer-review o pair programming
   3. Tecnologia i negoci son diferents coses
   4. Les dates límit són crucials
   5. Objectiu: saber quan abans la detecció d’endarreriments
   6. El producte pot canviar mentre està en desenvolupament
   7. Gran diversitat en l’equip
   8. Solapament de les diferents fases de desenvolupament (no cascada)



* 1. Rols:
     1. Negoci:
        1. Product Owner: representa als stakeholders, els interessats (externs o interns) del resultat final del producte. Res de management, alguna execució tècnica
     2. Tecnologia (no negoci):
        1. Scrum Master:
        2. Desenvolupadors: execució tècnica
  2. Treball organitzat a través de meetings:
     1. Sprint Planning, Sprint Retrospective i Daily Meetings
  3. Documentació: User Stories, Backlog i estimacions de temps
  4. Scrum: el com
     1. Planificació: escollir US més importants i estimar el seu temps
     2. Execució
     3. Cada matí stand-up Meeting
     4. US feta: testejar i mergejar
     5. Final del sprint:
        1. Release de la feina feta
        2. Retrospective
        3. Sprint Review: es mira, mesura la productivitat
     6. Ha d’estar adaptat a:
        1. L’estructura de l’organització
        2. La cultura de l’organització
        3. La mida de l’organització
        4. La complexitat del producte

1. Rols Scrum
   1. Separar responsabilitats, evitar conflictes d’interès
   2. Scrum Master & Product Manager: relacionarse amb els alts càrrecs i relacionar-se amb l’equip (mid-high management)
      1. Product Owners:
         1. Outter Gameplay
            1. balancejar els interessos de les parts interessades segons les hores de desenvolupament (escasses)
            2. MAXIMITZAR el resultat donat pels desenvolupadors

Tenen el mateix resultat quan:

El resultat és mesurable, els interessos estan alineats amb el resultat i el resultat i interessos són transparents respecte l’organització

* + - 1. Team Gameplay
         1. Product Backlog (cartera de productos)

Expressar-lo clarament, ordenar-los òptimament per maximitzar objectius i missions i fer-lo visible, transparent i clar per tothom. + En què treballarem a continuació. Desenvolupadors els han de tenir clar al nivell necessari.

* + - * 1. Optimitzar el valor del treballar realitzat pels desenvolupadors
    1. Scrum Master
       1. MAXIMITZAR la productivitat de l’equip
          1. Promoure més hores de feina efectives
          2. Incrementar la feina feta en hores de feina
       2. Team Gameplay
          1. Eliminar obstacles
          2. Establir un entorn de treballar on l’equip es efectiu
          3. Definir dinàmiques d’equip (Daily meetings, standup meetings...)
          4. Incentivar una bona relació entre l’equip (desenvolupadors) i el product owner y demes
          5. Protegir l’equip d’interrupcions i distraccions

1. Àmbits de treball
   1. Objectiu: equilibrar el tamany i el com de la feina a fer, jeràrquicament:
      1. Tema conté iniciatives (LLENGUATGE DE NEGOCIS)
         1. Iniciativa conté epics (objectiu)
            1. Epics conté User Storys

US contenen tasques (LLENGUATGE TÈCNIC)

Tasques contenen sub-tasquest a la vegada dividides en sets de sub-tasques

1. Git
   1. Agilitat, comunicació d’equip, prevenir desastres...
   2. Merge:
      1. Merge feature-branch a target branch (*Simple, alt risc)*
         1. Simple, 1 merge, però potser fas malbé la target branch
      2. Merge target branch a feature branch i després juntar les dos *(Al mig)*
         1. 2 merges, fas malbé feature branch
      3. Crear merge branch desde la target, merge feature branch into merge branch, merge branch a target (*Burocràcia, baix risc)*
         1. No et carregues cap branch, pots compartir la merge branch. Fas 3 merges
2. Beyond Scrum: Kanban
   1. Es Agile
   2. No
      1. US, Sprints, Canvis ni prioritat al JIT (on demand prioritat)
   3. Kanban Board: pot ser usada en qualsevol metodologia.
      1. Ofereix una visió global de què està passant.
      2. Backlog (tasques) – To Do – In Progress – Testing – Done
3. Pair Programming
   1. Dividir la feina d’una tasca entre dos programadors
   2. Setups:
      1. Dev – Test
      2. Dev – Quality Assurance
      3. Dev – Dev
      4. Dev – Code Review
   3. Beneficis:
      1. Tunnel – vision
      2. Errors que podria fer un programador – Erros que podrien fer 2 programadors
      3. Suport moral
      4. Menys distraccions
      5. Compartir coneixements
      6. Adquirir coneixements més ràpids pels newbies
      7. Rol Correcte?
      8. Empleats més contents
   4. Inconvenients
      1. Odiarnos
      2. Cansanci
      3. No coincidir
4. Code Review
   1. Activitat dins de QA
   2. Un o varis humans revisen el codi (si pot ser no propi)
   3. Després de la implementació
5. TDD (X Driven Development)
   1. Test, Domain, Feature, Acceptance Test, Behaviour)
   2. Proces
      1. Escriu el test
      2. Mira si falla
         1. Escriu codi perquè el passi
      3. Passa tots els tests
         1. Refractor codi i mira si passen
      4. Si passen: escriu nous tests
   3. Test per tot però un test no et cobreix qualsevol input.
6. Pull Requests
   1. Mantenir un ordre
7. SAFe (Scaled Agile Framework)
   1. Agile aplicat a una gran empresa
   2. Petits equips Scrum (3 a 8 persones)
   3. Dividir equips o responsabilitats quan són massa grans
   4. Coordinació entre Product-Owners i Scrum Masters
   5. Equips s’especialitzen en el procés amb una producció pipeline
8. Enginyer: soluciono problemes
9. Dependency Hell
   1. Quan el teu projecte te masses components, com codi, paquets, recursos inters/externs...
   2. Problemes amb les versions...
10. Versionament
    1. Les coses poden canviar amb el temps: hem de mantenir aquets canvis. Tipus de canvis:
       1. En com usem el codi
       2. Canvis en la lògica
       3. Canvis en la manera de solucionar errors
    2. Canvis Grans: els de com usem el codi (X)
    3. Canvis mitjans: els de com funciona la lògica (Y)
    4. Canvis petits: bugfixing (Z)
    5. X.Y.Z
    6. Més sobre versions:
       1. Paritat significa estabilitat i suport
       2. Versions amb nom (Android)
       3. Long Term Support versions
       4. Major-Minor canvis estan fixats per calendari
       5. Oferir diferents versionaments
       6. Qualificadors:
          1. Snapshot: una “foto” del codi, que li falta bastant
          2. Release
          3. Beta, alpha, nightly
          4. RC: Release Candidate
    7. Versionament i Git:
       1. Minor changes van a la dev branch
       2. Patch changes a la master
       3. Versions: tags a la master branch
11. Gestionament de paquets
    1. Els paquets son codi comprimit amb una versió a compartir
    2. Sovint hi ha masses
    3. Compartir codi de terceres persones no convé
    4. Solució:
       1. Pujar els paquets en un repositori públic
       2. Especificar en un fitxer les dependències. El Package Maneger les tindrà en compte
    5. PACKAGE MANAGER
       1. Tambè compilen i processen el teu codi
       2. Tambè pujen codi a repositories públics i privats
       3. Tambe gestionen versions del teu codi i de commits de git
       4. Tambè accepten plugins per
          1. Autogenerar documentació, codi, revisar codi, re-packejar, deploy codi...
       5. Package Shading
          1. Shade a llibreria es agafar els continguts d’una llibreria, posar-les en una de pròpia i canviar el paquet.
12. Automatització
    1. Situació: hi ha costos i accions humanes relacionades amb el desenvolupament que són repetitives:
       1. Corre test
       2. Merges
       3. Fer i distribuir builds
       4. Deploy enviroment?
    2. Problemes de no automatització:
       1. Merge moltes coses en poc temps
       2. Retards en les entregues
       3. Hores de treball en equip reduïdes per culpa
       4. Errors humans en el procés
    3. Industria
       1. El procés ideal per automatitzar és aquest:
          1. Té un alt cost (econòmics, riscos, hores humanes...)
          2. Es fa moltes vegades
             1. 1 i 2 ens ajuden a mesurar quins recursos salvarem
          3. No varia molt
          4. Es de tipus decisions objectives
             1. 3 i 4 ens ajuden a mesurar la simplicitat de la automatització (costos de fer-ho)
    4. Pasta (Economics)
       1. Concepte d’automatització es basa en:
          1. Fes un inversió fixa per reduir costos variables
             1. Inversió fixe: despesa que es fa una vegada i s’adjunta a un actiu?
             2. Cost variable: despesa que passa cada vegada que passa alguna cosa i no s’adjunta a cap actiu
       2. Formula
          1. I = despesa fixa de l’automatització
          2. C = cost variable del procés que ens estalviarem
          3. R = repeticions del procés durant un cert temps
          4. T = vida útil esperada de l’automatització
             1. Si I/T < C · R 🡪 Automatitza
             2. Com més gran sigui la C, R i T millor
             3. Com menys I millor
       3. Últimament hi ha moltes empreses que es dediquen a donar solucions d’automatització, de manera que el cost d’automatitzar ha baixat
          1. No vol dir que haguem d’automatitzar-ho tot
       4. Problemes:
          1. Sotaestimacio del preu
          2. Hype: automatitzem pq tothom ho fa
          3. Prioritats: a vegades els costos són grans i es podien dedicar a altres coses
          4. Canvis en la tecnologia
    5. Exemples
       1. Git, Package management, code style check, DDBB backups & server
    6. Integració continua
       1. Els merges impliquen sintaxis, compilació, unit tests, etc...
       2. Aquesta garantia es pot automatitzar: trobes bugs abans
       3. Principis:
          1. Un unic source repository
          2. Automatitzar la build
          3. Self-testing
          4. Versió executable fàcil darribar a tothom
          5. Automatitzar el deploy
       4. Com?
          1. Developers en el seu entorn 🡪 commit a repo
          2. CI server builds i run test
          3. CI informa de build succesfull. Si falla, alerta
    7. Continuus Delivery
       1. Automatitzar deploys: del repositori al server
       2. Software CAN be released to production at any time
       3. Quan?
          1. Software deployable en tot el cicle de vida
          2. L’equip prefereix fer-ho deployable a treballar en noves features
          3. Comentaris ràpids i automàtics sobre la disponibilitat de producció dels seus sistemes cada vegada que algú els hi fa un canvi?
          4. Pots fer un botó per fer el deploy
       4. Aqui pots triar si fer molts al dia o no.
       5. Beneficis
          1. Menys risc al fer deploy
          2. Progres “de veritat” ja que esta *live*
          3. User feedback
    8. Continuus Deployment
       1. No tries quan es fa el deploy, no hi ha butó
       2. Es fa quan s’accepta un pull request, un commit...
       3. Automaticamen va al pipeline i després a producció. Molts deploys al dia
       4. In order to do conitnuus deploy has de fer conitnuus delivery
    9. Feines dels desenvolupadors en l’automatització??
       1. Descobriment de serveis
       2. , orquestració dels deploys, backups automàtics, documentació automàtica, monitoring
13. Descobriment de serveis:
    1. Tens bastants serveis API. Cada un publicat en protocol HTTP en un domini. Cada API ha de saber on son els altres serveis
14. Deploy Orchestration
    1. Redueix riscos, temps, costos...
15. Monitoring
    1. Data driven X
       1. Marketing, decision making, management, journalism (periodismo de datos)
    2. Goals
       1. Controlar que està passant i que ha passat
       2. Donar suport al decision making
       3. Evitar fallos
       4. Automatitzar més coses
    3. Dos possibilitats
       1. Service Monitoring
          1. Requests (HTTP, SQL, BBDD, Queue)
          2. Oferim bona calitat?
          3. Falla el nostre servei?
          4. La BBDD fa que vagi tot lent? La cache?
       2. Resource Monitoring
          1. Memoria i temps de la JVM, Servidor, Xarxa, capacitat del disc, fils d’una app...
          2. S’usa el servidor?
          3. Necesitem més espai al disc?
          4. La RAM va bé? Mes ample de banda?
       3. Seguiment per stack
          1. Hardware Stack
          2. Container Stack
             1. Una sèrie d’apps juntes
          3. Application Stack
    4. Etapes
       1. Recolectar la data
       2. Enviarla a algun lloc
       3. Guardar-la en una BBDD
    5. Softwares: Elasticsearch, Splunk
16. Infrastucture provisioning
    1. Problema, necessitem
       1. Hardware
       2. Environments
       3. No gastar els diners
    2. És garantitzar que el nostre servei té prou hardware i ambients similars
    3. Trucar al hardware provider
    4. En comptes de fer-ho manual, pilla serveis automàticament. Condicions:
       1. Hardware provider té una API per llogar coses
       2. Sistemes monitoritzats
       3. Escriure condicions per decidir quan encendre/apagar servers
       4. Els deploys són automàtics
    5. A tenir en compte:
       1. Nit/Dia
       2. Us imprevist: Black Friday
17. CircleCI: continuus Integration and deploy
    1. Per cada commit volem:
       1. Que el codi compili i passi tests
    2. Evitar pushes amb errors fagin merge
    3. Feines: un procés de passos amb un seguit de feines en que si una falla, no s’executen més
    4. Workflow: un procés de feines executades amb dependències:
       1. Build and test
       2. Deploy if succés
       3. Notify