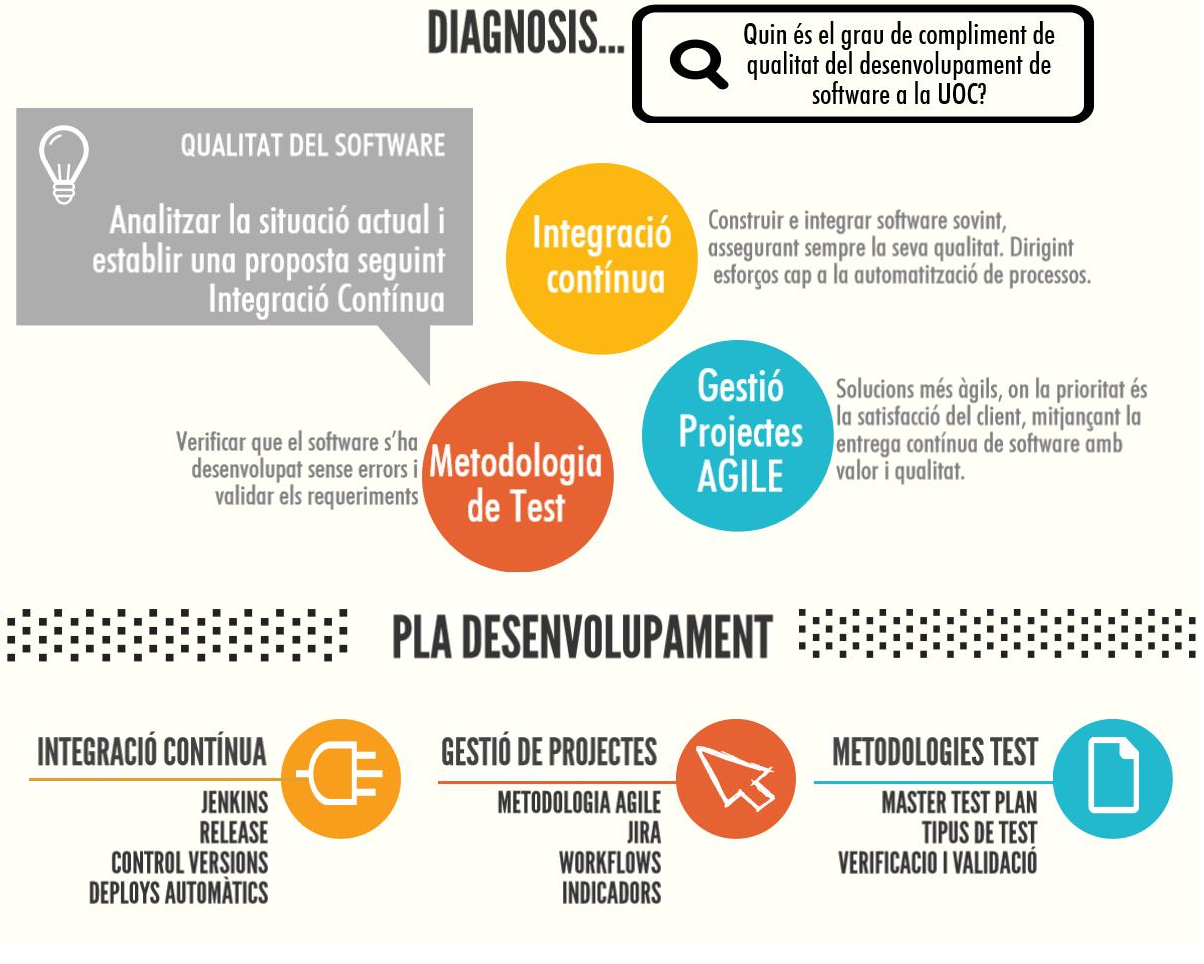
Informe de diagnòstic sobre Qualitat

Es proposa fer unaconsultoria de qualitat peridentificar quins són els objectius que hauria de seguir el departament de QA, per portar a terme un seguit de millores en el sistema actual, tenint en compte els següents objectius:



Actualment, l**’entorn de desenvolupament** és l’ideal per tal de garantir la qualitat del SW:

* Jenkins per a la Integració contínua.
* Sonar per l’anàlisi estàtic del codi.
* Artifact per injectar dependències.
* GIT/SVN per a controlar versions.
* Fisheye, per a fer cerques complexes en el repositori de codi.
* Gestió d’aplicacions i dependències amb el cmdb.

Encara que hi ha un entorn òptim, estem lluny de treballar sota el concepte d’**Integració contínua.** La idea bàsica de la CI és la de construir e integrar software sovint, assegurant sempre la seva qualitat. La transició cap a la CI no és quelcom que pugui succeir de manera natural, i per executar-ho amb èxit, es necessita monitoritzar la seva evolució, dirigint tots els esforços cap a la automatització de processos.

El mateix passa amb la **gestió dels projectes i aplicacions.** JIRA millora la qualitat del software gràcies als processos d’interacció amb els proveïdors, i serveix de recolzament en la gestió de requisits, i pel seguiment de bugs. Però a vegades un excés de gestió pot portar a la ineficiència. Sobretot tenint en compte el canvi constant de requisits i prioritats a la que es sotmeten els projectes. Per això la gestió de projectes hauria de proporcionar solucions més àgils a la demanda dels clients, on la prioritat es focalitzés en la satisfacció del client mitjançant la entrega contínua de software amb valor i qualitat.

Tenint en compte que les proves constitueix una de les activitats més importants dins del cicle de vida del sw, s’haurien de crear processos comuns relatius al **Testeig,** amb l’objectiu de verificar que el software s’ha desenvolupat sense errors en totes les plataformes i sobretot validar que s’hagi fet el que el client demanava.

Un cop analitzada la situació actual, caldria generar un **informe de diagnòstic**. Establir una proposta sobre el nivell mínim esperat de qualitat. Quines són les actuals polítiques de qualitat de la UOC?. Que es el que busquem?. Que es el que ens manca?. Quins són els problemes actuals?. Quin és el grau de compliment de qualitat del desenvolupament de software la UOC?. Quines són les mètriques/indicadors de qualitat?. Quins són els projectes que es volen desenvolupar sota el model Integració contínua?.

A partir d’aquest informe, establir un **pla de desenvolupament** del sistema de gestió de qualitat, amb una planificació i un calendari d’activitats.

Elaboració de la **documentació** del sistema de gestió de qualitat definit.

**Executar** el que s’ha establert amb el pla de desenvolupament.

Millores en la Integració contínua

El diagnòstic per millorar la integració contínua, el podríem enfocar sota el següent paraigües:

# A- Millores JENKINS

Jenkins és un servidor d'integració contínua sobre el qual es poden automatitzar certs processos (builds). Actualment, estem fent servir JENKINS per verificar si el codi compila correctament i l’estem preparant per poder monitoritzar la seva qualitat.

El que es recomana per començar, és organitzar tot els sistema actual:

* Definir quins són els projectes que hem **d’auditar amb JENKINS** i descriure quin són els builds que s’han de generar per cadascun d’ells.
* També seria important definir quins són els processos de manteniment que s’ha de seguir a nivell de qualitat de codi amb el SonarQube, per cadascun dels projectes.
* Un primer pas, per millorar en la integració contínua seria **integrar tots els projectes JENKINS amb JIRA**. D’aquesta manera tots els projectes construïts al JENKINS estarien disponibles a l’Administrador del JIRA i correctament sincronitzats amb els seus issues.

De totes maneres, no li estem treien el màxim rendiment al JENKINS sinó fem servir **automatitzacions**:

* Una de les claus per ser més àgils a la hora de desenvolupar, és detectar el màxim número d’errors el més aviat possible. Per això, es recomana la metodologia de desenvolupament per proves Unitàries (test-driven development), generant/executant **Unit Test**, en el mateix moment que s’està desenvolupant.
* El primer objectiu dels **Test automàtics**, és simplificar al màxim els test amb un mínim de scripts, ja que encara que és gaire bé impossible automatitzar tot el procés de test, tampoc és necessari. Per exemple, són bon candidats per automatitzar tant el Test de Regressió com els smoke test.
* Tenint en compte que cada repositori pot tenir diferents branques de desenvolupament, hauríem de construir al JENKINS builds per tal de que qualsevol canvi fet en una branca master, s’integrés automàticament a la resta. (**merges automàtics**)

B- Deploys automàtics

Encara que l’èxit de la entrega contínua radica més en el procés que no pas en l’eina. El que sí que podem assegurar, sobretot si tenim en compte que el procés de pujar codi nou a producció és molt mecànic, és que un cop el procés de la instal·lació està definit i funcionant correctament, ens estalviaríem molt de temps i esforços si el procés fos automàtic.

Per tant, l’ideal és arribar a fer les pujades de la Release a producció de manera automàtica, i s’hauria de donar únicament després de revisar amb èxit el nou codi comitejat.

Les eines més conegudes del mercat que ens ho permet són: **Jenkins** (Open Source) i **Bamboo** (Altassian). Totes dues cobreixen més o menys les mateixes funcionalitats, però un dels avantatges que tindria Bamboo seria la seva connexió més àgil amb el JIRA.

# C- Control de Versions ( CVS / GIT )

En el cicle de vida del software és molt habitual que es produeixin canvis de codi contínuament. Per tal d’evitar conflictes i facilitar que tothom treballi amb el codi el més actualitzat possible, no solament és necessari una bona eina de control de versions, sinó que també necessitem definir processos per a treure-li el màxim profit i minimitzar així problemes amb el codi.

* Definir procés de creació de branques.
* Quan fem merges up?
* Creació de tags per Release.
* Comentaris als commits …. relacionant amb JIRA
* Fer servir el servidor GitHub per a allotjar tots els repositoris GIT.

Links Interès

Model de branques, <http://nvie.com/posts/a-successful-git-branching-model/>

Guia senzilla GIT, <http://rogerdudler.github.io/git-guide/index.es.html>

# D- Release i versions

Per tal d’assegurar la qualitat del software, hauríem de treballar amb versions i Release.

La Release és una versió de llançament a producció, mentre que la versió és prèvia al llançament definitiu del software i no té perquè pujar a producció.

Es recomana la figura d’un **Release Manager** per a la gestió i planificació de les versions i les pujades a producció de totes les Release.

Per això, prèviament s’hauria de tenir en compte:

* Quan es fan versions i/o Release ?
* Quina nomenclatura seguim?
* Com diferenciar allò que es projecte, i allò que és incidència.
* Documentació amb Release Notes i versions …Afegir la informació la Release Notes al confluence.
* Per una millora en la gestió de versions, es recomana arxivar totes aquelles versions que estan fora del Road map.

Per planejar una versió

<https://confluence.atlassian.com/display/AGILE/Planning+a+Version>

Per crear Release Notes

<https://confluence.atlassian.com/display/JIRA/JIRA+Release+Summary>

Exemple com JIRA crea Release Notes

<https://jira.atlassian.com/jira/secure/ReleaseNote.jspa?projectId=10240&version=50907>

Gestió de Projectes més Àgil - SCRUM

# A- Qualitat del software amb Agile

Agile Testing és una part fonamental de la filosofia Agile, en el que l’essència de les funcions de Testing no canvien; solament els mètodes i rols, perseguint el mateix objectiu: contribuir a assegurar la qualitat des de l’inici.

**Variables** que afecten a la qualitat, fent servir agile:

* La qualitat del Software hauria de començar amb la **entrada de requisits**, ja que hauria de ser un dels passos explícits en la etapa d’anàlisis d’un projecte.
* Participació del client**, testejant en les primeres etapes**, fent comentaris curts i prioritzant requeriments.
* Scrum practice, integració contínua, refactoring, experiència dels membres de l’equip, èxit en els compliments, condicions de satisfaccions, desenvolupament guiat de les proves del software (*escriure les proves abans de començar a desenvolupar*), guies de test d’acceptació, proves de cobertura, compromís de sprint.

**Definició de requisits**

Amb el desenvolupament Agile, se li dóna valor al negoci tan aviat com es possible i en unitats funcionals petites. Per la definició de requisits, es fa servir el concepte de backlogs:

* Cada Projecte té assignat un backlog per si mateix.
* Cada backlog té assignat un product owner per a coordinar els diferents Items.
* Cada backlog té sprints per agrupar requeriments.
* Cada sprint té assignades històries, per tal de calendaritzar i prioritzar requeriments.
* Es recomana fer test per cada Sprint. I passar la UAT abans de la Release.

Llistat de **Bones pràctiques de qualitat, en un entorn Agile:**

* Sense cap **estratègia i pla de proves**, és difícil de gestionar les proves. El testing ha de seguir un Planning, unes especificacions i una execució seqüencial.
* **L’equip** de test hauria de ser **independent** per tal de ser efectius.
* **Metodologia de Test de UAT,** El User Acceptance Test ha d’estar clarament definit.
* **La gestió del defectes** es recullen correctament, així com el workflow per verificar la seva correcció.
* **Reporting**, S’han d’entregar periòdicament reports a les parts interessades.
* **Mètriques**: Conjunt de mètriques per seguir el progrés del projecte.

**L’Equip de Test està implicat amb la planificació de la Release** i desenvolupa les següents activitats:

* Escriu històries d’usuari per determinar les proves de cobertura. ( Validació ) - JIRA
* Sol·licita aclaracions sobre històries d’usuari, allà on hi manca informació.
* Defineix els criteris d’acceptació ( UAT). ( Verifica ).
* Determinar la estratègia de proves, Test plans i Test Cases per totes les Releases.
* Indica possibles riscos.
* Defineix el nivell de Test que s’ha dur a cap.
* Executa la planificació de proves.
* Analitza el pla de proves per si cal automatització

# B- Millores en la Gestió de projectes - JIRA

* Actualment, en el JIRA no hi ha una visibilitat Global del que s’està fent, ni quin és l’estat de tot plegat. Es recomana fer una auditoria de tots el projectes i verificar que estan ben categoritzats sota etiquetes ( estat, aplicacions - cmdb .... ).

o Per organització: PIA ( Campus/front i Gestió/back ) / Gestió i processos / Arquitectura / Operacions ... ??

o Per àrees: Aula nova / Bústies Compartides d’aula / Multilingüisme / Vídeo / Binari de Campus / Seguretat

o Per proveïdors

* Neteja tiquets oberts que estan al “limbo” ( analitzar OQ-159 ).
* Introducció de la metodologia Agile al JIRA, de manera que es pugui fer una gestió més AGIL, tant dels evolutius com dels projectes nous:
  + Planificant les tasques amb definició d’històries, sprints i Boards
    - Scrum per a les noves Release
    - Kanban per facilitar les millores incrementals en el procés existent.
  + Millorant la estimació, amb *story points*.
  + Prioritzant, amb noves funcionalitats que ajuden a endreçar i filtrar tiquets.
  + Millorant la eficiència amb Retrospectives, fent servir els reports predefinits (burndown chart, sprint report, velocity chart … )
* Ús del [SERVICE DESK](https://confluence.atlassian.com/servicedesk/getting-started-with-jira-service-desk-408453918.html), per facilitar a l’usuari la creació de tiquets al JIRA.

C- Definició d’Indicadors i mètriques de qualitat

Implantar un bon procés de qualitat, no s’aconsegueix en poc temps i donat que cada empresa es diferent, no es pot imposar cap estàndard. Per tant, es proposa seguir els processos establerts en els **cicles de millora contínua.** Que descriu quatre passos bàsics per aconseguir la millora de la qualitat: planificar, fer, avaluar o revisar i actuar en conseqüència, és a dir, millorar.

Per això es proposa:

* **Definició d’indicadors** de qualitat per entendre i controlar el desenvolupament del nou software i els projectes de manteniment.
* Les pràctiques de control de qualitat deurien estar **documentades** per a millorar la seva repetibilitat: Especificacions, dissenys, business rules, configuracions, canvis de codi, test plan, casos d’ús, report incidències
* **Manual d’usuaris.**
* **Generar informes de qualitat** per a tots els stakeholders, durant el desenvolupament, a partir indicadors i mètriques.

# D- Millores en els WorkFlows

Amb la entrada del concepte de la Release i la forma de treballar Agile, es recomana desvincular dels workflows les instal·lacions i replantejar-se les transicions.

Metodologia de test - Agile

A- Importància del Testing

La gran majoria de bugs s’haurien de detectar en la fase de desenvolupament i la resta en el moment de la UAT. El client final no hauria de trobar cap bug. Per això és necessari seguir un protocol de testeig.

B- Master Test Plan

Encara que sembli que amb la metodologia agile, el paper de proves ha de desaparèixer, ja que es treballa més amb el just in time: Trobo bug i l’arreglo. Quan estem parlant de fer una Release amb noves funcionalitats, hauríem de definir una estratègia de test.

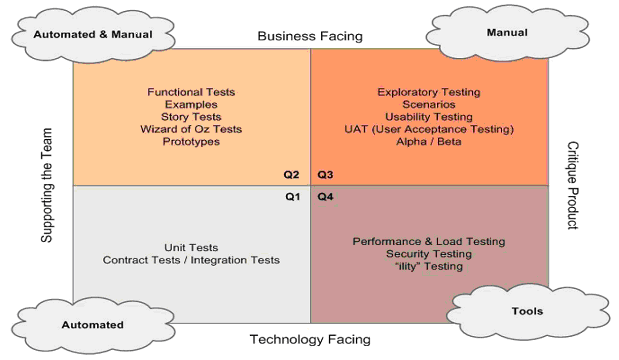
* Definir com **s’organitzaran** les proves
  + Que volem aconseguir amb el Test?
  + En quins mòduls farem les proves?. Que testejarem i que no testejarem.
  + Quins són els Rols de preparació, execució i verificació de les proves?
  + Tenint en compte que la UOC, treballa amb proveïdors. S’hauria de definir prèviament quin grau d’implicació en el test, tindrà cadascuna de les parts.
* **Detallar quins test funcionals s’executaran:**
  + Casos de test per cada sprint. Per validar que tot funciona com s’espera.
  + Llistat amb criteris d’acceptació ( UAT). Revisió i verificació de requeriments.
  + Test de regressió. Per comprovar que un cop pujada tota la Release, tot el sistema segueix funcionant correctament. Moment de fer regressió en tots els navegadors, plataformes i dispositius.

Com a eina per fer els test anteriorment descrits es proposa, en una primera fase, L’Excel ( definint un únic format ) i associar-los a la Release.

* **Definició de Test de Sistema i/o desenvolupament**
  + Quins test de sistema s’executaran: Test de rendiments, test de carga...
  + Unit Test. JENKINS, Sonarqube, FishEye.
  + És el moment de decidir, quins Test funcionals es poden automatitzar.
* Calendaritzar el **Pla de proves**
  + Ja amb la entrada de requisits, s’hauria de fer una **preparació de les proves.** Per tal de fer una revisió de requeriments i saber quines tècniques es faran servir.
  + Seguidament s’hauria d’**escriure els casos de test**. Per saber que provarem i que validarem.
  + Hauríem de **calendaritzar quan s’executen** cada un dels test. Podríem fer-ho a nivell de Historia o a nivell de Sprint, o inclús a nivell de Release.
* Com es **gestionaran les incidències**?
* Definir el **risc i la estimació** de les proves.
  + Quins riscos estem disposat a tenir?
  + Quant volem invertir?
  + Quines Dependències hi ha?
* Definir l’equip de Testing.
* Definir usuaris de proves tant a test com a producció.

**Exemple de Test Cases per a la Bustiaca**. [**Links Excel Test\_BUSTIACA**](https://docs.google.com/spreadsheets/d/1mVeppC-hAmObuXgpgJvAKQJu580cbWGca0Gky1NRxnI/edit#gid=477998914)

C- Fases de Test, per la verificació i validació



**Q1: Fase de Test tecnològic.**

A mida que els programadors acaben de desenvolupar les històries, portaran a cap proves a més baix nivell, per verificar que s’obtenen els resultats esperats.

* Tipus de Test: Unit tests ( Jenkins ), Qualitat de codi ( Sonarqube )
* Objectiu: Ajuden a assegurar que el software s’està construint sota uns principis bàsics. Prèviament, s’ha de valorar en quins desenvolupaments s’ha de fer servir Unit Test.
* Qui: Normalment els desenvolupadors. Però és aconsellable que els testers s’impliquin en aquest tipus de test, ajudant i guiant als desenvolupadors a dissenyar els test unitaris.

**Q2: Fase de Test orientat a la validació de les Històries**

Un cop la història es desenvolupa passa a QA per començar amb la validació de requeriments.

* Tipus de Test: Test funcional de les Històries dels Sprints.
* Objectiu: Assegurar que tot el sistema funciona.
* Qui: Programadors del mateix scrum team en col·laboració dels testers i clients.
* Reporting: Llistat amb l’estat de totes les Histories del Sprint i els seus requeriments. El marc de treball agile és ràpid en la resolució de les incidències, tant aviat es troba un bug es procedeix a arreglar-ho. Per això es convenient fer servir un llistat amb totes les històries i veure l’estat per cada una d’elles. ( Veure Excel )

**Q3: Fase de Test orientat al negoci. Pot estar fora del Sprint**

Una de las últimes fases, és la de User acceptant test (UAT). On s’entrega el projecte per a executar les proves finals i verificar el compliment de tots els requeriments.

Aquesta Fase es pot fer després del Sprint o bé just abans de fer la Release.

* Tipus de Test: UAT test. Test d’exploració.
* Objectiu: Verificació requeriments.
* Qui: Client Final

**Q4: Fase de Test orientat al Producte. Fora del Sprint**

Al final, fem la proves de regressió, on busquem garantir que els canvis no han introduït nous errors i afectin a altres parts del producte. En aquest punt , l’equip de control de qualitat fa un cross-browser complert, provant funcionalment en tots els idiomes, navegadors, plataformes i dispositius.

* Tipus de Test: Test de avaluació, Test de regressió, Test més tècnics ( ex. Performance tests, load i stress tests, security i recovery tests.)
* Objectiu: Avaluar el producte, a nivell d’escalabilitat, estabilitat, confiabilitat, manteniment, compatibilitat.
* Qui: Perfils tècnics amb domini de les eines específiques.

Estratègia QA per la Release Campus 2016

**Actualment**, no hi ha cap equip de Test destinat a verificar i validar. És el cap de projecte el que verifica la qualitat de cada pujada a producció. Per fer-ho es crea una finestra, per preparar l’entorn necessari per a la realització de les proves.

En el JIRA hi ha un apartat “Test Funcionament”, on es descriu quina és la aplicació que s’ha de testejar per cada tiquet. No hi ha definit quina és la bateria de proves que s’ha de realitzar. No es descriu el resultat final de les proves. Tampoc sabem en quins navegadors s’han provat. Ni si s’han verificat tots els requeriments. Alhora de pujar la instal·lacions d’una incidència, al JIRA, no hi ha constància de que el tiquet hagi estat solventat i provat i que no hi hagi hagut cap efecte colateral ( regressió )

**Pel conjunt de millores i peticions** recollides a la [Versió Campus 2015\_](http://confluence.uoc.edu/pages/viewpage.action?pageId=11076748)1, tant per la Gestió de processos com per part del campus, es proposa fer servir la metodologia Agile.

Tenint en compte que les millores s’agrupen principalment en les següents aplicacions, l’ideal seria fer una release per a cada un d’ells:

1. AULACA, Aula nova
2. BUSTIACA, Bústies Compartides d’aula
3. WEBCOMP, Framework Responsive basat en Web Comp
4. AULAMLING, Aula Multilingüe
5. MEDIA, Uoc Media
6. BIN, Binari de Campus
7. SEGUR, Seguretat

# JIRA

Si seguim amb mètode Agile, aleshores :

* L’ideal seria, saber quina relació de projectes JIRA té associats cada Release i fer un **Board** del tipus scrum per a cada un d’ells.
* Cada Board té associat un **Backlog** amb totes les històries que s’ha de fer.
* Les **Històries** de cada un d’ells corresponen als tiquets de JIRA assignat a cada project.
* Per tal de calendaritzar i prioritzar cada una de les històries, l’ideal seria agrupar les històries amb **Sprints**. La data inici i de fi dels sprints, s’haurien de definir a priori. Un sprint no hauria de durar més de tres setmanes.
* Per fer un seguiment del treball fet i del restant, hi ha la opció de fer **Reports**, això ajuda a l’equip a gestionar el progrés i respondre amb conseqüència.
* Tenint en compte, que un cop es defineixen els sprint, ja no es pot modificar, es proposa crear un altre Board del tipus Kanban: **Expedite**, per tal d’anar solventant totes aquelles incidències que queden fora de la Release de la [Versió Campus 2015\_](http://confluence.uoc.edu/pages/viewpage.action?pageId=11076748)1 i que són Urgents.
* Les versions haurien de correspondre amb els sprint i a la vegada amb la Release. Sempre definides a priori. Encara que si solament fem dues pujades a producció a l’any. Podem tenir diferents Releases.

# Creació de la Release

Coincidint amb l’inici del nous semestre, es fa el deploy a producció dels nous requeriments del Campus. Per això el model de Release que millor s’adapta, tenint en compte Agile seria la de treballar amb Sprints.



Cada sprint té associada una Versió. Però realment s’hauria de diferenciar la versió de la Release que pujarà a producció. Aquesta diferenciació la donarem amb la nomenclatura de la versió.

# Versionatge

Hauria de seguir un criteri comú

1. AULACA
   1. 1.3.20 -09/06/2015
   2. 1.3.21 -30/06/2015
   3. 1.4.00- 03/08/2015
   4. 1.5.00 - 05/02/2016
2. BUSTIACA, Bústies Compartides d’aula
   1. 1.1.1 01/07/2015
   2. 1.2.0 ????
3. MEDIA, Uoc Media
   1. [1.0.1](http://jira.uoc.edu/jira/browse/MEDIA/fixforversion/18784) Primera Install en OT, 23/05/2015

# Reports

Durant el desenvolupament del Sprint

Per saber l’estat de la Release en cada moment.

Per saber que s’ha fet amb anteriors Releases.

Saber la velocitat de cada equip.