


Gestion des développements

Développement logiciel

Processus développement

- Créer le logiciel selon les demandes du client (requirements)

- Gestion dév. et évolutions
 - Environnements
 - Configuration management
 - Contrôle des changements (Change control - CoC)
- 

Processus Management

- Planifier le travail
- Planifier les livraisons
- Allouer les ressources
- Gérer le budget, les coûts
- Surveiller l'avancement des travaux
- Gérer les risques.

Environnements de développement

Projet AE

- Outils de développement
 - Eclipse / IntelliJ
 - DataGrip / Pgadmin / ElephantSQL
 - Visual Studio Code
- Ne pas oublier
 - PostgreSQL
 - Browser
 - Serveur de fichiers
 - Maven (dépendances)
 - Token JWT (authentification)
- MAIS AUSSI et surtout:
 - Jakarta / Jersey
 - Jenkins
 - CPD/PMD
 - JUnit5
 - Check styles
 - Jacoco
 - Javadoc
 - Git (Gitlab)
 - www.coursinfo.ipl.be
 - Discord / facebook / Slack / Teams
 - ...

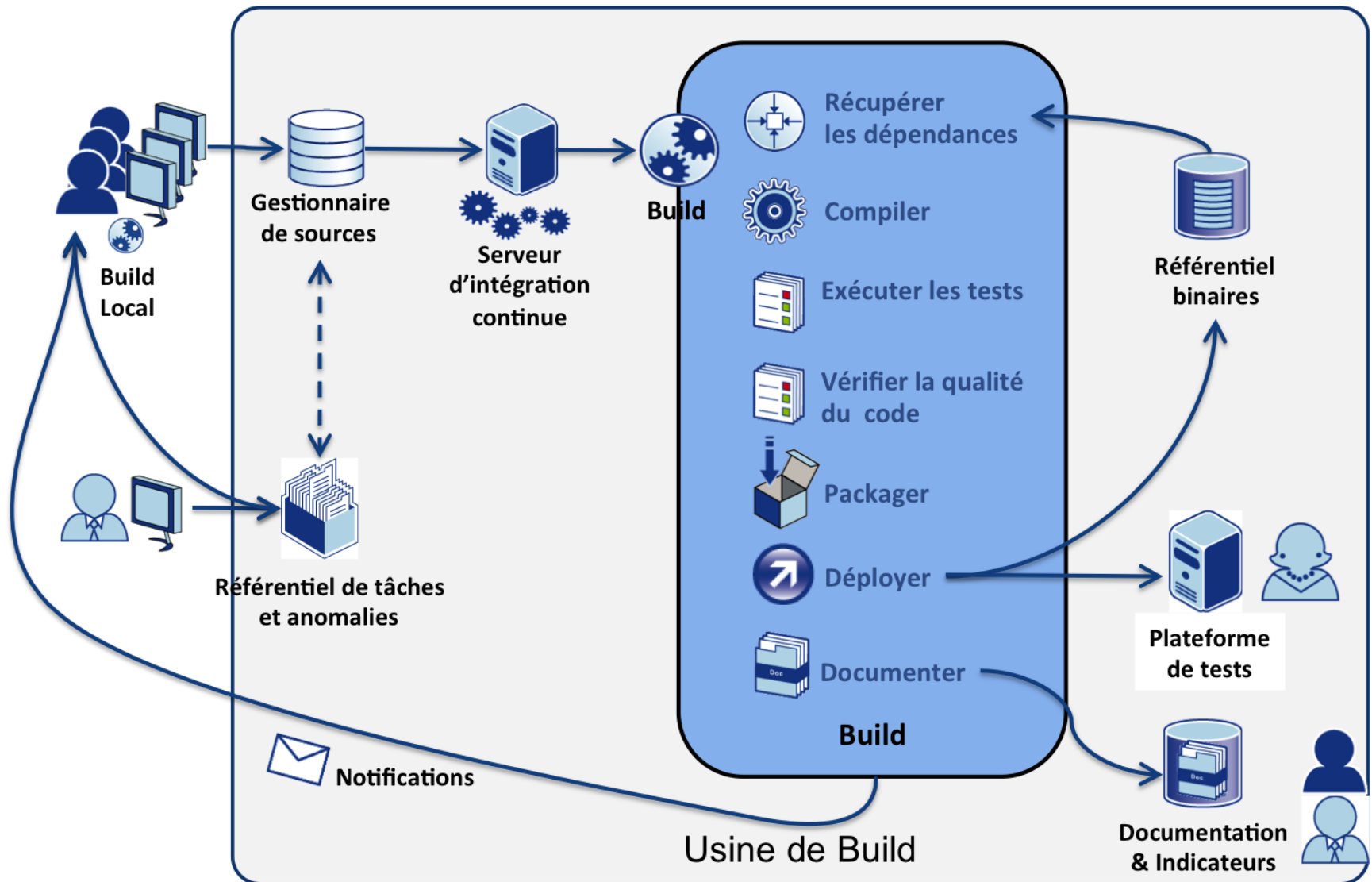
Usine logicielle

- Un ensemble structuré de logiciels **qui fonctionnent ensemble** pour permettre le développement.
 - Non seulement outils de programmation, compilation, tests, déploiement et sauvegarde (gestion de sources...) mais aussi
 - de gestion de documents
 - de gestion du cycle de vie d'un projet, de suivi des activités
 - de gestion des licences.
- Et **tableaux de bord**.

➔ **Outils connectés entre eux**

//
Intégration continue

Exemple



Langages utilisés



Pour la partie back-end, le langage Java



Pour la partie front-end

- JavaScript.
- HTML & CSS.

Frameworks utilisés



ARHS
Developments
Belgium – client
Proximus

Nombreux stages
d'observation

Base :
Ronsmans Thomas

2016-2017
& mises à jour

Qualité du code



- Software qui vérifie la qualité du code
- Exécute au moins quatre outils :
 - Checkstyle
 - PMD
 - Findbugs
 - Corbertura

Outils de développement

Jira Software



- Créer des user stories et des tickets
- Planifier des sprints
- Affecter des tâches
- Définir des priorités et états pour les tâches
- Visibilité totale

ARHS Developments Belgium

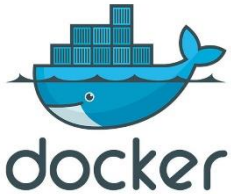
Outils de développement



IntelliJ IDEA est un environnement de développement pour le langage Java.



Maven est un outil libre qui facilite grandement la construction de projets Java. Il facilite l'automatisation de tâches telles que la compilation, les tests unitaires et le déploiement des applications du projet



docker est un logiciel dont le but est de faciliter les déploiements d'application. Il permet de créer des environnements isolés appelés containers, servant à contenir des applications et leurs dépendances. L'intérêt est que l'application embarquée fonctionnera au sein du container indépendamment du serveur ou de la machine sur laquelle elle se trouve

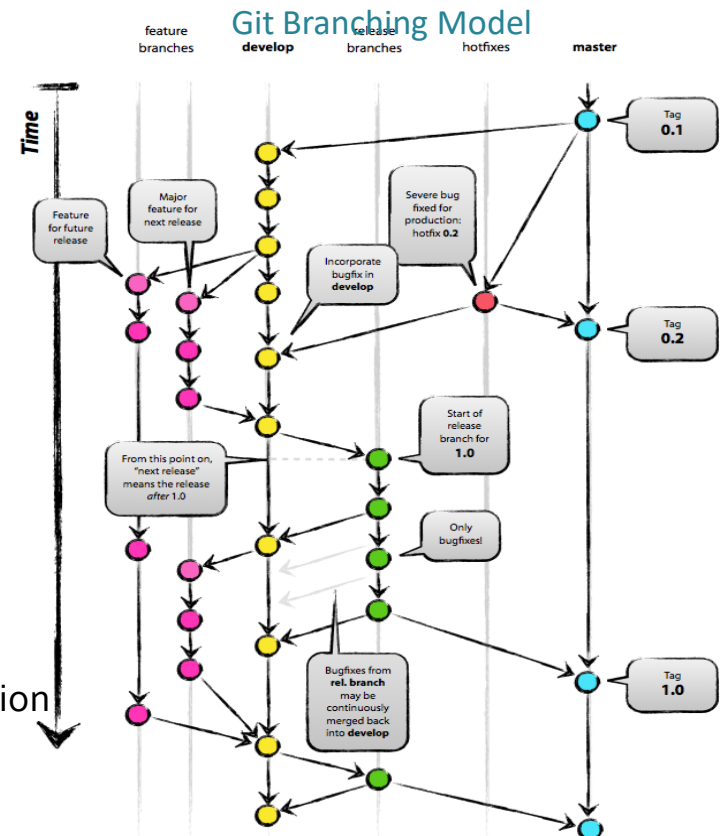
Outils de développement



- Branching model :
 - Branches
 - Commits
 - Updates
- Séparation du travail
- Chaque membre d'équipe peut travailler indépendamment sur sa version
- Pas de problèmes générés lors de la mise en commun

GitLab

- Repository central : contient le code source de l'application
- Composé de plusieurs directory
 - Directory principal (Projet)
 - Plusieurs autres directory => Modifications des programmeurs



ARHS Developments Belgium



Jenkins

Jenkins :

- Outil open source
- Intégration continue
- Rebuild le package lors d'un évènement
 - Commit
 - Update
 - ...
- Test d'intégration et unitaires automatiques
- Personnalisable

ARHS Developments Belgium



maven



Git

GitLab



Jenkins



Commentaire prof
+ autres logiciels non décrits dans le stage

Gestion de configuration

"Software Configuration Management is a disciplined approach to managing and controlling the evolution of software development and maintenance practices and their software products as they change over time" [© Ovum Ltd 1998](#)

Gestion de configuration

- Gérer la description technique d'un système et de ses divers composants ; gérer **l'inventaire de tous les éléments de la configuration** (configuration items).
- Gérer **l'ensemble des modifications** apportées au cours de l'évolution du système (→ contrôle des changements apportés au système).
- Gérer **les versions** (releases).

Pourquoi ?

Gestion de la complexité

- Nombreux développeurs.
- Sous-systèmes interdépendants.
- Différents langages.
- Évolution des systèmes.

Pour avoir un logiciel

- Fiable, robuste,
- Maintenable ...

Voir qualité du logiciel -
chapitre 2(partie 2)

Objectifs /1

Exemples d'objectifs poursuivis :

- Que les demandes de changement et les rapports de problèmes (bugs) soient gérés dans le système
 - Utilisation d'un programme de gestion de bugs.
 - Vérification que chaque demande de client soit introduite dans le logiciel.
- Que des bugs fixés (résolus) précédemment ne réapparaissent pas mystérieusement
 - Si cela arrive, le bug est rouvert et est marqué dans le programme de gestion de bugs.
 - Mesure : comptage du nombre de bugs rouverts dans une version : il devrait être égal à 0.

Objectifs /2

- Que les activités de développement soient sous contrôle
 - Mesure : il est possible de produire un état des développements en moins d'une demi-journée.
- Que la documentation soit à jour et correcte
 - Ceci est difficile à atteindre.
 - Une solution est de maintenir un numéro de version et la date de dernière mise à jour du document dans chaque document produit.
 - Mesure : chaque document doit avoir un numéro de version et la date lui correspondant.

Objectifs /3

- Que tous les « configuration items » soient identifiés.
- Que chaque release puisse être construite précisément.
- Que les différences entre deux releases soient facilement identifiées.
- **Qu'une release envoyée à un client soit complète** (pas de fichier manquant).

Gestion de configuration

1. Inventaire
2. Contrôle des changements
3. Gestion des versions.

1. Inventaire des éléments et leur gestion

Inventaire

- Informations

- Documentation.
- Méthodes de travail.
- Spécifications techniques.
- Définition des bases de données.
- Scénarios de tests.
- Enregistrement des tests.
- Dossiers de validation.
- Composants externes.
- Dossiers sécurité.

- Code source.
- Environnement.
- Exécutables.
- Ressources nécessaires au développement.
- Fichiers de paramétrisation (INI...).
- Scripts pour créer la base de données.
- ...

Inventaire (2)

- Organisation des informations.
- Conventions de nommage.
- Nomenclature de versions de fichiers.
- Documents relatifs à l'assurance et au contrôle qualité.
- Standard.
 - Un standard est une **spécification technique établie par l'équipe** et qui **repose sur les acquis** de la science, des technologies et de l'expérience.
- ...

Gestion de configuration

1. Inventaire
2. Contrôle des changements
3. Gestion des versions.

1.1. Organisation des informations

Organisation

- Définir l'arborescence pour stocker les différents items de configuration et le lien avec le numéro de version du logiciel:
 - les fichiers de documentation.
 - les exécutables.
 - les jeux de tests.
 - les enregistrements de l'exécution des tests...
- Définir la structure des numéros de version du logiciel.
- Définir la structure des documents.

Documents relatifs à l'assurance et au contrôle qualité










- Voir leur place dans l'organisation des informations.
- Faire le lien avec le plan qualité (voir chapitre 4).

Nomenclature

Exemple : n.m.ooooo.p

- n.m : release majeur
- ooooo : numéro de build
- p :
 - par défaut, valeur 0
 - si branchement, représente le numéro séquentiel de branchement sur le numéro de build

Exemple Microsoft 2016

 bfsvc.exe	16/07/2016 13:42	Application	60 KB	10.0.14393.0
 hh.exe	16/07/2016 13:42	Application	18 KB	10.0.14393.0
 notepad.exe	16/07/2016 13:43	Application	238 KB	10.0.14393.0
 winhlp32.exe	16/07/2016 13:42	Application	10 KB	10.0.14393.0
 write.exe	16/07/2016 13:42	Application	11 KB	10.0.14393.0
 splwow64.exe	22/11/2016 23:55	Application	128 KB	10.0.14393.351
 explorer.exe	04/03/2017 08:03	Application	4,565 KB	10.0.14393.953
 regedit.exe	04/03/2017 07:18	Application	313 KB	10.0.14393.953
 HelpPane.exe	28/03/2017 07:14	Application	953 KB	10.0.14393.1066

Gestion de configuration

1. Inventaire
2. Contrôle des changements
3. Gestion des versions.

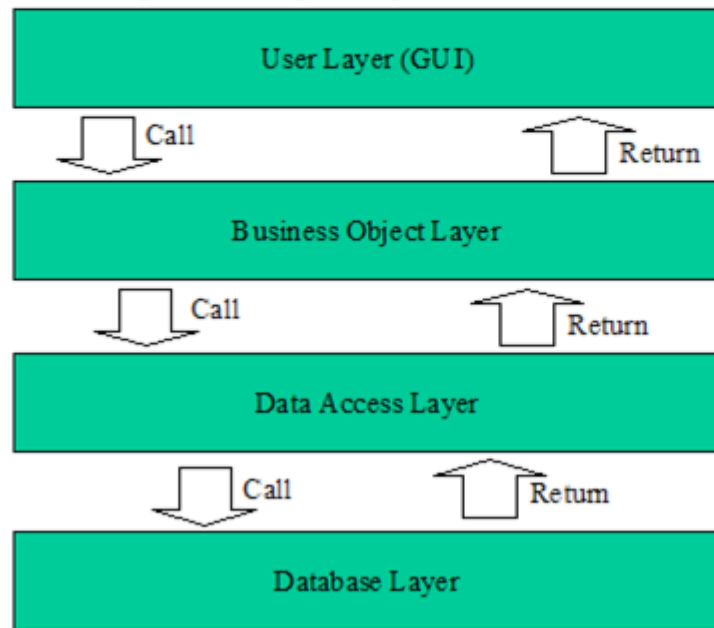
1.2. Standards

Les exemples sont inspirés des bonnes pratiques développées dans la société IRIS sa jusqu'en 2009.

Standards d'architecture

1 Program Structure Standards (physical & logical)

1.1 Logical



Objects and functions within one layer are only allowed to access objects and functions in the layer immediately below.

This greatly improves portability and re-use of objects - even more than that, though, is the fact that it makes the code much easier to understand.

Standards d'architecture

1.1 *Physical*

- Comment est organisé le code ?
- Comment est faite la découpe en composants ?
- Où aller chercher les composants externes ?
- Où sont stockées ces informations (fichier INI, properties...) ?

Standards de programmation

1 Coding Standards

1.1 General

- Définition du style de codage.
- Langue utilisée pour les commentaires du code.
- Stockage des textes en dehors du code pour permettre le multilinguisme**.
- Convention de nommage des fonctions (Ex : CamelCase - noms significatifs).
- Eviter l'utilisation de constantes hardcodées.
- Ne pas dupliquer du code : refactorise.

** ceci est repris dans une documentation des détails d'implémentation.

Standards de documentation

5 Documentation Standards

We are managing the documentation, version by version.

5.1 *Code Documentation*

- Souvent la documentation du code est trop détaillée et s'assurer qu'elle soit à jour devient un problème laborieux.
- Il faut définir ce qu'il faut documenter et à quel niveau de détails.
- Exemple : documenter les fonctions et les classes qui sont exportées ; les dépendances ; les fichiers, les tables de la base de données...

Standards de documentation

5.2 Database Documentation

- Que faut-il inclure ?
 - Un diagramme de structure de données.
 - Les triggers et les procédures stockées.
 - Le type de chaque champ, sa longueur et une courte description de son contenu
 - ...
- Il ne faut jamais reproduire le code lui-même dans ma documentation. Pas de duplication !

Templates

Les standards couvrent aussi le template de document.
Tous les documents d'un même type utilisent le même modèle.

- ➔ Confort &
- ➔ Automatisation possible.

Exemple :

- Template tests fonctionnels.

Gestion de configuration

1. Inventaire
2. Contrôle des changements
3. Gestion des versions.

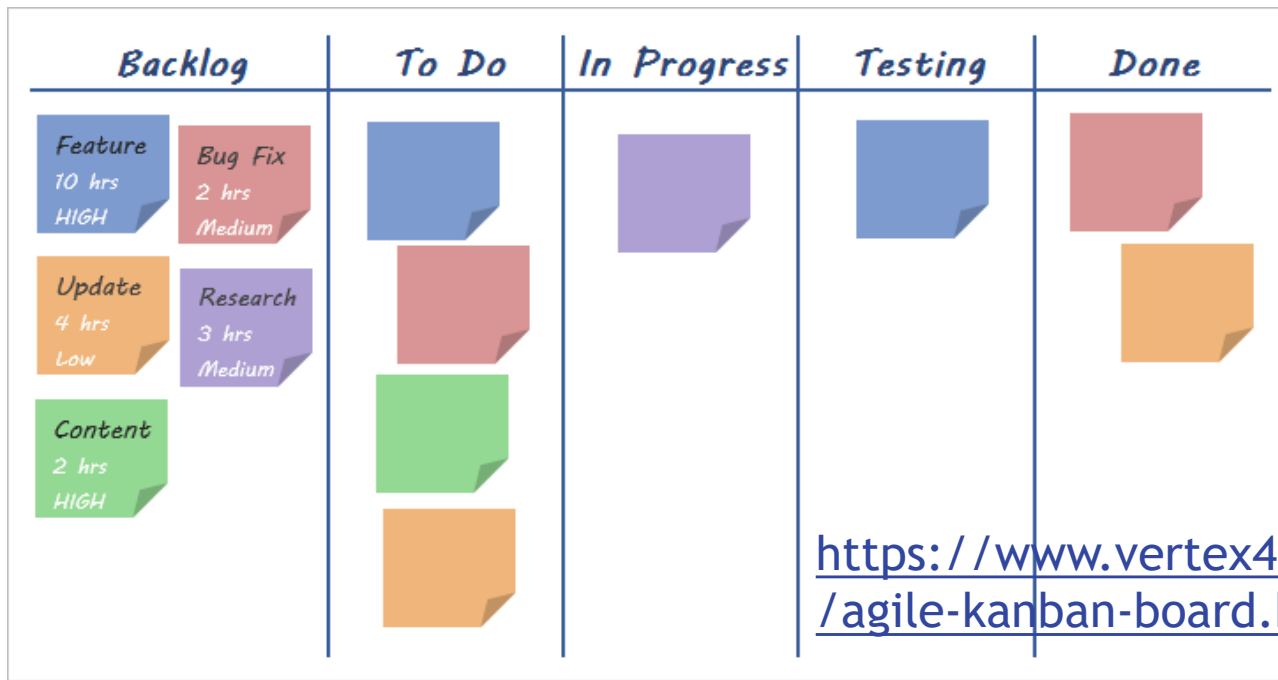
1.3. Éléments des méthodes de travail

Exemples

Les slides suivants vont tous présenter des exemples d'éléments que l'on définit dans une méthode de travail.

Chaque équipe décrira plus ou moins formellement ses méthodes de travail et insistera sur les points les plus importants.

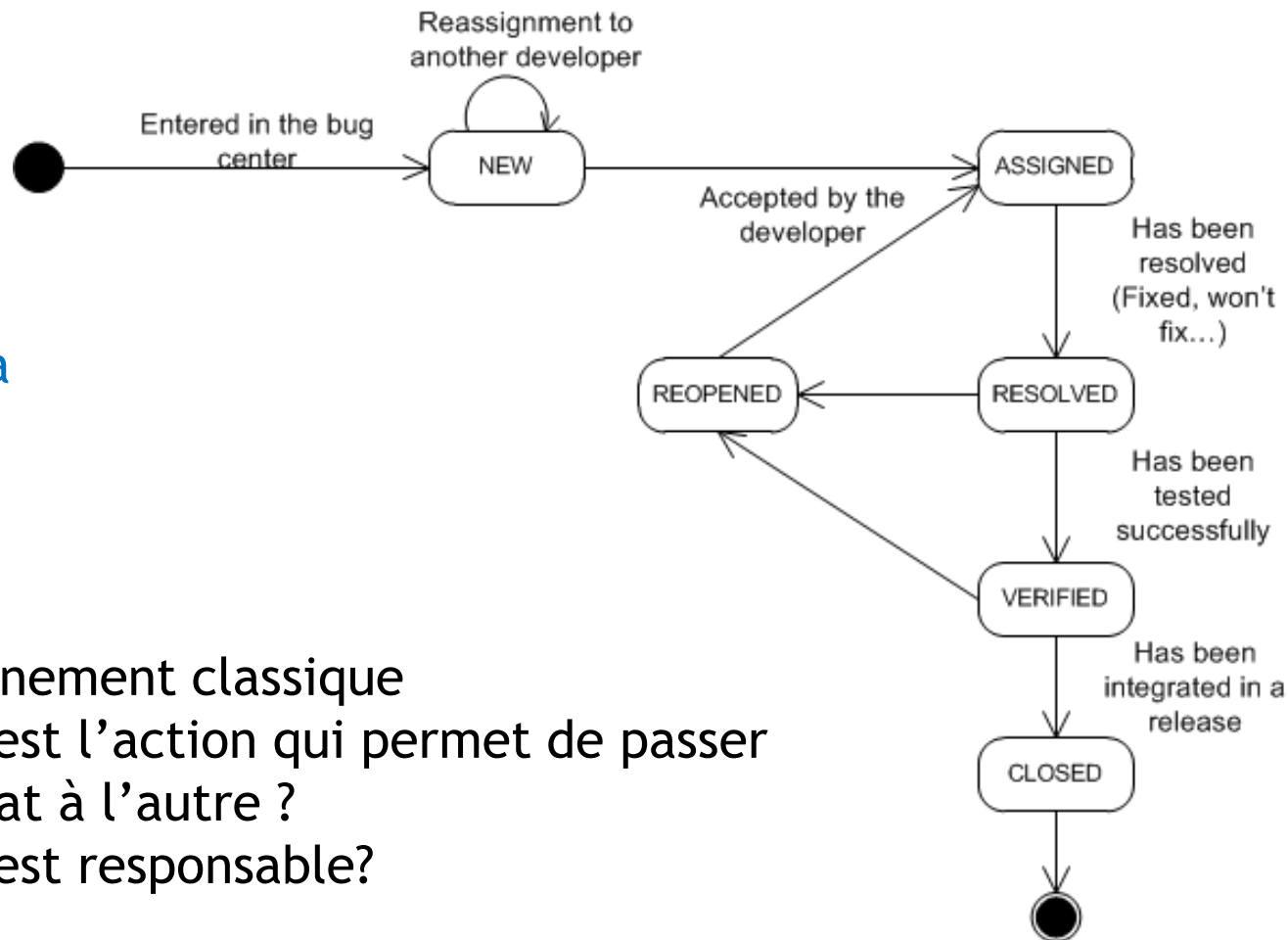
Suivi du backlog



Environnement Agile

- Quelle est l'action qui permet de passer de « Testing » à « Done » ?
Que se passe-t-il si le test a échoué ?
- Qui en est responsable ?

Suivi des demandes et des bugs



Ex: Jira

Environnement classique
Quelle est l'action qui permet de passer d'un état à l'autre ?
Qui en est responsable?

Criticité des bugs

La classification utilisée pour développer le logiciel :

- *Bloquant* : le problème bloque la production sur le site du client.
- *Majeur*: le problème bloque la production mais il existe un moyen de le contourner.
- *Modéré* : le problème n'empêche pas la production de travailler.
- *Mineur*: le problème est cosmétique.

Niveaux de tests

Connaître les responsabilités et ce que signifie chacun des niveaux :

- **Unit testing:** each programmer handles the Unit testing of what he has coded.
- **Link testing:** once its programmer has released a module, another programmer is in charge of integrating and testing it.
- **System testing:** it is done by the testers who review the Test Plan according to the specifications.
All module Test Plans are integrated together within the Global Test Plan who tests the system.
- **Acceptance tests:** the client plays the Acceptance Test Plan until he accepts the system.

Critères pour accepter une version

Les critères sont:

- Le produit suit les demandes qui étaient prévues pour cette version.
- Les demandes induites ont été testées.
- Il n'y a aucun bug bloquant.
- Les bugs majeurs d'une version précédente ont tous été corrigés.

Gestion de configuration

1. Inventaire
- 2. Contrôle des changements**
3. Gestion des versions.

2. Contrôle des changements

Contrôle des changements

Control of Changes (CoC)

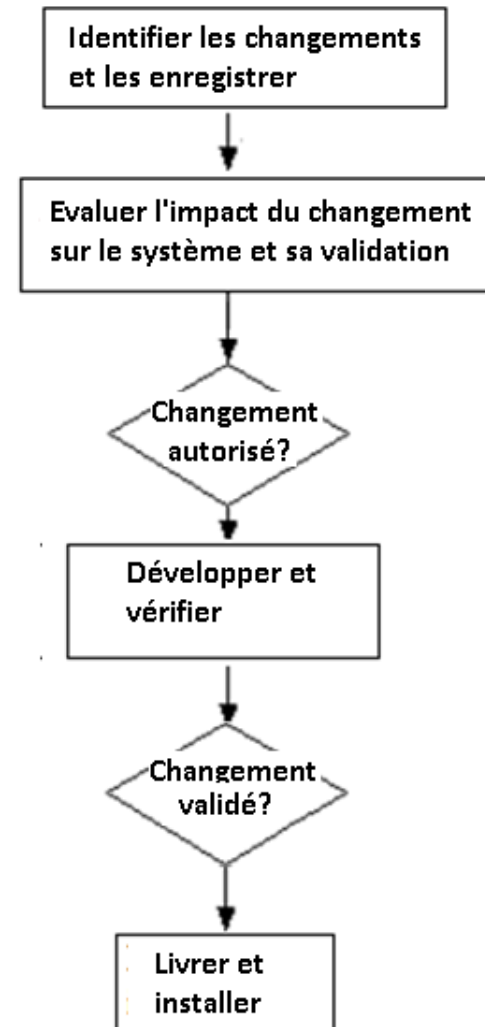
Pourquoi ?

- Première validation du software a décrit un état "validé".
- Exigence qualité : avoir un software validé tout au long des différentes versions.

Déterminer les actions nécessaires pour s'assurer que le **système soit maintenu dans un état validé.**

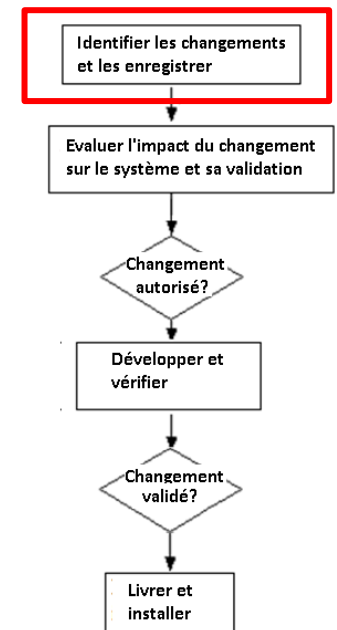
Contrôle des changements

1. Identifier les changements et les enregistrer.
2. Evaluer leur impact.
3. Décider de les implémenter ou non.
 - Si oui,
 - Les développer et les tester.
 - Livrer et installer.



1. Identifier les changements et les enregistrer

- Qui ?
- Comment sont reportées les demandes de changement ?
- Quel logiciel de suivi des changements ?
- Informations à enregistrer ?
 - Auteur de la demande (ex: client)
 - Auteur de l'enregistrement
 - Demande
 - Descriptif
 - Date
 - Etat « nouvelle demande »
 - ...



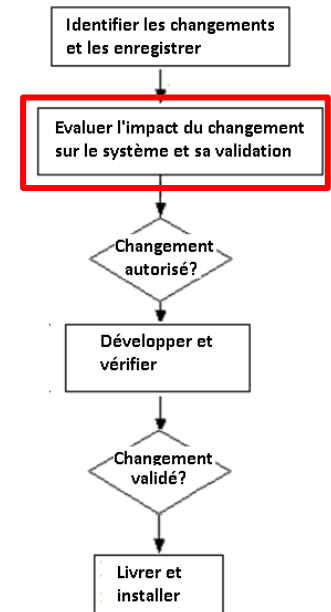
2. Evaluer l'impact des changements

Description impact

- Changements hardware.
- Changements logiciels.
- Changements d'équipement.
- Changements de processus.
- Besoin de formation.
- Changements dans la documentation.
- Impact sur les autres systèmes.
- Impact sur la validation.

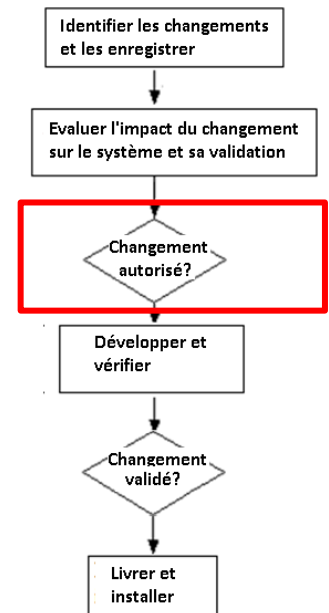
Evaluation

- Spécifications.
 - Délai.
 - Budget.
- &
- Impact :
 - Mineur.
 - Majeur.



3. Décider de les implémenter ou non

- En fonction de l'évaluation faite, les demandes de changements sont prises en compte dans les développements actuels ou non.
- Si les changements sont implémentés, ils suivent le cycle de développement.



Gestion de configuration

1. Inventaire
2. Contrôle des changements
3. **Gestion des versions.**

3. Gestion des versions

Préparer une version

- Démarrer le développement d'une nouvelle version :
Quand ? Comment ? Pourquoi ?
 - Où sauver l'information ?
 - Liste les demandes prises en compte.
- Préparer une version :
 - Changement de / nouvelle technologie?
 - Spécifications (correspondant aux nouvelles demandes).
 - Préparation des environnements.
 - Préparation du document de contrôle des changements (exemple).

Développer et terminer

- Pendant le développement de la version, **contrôle des changements** :
 - Suivi.
 - Nouveau code.
 - Nouveaux tests.
 - Revue du code.
- Préparation du déploiement.
- En fin de changement, décommissionning
 - Mettre à jour les documents.
 - Enregistrer la version dans le document de configuration.

[IRIS-SCM Releases.doc](#)

Questions ?