Plan Qualité 2014

Projet de BTS IRIS   
Version 3.0

Sommaire

1. But, domaine d'application, responsabilités 4

1.1. But 4

1.2. Domaine d'application 4

1.3. Responsabilités 4

2. Documents applicables et de référence 4

2.1. Documents applicables 4

2.2. Documents de référence 4

3. Démarche de développement 4

3.1. Cycle de vie itératif des 'petits projets' (que nous utiliserons) 5

3.2. Développement des objets de métier 5

3.3. Evaluation des itérations 6

3.4. Planification des itérations 6

3.5. Exemple de processus de modélisation UML simplifié 7

3.6. L'architecture 7

3.7. L'analyse objet 7

3.8. Test et validation 8

3.9. Condition de sortie de phase 8

4. Documentation 8

4.1. Style des documents 8

4.2. Organisation des dossiers sur le disque pour chaque projet 10

5. Documents à produire au jury final (2 exemplaires) 12

5.1. Dossier technique de projet 12

6. Activités complémentaires 12

6.1. Réunions d’avancement 12

6.2. Suivi d’activité 13

6.3. Réduction des risques 13

6.4. Suivi de l'application du plan qualité 14

6.4.1. Contrôle 14

6.4.2. Contrôle qualité 14

6.5. Gestion de version et de configuration 14

6.6. Gestion des modifications 15

7. Méthodes, outils et règles 15

7.1. Cahier des charges 15

7.2. Analyse 16

7.3. Conception d'un incrément 16

7.3.1. La conception logique 16

7.3.2. La conception physique 17

7.3.3. Algorithmique et Codage 17

7.3.4. Test et validation 17

7.4. Outils 18

7.5. Environnement de travail 18

8. Annexes 18

# But, domaine d'application, responsabilités

## But

Ce document a pour objet de décrire les règles à respecter pour la réalisation de la totalité d'un projet informatique dans le cadre d'un développement qui sans respecter au sens strict la norme actuelle, cherche à initier et à faire accepter la démarque qualité et ces principes de bases. Le but n'étant pas de contraindre mais persuader de la validité de cette démarche.

Le respect du plan qualité est devenu un critère principal dans le choix des entreprises fournisseurs par les clients institutionnels et de plus en plus par tous.

## Domaine d'application

Ce plan qualité concerne le logiciel développé lors des projets de BTS Informatique Industrielle.

## Responsabilités

Pour chaque projet seront définis :

* la personne tenant les rôles du demandeur et du représentant des utilisateurs,
* la personne responsable du suivi qualité et du développement,
* les personnes responsables du développement.

La responsabilité du plan qualité est à la charge de Mr Dumas.

# Documents applicables et de référence

## Documents applicables

Leur application est imposée et les documents sont fournis par la section :

* Cahier des charges v1.0,
* Plan qualité,
* Standards de codage du C++, en Java, en algorithmique, etc.
* Règle de description des IHM,
* Fiches de test, lecture croisée, de demande de modification, de suivi de modification, etc.

## Documents de référence

Ils sont utiles au développement mais leur application n'est pas imposée, ni vérifiable :

* Cahier des charges v1.0,
* Méthode UML
* Spécification du ou des sous système.

# Démarche de développement

La démarche de développement utilisera la modélisation objet. Le langage de description utilisé est UML. Le processus d'élaboration de ces modèles n'est pas décrit par UML.

UML précise que le processus devra être :

* piloté par les cas d'utilisation,
* centré sur l'architecture,
* à déroulement itératif (et incrémental).

## Cycle de vie itératif des 'petits projets' (que nous utiliserons)

Le cycle de vie itératif est basé sur l'évolution de prototypes exécutables et donc concrets : l'utilisateur peut utiliser les incréments déjà réalisés. Ce cycle de vie demande plus d'attention et d'organisation.

Chaque itération reproduit le cycle de vie en cascade à une petite échelle.

**Les phases de la première itération sont : la définition des besoins, l'analyse, la conception, le codage, l'intégration, l'évaluation et la livraison.**

**Dans les autres incréments, sauf situation de demande de modification de besoin ou d'analyse, les phases sont limitées à : la conception, le codage, l'intégration, l'évaluation et la livraison.**

Analyse

Implémentation

Validation

Définition de l’itération

Produit une version

Conceptionion

Définition

**L'analyse des besoins est donc complète au 1er incrément**. Si possible, la connaissance des besoins ne doit pas changer avec les itérations.

La première itération est plus difficile car elle demande la mise en place de l'environnement, de l'équipe et du déploiement. Elle sert à valider l'architecture.

**Le nombre d’incréments sera compris entre 2 et 3**

## Développement des objets de métier

Au travers des développements de divers projets et des connaissances de l'entreprise, il est essentiel d'avoir une réflexion sur les objets de métier que l'on doit développer ou acquérir.

Cette réflexion est lourde de conséquence puisqu'elle va influencer les développements à venir et donc leur coût.

**Des actions de Reverse Engineering seront menées afin d'intégrer des modèles de classes déjà développés ou de modéliser des bases de données existantes.**

Il faudra aussi créer des objets d'interface avec des applications existantes.

## Evaluation des itérations

Chaque itération doit durer entre 1 et 3 mois.

Pour chaque itération, il faut :

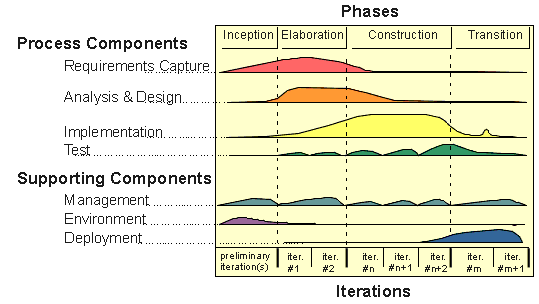
* une revue de départ qui fige les objectifs de l'itération, les critères d'évaluation et la liste des cas d'utilisation ou des scénarios à réaliser,
* une revue d'évaluation qui valide les résultats de l'itération par rapport aux critères des scénarios à réaliser et produit une application exécutable.

## Planification des itérations

Le cycle de développement est perçu comme une suite d'itération où le logiciel évolue par incréments, donnant à chaque fois un nouveau prototype.

Les critères de choix du plan des incréments sont :

* Priorité à l'architecture et aux risques lourds dans les premiers incréments,
* Répartition des autres risques au fil des incréments,
* Priorité au codage lourd dans les incréments intermédiaires,
* Transition vers l'exploitation pour les derniers incréments.

**Répartition des charges dans les divers incréments (source Rational)**

**Etude d’opportunité** : Etude de marché, spécification du produit final et définition de la portée du projet.

**Elaboration** : Spécification des particularités du produit, planification des activités, des ressources, conception et validation de l’architecture du logiciel.

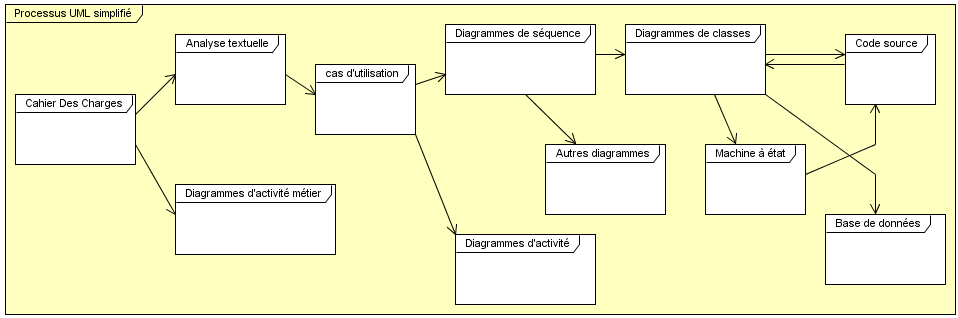
**Construction** : Réalisation du produit, adaptation de la vision, de l’architecture et des plans jusqu’à la livraison du produit.

**Transition** : Fabrication, formation, déploiement, support technique et maintenance.

Le premier prototype est souvent une maquette, qui permet de visualiser les concepts de l'application. Chaque itération est basé sur un nombre réduit de scénarios qui s'attaquent d'abord aux risques importants et détermine les catégories à construire dans l'itération. Les catégories forment un espace de travail bien adapté aux développeurs.

## Exemple de processus de modélisation UML simplifié

Exemple de séquencement des diagrammes UML dans un processus de développement :



Remarques : Les **dossiers d'analyse et de conception** seront élaborés à partir de descriptions textuelles, des diagrammes UML produit pour ce projet et d'annexes.

## L'architecture

Elle concerne plusieurs points de vue :

* la vue des cas d'utilisation (acteurs, cas d'utilisation)
* la vue logique (objets, classes, collaborations, catégories)
* la vue de réalisation (modules, sous-programmes, tâches, sous-systèmes)
* la vue des processus (tâches, threads, interactions)
* la vue de déploiement (temps de réponse, performances, bande passante et capacité de communication, contraintes physiques, tolérance de panne, etc..).

## L'analyse objet

Elle identifie les besoinset l'environnement du système sans décrire *comment*.

Le cahier des charges ne sera qu'un point de départ à la description des *cas d'utilisation* (uses cases) afin de déterminer les besoins réels de l'utilisateur. L'analyse se poursuit par la modélisation de l'application, c'est à dire l'identification des objets, des classes et des interactions entre ces objets.

Les modèles que l'on peut mettre en œuvre en analyse, dans un ordre quelconque, sont :

* les cas d'utilisation,
* les diagrammes de classes,
* les diagrammes de collaboration et de séquences,
* les spécifications de classes et d'objets,
* les diagrammes d'états (pour les classes à comportement fort).

Elle conduit à un premier modèle statique du système. C'est ce même modèle élaboré en analyse qui va être utilisé (modifié, enrichi, etc..) pendant tout le cycle de vie du système.

On utilisera un logiciel de gestion de version du modèle afin de conserver la trace du modèle d'analyse.

## Test et validation

On utilise les scénarios de l'incrément concerné pour construire les plans de tests et les procédures associées. La méthode incrémentale nécessite que l'on réalise les tests de non régression (rejeux des tests des incréments précédents).

**Les tests doivent donc être, si possible, automatisés.**

Les tests unitaires concernent les classes qui seront testées dans un ou plusieurs tests 'boite noire'.

Sans outil disponible, les tests des IHM (Interface Homme Machine) seront réalisés de manière interactive par un testeur. Dans ce cas une fiche de test complète, quand au déroulement du test, devra être produite.

Des tests d’intégration devront être réalisés à chaque intégration.

Un test de validation permettra la validation du produit en fin d’incrément.

## Condition de sortie de phase

* Développement de la phase terminé,
* Documentation de cette phase produite,
* Respect du plan qualité.

# Documentation

Chaque phase de développement (définition, analyse, conception, construction, validation) est caractérisée par la production de documents.

Ces documents sont organisés en arborescence, dont les intitulés correspondent à ces phases. Les documents obtenus à la sortie d'un incrément sont utilisés comme point d'entrée de l'incrément suivant.

**La notion d'incrément n'est pas visible dans les documents finaux.**

Pour en conserver la trace il faut faire appel à un logiciel de gestion de version.

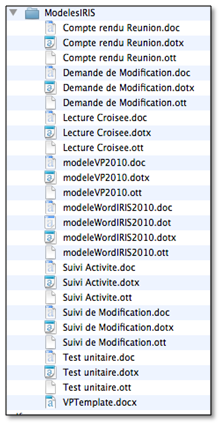
## Style des documents

Les modèles de documents sont fournis pour Word et pour Open Office et sont réunis sur le NAS de la section avec le plan qualité.

A chaque nouveau document, il faut commencer par compléter les entêtes et pied de page et la page de garde.

**Liste des documents :**

* Compte rendu de réunion,
* Demande de modification,
* Lecture croisée,
* Modèle générique pour tout document,
* Suivi d’activité,
* Plan de test unitaire,
* Template Word pour Visual Paradigm.

Aux formats :

* Word 2003,
* Word 2010,
* OpenOffice

## Organisation des dossiers sur le disque pour chaque projet

**Documentation  
 Besoins** cahierDesCharges.doc donné par le professeur responsable **Analyse** analyse.doc dossier d’analyse maquette.doc document de description de la maquette validée

planIncrement.doc plan des incréments

planValidation.doc plan de validation

…

**Conception**

conception.doc dossier de conception

construction.doc document de description de mise en œuvre de   
 technologies, de protocole, d'algorithmes importants.  
 configuration.doc document de configuration  
 …

**LecturesCroisées**

lectCroisee1.doc

lectCroisee2.doc

…

**Modifications**

dmdModif1.doc

dmdModif2.doc

suiviModif1.doc

suiviModif2.doc

…

**Tests** répertoire contenant les plans et les rapports de test

**Unitaires**

tuP1\_1.doc fiche du test unitaire 1 du package P1

tuP1\_2.doc fiche du test unitaire 2 du package P1

tuP2\_1.doc fiche du test unitaire 1 du package P2

tuP2\_2.doc fiche du test unitaire 2 du package P2  
 …

**Integration**

ti1.doc tests d’intégration des packages

ti2.doc

…

**Validation**

tv1.doc tests de validation de l’incrément

tv2.doc

…

**Manuels**

manuelUtilisateur.doc manuel utilisateur (sortie d'analyse et d'incréments)

manuelInstallation.doc manuel d’installation (sortie d'analyse et d'incréments)

**Gestion**

**Etudiant1**

suiviActiviteE1.doc description des activités par semaine de l'étudiant 1.

indexLectureE1.doc participation aux divers documents communs.

indicationE1.doc principales difficultés rencontrées

**Etudiant2**

…

**Modele**

analyse.vpp modèle d’analyse pour tout le projet

conception.vpp modèle de conception pour tout le projet

**Codage** répertoire contenant les sources validés

**Maquette** sources de la maquette, obtenue en fin d’analyse  
sources

**Version**

Sources sources non spécifiques à un package

**Aide** package de l’aide en ligne

sources

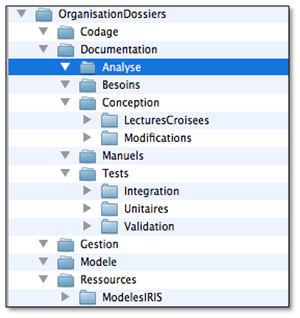
**Rep1**  (nom dépendant du projet)

Sources développement d’un package par un seul étudiant

**Rep2** idem

sources

**...**

**Ressources** documents donnés par le professeur  
planQualité.doc plan qualité de la section

ModelesIRIS modèles de documents

etc.

# **Documents à produire au jury final**

## Dossier technique de projet

À l'issue du projet, l'équipe d'étudiants remet au centre d'examen un dossier technique unique représentatif de l’ensemble du projet.

Ce dossier comprend une partie commune à tous les membres de l'équipe et la partie personnelle traitée par chacun d'entre eux.

Le dossier technique du projet réalisé par un groupe d’étudiants est constitué comme suit :

* **Partie commune : (de 20 à 30 pages)**
  + introduction, situation du projet dans son contexte industriel
  + dossier de spécifications
  + dossier d’étude préliminaire et plan de tests des performances au regard du cahier des charges. Certains éléments peuvent être présents dans les parties personnelles.
  + éléments nécessaires à la recette du prototype final
  + résultats des essais du prototype final
  + conclusion : test intégration, procédure et résultats de la recette.
* **Partie personnelle : (de 20 à 30 pages**)
  + situation de la partie personnelle dans l'ensemble du projet
  + dossier d’étude et de réalisation détaillée, essais unitaires.

Dans sa partie personnelle, il est souhaitable qu’une partie de ce qu’il rédige puisse montrer sa participation à une réflexion commune. L’autre partie contiendra les éléments qui permettront d'évaluer son action individuelle.

Des documents annexes peuvent être joints **sous forme électronique** (annexes techniques, programmes complets, manuel d'utilisation, notice de maintenance, sources complets, … ).

Les documents des projets signés en commission de validation seront joints au dossier de l’étudiant.

# Activités complémentaires

## Réunions d’avancement

**On devra organiser une réunion de coordination par semaine**

Avec le professeur responsable du projet afin de faire le point sur l’avancement, les questions non résolues et afin de mettre en évidence les risques non encore évalués. C’est l’occasion de planifier la semaine, personne par personne et de prendre des décisions indispensables à l’avancement du projet.

Lors de cette réunion, chaque personne devra exposer le travail réalisé la semaine passée, l’avancement de son travail et signaler les retards par rapport au planning. Un compte rendu de réunion sera écrit par un des intervenants de la réunion, à choisir en début de réunion.

## Suivi d’activité

**Chaque fin de semaine, chaque étudiant devra décrire dans un document, son travail de la semaine dans un document appelé ‘suivi d’activité’.**

## Réduction des risques

A chaque fois qu’un étudiant doit mettre en œuvre une nouvelle technologie (technologie non vue en cours : nouveau langage, nouvelle plateforme, nouveau protocole, etc.), celui ci devra réaliser un logiciel de validation de cette technologie afin de démontrer sa capacité à maîtriser cette nouvelle technologie. Cette validation fera partie du rapport final.

**Remarques**:

* la validation sera faite hors contexte du projet, donc non intégrable au projet.
* on ne validera pas des technologies déjà vues en cours ou en TP (exemples : environnement Netbeans, MySQL, socket C++, etc.).

## Suivi de l'application du plan qualité

**Tout document doit faire l'objet de lectures croisées (et écriture des fiches associées).**

### Contrôle

Une revue de projet contrôlera pour les phases suivantes :

1. L'analyse :

* La bonne prise en compte des besoins exprimés dans le cahier des charges.
* Que les dossiers de spécification et de validation sont complets et respecte le plan qualité.
* Le gel des produits de cette phase et la mise en configuration associée.

1. La conception d'un incrément :

* Que les éléments spécifiés ont été pris en compte.
* Que la décomposition en classe et package est cohérente et compatible avec les moyens.
* Que l'architecture du projet est cohérente.

1. La réalisation de cet incrément :

* Que les différents packages ont été implantés conformément à la conception.
* Que les outils sont clairement identifiés.
* Que les dossiers de tests et de validation sont complets.
* Que les autres dossiers de conception sont complets et conforme au plan qualité.
* Le gel des produits de cette phase et la mise en configuration associée.

### Contrôle qualité

Les contrôles portent sur :

* Les produits réalisés à l'occasion d'un projet,
* Le processus de développement.

Le contrôle d'un produit fait l'objet d'un rapport dans le "fiche contrôle produit".

Il consiste à vérifier :

* La conformité du dossier en fin de phases par rapport aux documents techniques de début de phase.
* Le respect des normes et standards de présentation.

Le contrôle du processus de production consiste à contrôler l'existence des documents :

* Fiches de lecture croisée,
* Fiches de demande de modification.

## Gestion de version et de configuration

La gestion de la configuration n’est pas une obligation mais une recommandation, à discuter avec votre chef de projet et les exigences du client.

L'arborescence du projet doit respecter le chapitre précédent.

Remarques :

* N’y sont rangé que les fichiers validés par des réunions officielles.
* Les documents de fin de phases qui ne doivent plus évolués sont verrouillés en écriture.
* Toute modification d'un document est mémorisée, identifiée et commentée.
* Chaque sortie de phase donne lieu à un étiquetage définitif des documents du projet.
* Plusieurs personnes peuvent obtenir un document pour consultation, une seule peut l'extraire pour modification. Une modification est annulable.
* L’exécutable n’est pas mis en gestion de version, il sera recompilé en cas de besoin.
* La configuration ne sera pas sauvée.
* Chaque action devra être commentée.
* On peut donc obtenir toutes les versions précédentes.

Pas de gestion de configuration de l'environnement, si ce n'est l'écriture d'un document de configuration précisant les configurations minimales matérielles et logicielles nécessaires au développement et au fonctionnement du logiciel :

* Environnement de fonctionnement (avant et après installation) :
  + Librairies et versions (DLL, OCX, etc.),
  + Répertoires,
  + Paramétrage (écran, réseau, etc.),
  + Périphériques,
  + Câblage,
  + Incompatibilité logicielle,
  + Etc.
* Environnement de développement :
  + Matériel (configuration et environnement),
  + Logiciel et version (SE avec sa configuration, compilateur, outils, etc.).

## Gestion des modifications

Seules les modifications appelées **non-conformités**, et concernant les phases précédentes à celle concernant cette procédure sont en prendre en compte ici.

Les erreurs concernant la phase courante sont traitées par une procédure de relecture (lecture croisée).

La procédure se déroule en trois temps :

1. établissement d'une fiche de demande de modification,
2. prise en compte de cette demande (conséquence, planning, budget, information, procédure, tests) et accord du chef de projet et création d'une fiche de suivi de modification,
3. réalisation de la modification.

# Méthodes, outils et règles

## Cahier des charges

Il est écrit et maintenu par le client.

Le premier travail à réaliser et la relecture de ce document afin d'en analyser : les exigences, la complétude, la cohérence, la précision et la clarté. En cas de problème demander des compléments d'information au client et si nécessaire des modifications du cahier des charges (impossibilité de réalisation, incompréhension, etc.). Il est rédigé dans le langage technologique du client (et sans référence informatique).

C'est lui qui sert de base à l'appel d'offres, à la négociation du coût, des délais de livraisons et de la recette.

## Analyse

Le début de l'analyse va consister à **formaliser le cahier des charges** afin de mieux cerner sa **faisabilité**, à identifier les incohérences, les liens dans sa globalité et à organiser le travail de développement. Il concerne la définition des acteurs et leur description, la définition des cas d'utilisation et leur description textuelle et sous la forme d'un ou plusieurs diagrammes de séquences mettant en scène les acteurs et une classe "**Systeme**" matérialisant le système à développer.

Elle doit être réalisée avec précision car elle va servir de base au chef de projet pour évaluer le coût de développement, les personnes et le matériel à mettre en œuvre et bien d'autres choses encore.

Il semble intéressant que le client soit capable de lire cette formalisation.

Cette spécification sera exprimée avec le langage UML et fera partie du dossier d'analyse du projet.

**Une revue valide cette analyse.**

La spécification précise ensuite : l'architecture du système, le(s) diagrammes de classes du domaine, les diagrammes de séquence et de collaboration décrivant les différents cas d'utilisation identifié par la faisabilité, des diagrammes états- transitions de certaines classes, un diagramme de déploiement). Tous ces éléments étant largement commentés.

L'analyse doit produire : un modèle UML d'analyse, un dossier de validation, un plan des incréments, une maquette (exécutable ou non, jetable ou non), les manuels d'installation et d'utilisation créés.

**L’objectif de la spécification UML documentée sera si possible la génération automatique du dossier d’analyse.**

## ****Conception d'un incrément****

Avant tout, nous allons créer un 'package' **"Conception"** dans lequel nous allons transférer tous les diagrammes UML qui concerne l'incrément courant et les classes associées. Si un diagramme d'analyse concerne plusieurs incréments, il est décomposé en deux diagrammes afin qu'un des deux diagrammes ne décrive que des scénarios de l'incrément courant. Toute la conception ne concernera ensuite que des éléments de ce package.

La conception d'un incrément va se dérouler en deux sous phases :

1. conception logique,
2. conception physique.

Sous phases, qui ne se retrouveront pas dans le dossier final de conception.

### La conception logique

Elle va consister à ré-architecturer la modélisation suivant un point de vue logique de ce projet et de cet incrément (nouveaux packages, par exemple), à ajouter des patterns de conception logique et d'autres informations indépendantes du langage de développement. Pour des sous systèmes répartis non objet, les protocoles de communication non encore définis (cahier des charges) donneront lieu à la définition très précise dans le dossier de construction.

### La conception physique

Elle va consister à définit l'organisation physique du matériel, à définir les composants logiciels, leur langage de développement, à remplacer la messagerie des diagrammes de séquence et de collaboration par des appels de méthodes (entre des objets répartis ou mono tâche), des envois de signaux et la mise en place de classes **proxy** pour les appels répartis non objet (protocole de communication par exemple), la mise en place de patterns de conception physique liées le plus souvent à l'environnement de développement (Vue/Document par exemple).

Elle va consister, si la maquette est non jetable, à réaliser une rétro-conception de cette maquette dans le modèle, puis à remplacer les acteurs par les IHMs obtenus dans les diagrammes de séquences et de collaboration.

On réorganise ensuite le modèle en package avec un point de vue de poste développeur (un ou plusieurs packages par personne).

La construction va consister ensuite pour chaque développeur à réaliser le package qui lui a été affecté : complément de description (de chaque classe et de chaque membre), de modélisation, de documentation, d'intégration à l'environnement de développement, ajout de classes ou de composants de métiers.

Classe par classe, écriture d'un document de test unitaire, écriture des algorithmes des méthodes, codage, debug, test unitaire.

Intégration au package et test d'intégration.

Intégration des packages entre eux, en respect du plan d'intégration et test associés.

Validation du produit obtenu et production des documents associés à cet incrément.

Ecriture, si nécessaire, du logiciel d'installation.

### Algorithmique et Codage

Ecrire dans l'esprit d'être relu par d'autres ou par vous même dans plusieurs mois ou années !

Pour l'algorithmique, voir en annexe le dossier des règles d'écriture de l'algorithmiques.

Pour le codage, voir en annexe le dossier des règles de codage en C++ et Java.

**Ne pas oublier les commentaires en début de chaque module de classe et à chaque début de méthode permettant si possible la génération automatique de la documentation d’utilisation de cette classe (comme avec la JavaDoc par exemple).**

### Test et validation

**Pour le dossier final remis au jury, vous devrez produire au moins un plan de test unitaire et réaliser ce test. Ce test sera présenté à l’oral dans la deuxième partie intitulée ‘démo’.**

Je vous conseille d’écrire plusieurs plans de test, même si vous n’en codez qu’un seul.

Je vous conseille aussi d’écrire un plan de d‘intégration que vous réaliserez ci possible.

Un plan de validation devra être écrit et la validation associée réalisée.

## Outils

Les outils mis en œuvre sont (sauf description contraire du manuel qualité spécifique au projet) :

* Visual Paradigm,
* Netbeans, .NET ou des plateformes spécifiques définies par le client,
* Windows 7, Linux,
* Microsoft Office ou Open Office pour la documentation,
* VSS ou autre pour la gestion de version,
* Explorer ou Firefox pour la navigation Web.
* MySQL ou SQLServer en fonction de l’environnement.

## Environnement de travail

Tous les travaux du projet sont sauvés sur le disque prévu à cet effet sur le serveur de la section.

Les machines de développement sont banalisées. A chaque début de phase, le développeur obtient à son environnement de développement qu'il range sous "C:\Temp".

Cette action crée une arborescence conforme au plan qualité et autonome par rapport à la machine de développement.

A la fin de chaque jour de travail cette arborescence est sauvée à partir de la racine "NomDuProjet" sur un répertoire de travail spécifique à ce projet et prévu à cet effet sur le NAS.

Le lendemain, le développeur peut transférer ce répertoire à nouveau sur le répertoire "C:\Temp" afin de continuer son développement. Après la validation d'une phase, les documents modifiés de cette arborescence sont mise à jour dans l’outil de gestion de version (si utilisation).

2. Sauvegarde

Répertoire de

Sauvegarde

3. Chargement

"C:\temp\NomDuProjet"

Répertoire de travail

4. Fin de phase

Outil de gestion de version

1. Début de phase

# Annexes

Dossier sur les règles de description des IHM.

Dossier sur les règles d'écriture en langage C++ et Java.

Dossier sur les règles d'écriture des algorithmes.

Dossier sur les tests unitaires, d’intégration et de validation.