Plan Qualité du projet SIRIUS

Projet	SIRIUS
Document	Plan qualité du projet
Version	2.0
Date	24 septembre 2003
Référence	SP1

Table des matières

Chapitre 1

Introduction

1.1 Objectif global du logiciel

Le projet SIRIUS est un projet open source de développement d'une application de gestion de la scolarité au niveau des établissements de l'enseignement supérieure au maroc. Initié par l'université Hassan II Mohammedia en partenariat avec l'association OSIM[1], ce projet a pour ambition de créer une synergie entre les compétences et les ressources au sein des universités et écoles marocaines avec les développeurs libres de part le monde.

1.2 Historique des versions du document

Version	Date	Auteur(s)	Apport
1.0	16/6/2003	Tarik Fdil	Création du document
2.0	24/9/2003	Tarik Fdil	Mise à jour règles de codage

1.3 Statut du document

Vérification :

Statut	non vérifié
Vérificateur(s)	
Date vérification	

Validation :

Statut	non validé
Validateur(s)	
Date validation	

[1] Association Open Source In Morocco http ://www.linux-maroc.org

Chapitre 2

Eléments de qualité

2.1 Cycle de vie du produit

Nous distinguons sept phases dans la vie d'un produit logiciel :

1. Planification : est la phase de planification du projet. Elle planifie tout le cycle de vie du produit.
2. Spécification : le but de cette phase est de transformer les besoins du client en spécifications fonctionnelles.
3. Conception : l'objectif de cette phase est de transformer les spécifications fonctionnelles en solution technique et en spécifications détaillées.
4. Développement : est la phase de production de programmes proprement dite. Elle produit également de la documentation technique.
5. Qualification : est la phase de vérification que le produit réalisé correspond bien aux spécifications et aux exigences qualité.
6. Installation : est la phase d'installation ou de déploiement du logiciel.
7. Support : correspond à la période de garantie et de maintenance du logiciel.

2.2 Documents produits

2.3 Traçabilité bidirectionnelle

Toute spécification dans le rapport des exigences doit se référer explicitement aux documents de base. Les changements des exigences doivent également se référer explicitement aux demandes faites par l'utilisateur avec de préférence des documents à l'appui.

Dans le rapport de spécification il est demandé de préciser les éléments suivants :

- Liste de toutes les fonctionnalités du système
- Description littéraire de chaque fonctionnalité

- Description des règles de gestion
- Liste des états de sortie ou des consultations

Chaque fonctionnalité décrite doit être numéroté avec un numéro de référence de style SPn où n est le numéro de la fonctionnalité. Chaque règle de gestion doit porter un numéro de type RGn où n est le numéro de la règle. Chaque état ou consultation décrite devra porter un numéro de type ETn où n est le numéro d'état. Pour garder une traçabilité des modifications dans les spécifications, chaque nouvelle version du rapport devra garder la même numérotation des SPn, RGn ou ETn. Si de nouvelles fonctionnalités sont rajoutés on leur donne de nouveaux numéros. Si au contraire des fonctionnalités ont été supprimées, on n'utilisera pas leur numéro et on indiquera clairement que telle fonctionnalité a disparu dans la version en cours du document.

Eléments à prendre en compte pour l'élaboration du document de conception :

- Le document de conception devra préciser clairement à quelle version du rapport de spécification il se réfère.
- Chaque écran, état, procédure sera identifié avec une référence unique. On utilisera la numérotation ECn pour les écrans, ETn, pour les états, PRn et FNn pour les procédures ou les fonctions. n désignant le numéro séquentiel de l'élément en question.
- On dressera un tableau de correspondance entre les éléments ECn, ETn, PRn et FNn de conception et leur origine SPn, RGn et ETn du rapport de spécification.

2.4 Suivi des bugs

Il est important durant la période qui suit immédiatement l'installation chez le client du logiciel de suivre les rapports de bug. Pour cela nous préconisons l'utilisation de l'outil de suivi des réclamations de sourceforge.net Un outil de suivi des bug permettant de suivre les informations suivantes :

- Identifiant unique du bug.
- Catgéorie de bug : ce champ permettra d'opérer un classement selon les besoins de l'application. Cela peut être par exemple l'origine de déclaration du bug : client, internet, mail, etc.
- Sévérité du bug : mineure, majeure, fonctionnalité, etc.
- Status du bug : corrigé, assigné, en instance, etc.
- Date de dernière mise à jour.
- Bref descriptif du bug.

Chaque projet a son propre système de suivi des bugs. Ce système peut être mis à la disposition du client via internat/intranet pour entrer lui même les bugs où bien le client remplit un formulaire de déclaration de bug et le responsable de la maintenance auprès du client saisit le bug dans le système de suivi.

Aucun bug déclaré ne doit rester plus de 48h sans être assigné à un membre de l'équipe du projet qui est chargé de l'instruire.

2.5 Normes de développement

2.5.1 Modularité

On essaiera autant que possible d'écrire un fichier par fonction, procédure ou classe. Un fichier ne devrait pas dépasser deux à trois pages de listings. On ne regroupera qu'exceptionnellement plusieurs fonctions dans un seul fichier. Dans ce cas, chaque fonction aura son propre entête comme cela est décrit dans le paragraphe qui suit.

Pour les classes C++ il est préférable d'observer les règles suivantes :

- La déclaration de la classe se trouve dans un fichier nomClasse.h où NomClasse est le nom de la classe définie.
- Le constructeur et le destructeurs doivent se trouver dans le fichier nomClasse.cpp
- Chaque fonction membre doit se trouver dans un fichier à part qui se nomme nomClasseMaFonctionMembre.cpp

2.5.2 L'entête du programme

Chaque fichier devra contenir une entête sous forme de commentaires. Cette entête devra contenir au moins les informations suivantes :

- le nom du fichier
- le nom de la fonction ou procédure ou classe
- une brêve description (une ligne) du but de la fonction
- une description plus détaillée de la fonction (5 à 6 lignes)
- l'auteur du fichier
- la date de création du fichier
- l'historique des mise à jour avec date, auteur et description du changement
- la référence au document de conception et au ECn, ETn, PRn ou FNn implémentée.
- les entrées/sorties de la fonction et éventuellement la valeur retournée.
- les effets de bord de la fonction (variable globale, fichier, base de données, e/s)

2.5.3 Commentaires et règles de nommage et de présentation

Les commentaires doivent être suffisant et utiles : ni trop ni trop peu. On suivra la notation de l'outil doxygen. Les identificateurs de fonctions, classes et variables devront suivre les règles Java. La présentation des programmes suivra la présentation des programmes GNU/Linux.

Livrables

Le processus de développement devra livrer plusieurs produits :

- Un guide utilisateur. C'est une présentation étape par étape de la réalisation des grandes fonctionnalités en présentant des copies d'écran avec des champs

renseignés et des exemples de session. Le Guide Utilisateur doit se présenter sous forme papier et html. Le fichier HTML est accessible via l'application.
- Un guide de référence. C'est une présentation détaillée de toutes les fonctionnalités, présentatées de préférence par ordre alphabétique. Le Guide de référence doit se présenter sous forme papier et html. Le fichier HTML est accessible via l'application.
- Un Guide d'installation. C'est une présentation détaillée de la procédure d'installation de l'application. Le Guide d'installation doit se présenter sous forme papier et html. Le fichier HTML est accessible via l'application.
- De la documentation technique produite par doxygen.
- Les deux fichiers README et INSTALL qui doivent être des fichiers Ascii texte. README contient la liste des fichiers de l'application installée et leur contenu. INSTALL est la forme textuelle du Guide d'Installation.
- L'ensemble des fichiers sources avec la procédure pour construire l'application (makefile, script, etc.)
- Tout fichier utilisé par la procédure d'installation

2.6 Gestion des configurations

Nous distinguons deux types de produits élaborés durant les diverses phases du cycle de vie d'un produit logiciel : les documents et les applications ou programmes. Un document est un texte, formaté ou non, sur une ou plusieurs pages, écrit sous forme électronique et qui est généralement imprimé sur papier. Les applications ou programmes sont des logiciels construits à partir de documents que sont les programmes sources. Pour ces deux types de produits, la gestion des configuration consiste à gérer la codification, les versions, le stockage et l'intégrité de ces produits.

2.6.1 Versions de produits

Tout produit final ou intermédiaire, qu'il soit un document ou un logiciel aura un numéro de version. Le numéro de version est de la forme x.y où x est le numéro de version majeure et y le numéro de version mineure. Quand des modifications importantes sont apporté à un produit, on change son numéro de version majeure. Exemple de modification majeurs : changement important des spécifications ou ajout de nouvelles spécifications pour un logiciel ou un rapport de spécification. Quand il s'agit de correction moins importante on change uniquement le numéro de version mineure. Exemple de changement mineur : correction de bugs dans un logiciel ou d'erreurs mineurs dans un document écrit.

Bien qu'un logiciel soit lié de manière intrinsèque à un document de conception d'une version bien déterminée, les numéros de version du document de conception et du logiciel évoluent différemment. En effet le document de conception peut ne pas avoir changé alors le logiciel a changé de version pour corriger un bug ou pour ajouter une nouvelle fonctionnalité en cas de livraion par étape,

etc. Par contre il y a une adéquation parfaite entre un logiciel construit (compilé et linké) et ses programmes sources. On doit pouvoir régénérer sans problème un logiciel à partir de ses sources et inversement on doit retrouver immédiatement les sources à partir de la version du programme construit. Pour cela on utilisera le logiciel de gestion de versions concurrentes cvs pour tous les programmes sources et pour la documentation. Celle-ci pour bénéficier des possibilités de cvs il est souhaitable qu'elle soit produite avec des outils générant des fichiers texte comme Latex, Lyx ou les outils à base de XML ou SGML.

2.6.2 Codification des documents

Chaque document est caractérisé par les éléments suivants :
- Référence du document.
- Titre du document.
- Version du document.
- Date de version du document.
- Historique des révisions du document.
- Statut de révision du document.
- Statut de validation de document.
- Auteurs du document.

Pour la forme électronique du document, il y a d'autres éléments caractéristiques :
- Nom du fichier.
- Extension du nom de fichier (qui détermine le logiciel avec lequel il est créé).
- Résumé MD5 du document.

Toutes les informations ci-dessus doivent apparaître clairement sur le document lui même. La référence du document est déterminée de la manière suivante : pppCCnnn où ppp est le code du projet dans le cadre duquel le document est produit, CC est le code du type de document et nnn un numéro de séquence unique pour un projet donné et un type de document donné. La liste des codes documents est la suivante :

Type de documents	Code associé	Type de documents
Plan de développement de projet	PL	fichiers sources
Document de plannification	DL	tarball's du projet
Rapport de spécification	SP	Résultat des tests qualité
Document de conception	CO	Rapport de recette provisoire
Plan Qualité du projet	PQ	client Rapport de recette définitive client
Rapports d'état d'avancement	RA	Rapport de support logiciel
Guide d'utilisateur	GU	compte rendu de réunion
Guide de référence	GR	Devis ou proposition technique ou commerciale
Guide d'installation	GI	Correspondance, lettre, mail, fax
README	RM	Procédures et standards
INSTALL	IN	Documenttation générale
Makefile	MF	Rapport de conclusion de projet

Le nom du fichier électronique contenant le source du document doit être sous la forme pppCCnnnVx-y_nom.ext où 'pppCCnnn' est la référence du document

explicité ci-dessus et 'nom' est un nom quelconque donné au fichier et 'ext' est l'extension du nom de fichier, correspondant en générale à la nature du fichier : .c pour fichier source C, .lyx pour fichier source lyx, etc. Quant à 'Vx-y' c'est le numéro de version du document explicité au paragraphe 2.6.1.

Dans le cas du projet Sirius, le numéro du projet 'ppp' peut être omis car il y a un seul projet en cours.

2.6.3 Codification des applications

Une application est forcément stockée sous-forme électronique. Elle est caractérisée par :

- Référence de l'application.
- Version de l'application.
- Date de version de l'application.
- Historique des versions de l'application.
- Nom du fichier.
- Extension du nom de fichier (qui détermine le logiciel avec lequel il est créé).
- Résumé MD5 du document.

Une référence de l'application est sous forme nnnVx.y où 'nnn' est le numéro du projet et 'Vx.y' est la version de l'application. Le nom du fichier correspondant à l'application est de la forme nomNNNvx-y.ext où 'nom' est le nom de l'application, 'NNN' est le numéro du projet et 'Vx.y' la version de l'application.

2.6.4 Stockage des produits

Chaque projet a son espace de stockage sur un serveur de fichier sous forme d'un répertoire nommé ARnnn où nnn est le numéro de projet et 'AR' signifie 'armoire'. Ce répertoire contient deux sous-répertoires 'Documents' et 'Applications', le premier contient tous les documents du projet et le second, lui même subdivisé en deux sous répertoires 'src' et 'runtime' contenant respectivement les diverses versions des fichiers sources et des applications construites.

En principe le répertoire consacré au projet ne contient que les documents validés et non pas les documents en cours d'élaboration, de vérification ou de validation. Idem pour les applications on ne stockera que les versions d'applications stables qui ont été publiées. Toute autre version en cours de développement ne sera pas stockée dans ce répertoire.

Concernant les fichiers sources des versions déjà publiées comme celles en cours de développement, ils seront tous gérés à l'aide d'un CVS. En principe une version d'une application extraite du CVS devra être rigoureusement identique à celle stockée dans le répertoire du projet.

Il n'est pas exclu que les versions papiers des documents de projets soient stockées dans une armoire physique consacrée au projet. Par ailleur tous les documents du projet, sauf les fichiers textes, doivent être publiés au format PDF pour être consulté sur internet/intranet.

2.6.5 Intégrité des produits

Chaque produit intermédiaire ou final, document ou application est signé à l'aide d'un résumé MD5. Ce résumé créé au moment de la validation du document permet de vérifier que le produit n'a pas été altéré. On créera une table dans une base de données où il y a la référence du produit et son résumé MD5. Quiconque pourra consulter cette base pour vérifier l'intégrité du poduit qu'il a entre les mains. La mise à jour de cette table est évidemment restreinte et tient compte des règles de sécurité permettant d'avoir confiance dans son contenu.

2.7 Outils de développement et plateformes utilisées

- SGBD : PostgreSQL
- Librairie et outil de développement : QT, KDevelop et QT-Designer
- Langages de développement : C++, SQL, XML, PHP, Perl
- Architecture : client/serveur
  - Plateforme serveur : Linux
  - Plateformes clients classique : Linux, Windows
  - Palteformes clients internet : toute machine disposant d'un navigateur internet : pc, mac, etc.

2.8 Revues formelles et Test

2.8.1 Revues de documents et de code

Chaque document écrit ou programme écrit doit être revu par une autre personne que celle qui l'a écrit. Le but est de le critiquer et en vérifier la conformité aux exigences qualité convenus. Cette vérification peut se faire de manière formelle dans une réunion ou informelle selon l'importance du document ou programme.

2.8.2 Tests fonctionnels

Nous distinguons trois types de test :

- Les tests unitaires qui servent à tester individuellement une fonction ou un module.
- Les tests d'intégration qui permettent de tester l'application dans son intégralité.
- Liste des tests de non regression qui sont utile pour vérifier que des fonctionnalités déjà existante ne sont pas altérées par une nouvelle fonctionnalité installée.