**Cahier des charges**

1. Contexte et définition de problème :

On a de plus en plus des modifications techniques à gérer dans les produits de la société Linedata. Aujourd'hui tous les ingénieurs sont conçus à télécharger le répertoire de travail qui est dans le système de version de contrôle ‘GIT’ ou ‘CVS’ pour faire la comparaison de ses travaux avec celles qui existent dans ces derniers et cela a bien été une grande perte de temps et ce n’est pas très aisé de faire toute cette opération pour chaque personne toute la période de travaille … De plus , lorsque on doit migrer, le travail fait, d’une version CVS a une autre GIT cela pose plein de problèmes de continuation des suivis associer aux fichiers comparés pour chaque version alors cela devient vite le fouillis : on ne retrouve pas le suivi des modifications lors de migration. Et comme pour les années à venir tous les versions de produit seront contrôlés par le GIT puisque il est plus efficace que le CVS, ce point est à travailler d’urgence.

1. Objectif :

Nous voulons offrir un meilleur service qui peut gérer mieux tout la difficulté déjà décrite à l'aide d'un véritable outil de comparaison des versions en assurant la continuation des suivi lors d’une éventuelle migration avec une possibilité de déploiement du projet et sa compilation à distance sans le recours à Eclipse. Aujourd'hui nous visons l’automatisation de toutes ses opérations après la mise en place du nouvel outil.

1. Description Fonctionnelle
2. **Besoins Fonctionnels :**
3. télécharger les répertoires à comparer.

Description :

Pour faciliter le traitement, il faudra éventuellement avoir l’accès aux fichiers donc on aura besoin à télécharger tous les repos qui contiennent les versions demandés.

1. comparer les fichiers avec les fichiers de l’autre répertoire.

Description :

Faire un algorithme qui aide à parcourir tout l’arborescence d’un répertoire et comparer les fichiers afin de trouver une correspondance entre les fichiers des deux répertoires données.

1. afficher les différences trouvées après la comparaison.

Description :

Après quand trouve les fichiers qui correspondent on va appliquer une comparaison de code et afficher les différences en faisant une coloration des parties différentes

1. intégrer les 2 fichiers différents.

Description :

Intégrer une version de code dans le fichier destinataire pour adapter les deux versions déjà comparées.

1. extraire les suivis de chaque fichier pour chaque répertoire.

Description :

Extraction des suivis de chaque fichier qui contient tous les activités déjà faites sur cet fichier.

1. attacher chaque suivi à son fichier qui le correspond.

Description :

Associer chaque suivi de l’ancienne version de fichier à la nouvelle version de fichier pour assurer la continuité des activités et la faciliter de les consulter après la migration.

1. déployer le projet dans le serveur de l’application.

Description :

Appliquer une configuration pour déployer le projet modifié sans le retour à Eclipse pour le faire en gagnant plus de temps et faciliter la compilation après.

1. compiler à distance pour tester le bon fonctionnement de projet modifié.

Description :

Gagner plus de temps pour tester le bon déroulement du projet après le changement qui avait lieu.

1. **Besoins non Fonctionnels :**
2. **Sécurité**

• Besoins de mot de passe – longueur, caractères spéciaux, expiration, politique de réutilisation  
• Déconnexion après temps morts d’inactivité – durées, actions

1. **Performance**• Temps de réponse – le chargement de l’application, ouverture d’écran et des délais de rafraîchissement, etc.  
   • En temps de traitement – fonctions, calculs, importations/exportations de données.
2. **Capacité**• Mémoire(Stockage) – combien de données le système doit-il être capable de stocker?  
   • Besoins de croissance d’année-en-année (croissance organique).
3. **Intégrité**• Le traitement des mauvaises données – import de données, marquer-et-continuer ou arrêt la politique d’importation.
4. **Compatibilité**• La compatibilité avec des applications partagées – À quels autres systèmes doit-il parler ?  
   • La compatibilité avec des applications tierces – Avec quels autres systèmes doit-il cohabiter ?  
   • La compatibilité sur des systèmes d’exploitation différents – sur lesquels doit-il être capable de fonctionner.
5. **Aptitude à la maintenance :**• La conformité aux standards d’architecture – à quels standards a-t-il besoin de se conformer ou en être exempté ?  
   • La conformité aux standards de design – Quels standards de conception doivent être suivis ou des exemptions obtenues.
6. **Ergonomie**• Les standards d’ergonomie – la densité d’éléments sur les écrans, la disposition et le flux, les couleurs, l’Interface Utilisateur, les raccourcis clavier  
   • langages, orthographe, claviers, formats de papier.
7. *Délais*
8. Diagramme de Gant :