

# Réussir un projet informatique : les bonnes pratiques

par Eric Reboisson (<http://ericreboisson.developpez.com>) (Blog)

Date de publication : 13/05/2007

Dernière mise à jour : 13/05/2007

Cet article présente quelques bonnes pratiques et conseils qui permettent de mener à bien un projet informatique. Sans la prétention d'être une table des commandements, vous trouverez des pistes éprouvées et basées sur le bon sens et l'expérience de nombreux développeurs et gestionnaires de projets.

- I - VRAC
- II - Un petit mot avant de commencer
- III - Au commencement, il n'y avait rien
- IV - Définition et gestion des tâches
- V - Gestion du référentiel des sources
- VI - Le développement au quotidien : rapidité, facilité
- VI - Le référentiel documentaire
- VIII - Conditions matérielles et humaines
- IX - Vérifier la qualité du projet
- X - Conventions et règles
- XI - Conception et modélisation
- XII - Communication
- XIII - Le bon sens
- XIV - Le mot de la fin
- XV - Remerciements

## I - VRAC

- Prendre une application fil rouge en exemple. - test de couverture, fonctionnel, non regression, unitaire. Adopter un rythme constant...laissons le sprint aux athlètes. L'article ne traitera pas : - des subtilités de la rentabilité et du jonglage des ressources...

## II - Un petit mot avant de commencer

L'Homme possède cette capacité d'apprendre de ses expériences, de ne pas reproduire les mêmes erreurs (au moins d'essayer) et par là même d'évoluer. Aussi, la bonne réussite d'un projet informatique passe par un ensemble d'invariants, d'étapes nécessaires au cycle de vie du projet informatique.

Dans cet article, j'ai voulu synthétiser mes expériences personnelles vécues sur le sujet et les différents échanges à ce propos sur les forums de Developpez.com.

Et pour finir l'introduction, je signale que le public visé peut aussi bien être le lecteur néophyte (je m'adresse là aux angoissés de la feuille vide démarrant un projet dans des structures possédant peu de normes en la matière) mais aussi aux confirmés (qui pourront par la même occasion partager leurs expériences afin de faire évoluer (là encore) en bien cet article).

Bonne lecture à tous !!!

### III - Au commencement, il n'y avait rien

Une idée, un besoin, une nécessité, autant de causes qui conduiront à planter la petite graine "Projet Informatique".

## IV - Définition et gestion des tâches

Ca y est, le projet du siècle va démarrer pour toi et tes collègues mais avant de foncer tête baissée (dans le mur???) , il faut Avez vous un gestionnaire de bugs ? - Parfois un simple fichier excel suffit - bug tracking pour les grosses applications et/ou commerciales - groupware Que fait on ? ==> Planning, emploi du temps En combien de temps ? Qui fait quoi ? ==> répartition des tâches Où en est on ? Réunions régulières pour connaître l'avancement (retards , difficultés).

ID	Description	Affectation	Statut	Commentaire	Version

## V - Gestion du référentiel des sources

Le référentiel des sources est l'ensemble du code et de tous les éléments qui constituent le coeur du projet. Maintenir correctement ce référentiel permet : l'interaction entre les acteurs du projet. Utilisez vous un gestionnaire de versions ? (CVS, VSS, SVN,...[lien vers la FAQ SCM](#)) Sauvegarder.

## VI - Le développement au quotidien : rapidité, facilité

Pouvez vous simplement générer une version de votre application ?

## VI - Le référentiel documentaire

Les spécifications Les spécifications sont elles suffisamment précises, compréhensibles par tous ? Manuels d'utilisation (technique et fonctionnel) Divers...

## VIII - Conditions matérielles et humaines

Ordinateur assez puissant. Les configurations sont elles suffisantes : eclipse assez gourmand nécessitent suffisamment de mémoire L'ensemble de l'équipe est elle impliquées. Remercier les intervenants.

## IX - Vérifier la qualité du projet

les test unitaires Avez vous une gestion des tests efficace : - Tests unitaires - Tests fonctionnels - Stress tests, tests de charge

## X - Conventions et règles

Définir un ensemble de conventions est absolument nécessaire (mise en page homogène du code entre les développeurs, terminologie unique sur un projet...) - conventions d'écriture (pour améliorer la lisibilité et la maintenance) perso. j'utilise les conventions GNU (il existe un document bien fait) <http://www.gnu.org/prep/standards/>

## XI - Conception et modélisation

Prendre une petite heure pour réfléchir est conseillé au lieu de foncer la tête dans le guidon. Avoir une analyse claire du besoin Il ne faut pas négliger la phase d'analyse du projet il est important d'avoir en début de projet une documentation du but que l'application finale devrait atteindre - conception / modélisation visuelle (UML, ...) la documentation de début de projet doit contenir une vue globale du processus métier à réaliser (courte description du projet, glossaire de termes technique, schéma illustrant globalement a quoi doit servir le logiciel)

## XII - Communication

Premièrement, trop d'information tue l'information. - espace collaboratif et de diffusion d'informations (pas que pour le code source, mais aussi pour la doc, les forums, les news, ...) avoir si possible une liste de contacts des personnes concernées par le projet pouvant être contactées pour des précisions prévoir des réunions "régulières" pour justifier de l'avancement des travaux et prendre en compte les nouvelles demandes En particulier il est ABSOLUMENT INUTILE de faire un dossier de maintenance de 500 pages. Personne ne le lira. Un maximum de 5 pages (et encore) sera le maximum qu'une personne chargée de trouver un bug passera avant d'aller fouiller dans le code.... Même chose pour une doc d'installation (ai dessus d'une page c'est déjà trop, surtout si on s'adresse à des informaticiens).. Mettre des commentaires dans le code Coder en pensant que 1) ce sera repris par d'autres, 2) personne n'est irremplaçable, mais aussi 3) quand même chacun est un peu irremplaçable..

## XIII - Le bon sens

avoir et garder du bon sens # ne pas choisir une techno et s'y maintenir alors que d'autres permettraient de faire la même chose plus simplement # ne pas choisir une techno en fonction du "buzz" à la mode # ne pas utiliser un marteau pour écraser une mouche (exemple prendre OracleTM pour gérer une BD à 4 clés) # ne pas utiliser tout un tas d'algos sophistiqués alors qu'une simple réflexion peut amener une solution simple (et donc compréhensible et maintenable facilement) # ne pas continuer à dire "c'est dans la spec on fait comme ça" si on s'aperçoit que la demande/la conjoncture a évolué # ne pas dire dans une réunion d'avancement "on discute pas de ça mais de la proposition machin truc" alors que le "ça" résout peut-être le problème # ne pas utiliser une lourde gestion de projet pour une équipe de 5 personnes # ne pas oublier que ce qu'on veut faire, c'est ce qui est demandé (eh oui, ça a l'air évident, mais dans la plupart des gros projets, au fur et à mesure du temps qui passe, on oublie ça..) pour les utilisateurs auxquels s'est destiné... # si un utilisateur ne comprend pas un mot, une fonction, etc... c'est qu'il faut réviser sa copie. Ce n'est pas à lui de s'adapter, mais à l'info. de l'aider..

## XIV - Le mot de la fin

@TODO conclusion...

## XV - Remerciements

Un grand merci à AAA, BBB, CCC et DDD pour la relecture de cet article et leurs bons conseils.

