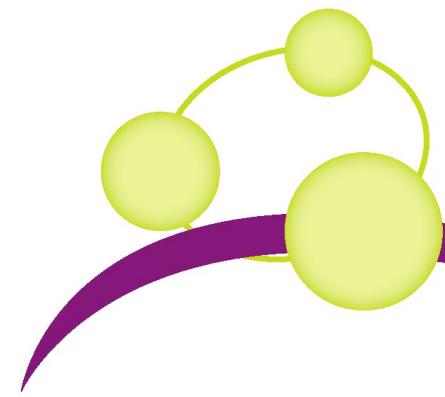


Expérience de la qualité dans nos développements.



21e [.net]
Logiciels Libres pour l'Entreprise



Qui sommes nous ?

21e [.net]
Logiciels Libres pour l'Entreprise

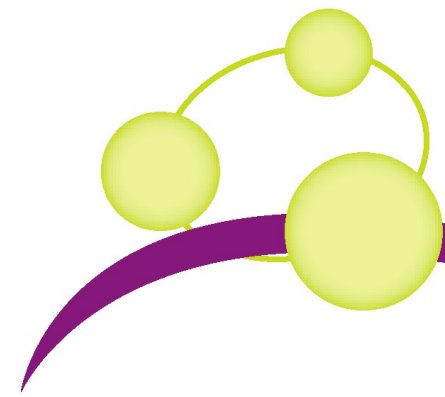
- Sébastien HEITZMANN - Gérant.
- Développement de logiciels sur mesure.
- 10 ans d'expérience.
- 5 personnes.

Typologie des projets.

- Création d'entreprise « internet »
- Budgets moyens : 20/30 jours de développements
- Cahier des charges le plus souvent inexistant
- Idées le plus souvent mouvantes.
- Moyens limités

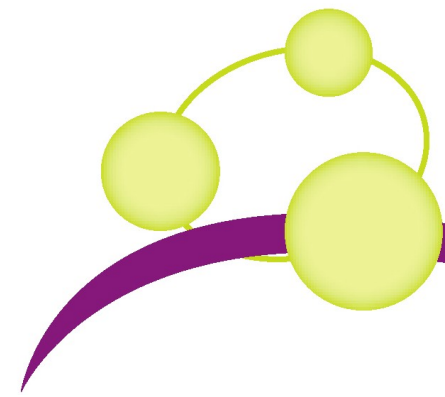
Quelles conséquences d'une mauvaise qualité.

- Projet au forfait : le client ne paye pas la mauvaise qualité
- Délai « explosé »
- Maintenabilité quasi nulle
- Rentabilité négative.
- Client insatisfait.
- La fin est proche ...



Méthodes qualités « théoriques »

- Spécifications précises
- Tests unitaires
- Tests fonctionnels par le client
- Intégrer le client à l'équipe
- Etc...
- Tout n'est pas toujours possible !!!



Mais avant cela ?

- Des méthodes de bases applicables par tous et simplement.
- Du bon sens.
- Un retour d'expérience de la vraie vie.
- A faire en plus du reste si possible.

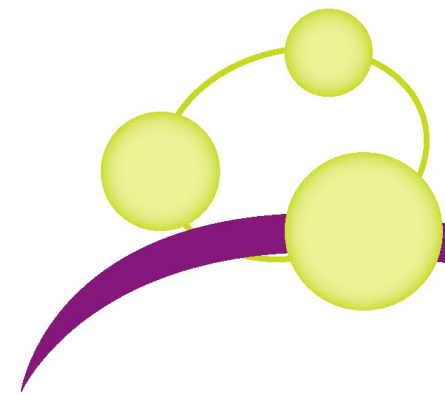
- Malheureusement pas appliquées par tous.

Utiliser un gestionnaire de code.

- Subversion est l'ami du développeur.
 - Une seule version « officielle »
 - Facilite le « boot » d'un développeur
 - Sécurisation du code
 - Collaboration facilitée avec des acteurs extérieurs
 - Facilite le déploiement / multi environnement
 - Traçabilité

Utiliser un gestionnaire de code.

- Les effets induits d'un gestionnaire de code.
 - Facilite le refactoring
 - Le droit de se tromper
 - Pas de code mis en commentaire
 - Réduction des « hack »
 - Revue de code simplifiée
- La base de tout un tas d'autres outils.
 - Trac, cruise control, etc...



Coder proprement.

- Le code est fait pour être lu.
- Un développeur passe 9/10ème de son temps à lire du code.
 - Modifier un code
 - Corriger un bug
 - Ajouter une fonctionnalité
 - Se documenter sur son fonctionnement
 - Accessoirement ajouter un ; manquant ou une ligne de code.

Coder proprement.

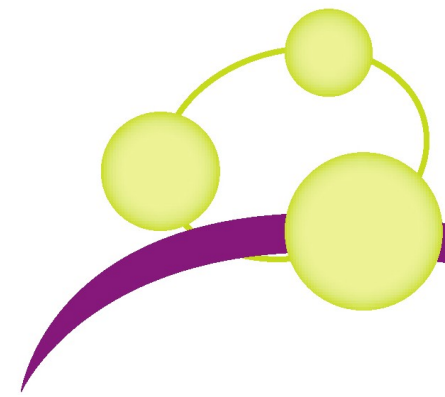
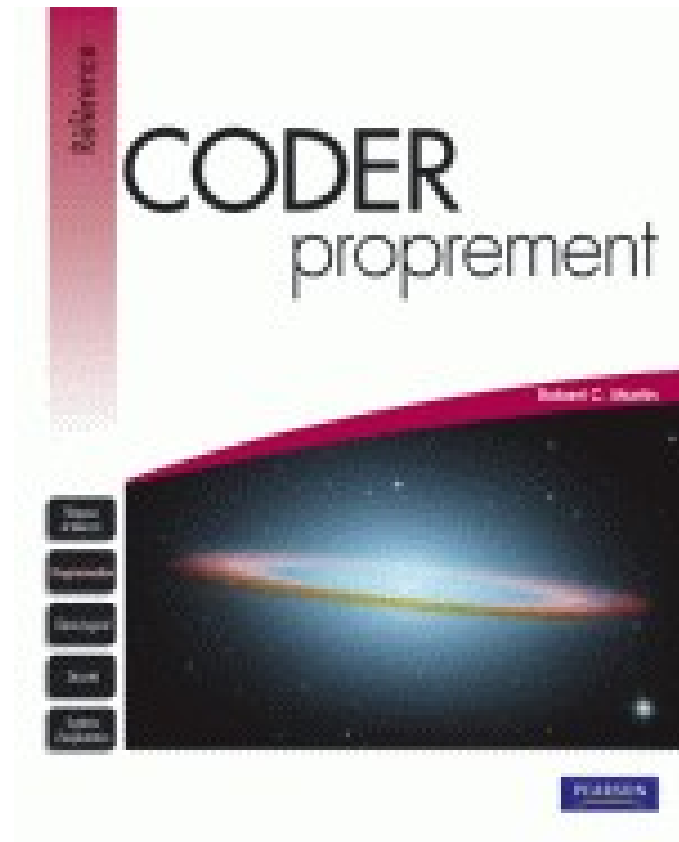
- Penser à celui qui va lire le code.
 - Nommage des variables / classes / méthodes
 - Réduire les commentaires. Si si !
 - Adopter une convention de codage pour l'équipe
 - Préférez la lisibilité à la concision.

```
If ( $customer->has_order() ) {  
    $customer-  
        >compute_stats();  
}
```

```
If ($c->nb_o > 0 ) {  
    $c->update_encours();  
    $c->update_impaye();  
}
```

Coder proprement.

- Lisez et faites lire :



Écrire moins de code.

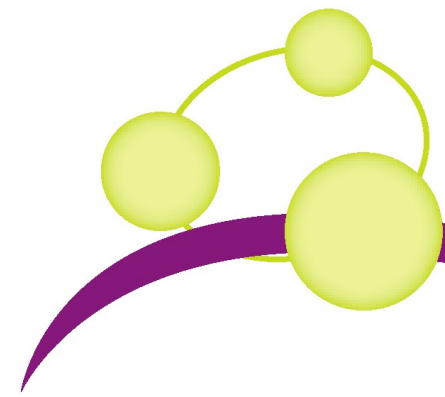
- Moins il y a de code, moins il y a de bugs.
- En moyenne, un bug toutes les 10 lignes de codes.
- C'est beau mais on est payé pour ça non ?

Comment écrire moins de code.

- Faire simple. Évitez de prévoir des fonctionnalités potentiellement utilisables par vos lointains descendants.
- Évitez la duplication de code. Le copier-coller est l'ennemi de la qualité.
- Choisir le bon langage de programmation / configuration.

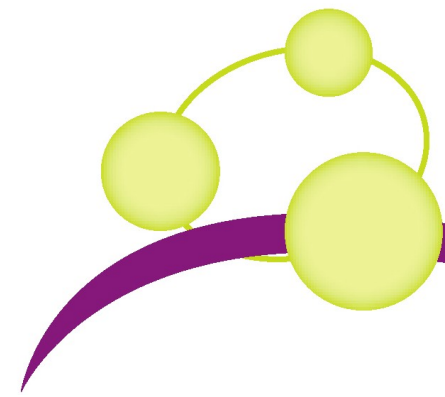
Comment écrire « encore moins de code »

- Utilisez des bibliothèques extérieures. C'est leur boulot de le maintenir et de gérer la qualité.
- Choisir des bibliothèques éprouvées.
- Privilégier la réutilisation **Si** ce n'est pas au détriment de la simplicité.



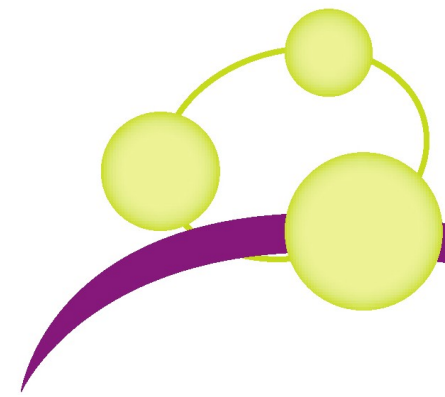
Comment écrire « encore moins de code »

- Utilisez des Frameworks de développement éprouvé
 - Symfony
 - Ruby on rails
 - ZendFramework
 - Struts
 - GWT
 - etc...



Comment écrire « encore moins de code »

- Avantage des frameworks.
 - Très peu de code à écrire.
 - La maintenance du framework n'est pas votre boulot.
 - Beaucoup de code généré par de petits fichiers de configurations.
 - Scaffolding, Générateur, ORM

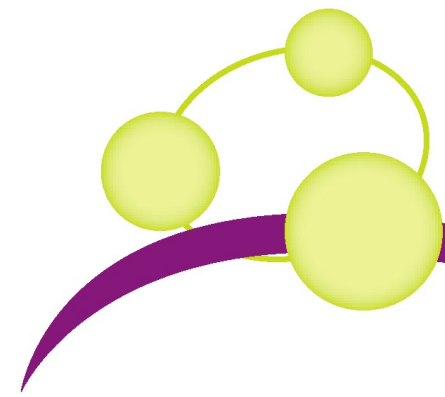


N' embauchez que des bons développeurs.

- N'employez que des développeurs parfaits qui font du code parfait sans aucun bug tout au long de l'année.
- Soyez infailible et soyez partout en même temps pour tout surveiller tout le temps.
- Ayez des clients parfaits qui savent parfaitement ce qu'il veulent.
- Ne communiquez qu'en binaire, au moins il n'y a pas de mauvaises interprétations possibles.

N' embauchez que des bons développeurs *qui font de leur mieux*

- Si vous n'avez pas que des développeurs parfaits :
 - Faites en sorte qu'ils soient guidés par des conventions et simplifiez leur la tâche
 - Règles de codages.
 - Règles d'écriture de code.
 - Framework structurant.
 - Tests automatisés.
 - Langage adapté.
 - Café à volonté et petite musique douce ...



Conclusion

- Les bugs font partie de la vie d'un logiciel.
 - Laissez leur le moins de place possible.
 - Laissez les autres les trouver.
 - Structurez et laissez le code propre.

La qualité est un tout