Bonnes pratiques de développement et valorisation

D. Arrivault[1]

[1] Laboratoire d'Excellence Archimède
Aix Marseille Université

1er Avril 2015 / Journée Mathrice - Marseille

Outline

Les phases du développement

Un ensemble de tâches.

- Définition du projet (spécifications, conception...).
- Gestion de projet (mise en place de l'équipe, de l'organisation, des outils...).
- Développement.
- Distribution (paquetage, installation).
- Diffusion / documentation.
- Maintenance / support.
- ...

Des méthodes de développement

Le fameux cycle en V...

- L'analyse des besoins et faisabilité / tests de recette.
- La conception architecturale / tests de validation.
- La conception détaillée / tests d'intégration.
- Le codage / tests unitaires.

- Forces : rassurant, facile à plannifier.
- Faiblesses : surprises à l'arrivée, figé, demande une grosse analyse amont.

Des méthodes de développement

Les méthodes itératives

- Réaliser des intégrations (plus ou moins) intermédiaires pour valider que le projet va dans le bon sens.
- Agile, Processus Unifié, XProgramming...

- Forces : réactivité, adhésion utilisateurs, gestion du risque.
- Faiblesses : nécessite une équipe formée et en interactions fortes.

Une réalité

Prenons l'exemple du LabEx Archimède :

- 4 Laboratoires (I2M, LIF, LSIS, CPT) + CIRM.
- 1 Cellule d'expertise en développement de logiciels :
  - MACAON, traitement automatique de la langue, indexation morpho-syntaxique de textes.
  - LTFAT-PYTHON, bibliothèque d'outils temps/fréquence en Python.
  - RR, plateforme de Recherche Reproductible pour les chercheur du labex.
  - GOOL, traduction de code objets (java, C++, C#, python).
  - SXP, développement d'un marché virtuel pour le troc multi-parties.
- Une équipe jeune et audacieuse : moi.

Comment faire du développement logiciel dans le monde réel de la recherche ?

Outline

Qu'est-ce qu'un bon logiciel ?

Du point de vue de l'utilisateur.

- Utile : il répond aux besoins.
- Ergonomique : facile à utiliser.
- Fiable : il ne plante pas dès qu'on change un paramètre, il passe le test de l'utilisateur novice.
- Performant : optimisez, si vous ne le faites pas pour vous, faites le pour la planète...

Pourquoi personne ne veut utiliser mon programme ?

1. « Et ça sert à quoi, sinon ? »
2. « Sérieux, il faut vraiment rentrer les 42 paramètres à la main ? »
3. « Marche pas sur mon mac ! »
4. « Je ne peux même plus regarder mes mails quand je lance ton truc. . . »

Qu'est-ce qu'un bon logiciel ?

Du point de vue du développeur

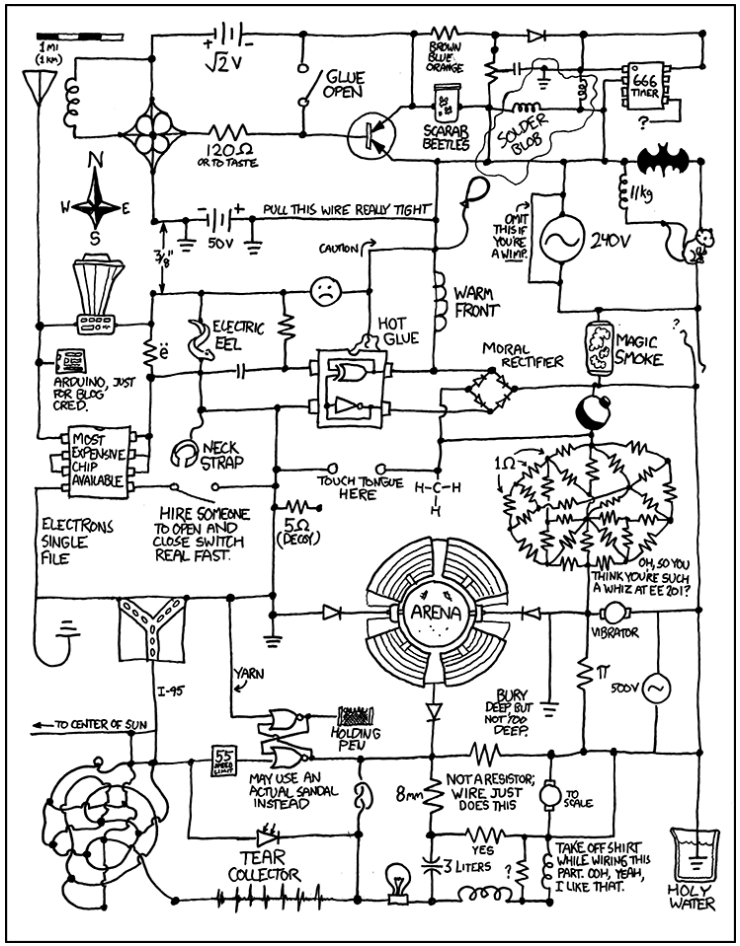
- Evolutif : la grosse fonction unique atteint toujours ses limites. . .
- Ouvert : utiliser de préférences des technologies (langages, interfaces) standards, ouvertes (Matlab c'est cher !) et maintenues.
- Maintenable : commenter le code.
- Exploitable : facile à installer, facile à mettre à jour, permettre de sauvegarder et de récupérer les données.
- Les 11 facteurs qualité de Mc Call : exactitude, fiabilité, efficacité, conviviabilité, intégrité, maintenabilité, flexibilité, testabilité, portabilité, reutilisabilité, interoperabilité.

Qu'est-ce qu'un bon logiciel ?

Tout code a potentiellement vocation à être repris.

1. Si je reprends mon code dans 5 ans, vais-je comprendre ce que j'ai fait ?
2. Si je donne mon code à mon collègue, va-t'il continuer à me parler ?
3. Pourquoi le stagiaire qui doit écrire une fonctionnalité de mon programme passe-t'il tant de temps à la cafet ?

Les spécifications.



Les spécifications.

En matière de logiciel, il est conseillé de faire précéder l'action par la réflexion...

Ecrire ce que le logiciel doit faire.

- Les cas d'utilisation : comment il va s'utiliser.
- Les besoins fonctionnels : comment les données seront modifiées.
- Les besoins non fonctionnels : environnement, performance, sécurité, support, maintenabilité...
- Les limites du logiciel : ce qu'il ne fera pas.
- Chaque spécification doit pouvoir être démontrée/testée.

Se donner des règles.

Réfléchir à comment on va le faire.

- Le ou les langage(s) à utiliser,
- les bibliothèques externes à utiliser (influence sur la licence),
- la chaîne de compilation (compilateur, outils),
- la structure du dépôt source,
- le ou les paradigmes de programmation,
- les règles de nommage,
- la gestion des erreurs,
- la gestion des commentaires,
- ...

Un peu de rigueur.

Quelques conseils.

- Rédiger un document d'architecture (expliquant les choix de programmation) qui sera mis à jour durant le développement.
- Développer par l'exemple : on commence par construire un exemple complet (code, test, documentation) qui sert de template pour la suite.
- Faire de la revue de code si on développe à plusieurs.
- Documenter au plus près du code (Génération automatique de documentation).
- Rédiger un document utilisateur et des tutoriels.
- Ecrire des tests et les automatiser : unitaires (structurels), intégration, fonctionnels (système global), recette (acceptation).
- Ecrire des classes et des méthodes de taille raisonnable.

En résumé

- YAGNI : You Ain't Gonna Need It, pas de fonctionnalités qui ne sont pas spécifiées.
- KISS : Keep It Simple, Stupid.
- DRY : Don't Repeat Yourself, le copier/coller est à éviter.
- Réutiliser du code, ne pas réécrire ce qui existe déjà.

Outline

Les gestionnaires de version.

Permet de stocker les fichiers source en conservant la chronologie de toutes les modifications qui ont été effectuées.

Les plus courants.

- GIT
- Subversion ou SVN
- Mercurial
- CVS

Les Forges.

Ensemble d'outils qui facilitent le développement collaboratif. Une forge intègre généralement : un gestionnaire de version ; un gestionnaire de listes de discussion (et/ou de forums) ; un outil de suivi des bugs ; un gestionnaire de documentation (souvent sur le principe du wiki) ; un gestionnaire des tâches.

Quelques exemples

- Souce Sup de Renater : https://sourcesup.renater.fr/
- GForge Inria : https://gforge.inria.fr/
- GitLab du LIF : https://gitlab.lif.univ-mrs.fr
- SouceForge : http://sourceforge.net/
- GitHub de GiHub Enterprise : https://github.com/

Pour les deux dernières le code doit être open source.

D'autres outils bien utiles

Intégration Continue

Adossée à une forge, elle compile sur différentes architectures, lance les tests et rapporte les résultats à chaque nouveau commit. Jenkins, GitLab CI, travis-ci.

Les outils de build

Gestion de toute la chaîne de compilation, des tests et de la distribution.

- Pour Java : Maven, Gradle.
- Pour C++ : CMake/CTest/CPack.
- Pour Python : distutils/pip

Et encore...

- tests : cunit, cppunit, junit, pytest, Funit...
- IDE : emacs, Eclipse, Netbeans, Spyder...
- Debuggers : gdb, jdb, pdb.
- Profiling : gprof, valgrind.

Outline

La valorisation

Pourquoi valoriser ?

La recherche publique a pour objectifs :

- Le développement et le progrès de la recherche, dans tous les domaines des connaissances ;
- La valorisation des résultats de la recherche au service de la société, qui s'appuie sur l'innovation et le transfert de technologie ;
- Le partage et la diffusion des connaissances scientifiques en donnant priorité aux formats libres d'accès ;
- Le développement d'une capacité d'expertise et d'appui aux associations et fondations, reconnues d'utilité publique, et aux politiques publiques menées pour répondre aux défis sociétaux, aux besoins sociaux, économiques et du développement durable ;
- La formation à la recherche et par la recherche ;
- L'organisation de l'accès libre aux données scientifiques.

(L.112-1 du Code de la recherche)

Protection des logiciels

Propriété intellectuelle

- Le logiciel est couvert par le droit d'auteur des œuvres littéraires et des créations artistiques.
- Il peut être indirectement protégé s'il est intégré à un dispositif breveté. En Europe le logiciel n'est pas brevetable en tant que tel.

L'originalité

Le droit d'auteur s'applique uniquement si le logiciel est original :

> *L'originalité d'un logiciel consiste dans un effort personnalisé allant au delà de la simple mise en œuvre d'une logique automatique et contraignante.*

Les droits de l'auteur

Les droits patrimoniaux [1] : employeur

- Ils couvrent la reproduction, La traduction, l'adaptation, l'arrangement (ou toute autre modification d'un logiciel), la reproduction du logiciel, la mise sur le marché à titre onéreux ou gratuit (y compris la location), la décompilation.
- Ils ne couvrent pas les droits de corriger les erreurs, de copier, de sauvegarder, d'étudier et de rendre interopérable.

Les droits moraux [2] : auteur

- Pour une œuvre : droit à la paternité, à la divulgation au respect de l'œuvre et au repentir.
- Application au logiciel (L121-7 du Code de la propriété intellectuelle) : seulement le droit à la paternité (incessible) et à la divulgation (cessible).

1. L122-6 et L122-6-1 du Code de la propriété intellectuelle.
2. L121-1 à L121-4 du Code de la propriété intellectuelle

La mise en œuvre du droit

Le droit d'auteur existe en dehors de tout dépôt MAIS le dépôt fait la preuve.

Où déposer ?

- Notaire, Huissier
- Agence de Protection des Programmes (APP)
- Entreprises spécialisées : LOGITAS.

Comment déposer ?

- Il existe des contrats standards (voir site de l'APP).
- Attention : en pratique, quand l'ouverture du dépôt est nécessaire (notamment en cas de défaillance de l'éditeur du logiciel), la majorité des dépôts sont inexploitables : se faire aider.

La licence

- La licence est un contrat c'est-à-dire un accord de volonté entre deux personnes : l'auteur et l'utilisateur.
- Elle précise la paternité (les auteurs), le droit de distribuer, le droit de commercialiser, le droit de modifier.
- Elle peut être propriétaire (accès au code source interdit) ou libre (liberté d'usage, d'étude, de modification et de distribution : implique la divulgation des sources).

Un logiciel sous licence libre peut être commercialisé.

La licence

Les licences libres

- GPL (Gnu Public Licence) : licence virale, tout code dérivé hérite de la licence.
- LGPL (Lesser Gnu Public Licence) : licence non virale peu contraignante.
- CeCill (Ce(A)C(nrs)I(NRIA)L(ogiciel)L(ibre)) : adaptées au droit français. Plusieurs types :
  - la CeCill est virale : compatible avec la GPL
  - la CeCill-B est comparable aux licences de type BSD
  - la CeCill=C est intéressante pour les composants logiciels

Etapes d'une valoristion

- Déterminer les auteurs (Attention les stagiares et intérimaires conservent leurs droits patrimoniaux sauf contrat le stipulant explicitement) et les titulaires des droits patrimoniaux.
- Déterminer une licence ou autre contrat : inventaire des dépendances.
- Préparation des sources et de la documentation.
- Décider d'une diffusion : académique (dépôt + licence libre), recherche d'un exploitant (dépôt + licence), recherche d'un partenaire ou création d'une start-up (dépôt + contrat de collaboration), diffusion interne (dépôt).

Outline

Conclusions

- La qualité doit être un principe dès le début du projet.
- Penser à la valorisation le plus tôt possible dans le cycle de développement.
- Faire simple dans toutes les étapes !
- 1/3 de réflexion, 1/3 de programmation, 1/3 de tests.

Outline

Credits

- Comics under a Creative Commons licence : http://xkcd.com/