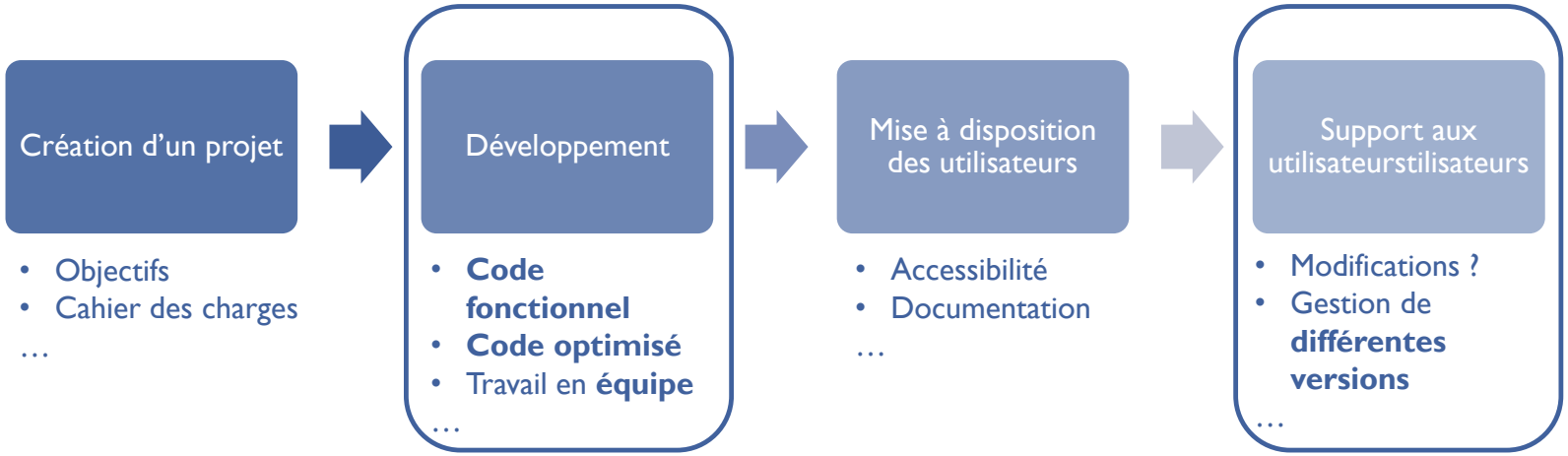
CODES COLLABORATIFS

À partir du cours de M. Julien Mathiaud

CODE COLLABORATIF ?

- Définition : projet informatique faisant intervenir plusieurs intervenants afin de remplir une tâche
- Étapes de création d'un code :



OBJECTIFS & PLAN

- Objectifs
  - Gérer un code élaboré par plusieurs personnes simultanément
  - Débugger un code efficacement
  - Optimiser un code
  - *Appliquer directement sur le projet S6 les méthodes étudiées*
- Plan
  - Logiciel de gestion de versions
  - Outils de débug
  - Outils d'optimisation

OUTILS DE DÉVELOPPEMENT EN COMMUN

RÈGLES DE DÉVELOPPEMENT

- Chaque langage de programmation a ses spécificités.
- Chaque langage a aussi ses permissivités.
- Il faut que les règles **soient les mêmes pour toute l'équipe**. Exemples :
  - Les noms de classes commencent par une majuscule.
  - Les noms de variables et de fonctions commencent par une minuscule.
  - Une seule instruction par ligne
  - Indentation de bloc : tabulations uniquement
  - Les variables sont typées dès que possible
- Voir https://google.github.io/styleguide/cppguide.html

GESTION DE VERSION

- Slides A. Gautier : https://silica.io/tutorials

CONFIGURATION

- Configuration des couleurs
  - git config --global color.diff auto
  - git config --global color.status auto
  - git config --global color.branch auto
- Pseudo
  - git config --global user.name 'votre_pseudo'
- E-mail
  - git config --global user.email moi@email.com

OUTILS DE DÉBUGGAGE

DÉBUGGAGE

- Bug ?
  - Mauvaise implémentation du programme.

- Exemples courants
  - Mauvais typage de variable (entier/réel)
  - Mauvaise allocation mémoire
  - Numérotation C et Fortran confondues
  - Coquilles dans le code (utilité de travailler à plusieurs...)
  - Mauvaise logique dans le développement
    - Nécessité de **réfléchir** à la structure globale **avant de coder**

- Solutions ?

SOLUTIONS

- Print/write
  - Outil le plus simple pour les bugs les plus simples (bien localisés)
- Points d'arrêts, conditionnés ou non
- Compiler avec tous les warnings (-Wall sur gcc)
- …

- Pour limiter les débuggages :
  - Commenter le code
  - Compiler régulièrement
  - Fractionner les fonctions
  - Mettre en place des messages d'erreurs clairs
  - Mettre en place des tests unitaires de façon systématique
    - Peut paraître superflu au premier abord, mais **grand gain de temps** par la suite
  - Erreur à éviter en C++ http://www.viva64.com/en/b/0391/

DÉBUGGERS

- Logiciels aidant à l'analyse de bugs,
  - Exécution pas à pas,
  - Affichage de variables à tout moment,
  - Mise en place de points d'arrêts
  - …
- Exemples
  - Gdb, gratuit, sans interface
    - http://www.gnu.org/software/gdb/
  - Idb, gratuit, avec interface
    - https://software.intel.com/en-us/articles/idb-linux
  - TotalView, très complet mais très cher
  - …

CODE PROFILING

PROFILAGE DE CODE

PROFILING ?

- Optimisation de code
  - Allocation mémoire
  - Rapidité d'exécution
  - …

En analysant le fonctionnement de chaque fonction

GPROF

- https://sourceware.org/binutils/docs/gprof/
- Outil d'analyse ligne à ligne d'un code en Fortran/C/C++ afin d'en estimer le coût.
- Fournit un fichier d'analyse qui guide le développeur pour améliorer les résultats.
- Ne dispose pas d'interface graphique.
- Utilisation : recompiler le code en mode débug (option –pg)

AUTRES OUTILS

- Gcov
  - Outil remplaçant gprof avec les dernières versions de gcc.
  - https://gcc.gnu.org/onlinedocs/gcc/Gcov.html#Gcov
  - Exemple : d'utilisation gcc -fprofile-arcs -ftest-coverage -g sample.c -o sample
- JVMTI
  - Pour Java
- Valgrind, analyse l'utilisation mémoire
- …

POUR PROGRESSER

- Pratiquer
- Échanger