Travail d'équipe et versionnement

Chacune des étapes implique une division des responsabilités entre les coéquipiers qui est définie de sorte que les tâches soient complémentaires et que le travail total soit complet. Ensuite chaque participant s'acquitte de sa part des responsabilités et partage **très rapidement** son travail avec ses coéquipiers via git afin de réaliser une **intégration continue**.

COMMIT RELATIF AUX FONCTIONNALITÉS

- Chacun doit faire au minimum un **update** au début de la séance et un **update & commit** à la fin de la séance.
  - En **moyenne**, une séance de travail devrait générer environ 5 à 10 **commit**.
  - En conséquence, il est attendu de la part de chaque programmeur environ 20 **commit** par semaine incluant des **commit** en-dehors des cours.
  - Une équipe de 3 devrait au bout de 8 semaines avoir généré au moins 300 **commit**.
- Ne pas faire de commit si le projet génère des erreurs. Ceci ne vous dispense pas de partager mais il faut alors procéder ainsi : mettre en commentaire ce qui fait des erreurs afin de permettre aux autres de continuer à tester leur travail puis effectuer le commit.
- Un **update/commit** doit être effectué à chaque avancement dans le code même si la fonctionnalité n'est pas terminée. Ce type de commit sera utilisé avec l'instruction trac **references**. Les expérimentations doivent aussi être versionnées dans une 'branch' du dépôt.
- Un **update/commit** doit obligatoirement être effectué à la fin de chaque fonctionnalité significative. Ce type de commit sera utilisé avec l'instruction trac **closes**.

COMMIT RELATIF À L'ARCHITECTURE

- Un **update/commit** doit être effectué à la fin de chaque changement central dans le code partagé. Dans ce cas le participant indique à son équipe de se mettre à jour.
- En cas de réarchitecture ('refactoring'), toute l'équipe doit effectuer un **commit** avant, cesser de coder puis exécuter un **update** après. Les réarchitectures doivent être limitées au minimum et réalisées tôt dans le projet, tôt dans les phases de développement ou en cas de besoin. Cette décision d'équipe doit être partagée avec l'enseignant.

Si d'aventure un participant devait réarchitecturer l'application, il aviserait toute l'équipe un peu en avance et leur demanderait de faire les commit de tout leur travail, puis de cesser de travailler pendant toute l'opération. Ensuite il ferait le suivi sur chaque participant si le projet se recharge tel qu'attendu.

COMMIT RELATIF À LA DOCUMENTATION

- Toutes les documentations sont partagées via Github dans un répertoire du projet qui s'appelle **doc** en minuscules, autant celles produites que celles utilisées.
- Une étiquette git (RELEASE) est préparée pour chaque remise. Ces étiquettes portent les noms indiqués dans le calendrier sinon les points ne seront pas accordés : **ITERATION_ 1_FONCTIONNEL**, **ITERATION_1_DONNEES**, **ITERATION_2_FONCTIONNEL, etc.**

IL EST INTERDIT DE

- Demander à un membre de votre équipe de ne pas **committer** dans le projet. Si l'environnement d'un participant est dysfonctionnel ou si le participant n'est pas compétent avec git ou le langage, vous veillerez à le former, à l'assister et à lui donner des tâches à sa mesure.

- Faire pointer vos espaces de travail (sandbox) vers des répertoires différents de git ou des branches diff/rentes de manière à ne pas travailler sur la même copie du projet. Toute semaine où l'équipe est dans cette situation amènera un retrait de 10% des points du projet.