a) Versions dans l'assemblyInfo

```
//============================
// [assembly: AssemblyVersion("MAJOR.MINOR.PATCH.SourceVersion")]
// Étant donné le numéro de version MAJOR.MINOR.PATCH.SourceVersion, incrémentez le:
//  Version MAJOR lorsque vous effectuez des modifications avec ruptures de compatibilités,
//  Version MINOR lorsque vous ajoutez des fonctionnalités de manière rétrocompatible, et
//  Version PATCH lorsque vous effectuez des corrections de bogues rétrocompatibles.
// SourceVersion : - est écrasé automatiquement lors de la compilation par le build
automatisé
//                 - est l'identifiant du rangement dans le gestionnaire de code source
//============================
[assembly: AssemblyVersion("2017.1.1.0")]
```

Si on s'interdit les ruptures de compatibilités au cours d'une année, les versions peuvent suivre le format suivant :

[**Année**].[**Livraison**].[**Correctif**].[**SourceVersion**]

- **Année** : Année du développement
- **Livraison** : Numéro de la livraison dans l'année
- **Correctif** : Numéro de la livraison du correctif pour la livraison en cours

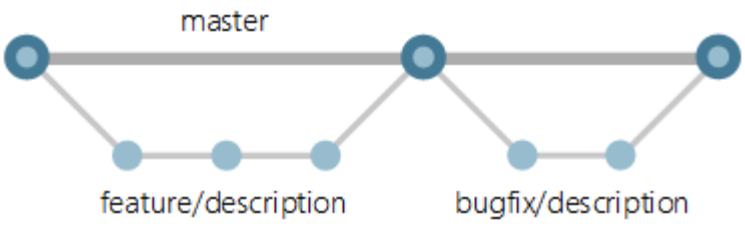
1. Branche « master »

Le niveau de qualité du code doit y être élevé et toujours à jour

L'application générée depuis cette branche doit toujours être dans un état livrable

Les versions PATCH dans master doit toujours être 0.

2. Utiliser des branches par « feature » pour votre travail



b) Conventions de nommage des branches

Feature/[Nom / description de la fonctionnalité]

bugfix/[Nom / description de la fonctionnalité]

c) Revue de code par Pull requests

- Ne pas ramener de code dans master sans pull requests, toujours avoir la branche master
- Avant de demander une revue de code :
    - Ré-examiner le changement comme si nous n'en étions pas l'auteur et en se conformant aux règles d'équipe
    - En cas de doute, aller poser des questions
    - Faciliter au maximum le travail de celui qui fait la revue de code
    - Le code soumis doit être petit, incrémental et complet
    - Décrire le contexte du changement pour aider le reviewer
- Pour une revue de code efficace :
    - Se limiter dans le temps : pas plus de 6 heures par semaines
    - Se limiter dans le scope :
        - Y a-t-il des bugs ? Des cas non gérés ? Des vérifications non-faites ?
        - Le code est-il propre ? Compréhensible ?
        - Inutile :
            - Proposer d'autres manières de faire alors que celle en place fonctionne et est propre, compréhensible
            - Commenter du code non-modifié par le demandeur
            - Faire des louanges
            - Ne pas créer de nouvelles tâches non planifiées
- Pour éviter les effets ping-pong :
    - Relire ses commentaires avant de les publier
    - Aller demander oralement des précisions si vous ne comprenez pas des commentaires
    - Si du code n'est pas compris, privilégier la modification du code plutôt qu'un message explicatif

3. Branches « Release »

Les branches « Release » permettent d'effectuer des corrections de bugs sur une application en acceptation ou en production alors que des développements incompatibles sont en cours au niveau de la branche master.

Ces branches ne sont pas créées au moment de la mise en production mais à la première demande de correction pour une version en production donnée.

Au moment de la 1ère demande de correction, il est tout de même nécessaire de vérifier si la création de cette branche est nécessaire. Les développements en cours sont-ils vraiment incompatibles ? Quels sont les risques ? Pouvons-nous vérifier l'impact dans un environnement non risqué (en dev) ? En analysant les changements effectués dans le code source ?

d) Conventions de nommage des branches Release

Release/[Nom / description de la fonctionnalité]

Identifiant d'un rangement TFS :

[**Année**].[**Livraison**].[**Correctif**].[**Changeset**]

- **Année** : Année du développement
- **Livraison** : Numéro de la livraison dans l'année
- **Correctif** : Numéro de la livraison du correctif pour la livraison en cours
- **Changeset** : numéro de rangement TFS