Proposition d’une nouvelle organisation pour la gestion du code du modèle SURFEX

Plan

A Contexte

B Organisation proposée

- B.1 Tests des nouveaux développements et débogage
- B.2 Préparation d'une nouvelle version
- B.3 Assistance aux utilisateurs

C Rappel des normes de codage

D Synthèse

A Contexte (1)

- Gestion fortement centralisée pour :
  - Phasages et tests des nouvelles versions, corrections de bugs, assistance aux utilisateurs, interactions avec les développeurs...

- Evolution du modèle Surfex observée :
  - Complexité du code croissante, nombre de développeurs et d’utilisateurs croissant, utilisations opérationnelles et recherche couplées appelant une plus grande fiabilité, grand nombre d'allers et retours en phase de tests, développements arrivant tardivement dans le processus de création d'une nouvelle version...

A Contexte (2)

- Objectifs identifiés :
  - Réduire le temps passé en aval dans la recherche et la correction des bugs
  - Réduire le nombre de phasages intermédiaires des contributions entre deux versions
  - Accroître l'implication des développeurs et utilisateurs dans la vie du code

=> Versions moins fréquentes et plus robustes de SURFEX

B Organisation proposée

B.1 Tests des nouveaux développements et débogage (1)

- Développement par l'équipe Surfex d'une **batterie de tests automatiques** (en cours) qui parcourent les options et configurations de Surfex, à lancer :
  - Par chaque contributeur avant la proposition d'un développement pour une nouvelle version officielle
  - Par chaque utilisateur qui corrige un bug non trivial, entre deux versions officielles
  - Par les phaseurs à la fin de leur étape de phasage

=> cette base devrait permettre de détecter beaucoup plus de bugs en amont de la préparation des nouvelles versions

B Organisation proposée

B.1 Tests des nouveaux développements et débogage (2)

- Une **branche BUG** est mise à disposition de tous les utilisateurs et développeurs.
  - Ils confronteront leurs bugs et leurs corrections **sur le forum du site surfex-lab**, au lieu d'**informer uniquement et directement l'équipe Surfex**.

    L'équipe Surfex officialisera les corrections depuis la branche BUG sur le TRUNK quand elles auront été testées et validées par les utilisateurs développeurs.

=> d'où l'importance d'indiquer son nom et une description de la modification effectuée dans le code, afin qu'on sache à qui s'adresser en cas de problème.

=> L'instauration d'une discussion entre les utilisateurs sur le forum à propos des bugs permettra petit à petit à chacun de mieux comprendre la façon dont la partie du modèle qui l'intéresse s'articule avec le reste du code.

B Organisation proposée

B.2 préparation d'une nouvelle version (1)

- Création d'un **groupe de phaseurs**, constitué de l'équipe Surfex, des contributeurs à la future version + 1 utilisateur clé par grand contexte d'utilisation (Mésonh, Arome, ARPEGE-CLIMAT...) :
  - En amont, analyse de l'impact de chaque développement sur les autres parties du code, afin de renvoyer toute contribution qui ne respecte pas l'exigence de reproductibilité des résultats d'une version à l'autre
  - Détermination d'un ordre de phasage en vue de minimiser le travail de chaque contributeur. Puis, phasage sur la branche DEV+ tests complets de sa partie par chaque contributeur (via la batterie de tests automatiques)

=> Entre deux versions officielles, le développeur est totalement autonome sur son développement, il n'a qu'un phasage final à effectuer, celui entre sa version de développement et la version en cours de construction (branche DEV)

B Organisation proposée

B.2 préparation d'une nouvelle version (2)

- Pour le **mode couplé**, l'utilisateur clé concerné détermine avec le groupe de phasage à quel moment les tests doivent être effectués (avant ou après la sortie officielle de la version OFFLINE) et les prend en charge.
- Réalisation d'un **bilan de tests final** sur la version OFFLINE, et officialisation de la version par le groupe.

=> Il serait souhaitable d'avoir des tests couplés à chaque nouvelle version afin d'éviter les trop grands écarts de versions ensuite

B Organisation proposée

B.3 assistance aux utilisateurs (1)

⇒ *Sur un **plan pratique**, l'équipe SURFEX a d'autres charges que l'assistance aux utilisateurs (développement, optimisations) et n'a plus les capacités d'assurer la totalité de l'assistance de premier niveau.*

- **Utilisateurs débutants** qui travaillent avec une personne plus expérimentée sur Surfex:
  - Cette personne expérimentée **assure le support** pour le nouvel utilisateur: environnement, compilation, exécution des cas tests, sensibilisation au débogage (lecture des messages d'erreur, print dans le code…)
- Avant de **contacter l'équipe Surfex**:
  - Regarder dans la branche BUG et/ou sur le forum s'il ne s'agit pas d'un bug résolu ou en cours de résolution
  - Essayer de comprendre le problème et de le résoudre
  - Consulter la **documentation** présente sur le site Surfex-lab

B Organisation proposée

B.3 assistance aux utilisateurs (2)

⇒ *Le site internet surfex-lab contient de **nombreuses documentations** qui peuvent contenir les informations recherchées:*

- Comptes rendus des réunions de coordination
- Browser de code (généré par Doxygen)
- Documentation scientifique
- Documentation technique:
  - Guide du développeur
  - Modifications induites par la parallélisation du driver OFFLINE
  - Guide de l'utilisateur avec documentation de toutes les namelists
- Documentations des versions:
  - Changements d'une version à la suivante
  - Tests de productibilité des résultats
- Bugs corrigés sur la version officielle courante (trunk SVN)
- Téléchargements:
  - Des versions officielles
  - D'outils divers d'aide au développements
  - De cas tests spécifiques à certaines options du code
  - De cartes de physiographie
- Forum de discussion

B Organisation proposée

B.3 assistance aux utilisateurs (3)

⇒*Pister un bug en mettant des print ou en commentant des lignes amène à découvrir une partie du code, ce qui amène à mieux connaître l'architecture du modèle, ce qui amène à comprendre plus vite l'origine des problèmes rencontrés et à coder plus facilement, et ainsi de suite…*

⇒*Le travail de débogage général profite à l'ensemble des utilisateurs*

C Normes de codage (1)

- Les noms de variables commencent par une lettre différente selon leur type et l'endroit où elles sont déclarées (cf norme DOCTOR)

- Les modules de Surfex sont séparés en modules « n » (variables différentes d'un sous-modèle à l'autre) et modules simples. Avant de créer un nouveau module ou d'ajouter une nouvelle variable dans un module existant, bien réfléchir à sa catégorie et copier les syntaxes à partir de modules existants.

- Ne pas écrire les mêmes lignes de code plusieurs fois (sources d'erreur), les regrouper en subroutines

- Les USE MODI des routines appelées doivent être introduits (sinon plantage à la compilation avec GMKPACK), ceux des routines non appelées supprimés

C Normes de codage (2)

- DR_HOOK est implémenté dans chaque routine (5 instructions à ajouter par routine)
- Faire un profiling temporel (avec DR_HOOK) sur son développement afin de déterminer si on n'introduit pas de ralentissement excessif du code
- Les modifications doivent être documentées dans le code
- Les développements qui changent les résultats doivent être mis sous clé

=> *Une contribution qui ne respecte pas ces règles est susceptible d'être rejetée à la première étape du phasage*

D Synthèse

- Deux objectifs à atteindre:
  - Renforcer l'efficacité des étapes de phasage, de tests et de corrections de bugs:
    - En éliminant le maximum de bugs en amont grâce à la batterie de tests automatiques
    - En organisant plus finement l'étape de phasage afin d'éviter les allers et retours entre versions de développements
  - Répartir les tâches d'évolution du code, afin que :
    - Le bon fonctionnement général du code soit pris en charge par plusieurs intervenants => contrôles plus fiables
    - l'équipe Surfex puisse consacrer plus de temps à des activités telles que l'optimisation ou l'amélioration de l'ergonomie générale du code.