```
/*****************************************************************************
 *  Logiciel portageMM2CPP de portage de code ModelMaker en langage C++      *
 *  Copyright INRA, février 2006                                             *
 *****************************************************************************/

/*****************************************************************************
 *
 * Fichier     : help_testsDuLogiciel
 *
 * Auteur(s)   : Nathalie Rousse, Nathalie.Rousse@toulouse.inra.fr
 *               de l'INRA - Institut National de la Recherche Agronomique -
 *               ( Département MIA, UMR AGIR, RMT modelia ).
 *
 * Description : Fichier faisant partie de la documentation du logiciel
 * portageMM2CPP. L'ensemble des fichiers de documentation est récapitulé
 * dans le fichier help.
 *
 *****************************************************************************
 * Historique :
 *
 * 23/11/07, Nathalie Rousse : création du fichier.
 *
 * Le fichier est créé par déplacement de texte qui se trouvait jusque là
 * dans le fichier help.
 *
 * 30/11/07, Nathalie Rousse : mise à jour du fichier conformément à la
 * modification/réorganisation des répertoires (cf m_repertoires).
 *
 * 17/09/08, Nathalie Rousse : mise à jour pour la version 080917.
 *
 *****************************************************************************/
/*

****** TESTS DU LOGICIEL ***************************************************

Il s'agit de vérifier le logiciel écrit en C++, c'est à dire qu'il se comporte comme attendu/prévu, sort des résultats
attendus/cohérents. Il n'est pas du tout question ici de faire de la validation du modèle (ajustement de paramètres, an
alyse de sensibilité ...).

Exemple (appelé par la suite "testComparaisonMM") : Il peut s'agir par exemple de vérifier que le logiciel écrit en C++
 est conforme au modèle sous sa forme ModelMaker ("modele2.mod"), c'est à dire qu'il se comporte à l'identique, sort le
s mêmes résultats. Les tests sont alors mis en place et en oeuvre selon le principe suivant : après avoir programmé en
C++ l'appel des simulations "équivalentes" à celles qui sont effectuées dans ModelMaker, comparer les sorties ainsi obt
enues avec les résultats des simulations effectuées intra-ModelMaker ; voir en illustration la "Séquence de constructio
n et déroulement d'un scénario de test" dans les représentations UML, voir aussi son complément dans « UMLmaj_portageMM
2CPP.ppt ».
```

D'une façon plus globale, mettre un test en place et en oeuvre consiste à : programmer en C++ l'appel des simulations m
ettant dans la situation à tester, exécuter le scénario de test ainsi créé, et analyser le comportement du logiciel et
ses sorties ainsi obtenues. Voir le complément de la représentation UML "Séquence de construction et déroulement d'un s
cénario de test" dans « UMLmaj_portageMM2CPP.ppt ».

Le répertoire des tests est "lesTestsCPP". Il contient :

- le fichier "tester" qui est le script de compilation et d'exécution des tests (plus exactement : exécution de la part
ie automatique des tests qui la plupart du temps est accompagnée et suivie de vérifications manuelles).

- les répertoires des scénarios de test. Chacun de ces répertoires regroupe les informations du scénario de test auquel
 il est associé, c'est à dire ce qui permet de dérouler le test et les résultats du test.

Parmi les répertoires des scénarios de test, le répertoire "scnParDefaut" est particulier dans le sens où c'est le répe
rtoire du scénario de test par défaut.

- les fichiers "rapportGlobalDesTests" et "genererLeRapportGlobalDesTests" : le fichier "genererLeRapportGlobalDesTests
" est un script de génération de documentation, qui regroupe dans un seul fichier "rapportGlobalDesTests" le rapport de
 test de chacun des scénarios de test (plus exactement le contenu des fichiers "rapportTest" et "statutTest" de chaque
scénario).

Le répertoire "scnParDefaut" contient :

(a) Le répertoire "leSourceTest" dédié au code source propre aux tests du scénario :

Le répertoire "leSourceTest" contient le fichier "mainTestPortageMM2CPP.cpp" du programme principal des tests du scénar
io. Ce fichier est écrit par le testeur.

(b) Des répertoires de recopie/sauvegarde du code source et des entrées du logiciel testé, dans l'état où ils se trouva
ient lorsque le logiciel a été testé (ie le test exécuté) :

La recopie/sauvegarde de ces répertoires est effectuée automatiquement lors de l'exécution du test (cf commande "tester
").

Ces répertoires sont :
- le répertoire "svg_leSourceCPP" : recopie/sauvegarde du répertoire "leSourceCPP".
- le répertoire "svg_laConfigCPP" : recopie/sauvegarde du répertoire "laConfigCPP".
- le répertoire "svg_lesEntreesCPP" : recopie/sauvegarde du répertoire "lesEntreesCPP".

ATTENTION, ces répertoires "svg_..." ne sont pas tous forcément utilisés à l'exécution du test :
- le répertoire "svg_leSourceCPP" : ce répertoire est utilisé à l'exécution du test.
- le répertoire "svg_laConfigCPP" : ce n'est pas forcément cette configuration qui est utilisée/appelée dans le test.
- le répertoire "svg_lesEntreesCPP" : ce ne sont pas forcément ces fichiers d'entrée qui sont utilisés/appelés dans le
test.
Pour plus d'explications, voir (c).

(c) Des répertoires nécessaires au logiciel testé (nécessaires à sa génération car en faisant partie, ou nécessaires à son exécution car invoquées lors de l'exécution) :

- le répertoire "laConfigCPP" contient le code source de la configuration qui est utilisée/appelée dans le test. Il est utilisé lors de la compilation du test. Dans le cas de tests qui ne nécessitent pas de retoucher la configuration du logiciel testé, "laConfigCPP" peut pointer le répertoire "svg_laConfigCPP" (voir (b)).
- le répertoire "lesEntreesCPP" est le répertoire dans lequel le logiciel testé va chercher ses fichiers d'entrée lors de l'exécution de la simulation (cf specificiteConfigPortageMM2CPP.h). Dans le cas de tests qui ne nécessitent pas de retoucher les fichiers d'entrée du logiciel testé, "lesEntreesCPP" peut pointer le répertoire "svg_lesEntreesCPP" (voir (b)).
- le répertoire "lesSortiesCPP" est le répertoire dans lequel le logiciel testé envoie ses résultats/sorties lors de l'exécution de la simulation (cf specificiteConfigPortageMM2CPP.h). Le test récupère (de manière automatique, cf commande "tester") des informations de "lesSortiesCPP" dans "lesSortiesTest".

(d) Les répertoires contenant les informations propres aux tests du scénario :

Ces répertoires sont :
- le répertoire "lesEntreesTest" pour les entrées.
- le répertoire "lesSortiesTest" pour les sorties.

Exemple "testComparaisonMM" :
- le répertoire "lesSortiesTest" : contient les résultats des tests (ie les fichiers .res résultats de la simulation en C++), qui sont rangés là automatiquement lors de l'exécution du test (ils sont récupérés dans "lesSortiesCPP").
- le répertoire "lesEntreesTest" : contient les données qui serviront d'éléments de comparaison lorsque le testeur fera ses vérifications. C'est le testeur qui remplit "lesEntreesTest". A priori il y dépose des données issues de ModelMaker (résultats des simulations effectuées intra-ModelMaker).

(e) Le répertoire "executionTest" dédié à l'exécution des tests du scénario :

Le répertoire "executionTest" contient le fichier "testPortageMM2CPP.exe" qui est l'exécutable du test (fichier généré et appelé automatiquement lors de l'exécution du test), ainsi que son état précédent (fichier "testPortageMM2CPP.exe.prec").

(f) Les fichiers rendant compte du test :

- le fichier "rapportTest" où le testeur décrit le test et où il notifiera les conclusions du test (la trame du fichier "rapportTest" est créée automatiquement lors de l'exécution du test si le fichier n'existe pas encore).
- le fichier "statusTest" où le testeur précise quelle utilisation faire du test au cours du temps (si c'est un test à rejouer ou bien transitoire...).
- le fichier "testPortageMM2CPP.ficheGen" : la fiche associée à la génération de testPortageMM2CPP.exe (fiche générée automatiquement lors de l'exécution du test), ainsi que son état précédent (fichier "testPortageMM2CPP.ficheGen.prec").

Les autres répertoires de scénarios de test suivent le même schéma que "scnParDefaut".

Procédure de développement puis de déroulement du scénario de test par défaut :

- Dans "scnParDefaut/leSourceTest", écrire le programme du scénario de test "mainTestPortageMM2CPP.cpp".

- Préparer "scnParDefaut/laConfigCPP" : si le scénario de test ne nécessite pas de retoucher la configuration du logici el testé, alors faire pointer "laConfigCPP" sur "svg_laConfigCPP" (ie créer lien, commande "ln -s"), et sinon écrire da ns "laConfigCPP" la configuration adaptée au test.

- Préparer "scnParDefaut/lesEntreesCPP" : si le scénario de test ne nécessite pas de retoucher les fichiers d'entrée du logiciel testé, alors faire pointer "lesEntreesCPP" sur "svg_lesEntreesCPP" (ie créer lien, commande "ln -s"), et sino n écrire dans "lesEntreesCPP" les fichiers d'entrée adaptés au test.

- Dans "scnParDefaut/lesEntreesTest", déposer les données nécessaires en entrée du test. Dans l'exemple "testComparaiso nMM", ce sont les données de comparaison issues de ModelMaker. Dans d'autres cas ça peut être des résultats attendus pr é-calculés, etc.

- Compiler/exécuter le scénario de test par défaut, en lançant dans le répertoire "lesTestsCPP" la commande en ligne : "./tester". Les résultats du test sont alors rangés dans le répertoire "scnParDefaut/lesSortiesTest".

- Dans "scnParDefaut", faire les vérifications voulues par le test (cf "Partie description de test" dans "rapportTest") . Il s'agit d'exploiter/analyser les informations de "scnParDefaut/lesEntreesTest" et "scnParDefaut/lesSortiesTest". Da ns l'exemple "testComparaisonMM" : comparaison des fichiers résultats de la simulation en C++ (contenus dans "lesSortie sTest") avec les données de comparaison issues de ModelMaker (contenues dans "lesEntreesTest").

- Dans "scnParDefaut", notifier les conclusions du test dans le fichier "rapportTest" (dans la "Partie rapport de test").

Développement de plusieurs scénarios de test scni=scn1,scn2,...,scnN :

- Dans le répertoire "scni/leSourceTest", écrire le programme du scénario de test scni : "mainTestPortageMM2CPP.cpp" (o n peut aussi particulariser son nom, par exemple : "mainTestPortageMM2CPP_scni.cpp"),

- Préparer "scni/laConfigCPP" : si le scénario de test scni ne nécessite pas de retoucher la configuration du logiciel testé, alors faire pointer "laConfigCPP" sur "svg_laConfigCPP" (ie créer lien, commande "ln -s"), et sinon écrire dans "laConfigCPP" la configuration propre au test scni.

- Préparer "scni/lesEntreesCPP" : si le scénario de test scni ne nécessite pas de retoucher les fichiers d'entrée du lo giciel testé, alors faire pointer "lesEntreesCPP" sur "svg_lesEntreesCPP" (ie créer lien, commande "ln -s"), et sinon é crire dans "lesEntreesCPP" les fichiers d'entrée adaptés au test scni.

- Dans le répertoire "scni/lesEntreesTest", déposer les données nécessaires en entrée du test. Dans l'exemple "testComp araisonMM", ce sont les données de comparaison issues de ModelMaker. Dans d'autres cas ça peut être des résultats atten dus pré-calculés, etc.

- Compiler/exécuter le scénario de test scni, en lançant dans le répertoire "lesTestsCPP" la commande en ligne : "./tes ter scni" (ou "./tester scni mainTestPortageMM2CPP_scni.cpp" si on avait donné au fichier un nom particulier). Les résu

```
ltats du scénario de test scni sont alors rangés dans le répertoire "scni/lesSortiesTest".

- Dans le répertoire "scni", faire les vérifications voulues par le test (cf "Partie description de test" dans "rapport
Test"). Il s'agit d'exploiter/analyser les informations de "scni/lesEntreesTest" et "scni/lesSortiesTest". Dans l'exemp
le "testComparaisonMM" : comparaison des fichiers résultats de la simulation en C++ (contenus dans "lesSortiesTest") av
ec les données de comparaison issues de ModelMaker (contenues dans "lesEntreesTest").

- Dans "scni", notifier les conclusions du test dans le fichier "rapportTest" (dans la "Partie rapport de test").

*****************************************************************************

*/
```