Modèles vs. instances de données

Un script Computree, soit une série d'étapes, n'est pas exécuté au moment où il est créé; autrement dit, les étapes sont ajoutées avant d'être exécutées. Afin de valider qu'une étape peut être ajoutée à la suite d'une autre, il faut que les sorties de la première et les entrées de la seconde soient connues et compatibles. Il est donc nécessaire de connaître leur format avant que quelques données que ce soit aient été traitées. Pour ce faire, Computree utilise une stratégie en deux temps : au moment de l'ajout de l'étape à l'arbre de traitement, l'étape spécifie le *modèle* de ses données; au moment du traitement, l'étape traite les *instances* de ses données.

Modèles

Les modèles de données doivent être spécifiés pour les entrées et les sorties de l'étape, soit dans les méthodes createInResultModelListProtected et createOutResultModelListProtected. Les méthodes utilisées indiquent d'ailleurs qu'elles traitent le modèle des données : addGroupModel et addItemModel.

```
void LVOX_StepNdNtGrids::createInResultModelListProtected()
{
    CT_InResultModelGroupToCopy* in_res = createNewInResultModelForCopy(DEF_in_res, tr("Grilles"
));
    in_res->setZeroOrMoreRootGroup();
    in_res->addGroupModel("", DEF_in_grp);
    in_res->addItemModel(DEF_in_grp, DEF_in_nd, CT_Grid3D<int>::staticGetType(), tr("nb (before)"
));
    in_res->addItemModel(DEF_in_grp, DEF_in_nt, CT_Grid3D<int>::staticGetType(), tr("nt (theo)"));
}

void LVOX_StepNdNtGrids::createOutResultModelListProtected()
{
    CT_OutResultModelGroupToCopyPossibilities* out_res = createNewOutResultModelToCopy(DEF_in_res)
;
    if (res != NULL)
    {
        out_res->addItemModel(DEF_in_grp_ndnt, _ndnt_ModelName, new CT_Grid3D<float>(), tr("nd/nt"
));
    }
}
```

Instances

Les données elles-mêmes sont uniquement accessibles au cours du traitement des données, soit au sein de la méthode compute et leur structure peut être modifiée à l'aide de méthodes comme addGroup et addItemDrawable.

```
void LVOX_StepNdNtGrids::compute()
{
    // Get inputs and outputs...
    // ...

    // Iterate...
    CT_ResultGroupIterator it(out, this, DEF_in_grp);
    while (it.hasNext() && !isStopped())
    {
        // Various treatments...
     // ...

         // Add actual items and groups to result
        out_group->addItemDrawable(nbnt);
        out_res->addGroup(out_group);
        nbnt->computeMinMax();
```

```
        }
}
```