Règles concernant la structuration des données

Résumé :

Nous indiquons ici les règles (et conseils) concernant la Structuration des Données (SD) :

- définir de nouveaux types de SD,
- avec quels objets JEVEUX ?

La lecture de ce document suppose la lecture préalable des documents [D4.01.01] et [D5.01.03].

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Table des matières

1 Définir et utiliser de nouveaux type_SD....3
2 Objets JEVEUX de base....5

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

1 Définir et utiliser de nouveaux type_SD

- 1 - Penser “Structures de Données” : pas seulement pour les Structures de Données “utilisateurs” mais aussi pour tous les objets de travail. Dès que l’on doit manipuler plusieurs objets “en même temps”, les regrouper dans un type de Structures de Données que l’on documentera.

- 2 - “Consulter l’acquis” :
  - connaître les types de SD existants,
  - mesurer l’adéquation des types de SD existants à son besoin,
  - se poser la question (et la poser aux autres) : faut-il modifier une SD existante ou en créer une nouvelle. En particulier, il faut éviter la multiplication des types de SD “utilisateur”,
  - toute modification (ou introduction) de type de SD doit être discutée en EDA.

- 3 - Utiliser les noms des types de Structures de Données dans les commentaires des routines. Respecter pour cela scrupuleusement l’orthographe de ces noms [D2.01.02].

  Écrire par exemple :

```
    SUBROUTINE TOTO (CART, MAILLA)
C IN CART   : NOM D'UNE SD CARTE
C IN MAILLA : NOM D'UNE SD MAILLAGE
```

- 4 - Utiliser de nouveaux suffixes pour tout nouveau type de Structures de Données. La liste des suffixes actuellement utilisés est donnée dans [D4.01.02]. Pour des raisons de lisibilité, on peut également commencer ses suffixes par une chaîne de caractères commune.
  Exemple : SD LISTE_RELA (K19) :

  suffixes : '.RLCO', '.RLDD', '.RLNO', … (“.RL” –> “relation linéaire”).

- 5 - Un suffixe doit commencer par un “.” (lisibilité).
  On peut utiliser les caractères : “A, B, … Z, 0, 1, … 9, _ , &, .”

- 6 - Respecter les longueurs “standard” pour les noms des Structures de Données :
  K8, **K14** ou **K19**.

- 7 - Ne pas multiplier les objets JEVEUX pour faire “plus joli” :
  - on peut stocker 1 K8 et 1 K24 dans un vecteur de 2 K24
  - un booléen isolé peut être codé dans un entier (0 ou 1),
  - …

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

- 8 - Quand on “commence” la structuration (pour les types de Structures de Données de plus bas niveau), utiliser les noms les plus longs (K19) pour pouvoir ultérieurement créer de nouveaux type de Structures de Données les contenant. (i.e. commencer les suffixes “par la droite“).

  Exemple :

```
type_1er      (K19)     record
   '.T1AA'     :        OJB    …
   '.T1BB'     :        OJB    …
type_2nd      (K14)     record
   '.T2AT1'    :        type_1er
   …
```

  Le type de SD `type_1er` peut être utilisé comme article du type `type_2nd`.

- 9 - Penser aux références “amont” : il est parfois utile de stocker dans une Structure de Données les noms des Structures de Données qui lui ont donné naissance. Ceci suppose que ces Structures de Données sont pérennes (possèdent une image dans la base ‘`GLOBALE`‘).

- 10 - Éviter les redondances au sein d’un type de Structures de Données, car la mise à jour devient plus problématique. Un mauvais exemple : `listr8` (liste de réels) où l’on stocke à la fois la liste complète des réels et la liste des intervalles de pas constant (l’un des objets (`.VALE`) peut être à tout moment recalculé à partir des autres).

- 11 - Quand on déclare des types de Structures de Données “chapeaux”,

```
type_1 (K8) ::= record
        / $VIDE    :       type_2
        / $VIDE    :       type_3
```

  il faut s’assurer qu’il existe un moyen de “traverser” les “/” ; c’est-à-dire de savoir reconnaître si la Structure de Données est de type `type_2` ou `type_3`.

  Par exemple :

- 12 - Problème des SD “cachées” :
  Il arrive parfois qu’une SD stocke le nom d’autres SD (par exemple, la `SD_RESULTAT` stocke dans son objet `.TACH` le nom des champs qui composent la `SD_RESULTAT`). Quand les SD référencées sont inconnues (ou cachées) de l’utilisateur (c’est le cas des champs des `SD_RESULTAT`), il faut trouver un nom pour ces SD référencées.

  On adoptera la règle suivante :

  Si on doit nommer une SD cachée que l’on référence dans une SD utilisateur nommée `NOMU` (K8), on donnera à cette SD cachée un nom commençant par `NOMU`.

  Grâce au respect de cette règle, la commande `DETRUIRE/CONCEPT=NOMU` détruira correctement TOUS les objets JEVEUX créés par la commande ayant créé `NOMU`.

  Pour obtenir un nom de SD cachée respectant cette règle, on peut utiliser la routine `GNOMSD`.

- 13 – Écrire le catalogue Python associé à la nouvelle structure de données. Voir D5.01.03

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2 Objets JEVEUX de base

- 1 - Ne pas utiliser l'attribut **'DOCU'** des OJB.

- 2 - Essayer de ne pas utiliser 'LONUTI': on cherchera en général à allouer "au plus juste" les objets. Dans ce cas, **'LONUTI'='LONMAX'**.

Utiliser à bon escient l'attribut **'LONUTI'** : c'est à l'utilisateur de le mettre à jour ; il ne faut pas lui donner une autre signification que la sienne : longueur réellement utilisée d'un vecteur.

- 3 - Pour les collections dont on sait qu'elles ne seront jamais très grosses, il est préférable de les créer contiguës. Dans le cas contraire, il faut les créer dispersées. On dira qu'une collection (ou un objet) est grosse si :
  - son volume peut être supérieur à 1Méga mot,
  - ou si son volume peut être supérieur à 10 fois le nombre de ddls du modèle.

- 4 - Ne pas utiliser les pointeurs (de nom ou de longueur) partagés entre plusieurs collections. Si (par exemple) 2 collections doivent être accédées par les mêmes noms, on peut faire :
  - créer un pointeur de noms,
  - créer les collections en accès "numéroté"
  - faire CALL JENONU avant l'accès aux collections.

- 5 - Ne pas stocker dans des OJB des adresses mémoire d'autres OJB (car une adresse mémoire est par définition "temporaire"). On le fait pour le type de Structures de Données mater_code et pour des raisons bien identifiées de performance, mais cela doit rester exceptionnel.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)