Annexe C : BNF du langage VODEL-D

Définitions préliminaires

Dans ce document nous utilisons les conventions suivantes :

1) les majuscules et minuscules sont équivalentes en VODEL-D ;
2) pour améliorer la lecture de cette BNF, les mots clefs du langage sont en **MAJUSCULE GRAS** ;
3) les lignes de commentaire commencent par les caractères "--".

Nombres

nombre	::=	car_num <nombre> | car_num
car_num	::=	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Identificateurs

identificateur	::=	alpha_car {car_num | _ | car_alpha}
car_alpha	::=	a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u |
		v | w | x | y | z | A | B | C | D | E | F | G | H | I | J | K | L | M |N |
		O | P | Q | R | S | T | U | V | W | X | Y | Z

Définition d'une classe

Trois éléments de base composent les classes de VODEL-D : le type de la classe, son caractère (abstraite ou concrète) et les dépendances de la classe.

Lors de l'écriture d'une classe le premier champ définit le *caractère* de la classe.

TypeCl	::=	**ABSTRAITE**	-- champ spécifiant une classe abstraite
		CONCRETE	-- champ spécifiant une classe concrète

Le type de l'entité, modélisé par une classe, se retrouve dans la définition de la classe par le champ type_ENT.

Type_ENT	::=	**ENT_PAS_PHY**	-- entité passive physique
		ENT_PAS_LOG	-- entité passive logique
		ENT_ACT	-- entité active

Toute classe dans VODEL-D est typée. Le type permet de regrouper les différents objets. Tout objet appartient à un groupe, composé de tous les objets de même type. Comme un objet possède un type unique, celui-ci n'appartient qu'a un seul groupe. Le type de la classe doit être prédéfini au moyen d'un identifiant de classe (*Identifiant_classe*). En reprenant les définitions du caractère et de la dépendance, toutes les classes de VODEL-D sont de la forme :

Nom_classe	::=	TypeCl Type_ENT Identifiant_classe identifiant_objet
		DEPEND Liste_identifiant ;
		FIN identifiant_classe ;

Entités passives physiques

Les entités passives physiques modélisent des ressources physiques disponibles sur les sites d'exécution des applications réparties ou parallèles.

la classe machine

La classe modélisant les *machines* peut être soit concrète soit abstraite. Une classe machine *concrète* modélise une machine physique. Deux attributs ont été définis pour cette classe : la vitesse de communication entre deux processus (communication locale) et la vitesse d'accès disque (bus disque). Les machines décrites par une classe *abstraite* sont des machines virtuelles et ne correspondent pas à des machines réelles. Elles existent pour donner un lien d'attachement aux machines concrètes ou virtuelles et ainsi former des groupes de machines. Les machines virtuelles permettent également de modéliser les liens de communication entre les machines. Le champ vitesse de communication présent dans la classe, définit dans ce cas la vitesse de communication entre les classes filles.

ProprieteMachine ::= **VITESSE_ACCES_DISQUE** [nombre | **INDEFINI**]
VITESSE_COMMUNICATION [nombre | **INDEFINI**]

Machine_description ::= TypeCl **ENT_PAS_PHY** ClassMach id_Mach
DEPEND idMach{, idMach}
ProprieteMachine
FIN ClassMach

La classe CPU

La classe CPU est un raffinement de la classe machine. Elle définit les caractéristiques d'un processeur. Les propriétés prédéfinies correspondent aux informations servant aux systèmes sous-jacents. On peut ainsi retrouver la puissance du processeur en MIPS, sa faculté à répondre aux exigences des calculs scientifiques, ainsi que le temps du changement de contexte. Le CPU est une entité passive physique. Son critère de dépendance vis à vis d'une autre entité correspond à sa localisation, donc à un identifiant machine.

ProprieteCPU ::= **FAMILLE** identificateur -- Intel, SParc, etc.
TYPE [identificateur | **INDEFINI**] -- PENTIUM 100 ou SPARC2
MIPS [nombre | **INDEFINI**] -- vitesse processeur
TAILLE_CACHE [nombre | **INDEFINI**] -- en KiloOctet
FPU [**OUI** | **NON** | **INDEFINI**] -- unité de calcul flottant ?
TPS_CONTEXT [nombre | **INDEFINI**] -- vitesse de commutation

CPU_description ::= TypeCl **ENT_PAS_PHY** ClassCPU idCPU
DEPEND idMach {, idMach}
ProprieteCPU
FIN ClassCPU

La classe Disque

La classe disque définit les unités de stockage accessibles sur le réseau. Le champ temps d'accès est en milliseconde. La classe disque peut dépendre d'une classe machine virtuelle ou concrète. Dépendre d'une classe machine virtuelle signifie que le disque est monté virtuellement sur tous les classes machines concrètes dépendant de la classe machine virtuelle. Dans le cas inverse, cette dépendance indique que le disque est monté localement et que seule la machine concrète père peut accéder aux données présentes sur

le disque. Les données d'un disque sont regroupées dans un objet persistant modélisant les entités passives logiques.

```
ProprieteDisque     ::=  TAILLE [nombre | INDEFINI]          -- Taille disque en MegaOctet
                         TEMPS_ACCES [nombre  | INDEFINI]    -- Vitesse de transfert
                         BUS [identificateur | INDEFINI]     -- SCSI, IDE, etc.
Disque_description  ::=  TypeCl  ENT_PAS_PHY ClassDisque idDisque
                         DEPEND idMach {, idMach}
                         ProprieteDisque
                         FIN ClassDisque
```

La classe Mémoire Physique

Dans cette modélisation deux valeurs sont présentes : la taille mémoire et le temps d'accès à la mémoire. Cet objet est du type entité passive physique.

```
ProprieteMem        ::=  TAILLE [nombre | INDEFINI]          -- Taille mémoire
                         TEMPS_ACCES [nombre  | INDEFINI]    -- Vitesse d'accès
Mem_description     ::=  TypeCl  ENT_PAS_PHY ClassMemPhy
                         DEPEND idMemPhy{,idMemPhy}
                         ProprieteMem
                         FIN ClassMemPhy
```

Entités passives logiques

Les entités passives logiques modélisent des ressources logiques telles que des données (fichiers ou code) ou de la mémoire virtuelle.

La classe Fichier

Le fichier est une entité passive logique. Le caractère abstrait ou concret de la classe fichier correspond à la production d'un fichier. Une entité de type fichier en attente de production, donc sans information de localisation (et donc sans dépendance envers une entité passive physique), sera déclarée de type abstrait. Lorsque le fichier est créé, la classe est remplacée par une classe concrète. Les différents attributs de la classe sont : le nom du fichier, le mode d'accès au fichier et la localisation complète du fichier.

```
Identifiant         ::=  nombre | identificateur
Chemin              ::=  {/}identificateur{/identificateur}-- localisation du fichier sur le disque
TypeMode            ::=  L  |   -- accès en mode écriture
                         E  |   -- accès en mode lecture
                         P      -- accès en mode partagé
                         T  |   -- tous les modes possibles
Fich_description    ::=  TypeCl  ENT_PAS_LOG ClassFichier idFic
                         DEPEND idEnt_Pass{, idEnt_Pass}
                         NOM_FICHIER Identifiant   -- identifiant logique du  fichier
                         CHEMIN Chemin             -- identifiant logique de localisation
                         MODE TypeMode             -- mode d'accès possible du fichier
                         FIN ClassFichier
```

La classe Mémoire Logique

La mémoire logique est équivalente à de la mémoire virtuelle. Un objet particulier nommé *root* définit les besoins en mémoire du système. Ces besoins caractérisent la mémoire virtuelle maximale du système. Les objets héritant de cette classe permettent de modéliser la mémoire nécessaire (champ *capacité*) pour le bon fonctionnement des processus de l'application parallèle.

```
Memory_description ::=  TypeCl  ENT_PAS_LOG ClassMemL idMem
                        DEPEND idMem{, idMemPhy}
                        CAPACITÉ [nombre  | INDEFINI]      -- taille mémoire
                        FIN ClassMemL
```

La classe Code

L'objet associé au code donne le nom de l'application et le chemin localisant le code de celle-ci, en donnant l'identifiant d'un objet de type fichier. Une application recompilée sur plusieurs machines possède plusieurs codes disponibles attachés à un objet CPU distinct. Ainsi, un objet modélisant le processus pourra posséder plusieurs dépendances envers des objets de type code.

```
Identifiant        ::=  nombre | identificateur
Code_description   ::=  TypeCl ENT_PAS_LOG ClassCode idCode
                        DEPEND idMachine {, idMachine}
                        CHEMIN Identifiant    -- identifiant VODEL-D d'un objet fichier
                        NOM_APP Identifiant   -- identifiant logique
                        FIN ClassCODE
```

Les entités actives

Les différents champs composant la classe entité active sont :

1) les descripteurs (point entrant ou sortant de communication) de l'entité ;
2) les identifiants référençant des entités de la classe mémoire, qui définissent les besoins en mémoire de l'entité active ;
3) les identifiants référençant les entités de la classe code, qui permettent d'identifier sur quel type de machine l'entité active peut être exécutée.
4) des identifiants référençant d'autres entités actives, pour modéliser la communication de groupe.

```
idDescripteur      ::=  nombre | identificateur
Ent_Act_description ::= TypeCl  ENT_ACT ClassE_A idE_A
                        DEPEND idE_A{, idE_A}
                        IN idDescripteur {, idDescripteur}          -- descripteur entrant
                        OUT idDescripteur {, idDescripteur}         -- descripteur sortant
                        MEMOIRE [idMem{, idMem} | INDEFINI ]        -- entité mémoire
                        CODE idCode{, idCode}                       -- entité code
                        SYNCHRO  [MONITEUR | RENDEZ_VOUS] -- synchronisation
                        FIN ClassE_A
```

Liaisons entre entités

Les liens entre entités sont basés sur une classe unique représentant un canal de communication.

La classe canal

Comme pour la classe entité active, les canaux possèdent des descripteurs présentés ci-dessous :

```
idDescripteur       ::=  nombre | identificateur
proprieteCom        ::=  SYNCHRONE |          -- mode synchrone
                         ASYNCHRONE |         -- asynchrone
                         GENERAL |            -- quelquonque
                         RPC                  -- rpc
proprieteAlign      ::=  [OUI | NON]          -- adaptation des données si machine hétérogène
proprieteGestion    ::=  FIFO |               -- liste FIFO
                         LIFO |               -- liste LIFO
                         RAND                 -- ordonnancement quelquonque
proprieteEvolution  ::=  PARTAGEABLE |       -- canal partageable
                         MIGRABLE |           -- migrable
                         DUPLICABLE [AVEC COHERENCE | SANS COHERENCE]
                         AUCUNE
proprieteEP         ::=  CREATION |          -- création d'un fichier par exemple
                         MODIFICATION         -- accès en modification
                         LECTURE              -- accès en lecture uniquement
                         AUCUN
Canal_description   ::=  TypeCl ClassCanal idCanal
                         DEPEND idCanal{, idCanal}
                         IN idDescripteur {, idDescripteur}        -- Descripteur entrant
                         OUT idDescripteur {, idDescripteur}       -- Descripteur sortant
                         CAPACITE [nombre | INDEFINI]              -- Capacité du canal
                         TYPE_COM proprieteCom                     -- propriété de la communication
                         ADAPTE proprieteAlign                     -- alignement des données
                         GESTION proprieteGestion                  -- ordonnancement des messages
                         EVOLUTION proprieteEvolution              -- évolution duarnt l'exécution
                         ACCES_EP proprieteEP                      -- mode d'accès aux entités passives
                         FIN ClassCanal
```

Description d'un DVSL

Un DVSL (*Distributed Virtual Software Link*) définit le lien virtuel entre deux classes entités actives ou une classe entité active et une classe entité passive via un canal :

1) Identification de l'entité active 1 (EL1) ;
2) Identification de la classe canal ;
3) Identification de l'entité active 2 (EL2) ;
4) Identification de la direction de la communication (de EL1 vers EL2, de EL2 vers

EL1, dans les deux sens à la fois) ;

5) le poids associé au DVSC.

```
DVSL_description ::= TypeCl DVSLClass idDVSL
                     EL1 ELD_nom                          -- entité active 1
                     LIEN idCanal                         -- canal
                     EL2 ELD_nom                          -- entité active 2
                     DIRECTION [EL12 | EL21 | BI]         -- sens de la communication
                     POIDS  [nombre | INFINI | INDEFINI]
                     FIN DVSLClass
```

Description d'un DVGL

Les DVGL (*Distributed Virtual Group of Links*) regroupent des DVSL et des DVGL. La définition d'un DVGL est donnée ci-dessous.

```
DVGL_description ::= TypeCl  DVGLClass idDVGL
                     IMPORTE idDVSL{, idDVSL}             -- DVSL intégrés
                     CONTIENT idDVGL {, idDVGL}           -- DVGL intégrés
                     DIRECTION [EL12 | EL21 | BI]         -- sens de la communication
                     POIDS  [nombre | INFINI | INDEFINI]
                     FIN DVGLClass
```

Description des entités spécifiques à VODEL-D

Nous précisons maintenant les éléments spécifiques de VODEL-D.

La classe Initialisation

L'initialisation d'entités actives est réalisée au moyen de l'objet initialisation. Celui-ci est spécifique à l'application utilisant VODEL-D.

```
Init_description ::= TypeCl ENT_PAS_LOG ClassInit idInit
                     DEPEND idInit{, idInit}        -- on construit un  graphe d'initialisation
                     ARGUMENTS identificateur       -- liste d'arguments
                     FIN ClassInit
```

Entités logicielles de définition

Une entité logicielle de définition est composée d'un ensemble d'instances de la classe entité active et d'entités logicielles. Les attributs que nous avons définis sont les suivants :

- **type** : un composant logiciel est *actif* s'il contient au moins un processus, sinon il est *passif* ;
- **évolution** : un composant logiciel est *libre* s'il peut être déplacé, sinon il est *fixe* ;
- **hiérarchie** : un composant logiciel peut être *primitif* ou *hiérarchique*.
- **regroupement** : un composant logiciel est *dissociable* si on peut le migrer par partie sinon, il est *indissociable* ;
- **répartition** : un composant logiciel est *réplicable*, *migrable* ou *centralisé*.
- **attache et repousse** : deux composants logiciels supposés communiquer intensivement doivent rester proches pour éviter la dégradation des performances. On définit alors un lien d'*attraction* entre eux. On peut aussi définir un lien de *répulsion*

qui provoque l'effet inverse.

```
proprieteEL        ::=  TYPE [ACTIF | PASSIF]
                        EVOLUTION  [LIBRE | FIXE]
                        REGROUPE  [DISSOCIABLE | INDISSOCIABLE]
                        REPARTITION  [MIGRABLE | REPLICABLE | NON]
Instance           ::=  (idE_A, [id_DVSL {, id_DVSL}], Id_init)-- identifiants des instances
ELD_description    ::=  EL DEFINITION ELD_nom
                        proprieteEL
                        ATTACHE [ELD_nom  {, ELD_nom} | NON]      -- attraction
                        REPOUSSE [ELD_nom  {, ELD_nom} | NON]     -- répulsion
                        CONTIENT [ELD_descritption | NON]         -- hiérarchie
                        IMPORTE [Instance {, Instance} | NON]     -- autres entités
                        FIN EL
```

Entités logicielles d'exécution

Les entités logicielles d'exécution sont composées des instances de la classe entité active et donc de processus. Chaque processus appartient à un seul groupe de même niveau. Un processus peut migrer dynamiquement d'un groupe à un autre. Le fait d'appartenir à un groupe signifie que l'entité active réalise des communications importantes avec les autres entités actives de ce groupe. Un groupe possède un *indice de liaison*. Cet indice correspond au coût de la communication entre entités logicielles. Si cet indice est égal à l'infini cela revient dans le langage VODEL-D à l'attachement d'entités actives. Nous avons aussi défini un *indice de charge* qui correspond à la charge totale du groupe et de ses sous-groupes. Le champ *placement* est utilisé pour indiquer la machine ou le processeur sur lequel l'entité logicielle sera placée. Enfin, le champ *modif_eld* indique si une modification a eu lieu lors de l'exécution et le champ *raison*, pourquoi elle a eu lieu (équilibrage de charge ou d'application).

```
proprieteEL        ::=  EVOLUTION  [LIBRE | FIXE]
                        REGROUPE  [DISSOCIABLE | INDISSOCIABLE]
                        REPARTITION  [MIGRABLE | REPLICABLE | NON]
Instance           ::=  (idE_A, [id_DVSL {, id_DVSL}], Id_init)-- identifiants des instances
ELE_description    ::=  EL EXECUTION ELE_nom
                        GROUPE nombre                  -- numéro du groupe
                        CHARGE nombre                  -- indice de charge
                        LIAISON nombre                 -- indice de liaison
                        PLACEMENT [idCPU | idMachine]  -- processeur ou machine
                        MODIF_ELD  [OUI RAISON [EQ_CHARGE
                                    | EQ_APPLICATION] | NON]
                        proprieteEL
                        ATTACHE [ELE_nom  {, ELE_nom} | NON]      -- attraction
                        REPOUSSE [ELE_nom  {, ELE_nom} | NON]     -- répulsion
                        CONTIENT [ELE_nom | NON]                  -- hiérarchie
                        IMPORTE [Instance {, Instance} | NON]     -- autres entités
                        FIN EL
```

Plan d’exécution

Le plan correspond à un ensemble d’entités logicielles d’exécution. Celui-ci a pour but de faciliter l'espionnage d'application et de déterminer les processus à exécuter à l'instant T.

Plan_description	::=	**PLAN** idIPlan	-- Identifiant logique VODEL-D
		SUIVANTS {IdPlan {, IdPlan}}	-- indique le(s) plan(s) suivant(s)
		ENTITES idELE{, idELE}	-- Identifiant d’entités log. d’exécution
		CONTRAT String	-- Contrat d’exécution
		FIN PLAN	