Type : **SPEC**	Projet RNTL **MORSE**	Réf : **SPEC-UML-LfP**
Date : **21/06/2004**		Sous-projet n° :
Version : **1.0**		Tâche n° :
Auteurs : **Maurice Assouline**		
Relecteurs :		

Représentation en UML des éléments de modélisation LfP

Liste des révisions

	Paragraphe	Commentaire
Version : 1.0 *Date* : 21/06/04 *Auteur* : Maurice Assouline		Version initiale

Table des matières

1 Introduction

1.1 Objet de ce document

Ce document décrit pour le projet MORSE la correspondance entre les concepts et les éléments de modélisation du langage LfP, et ceux d'UML.

Son objectif principal est de spécifier un ensemble de règles de correspondance qui pourront être implémentées par des transformations automatiques de modèles (de UML vers LfP), réalisées au moyen de l'outil Ameos/ACD d'Aonix (méta-générateur de code).

Des modèles UML pourront ainsi être automatiquement transformés en modèles LfP.

1.2 Conventions

Les règles spécifiées dans ce document supposent que les modèles UML sont corrects au regard de la syntaxe et de la sémantique UML, et qu'ils sont conformes aux directives de modélisation spécifiées dans le «Profil UML pour MORSE» [2]. Ce profil définit d'une part des restrictions au langage UML, d'autre part des extensions et des éléments particuliers de sémantique, au moyen de stéréotypes, étiquettes (tagged values), contraintes, et annotations.

(ce profil, en cours de définition, sera décrit dans le document [2])

Dans la suite de ce document, le terme « UML » sera utilisé pour désigner le langage UML spécifique défini par le Profil UML pour MORSE.

1.3 Organisation de ce document

Ce document classe les règles de correspondance par concept LfP : un paragraphe est consacré à chaque concept LfP, et à la représentation correspondante en UML.

Les concepts LfP sont regroupés en 2 grandes sections : Diagrammes d'architecture et Diagrammes de comportement.

Puis des exemples sont présentés, pour illustrer la représentation en UML de modèles élémentaires en LfP.

1.4 Documents de référence

[1] OMG. UML 2.0 Superstructure Specification, 18/05/2004.

[2] Maurice Assouline. Profil UML pour MORSE (*document en cours de rédaction*).

[3] Frédéric Gilliers. BNF de la grammaire des attributs des diagrammes L*f*P, 12/06/2003.

[4] Frédéric Gilliers. Description de la sémantique du langage L*f*P, 12/06/2003.

[5] Projet MORSE, Annexe Technique.

2 Diagrammes d'architecture LfP

Un diagramme d'architecture LfP sera représenté par un *diagramme de composants* UML.

Remarque : le terme « d*iagramme de composants* » (*Component Diagram*) a été redéfini en UML2.0, il n'a plus la même signification qu'en UML 1.x. C'est une sorte de diagramme de classes, où sont introduites les notions de *composants*, *ports*, *interfaces*, *connecteurs*, *parties* d'une classe.

En UML2.0, *Component* est un sous-type de *Class*.

Les principales entités LfP que l'on trouve dans les diagrammes d'architecture, et que nous devrons représenter en UML sont :

- Les Classes,
- Les Médias,
- Les Binders,
- Les Ports,
- Les Arcs de liaison.
- Les Déclarations (attributs textuels)

2.1 Représentation d'une Classe LfP

Une Classe LfP sera représentée par un *composant* UML (*component*), muni du stéréotype <<lfp_class>>.

Particularités sémantiques attachées au stéréotype <<lfp_class>> :

- Un composant <<lfp_class>> possède une exécution active, il est doté de sa propre source d'animation (« thread of control »).
- Un composant <<lfp_class>> a une exécution séquentielle (sans parallélisme interne) : aucun de ses sous-composants ne doit être actif.
- Le stéréotype <<lfp_class>> est défini comme une spécialisation du stéréotype <<component>>. Cela implique que les composants <<lfp_class>> possèdent les caractéristiques des *composants* UML. Ces derniers sont définis comme des parties modulaires d'un système, qui encapsulent leur contenu, et possèdent un ensemble de ports qui sont leurs points d'interaction avec d'autres composants, à travers lesquels ils offrent des services (*interface offerte*) et en sollicitent (*interface requise*).
- A travers ses ports, un composant <<lfp_class>> ne peut pas être connecté à d'autres composants <<lfp_class>>, il ne peut être connecté qu'à des composants <<lfp_media>> (voir paragraphe ci-dessous). Il possède des ports conjugués de ceux des composants <<lfp_media>> auxquels il est connecté.

Notation :

Au titre de sa qualité de classe active, un composant <<lfp_class>> est représenté par un rectangle avec les côtés gauche et droit en traits doubles. (Voir Figure 1 ci-dessous).

Le stéréotype <<lfp_class>> étant une spécialisation du stéréotype <<component>>, il est inutile de surcharger le dessin du composant en y insérant les 2 stéréotypes : le stéréotype <<lfp_class>> suffit. D'autre part, l'omission des traits doubles sur les côtés du rectangle peut éventuellement être admise, car la présence du stéréotype <<lfp_class>> implique qu'il s'agit d'une classe active.

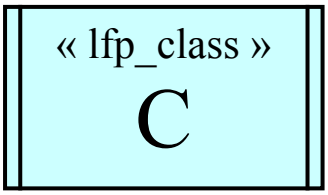

Figure 1 : Représentation d'une Classe LfP

2.2 Représentation d'un média LfP

Un Média LfP sera représenté par un *composant* UML (*component*), muni du stéréotype <<lfp_media>>.

Particularités sémantiques attachées au stéréotype <<lfp_media>> :

- Un composant <<lfp_media>> possède une exécution active, il est doté de sa propre source d'animation (« thread of control »).
- Un composant <<lfp_media>> a une exécution séquentielle (sans parallélisme interne) : aucun de ses sous-composants ne doit être actif.
- Le stéréotype <<lfp_media>> est défini comme une spécialisation du stéréotype <<component>>. Cela implique que les composants <<lfp_media>> possèdent les caractéristiques des *composants* UML.
- Un composant <<lfp_media>> ne possède pas de méthodes. Il peut posséder des attributs (un état).
- A travers ses ports, un composant <<lfp_media>> ne peut pas être connecté à d'autres composants <<lfp_media>>, il ne peut être connecté qu'à des composants <<lfp_class>>. Il possède des ports conjugués de ceux des composants <<lfp_class>> auxquels il est connecté.
- Un protocole, décrit sous forme de diagramme d'état, est associé à tout composant <<lfp_media>> (voir section "Diagrammes de comportement LfP" ci-dessous).

Notation :

Au titre de sa qualité de classe active, un composant <<lfp_media>> est représenté par un rectangle avec les côtés gauche et droit en traits doubles. (Voir Figure 2 ci-dessous).

Le stéréotype <<lfp_media>> étant une spécialisation du stéréotype <<component>>, il est inutile de surcharger le dessin du composant en y insérant les 2 stéréotypes : le stéréotype <<lfp_media>> suffit. D'autre part, l'omission des traits doubles sur les côtés du rectangle peut éventuellement être admise, car la présence du stéréotype <<lfp_media>> implique qu'il s'agit d'une classe active.

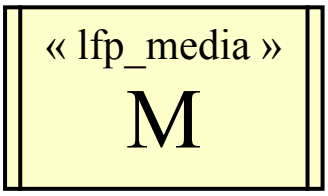

Figure 2 : Représentation d'un média LfP

2.3 Représentation des Binders et des Ports de LfP

Un Binder LfP sera représenté en UML par un connecteur d'assemblage (*assembly connector*) entre un composant <<lfp_class>> et un composant <<lfp_media>> (voir Figure 3 ci-dessous).

Un stéréotype <<lfp_binder>> facultatif peut être associé au connecteur d'assemblage. Si ce stéréotype n'est pas indiqué il est considéré comme existant implicitement.

> En UML, un connecteur d'assemblage (*assembly connector*) est un connecteur entre 2 composants qui exprime que l'un de ces deux composants fournit les services que l'autre demande (ou reçoit les messages que l'autre émet). Un connecteur d'assemblage relie donc une interface requise (ou le port correspondant) à une interface fournie (ou le port correspondant).
>
> Une interface fournie est symbolisée en UML par un petit cercle (une « boule ») relié au composant fournisseur par une ligne, une interface requise est symbolisée par un petit demi-cercle (« socket ») relié au composant demandeur par une ligne. Un connecteur d'assemblage est symbolisé par l'ensemble « boule enveloppée dans socket », relié du côté boule au composant fournisseur et du côté socket au composant demandeur.

Notation :

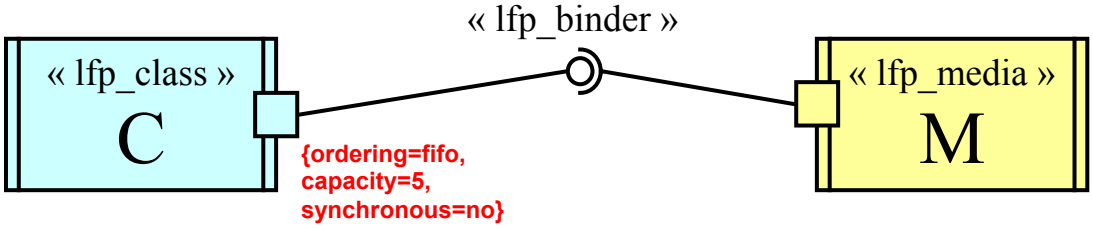

Figure 3 : Représentation d'un binder LfP

Le connecteur d'assemblage <<lfp_binder>> doit relier 2 ports UML, l'un appartenant au composant <<lfp_class>>, l'autre au composant <<lfp_media>>. Ces deux ports sont « conjugués », ce qui signifie que les messages et demandes émises par l'un correspondent aux messages et demandes reçues par l'autre.

L'orientation des arcs LfP est représentée par la socket et la boule (les demandes transitent de la socket vers la boule).

Pour représenter les attributs **ordering**, **capacity**, et **synchronous** des étiquettes (*tagged values*) **ordering**, **capacity**, et **synchronous** qualifiant le port du composant <<lfp_class>> doivent être spécifiées (les mêmes valeurs s'appliquent au port conjugué du composant <<lfp_media>>, sans qu'il soit nécessaire de dupliquer les étiquettes).

Pour la représentation de l'attribut **multiplicity** des binders LfP : voir paragraphes 2.3.1 et 2.3.2 ci-dessous.

Un port LfP correspondant à des demandes entrantes (ex. : port C.input pour une classe LfP nommée C) est représenté par un port UML relié à une interface fournie (symbole boule du connecteur).

Un port LfP correspondant à des demandes sortantes (ex. : port M.output pour un média LfP nommé M) est représenté par un port UML relié à une interface requise (symbole socket du connecteur).

L'interface fournie par un composant via un port est constituée de l'ensemble des méthodes offertes et des messages attendus en entrée du composant via ce port. L'interface requise correspondant au port conjugué n'a pas besoin d'être décrite, son contenu est considéré comme identique.

L'interface requise par un composant via un port est constituée de l'ensemble des méthodes d'autres composants appelées par le composant et des messages émis par le composant via ce port. L'interface fournie correspondant au port conjugué n'a pas besoin d'être décrite, son contenu est considéré comme identique.

Particularités sémantiques attachées au stéréotype <<lfp_binder>> :

- Les connecteurs d'assemblage <<lfp_binder>> représentent des buffers qui servent de support à la communication entre les composants <<lfp_class>> et les composants <<lfp_media>>.
- Dans un tel buffer, le composant émetteur écrit des messages, que le composant récepteur lit.
- Les messages pouvant être écrits dans le buffer par le composant émetteur sont ceux qui sont décrits dans son interface requise. Les messages pouvant être lus dans le buffer par le composant récepteur sont ceux qui sont décrits dans son interface fournie.
- Lorsque parmi les messages véhiculés certains supposent un message de retour, (comme dans le cas d'un message d'appel de fonction) le buffer est bidirectionnel (double).

2.3.1 Binder de multiplicité 1

En l'absence de stéréotype autre que <<lfp_binder>> qualifiant le connecteur d'assemblage, celui-ci représente un binder de multiplicité 1, ce qui signifie qu'il existe une instance du binder pour chaque instance de la classe LfP liée.

La Figure 3 ci-dessus indique donc qu'il existe un buffer de transmission de messages pour chaque instance du composant C.

Le déploiement au niveau instances du diagramme de la Figure 3 correspond donc à la Figure 4 ci-dessous.

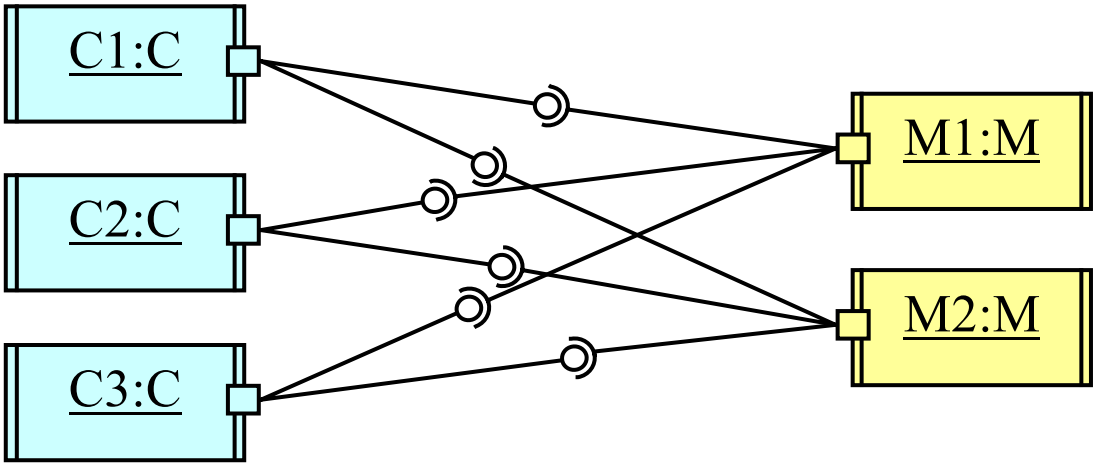

Figure 4 : Déploiement au niveau instances pour un binder de multiplicité 1

2.3.2 Binder de multiplicité all

Un binder LfP de multiplicité "all" sera représenté par un connecteur d'assemblage muni du stéréotype <<static_binder>> (en plus du stéréotype facultatif <<lfp_binder>>).

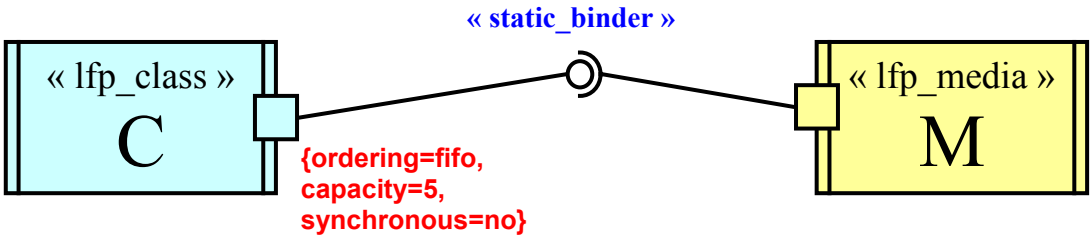

Figure 5 : Représentation d'un binder LfP de multiplicité "all"

2.4 Représentation des Arcs de liaison de LfP

2.5 Représentation des attributs déclaratifs de LfP

3 Diagrammes de comportement LfP

4 Exemples

4.1 Le modèle Client-Serveur