



**Aspect dynamique du système :
diagramme d'interaction (de
séquence ou de communication)**

Aspect dynamique d'un système : diagrammes d'interaction

- Pour documenter les cas d'utilisation par la description de leurs scénarios
- Pour montrer les interactions entre objets
 - **Diagramme de séquence**
 - **Diagramme de communication**



Les objets travaillent en synergie afin de réaliser les fonctions de l'application



**Aspect dynamique du système :
diagramme de séquence**

Diagramme de Séquence (DSQ)

Permet de décrire des interactions entre objets selon un point de vue temporel

Montre :

- les interactions entre les objets
- les messages ordonnancés dans le temps

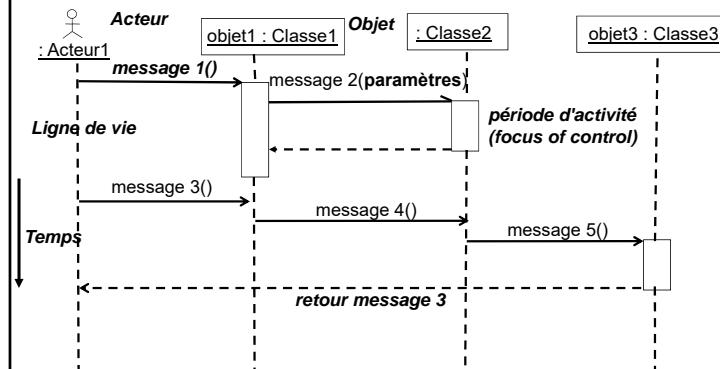
Ne montre pas :

- les associations entre objets

Représentation en deux dimensions :

- axe vertical : temps
- axe horizontal : objets qui interagissent dans la séquence

Concepts de base



82

UML - © Christine Bonnet

Concepts de base - suite

Ligne de vie : ensemble des opérations exécutées par un objet. Représentation de l'existence d'un objet

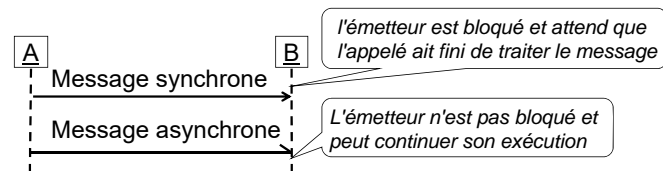
Période d'activité (focus of control) : temps pendant lequel un objet effectue une action (directement ou par l'intermédiaire d'un autre objet qui lui sert de sous-traitant)

Message : unité de communication entre objets
 → Reconstitution d'une fonction de l'application par la mise en collaboration d'un groupe d'objets
 La réception d'un message est un événement générateur d'une activité chez l'objet récepteur



83

UML - © Christine Bonnet

Catégories de message : signal (asynchrone) et appel d'une opération (synchrone)



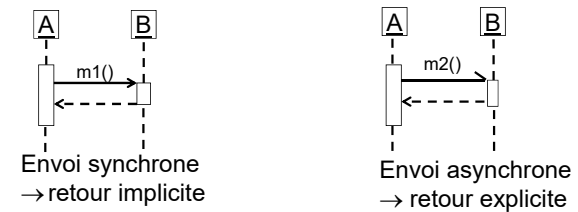
Autre notation (UML 2) :

Message synchrone : 
 Message asynchrone : 

84

UML - © Christine Bonnet

Activation et retour :

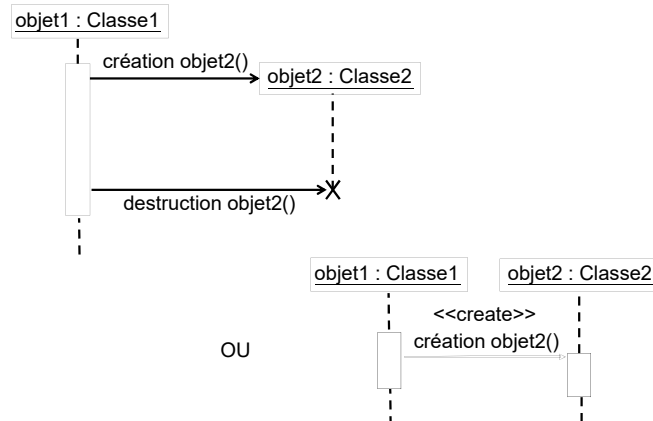


(N.B. : m1, m2 messages imbriqués)

85

UML - © Christine Bonnet

Messages de création et de destruction d'objet :



86

UML - © Christine Bonnet

Classes d'objets : stéréotypes de I. Jacobson (partie conception)

→ Modèle MVC



- **<<boundary>>** : classes qui servent à modéliser les interactions entre le système et ses acteurs (fenêtres, boîtes de dialogue, menus, ...)



- **<<control>>** : classes utilisées pour représenter la coordination (transfert d'informations), l'enchaînement et le contrôle d'autres objets

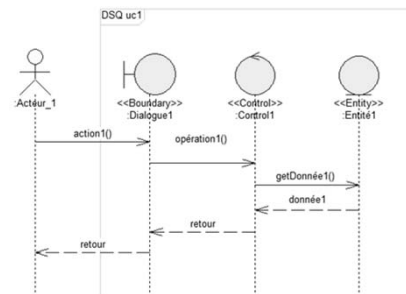


- **<<entity>>** : classes qui servent à modéliser des informations durables et souvent persistantes, ou tout type d'objet temporaire tel qu'un résultat de recherche

87

UML - © Christine Bonnet

Les 3 stéréotypes de Jacobson



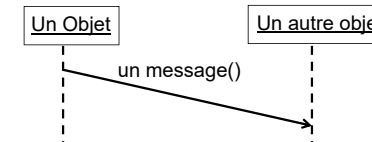
- Les acteurs ne peuvent interagir qu'avec les boundary
- Les boundary peuvent interagir avec les control ou exceptionnellement d'autres boundary
- Les control peuvent interagir avec les boundary, les entity, ou d'autres control
- Les entity ne peuvent interagir qu'entre elles

88

UML - © Christine Bonnet

Contraintes temporelles

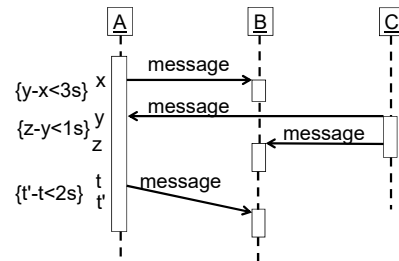
- **Délai de propagation :**



89

UML - © Christine Bonnet

- **Contraintes temporelles construites à partir de noms de transitions :**



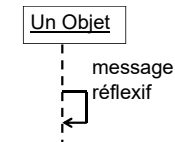
L'instant d'émission d'un message est une **transition**
Lorsque la propagation d'un message prend un temps significatif, les instants d'émission et de réception des messages sont matérialisés par un **couple (nom, nom')**

90

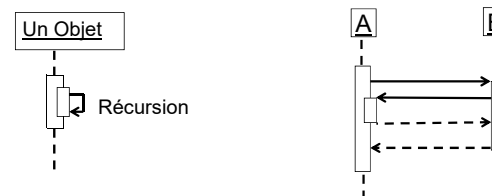
UML - © Christine Bonnet

Autres représentations

- **Message réflexif :** un objet s'envoie un message



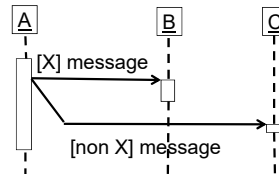
- **Message récursif, exécution simultanée :**



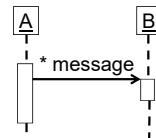
91

UML - © Christine Bonnet

- **Message conditionnel :**



- **Itération :** * [clause d'itération]



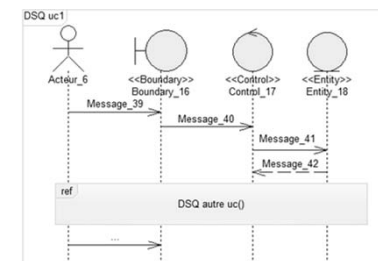
92

UML - © Christine Bonnet

Référence d'interaction

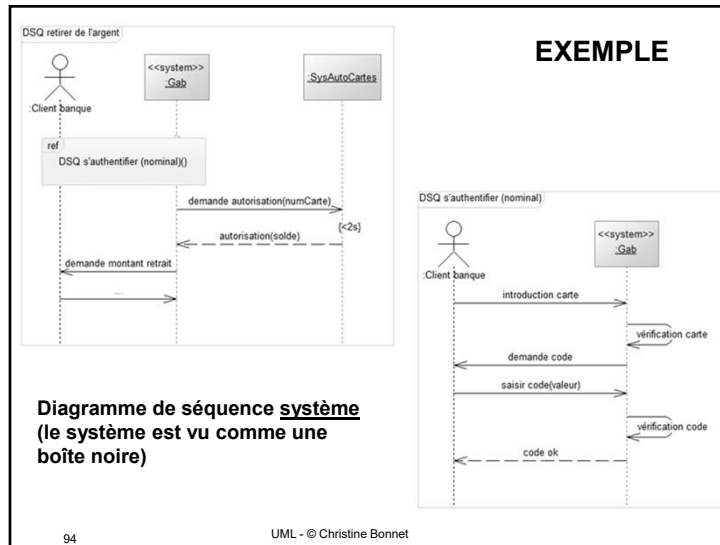
Référence à une interaction (autre diagramme de séquence) dans la définition d'une interaction
→ réutilisation d'une définition dans des contextes différents (relation <<include>> et <<extend>>)

Notation :



93

UML - © Christine Bonnet



Fragments (ou cadres) d'interactions

- Regroupement de **sous-ensemble d'interactions** (par exemple pour représenter plusieurs scénarios d'un cas d'utilisation)
- Un fragment est défini par un opérateur et des opérandes
- Les opérandes d'un opérateur sont séparés par une ligne pointillée
- Les conditions de choix des opérandes sont données par des expressions booléennes entre crochets [] appelées **conditions de garde**

95 UML - © Christine Bonnet

- Il existe **12 types de fragments prédéfinis** :
 - ✓ opérateurs de choix et de boucles : alternatives **alt**, option **opt**, Loop **loop**, Break **break**;
 - ✓ opérateurs contrôlant l'envoi en parallèle de messages : parallèle **par** et critical region **critical**;
 - ✓ opérateurs contrôlant l'envoi de messages : **ignore**, **consider**, assertion **assert**, négative **neg**;
 - ✓ opérateurs fixant l'ordre d'envoi des messages : weak sequencing **seq**, strict sequencing **strict**

NB : seuls les fragments soulignés seront détaillés par la suite

96 UML - © Christine Bonnet

L'opérateur alternatives (alt)

Instruction de test avec plusieurs alternatives (opérandes) possibles

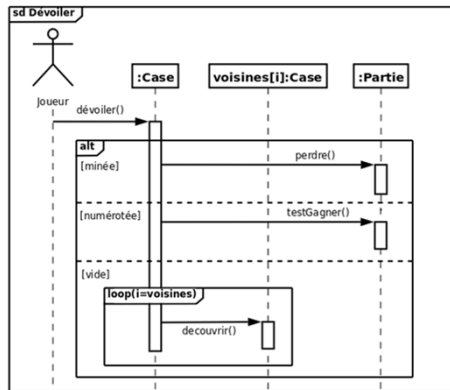
Chaque opérande détient une condition de garde ([]). Il est possible d'utiliser une condition "else"

Les opérandes sont séparés par une ligne pointillée

97 UML - © Christine Bonnet

Exemple

[<http://laurent-audibert.developpez.com/Cours-UML>]



Choix dans un diagramme de séquence

98

UML - © Christine Bonnet

L'opérateur Loop (loop)

Instruction de boucle

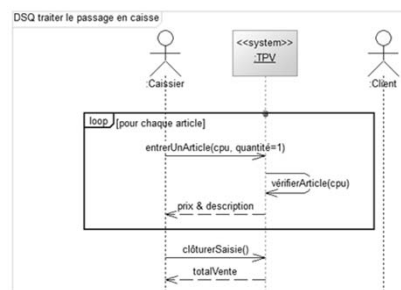
Possède une seule partie (opérande)

Il est possible de spécifier un nombre minimum et maximum de répétition ainsi qu'une condition de garde :
loop (minInt, maxInt) [condition]

99

UML - © Christine Bonnet

Exemple



100

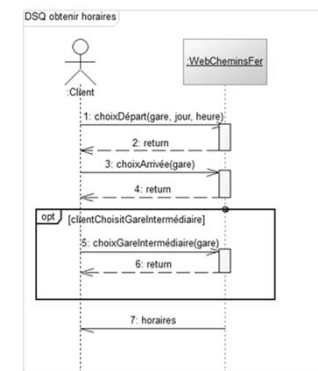
UML - © Christine Bonnet

L'opérateur option (opt)

Instruction de test sans alternative

Possède un opérande et une condition de garde associée

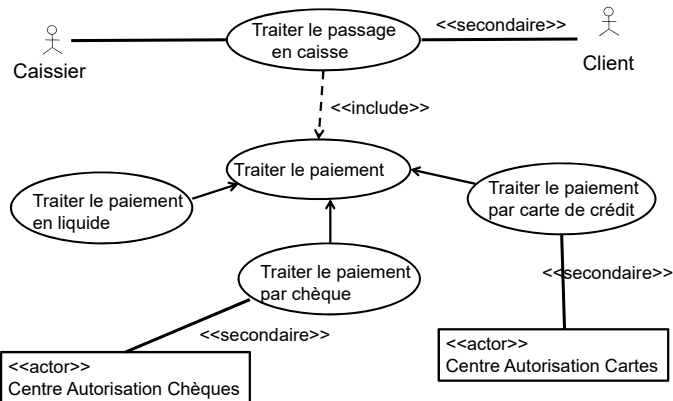
Exemple :



101

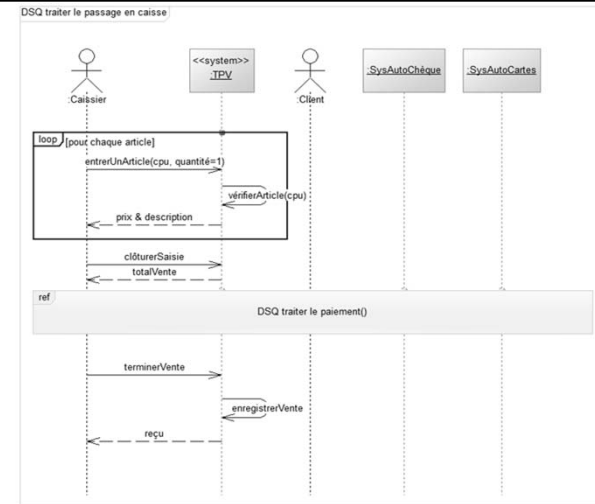
UML - © Christine Bonnet

Exemple de DSQ : retour sur l'exemple du Terminal de Point de Vente (TPV) [Pascal Roques, UML 2 par la pratique, Eyrolles]



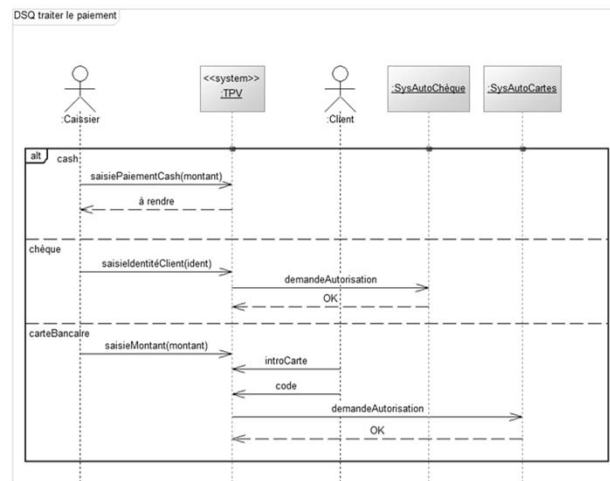
102

UML - © Christine Bonnet



103

UML - © Christine Bonnet



104

UML - © Christine Bonnet

Utilisation de notes

- Pour affiner la description : contexte d'utilisation du message, de l'objet, etc
- Pour repérer les scénarios alternatifs, les scénarios d'erreur ainsi que les conditions de poursuite du scénario nominal

105

UML - © Christine Bonnet

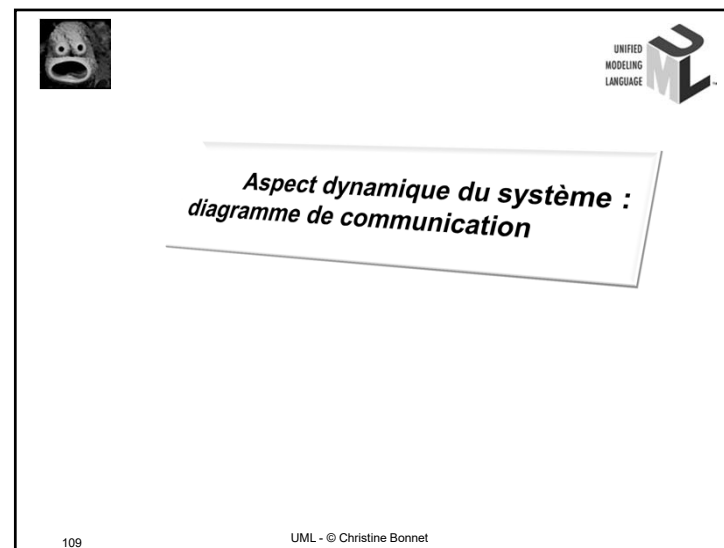
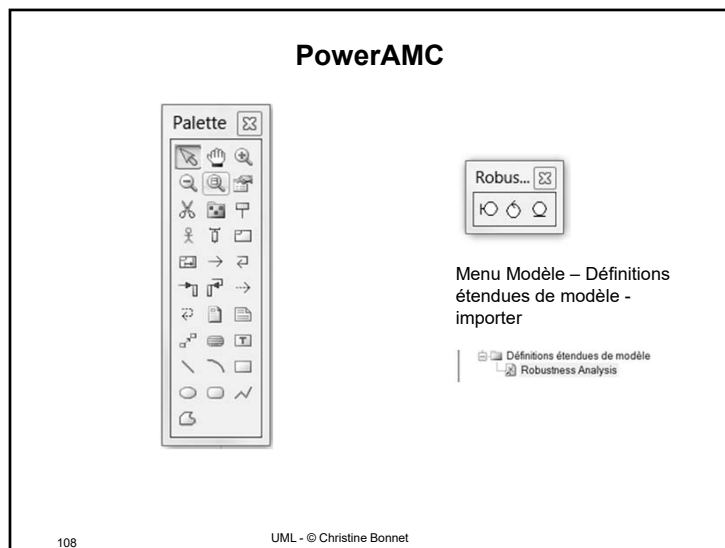
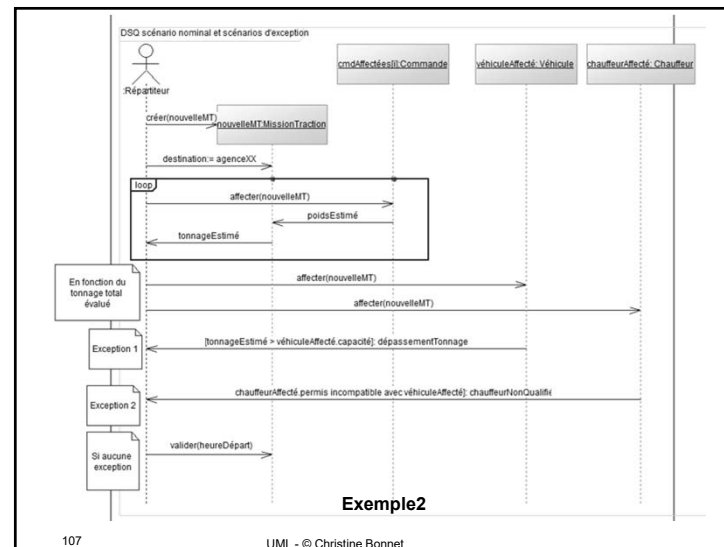
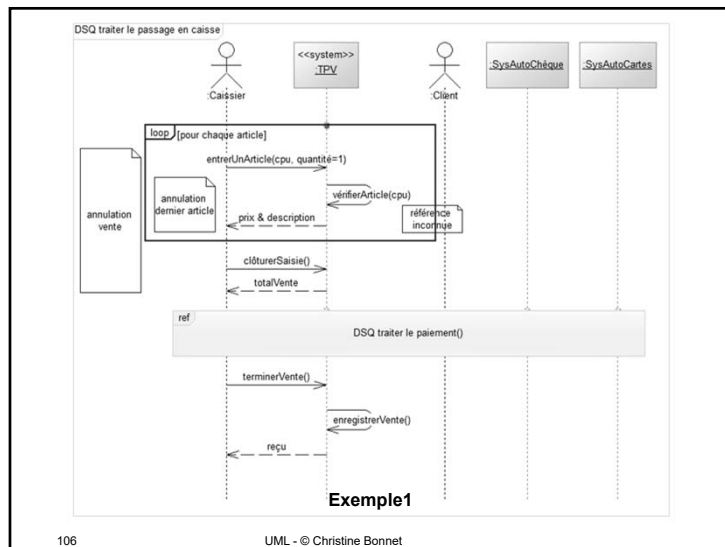


Diagramme de COmmunication (DCO)

Permet de décrire comment un ensemble d'objets collaborent pour répondre à une sollicitation. Représentation spatiale

Montre :

- les interactions entre les objets
- les relations entre les objets (classes)

Ne montre pas :

- la dimension temporelle

Représentation :

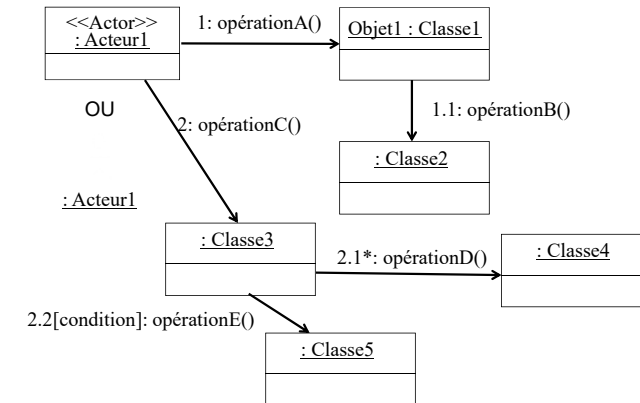
- objets et liens entre objets
- envois de messages

110

UML - © Christine Bonnet

DCO - suite

Notation :



111

UML - © Christine Bonnet

Messages

Un message regroupe :

- les flots de contrôle (description de la répartition de l'activité entre les objets)

Notation : \longrightarrow

- les flots de données (description des données qui transitent d'un objet vers un autre objet)

Notation : $\circ\longrightarrow$

112

UML - © Christine Bonnet

Les principales catégories de messages

- les constructeurs : créent des objets



Objets créés, puis détruits au sein de la même interaction :



- les destructeurs : détruisent des objets
- les sélecteurs : renvoient tout ou partie de l'état d'un objet
- les modificateurs : changent tout ou partie de l'état d'un objet
- les itérateurs : visitent l'état d'un objet ou le contenu d'une structure de données qui contient plusieurs objets



113

UML - © Christine Bonnet

Forme générale d'un message

synchronisation séquence ':' résultat ':= ' nom arguments

Synchronisation

synchronisation ::= rang {' synchronisation} '/'

Point de synchronisation d'un message = séquence d'envoi de message(s) terminée par /

Exemple : A.1, B.3 / message

→ Message envoyé lorsque les envois A.1 et B.3 ont été satisfaits

rang ::= [entier | nom de flot d'exécution] {' rang}

Ordre des envois

entier : rang de l'envoi de message au sein de l'emboîtement englobant

nom de flot : identifie un flot d'exécution parallèle au sein d'un emboîtement

Exemples : l'envoi 3.1.3 suit 3.1.2 au sein de l'emboîtement 3.1

l'envoi 3.1.a est effectué simultanément à l'envoi 3.1.b

114

UML - © Christine Bonnet

Forme générale d'un message (suite)

synchronisation séquence ':' résultat ':= ' nom arguments

◦ Séquence

séquence ::= rang [récurrence]

Indique le niveau d'emboîtement de l'envoi de message au sein de l'interaction (rang)

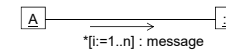
récurrence ::= '' [' clause d'itération ']' bloc*

récurrence ::= '[' clause de condition ']' bloc

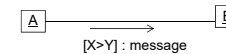
Indique l'itération et les branchements conditionnels

La clause d'itération est optionnelle ; elle est exprimée dans un format libre

L'envoi parallèle (appelée diffusion) est noté : * | |



La clause de condition valide ou non l'envoi des messages contenus dans le bloc ; elle est exprimée dans un format libre



115

UML - © Christine Bonnet

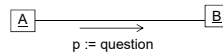
Forme générale d'un message (suite)

synchronisation séquence ':' résultat ':= ' nom arguments

◦ Résultat

Liste de valeurs retournées par le message. Le format de ce champ est libre. [paramètres des autres messages compris dans l'interaction]

Ce champ n'existe pas en l'absence de valeurs retournées



◦ **Nom** : nom du message ; il correspond à une opération définie dans la classe de l'objet récepteur

◦ Arguments

Liste des paramètres du message

Nom du message + arguments → action qui doit être déclenchée dans l'objet récepteur

116

UML - © Christine Bonnet

Exemples de la syntaxe d'envoi des messages

4: afficher (x, y) ← message simple

3.3.1: afficher (x, y) ← message imbriqué

4.2: âge := soustraire (aujourd'hui, dateDeNaissance) ← message imbriqué avec valeur retournée

6.2 [Age ≥ 18 ans] : voter () ← message conditionnel

4.a, 6.b / c.1: allumer (lampe) ← synchronisation avec d'autres flots d'exécution

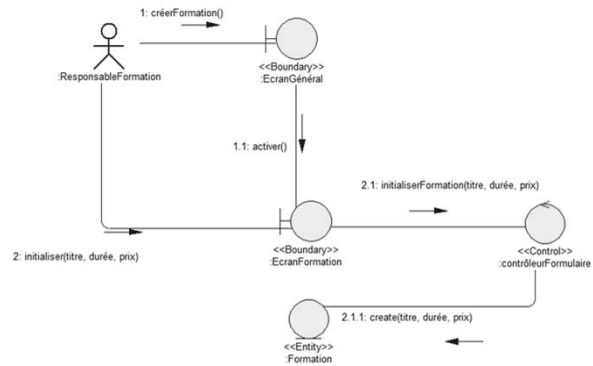
1 *: laver () ← itération

3.a, 3.b / 4 * | | [i := 1..n]: éteindre () ← itération parallèle

117

UML - © Christine Bonnet

Exemple 1 de DCO [Pascal Roques, UML 2 par la pratique, Eyrolles]



118

UML - © Christine Bonnet

Exemple 2 de DCO [<http://laurent-audibert.developpez.com/Cours-UML>]

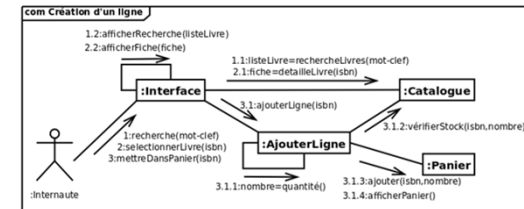
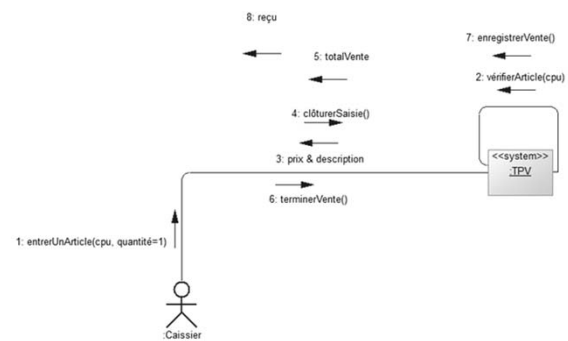


Diagramme de communication illustrant la recherche puis l'ajout, dans son panier virtuel, d'un livre lors d'une commande sur Internet.

119

UML - © Christine Bonnet

Transformation DSQ → DCO avec PowerAMC



120

UML - © Christine Bonnet