

UML

Diagramme de Séquence (DSEQ)

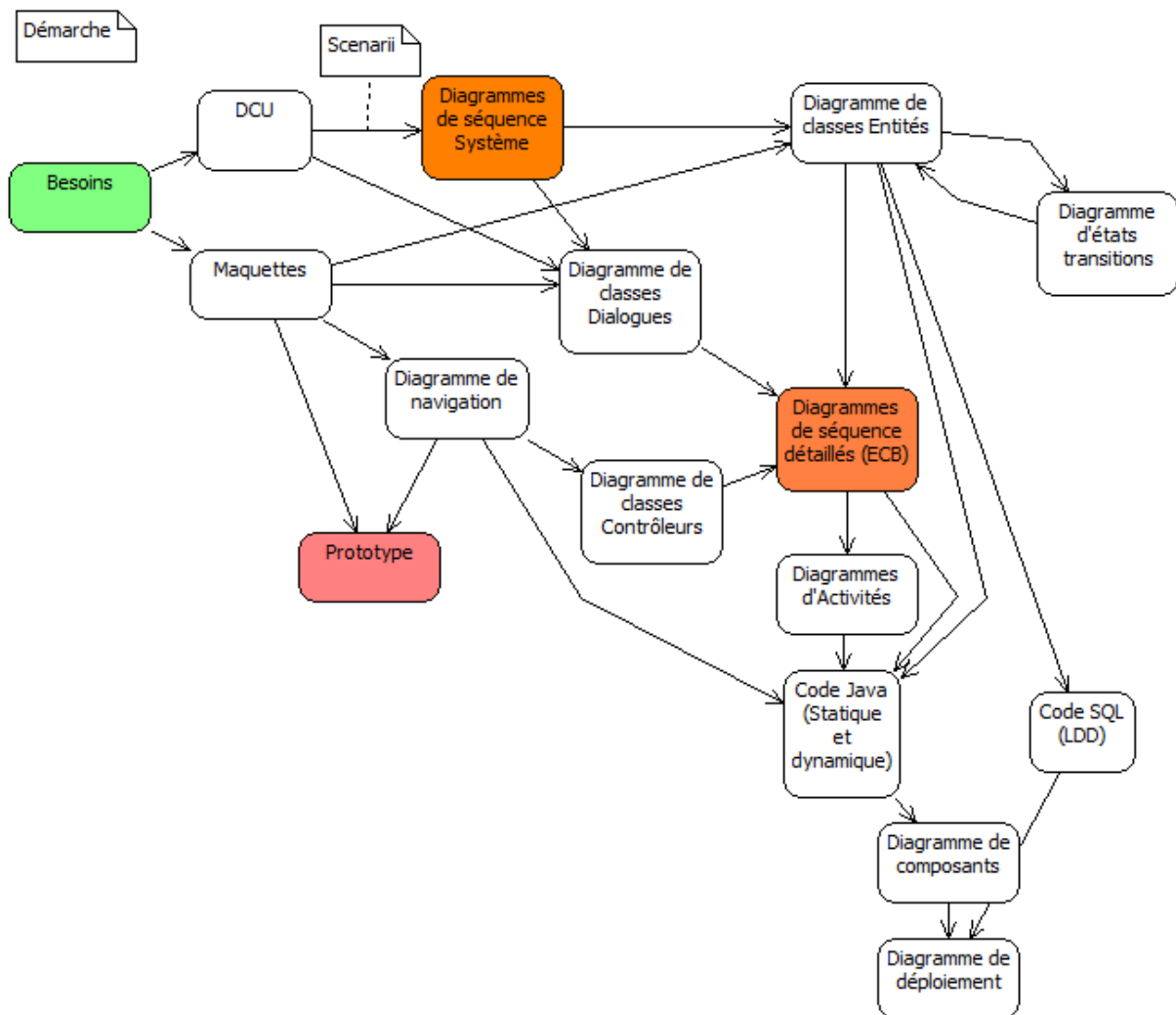


Table des matières

1.1 - Contexte.....	3
1.2 - Présentation.....	4
1.2.1 - Exemple : l'authentification.....	4
1.2.2 - Définition.....	5
1.2.3 - L'acteur, la classe ou l'objet.....	6
1.2.4 - Les messages de base.....	7
1.2.5 - L'activation.....	9
1.2.6 - La boucle locale.....	10
1.3 - Diagramme de Séquence Système.....	11
1.3.1 - Contexte.....	11
1.3.2 - Définition.....	12
1.3.3 - Exercices.....	13
1.4 - Diagramme de Séquence Détaillé.....	15
1.4.1 - Définition.....	15
1.4.2 - Représentations iconiques des acteurs et des classes stéréotypés.....	16
1.4.3 - Les messages CREATE et DESTROY.....	18
1.4.4 - Exemple de Diagramme de Séquence Détaillé.....	20
1.5 - Les fragments combinés.....	21
1.5.1 - Définition.....	21
1.5.2 - L'alternative (alt).....	22
1.5.3 - La boucle (loop).....	24
1.5.4 - La rupture (break).....	25
1.5.5 - La référence (ref).....	26
1.5.6 - DSS, Fragments combinés et CRUD Produit.....	28
1.6 - DSD Ajout Produit 4 couches !.....	29
1.6.1 - Exemple.....	29
1.6.2 - Exercice.....	29
1.7 - Mots-clés.....	30

1.1 - CONTEXTE

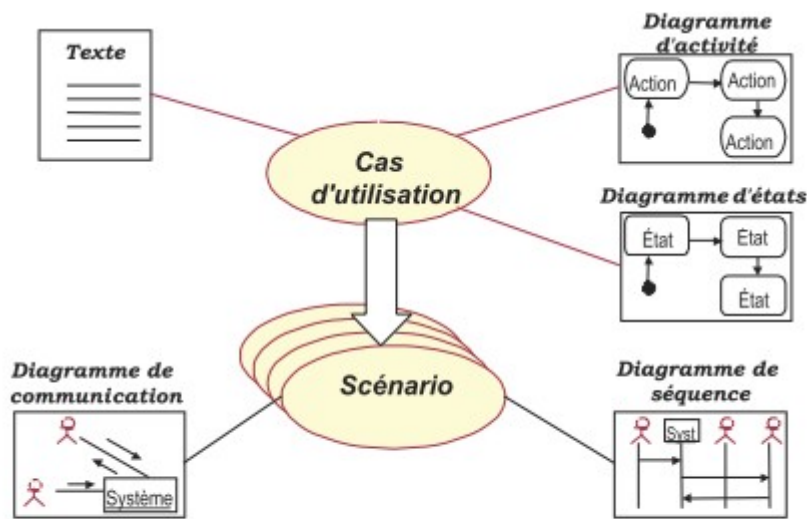


Figure 4-5 : Types de diagrammes dynamiques utilisables pour documenter les cas d'utilisation

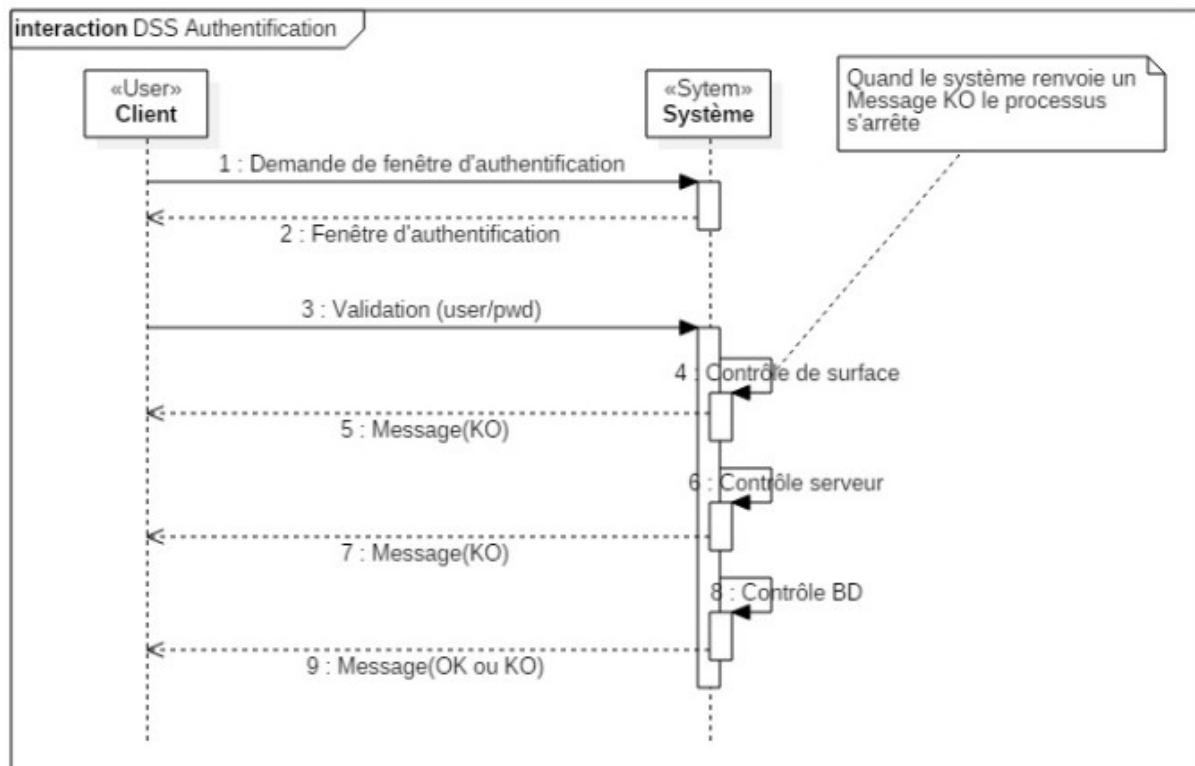
Figure empruntée à Pascal Roques « UML 2 en action », page 87 pour la version papier, et page 75 pour la version PDF.

Chaque scénario sera représenté par un Diagramme de Séquence Système (DSS). Mais il est possible pour éviter la multiplication des Diagrammes de Séquence de représenter le scénario nominal et les scénarii alternatifs avec un seul DSS. Il en sera de même pour le Diagramme de Séquence Détaillé.

1.2 - PRÉSENTATION

1.2.1 - Exemple : l'authentification

Diagramme de Séquence Système de l'authentification.

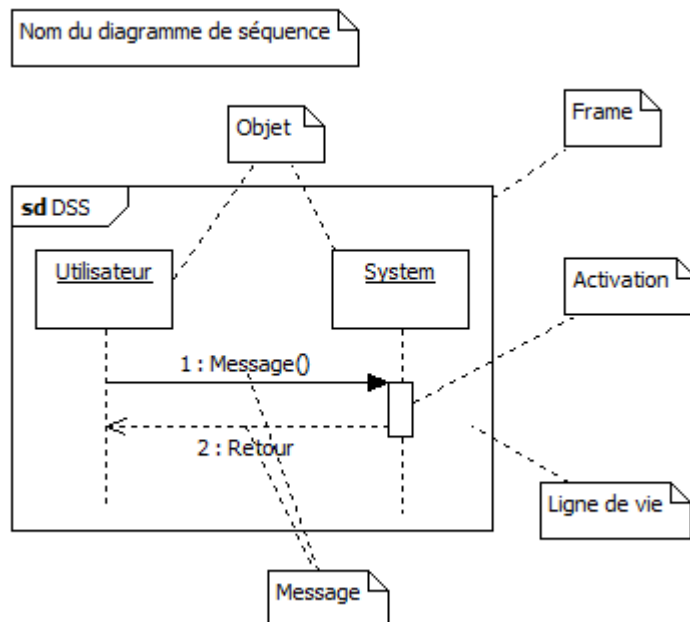


Key words !!!

Objet
 Stéréotype
 Ligne de vie
 Message synchrone
 Message de retour
 Message réflexif ou boucle locale (Self message)
 Paramètre(s)
 Activation

[message asynchrone, create, destroy]

1.2.2 - Définition



Le diagramme de séquence (DSEQ) permet de décrire et de formaliser les interactions entre les acteurs et le système ou les classes, par l'intermédiaire de messages, d'un point de vue chronologique et spatial. Le temps est représenté à la verticale et l'espace à l'horizontal.

Les DSEQ sont utilisés différemment, de façon plus ou moins détaillée, au cours du cycle de vie d'un projet.

Au début – en analyse - ils sont utilisés pour spécifier les cas d'utilisation (Diagrammes de Séquence Système où deux objets – utilisateur et système – sont représentés).

Un diagramme de séquence correspond à un seul scénario.

Par la suite – soit en analyse, soit en conception - , ils seront plus détaillés, et les messages correspondront aux messages des langages informatiques (Diagrammes de Séquence détaillés où n objets – utilisateur, dialogues, contrôleurs et entités - sont représentés).

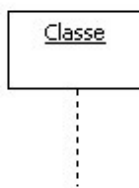
Le DSEQ utilise quatre concepts clés : l'acteur, la classe ou l'objet, l'activation, le message.

La structure de base est la structure séquentielle. Mais il est possible de représenter des structures alternatives et itératives.

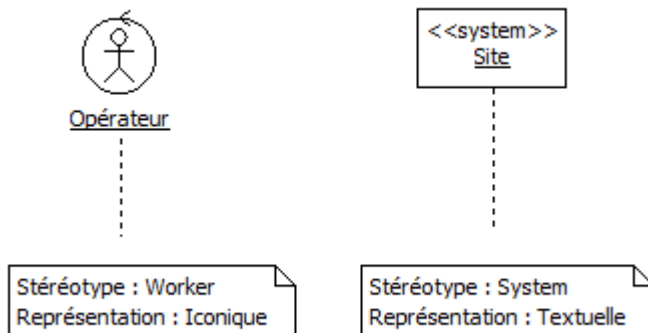
1.2.3 - L'acteur, la classe ou l'objet

Object

Les acteurs et les classes sont représentés par des rectangles ou par des icônes et une ligne discontinue verticale appelée ligne de vie (lifeline). La ligne de vie représente la durée de la présence de l'objet – l'instance de la classe – en mémoire.



par exemple



1.2.4 - Les messages de base

→ Stimulus

Le message est représenté par une flèche horizontale unidirectionnelle orientée de l'émetteur vers le destinataire.

Il peut être nommé (**le plus souvent avec un verbe pour un message CALL – synchrone - ou SEND – asynchrone - , et un nom pour un message RETURN**).

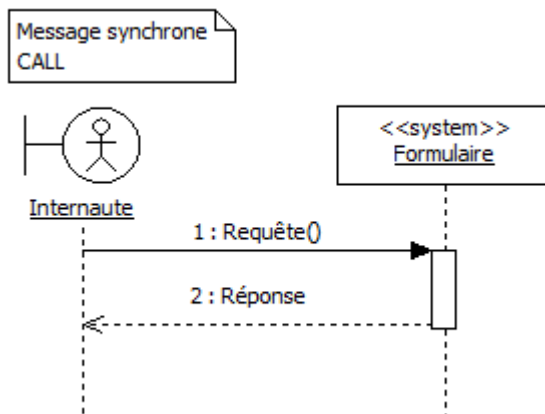
Il peut être numéroté.

Il peut être stéréotypé.

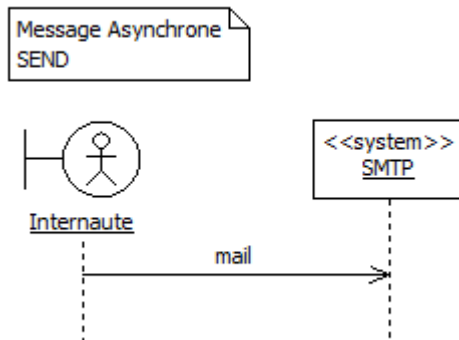
Il y a 5 types de message : CALL, RETURN, SEND, CREATE et DESTROY.

Un message de type **CALL** (synchrone) est représenté par un trait plein avec une flèche fermée et pleine. Le CALL est bloquant tant que le RETURN n'est pas envoyé.

Un message de type **RETURN** (synchrone) est représenté par un trait discontinu avec une flèche ouverte.



Un message de type **SEND** (asynchrone : un envoi de mail, la création d'un thread, requête AJAX, ...) est représenté par un trait plein et une flèche ouverte.



1.2.5 - L'activation

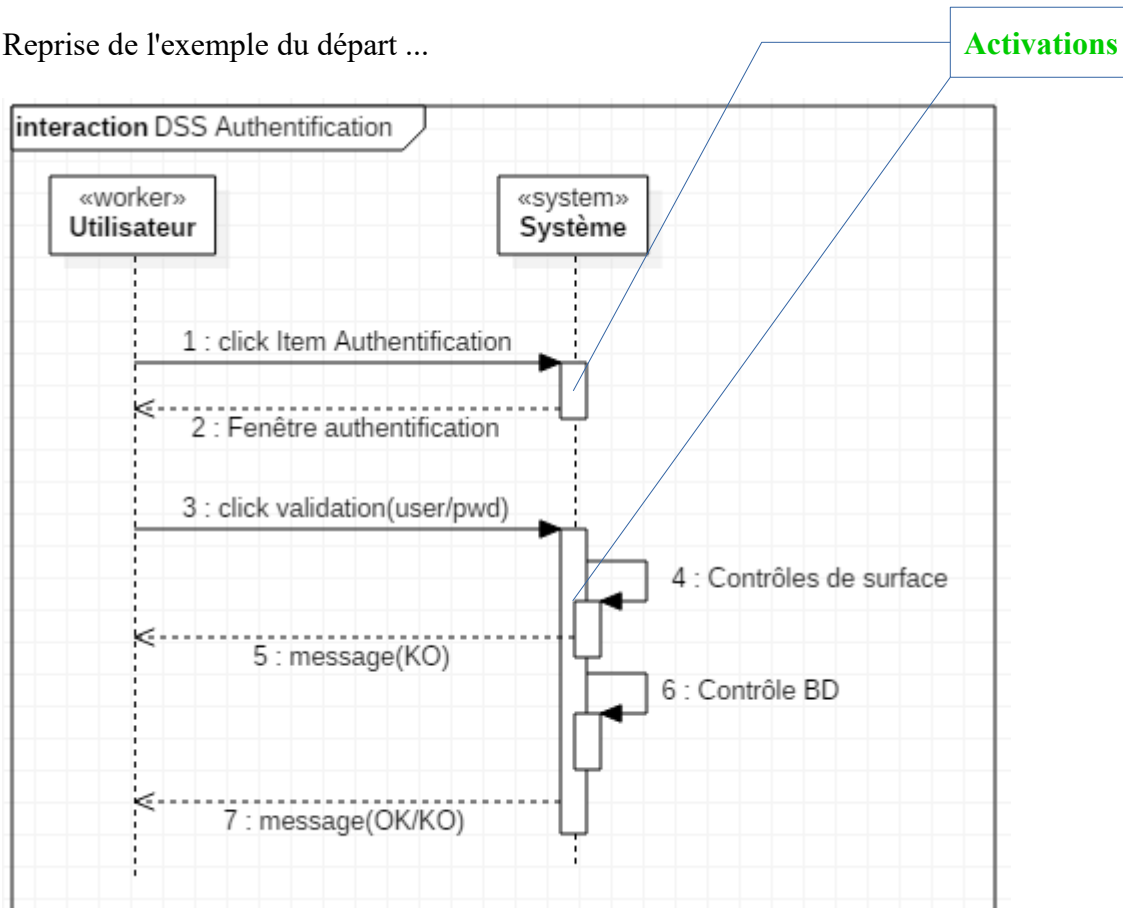
L'activation représente la durée pendant laquelle l'objet effectue une opération de la classe suite à un envoi de message au cours du scénario représenté.

L'activation est **bloquante**.

L'activation est représentée par un rectangle vertical le long de la ligne de vie de la classe.

⚠ Attention ! Ne pas confondre "la ligne de vie" et "l'activation".

Reprise de l'exemple du départ ...



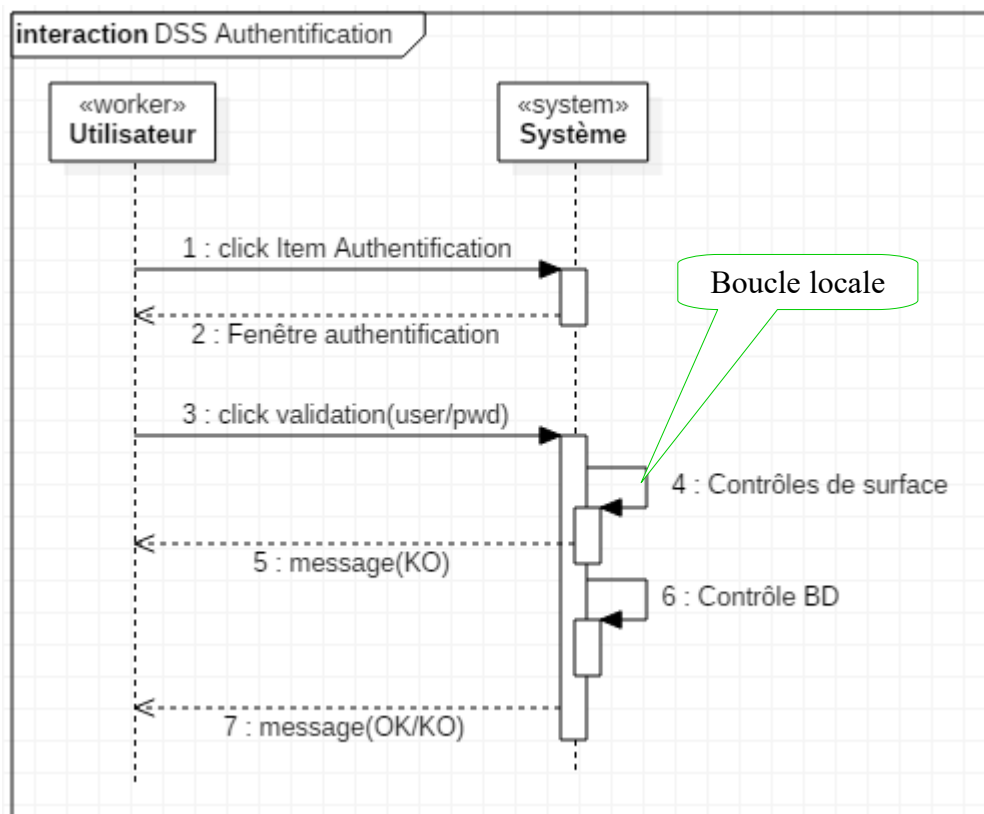
1.2.6 - La boucle locale

SelfStimulus

Permet de décrire une boucle sur la ligne de vie d'une classe.
Cela fait appel à une opération interne (vérification, tri, ...).

Le cas d'une vérification locale de la saisie.

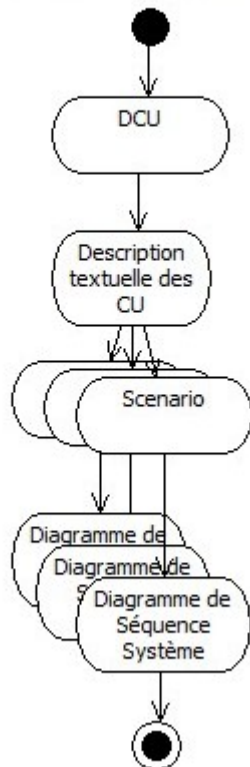
Reprise de l'exemple du départ ...



1.3 - DIAGRAMME DE SÉQUENCE SYSTÈME

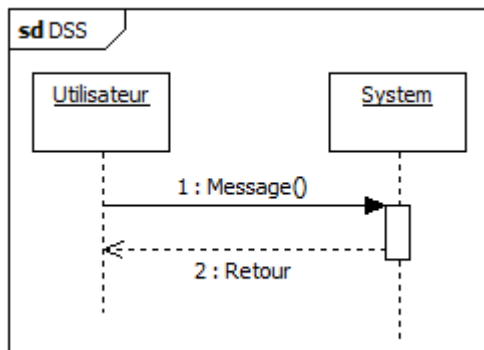
1.3.1 - Contexte

Démarche de création
des diagrammes de séquence Système



1.3.2 - Définition

Le Diagramme de Séquence Système (DSS) représente **deux objets** : l'utilisateur du système et le système. L'utilisateur envoie des messages et le système retourne des messages. Le diagramme est représenté à l'intérieur d'une frame de type sd (Sequence Diagram). Les diagrammes de séquence système sont plutôt du domaine de l'analyse.



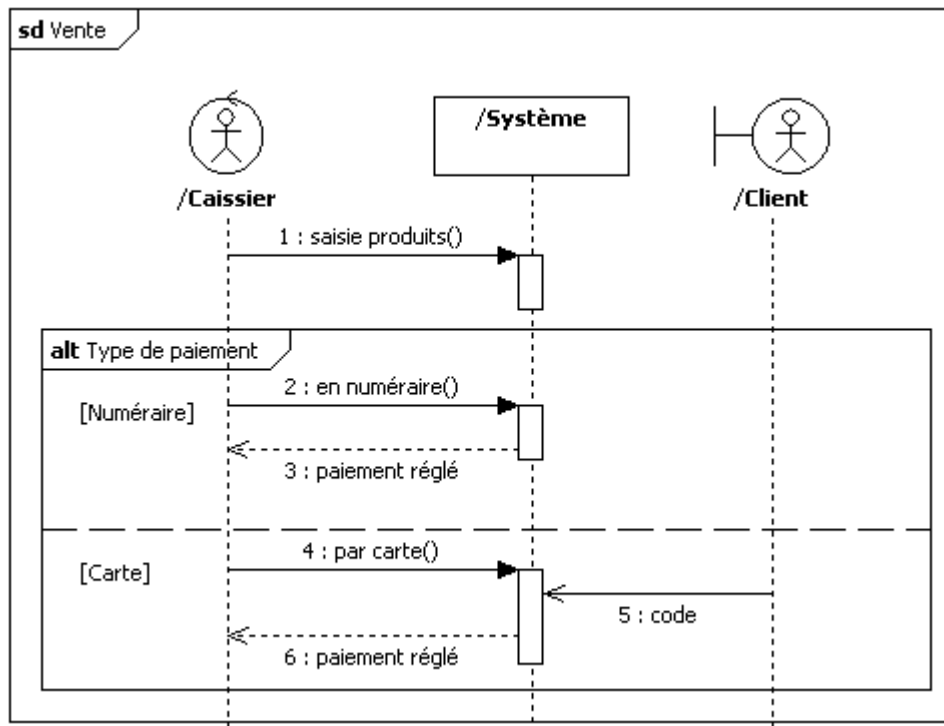
1.3.3 - Exercices

DSS Surfer !

DSS Gérer le panier.

DSS Gérer les commandes.

Note : il est possible, mais **rare**, d'avoir plus de 2 lignes de vie si un autre acteur (secondaire) intervient.



1.4 - DIAGRAMME DE SÉQUENCE DÉTAILLÉ

1.4.1 - Définition

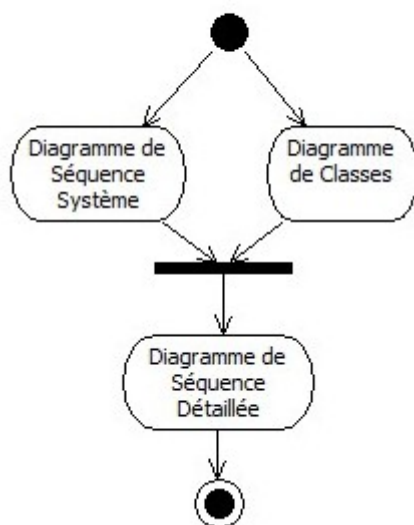
Le Diagramme de Séquence Détaillé (DSD) représente plusieurs classes ou objets en fonction des cas : l'utilisateur du système et les objets du système (Des pages, des fenêtres, des boîtes de dialogue, des frames, etc – **boundary** -, des contrôleurs – **control** -, des entités – **entity** -). Les objets envoient et retournent des messages.

Ces diagrammes, selon la terminologie de Jacobson (un des fondateurs d'UML et de UP) et de son pattern **ECB**, permettent de représenter les classes d'analyse.

Alors que les diagrammes de séquence système étaient dérivés des scénarii des cas d'utilisation, les diagrammes de séquence détaillés dérivent des diagrammes de séquence système et du diagramme de classes.

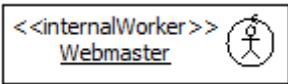

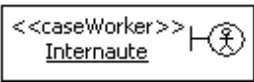

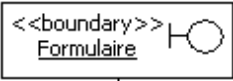

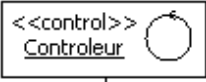
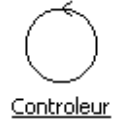


Les diagrammes de séquence détaillés sont plutôt du domaine de la conception détaillée alors que les Diagrammes de Séquence Système étaient plutôt du domaine de la conception architecturale.

Démarche pour la création du
Diagramme de Séquence Détaillé

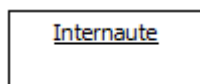


1.4.2 - Représentations iconiques des acteurs et des classes stéréotypés

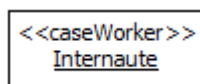
Utilisées lors de la création des DSEQ détaillés représentant les classes d'analyse (cf UP de Ivar Jacobson – un des fondateurs d'UML - et le pattern ECB).

Stéréotype	Décoration	Icône
Opérateur interne		
Opérateur externe		
Vue (Boundary/View)		
Contrôleur (Control/Controller)		
Entité (Entity/Model)		

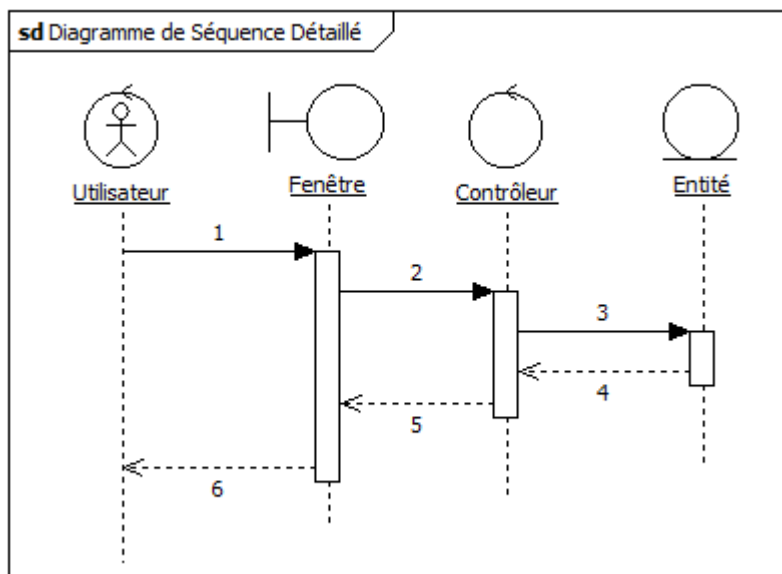
Représentation sans icône ni décoration ni stéréotype.



Représentation textuelle avec le stéréotype.



Syntaxe

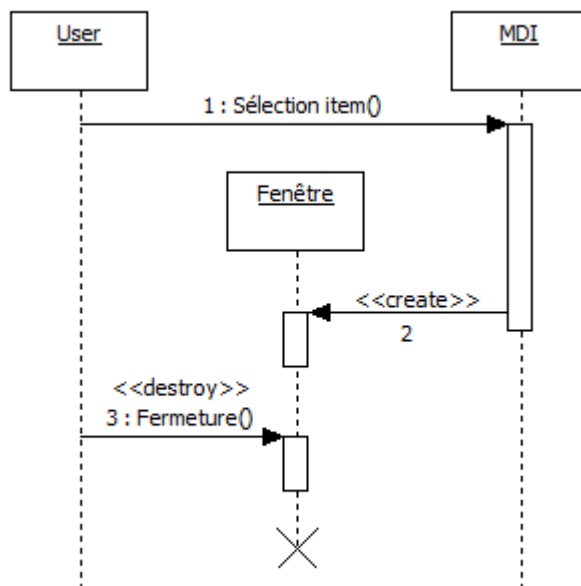


1.4.3 - Les messages CREATE et DESTROY

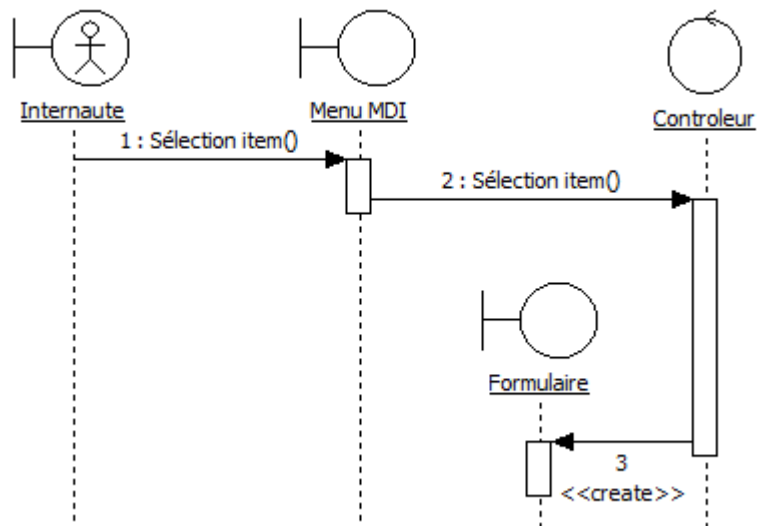
Un message de type **CREATE**, qui correspond à l’instanciation d’un nouvel objet (de type boundary, entity ou autre), est représenté par un trait continu terminé par une flèche fermée et pleine. Le trait est surmonté du mot create entre chevrons (<<create>>) ; c'est un stéréotype.

Un message de type **DESTROY** est représenté par un trait continu terminé par une flèche fermée et pleine. Le trait est surmonté du mot destroy entre chevrons (<<destroy>>) ; c'est un stéréotype. La ligne de vie est terminée par une croix de saint-André.

Exemple d'ouverture – donc d'instanciation - et de fermeture d'une fenêtre avec stéréotypes.



Exemple d'ouverture – donc d'instanciation - d'une fenêtre avec stéréotypes (Case Worker, Dialogue et Contrôleur) et représentations iconiques.



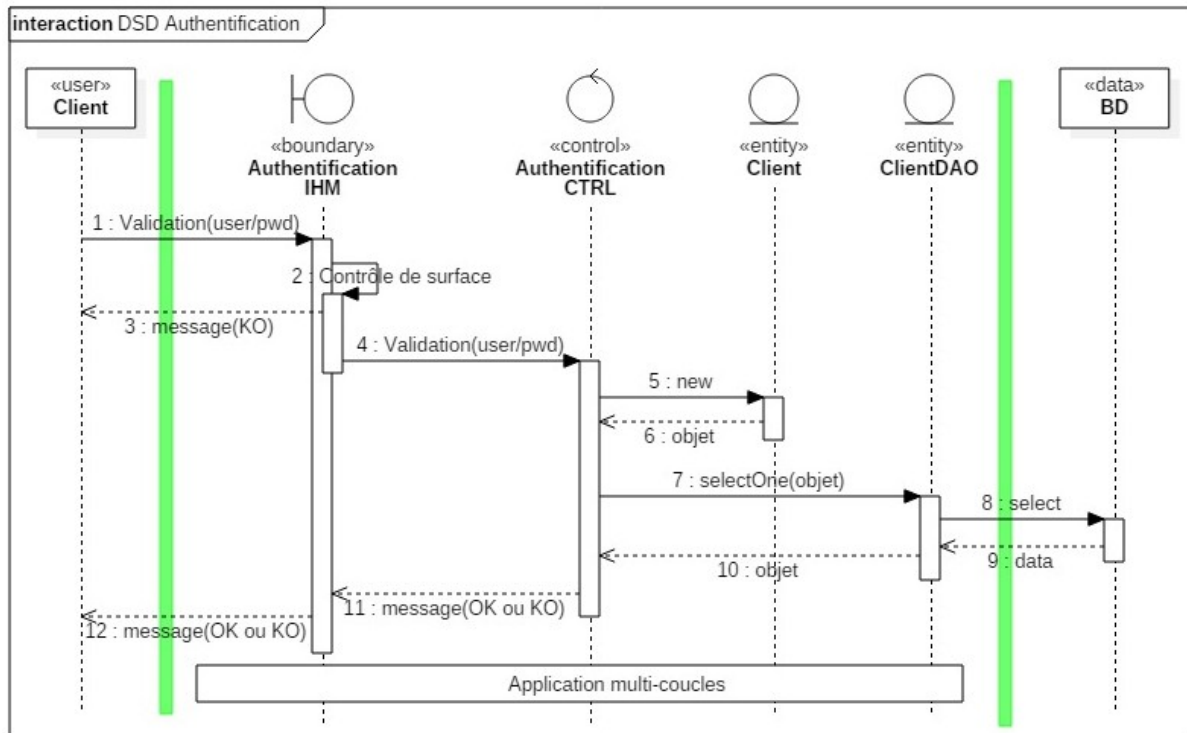
Notes :

Les CALL et les SEND sont plutôt des verbes (Valider, annuler, envoyer mail, ...).
Les CREATE et DESTROY aussi (ouvrir, fermer, quitter, ...).

Les RETURN sont plutôt des noms (création, authentification, ...).

mais ...

1.4.4 - Exemple de Diagramme de Séquence Détaillé

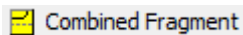


1.5 - LES FRAGMENTS COMBINÉS

1.5.1 - Définition

Un combined fragment (fragment d'interactions ou fragment combiné) permet de représenter une combinaison d'interactions. Il est défini par un opérateur et un nom.

Il est représenté par un rectangle avec en haut à gauche l'opérateur et le nom de l'opération.

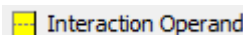


UML 2 propose plus d'une dizaine d'opérateurs.

La structure séquentielle (**seq**) qui est l'opérateur par défaut et signifie que l'ordre d'exécution n'a pas d'importance (pensez à un distributeur de boissons où vous pouvez soit insérer une pièce en premier puis choisir la boisson et inversement), l'alternative (**alt**), la boucle (**loop**), l'option (**opt**), la référence à un autre DSEQ (**ref**) pour alléger la représentation, l'interruption de la séquence (**break**), les traitements en parallèle (**par**) dans la catégorie des fragments pour la description des flux de contrôle.

Et aussi strict (ordonnancement strict), assert (tout ce qui n'est pas présent dans la séquence provoque un break), ignore (les messages peuvent être ignorés, les autres doivent être impérativement considérés), consider (les messages doivent être considérés, les autres peuvent être ignorés) et la négation pour les séquences invalides (**neg**) dans la catégorie des fragments concernant l'interprétation de la séquence.

En fonction des opérateurs on ajoutera un opérande (dans le cas d'une alternative par exemple).



cf http://support.objectteering.com/objectteering6.1/help/fr/objectteering_uml_modeler/diagrams/sequence_diagrams.htm

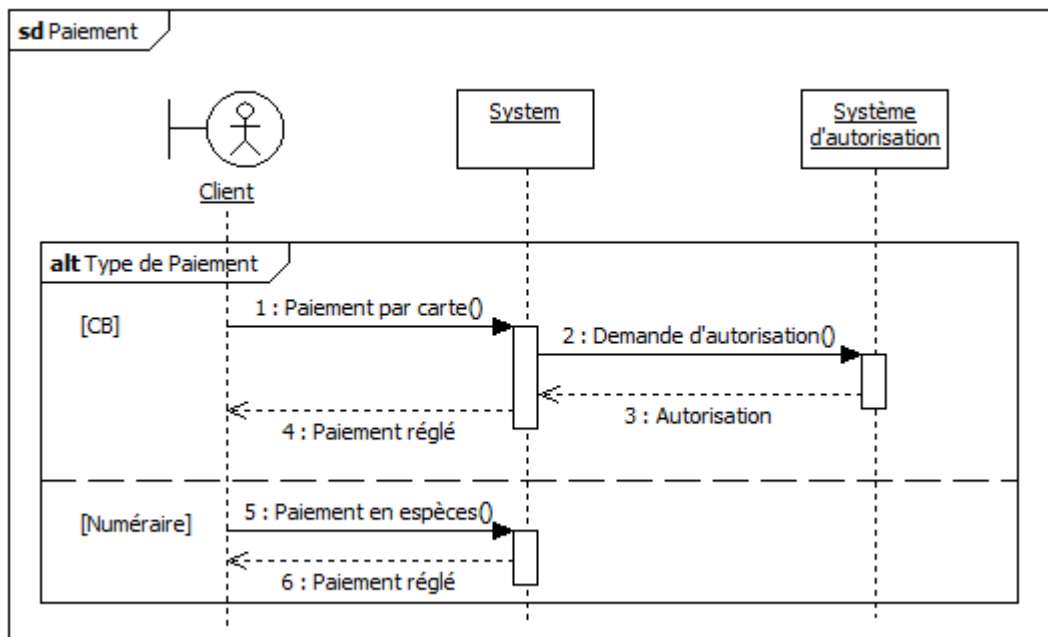
et <http://msdn.microsoft.com/fr-fr/library/dd465153.aspx>

Note : pour alléger les diagrammes les frames sont souvent omises ainsi que les fragments [seq].

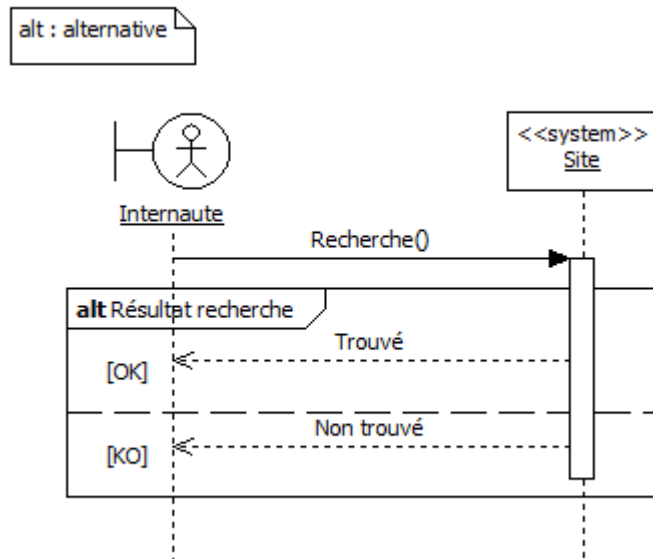
1.5.2 - L'alternative (alt)

Représentation d'une alternative (SI ... SINON) mais pas de SWITCH.

Exemple : L'utilisateur a le choix entre 2 options : payer par carte ou en espèces.



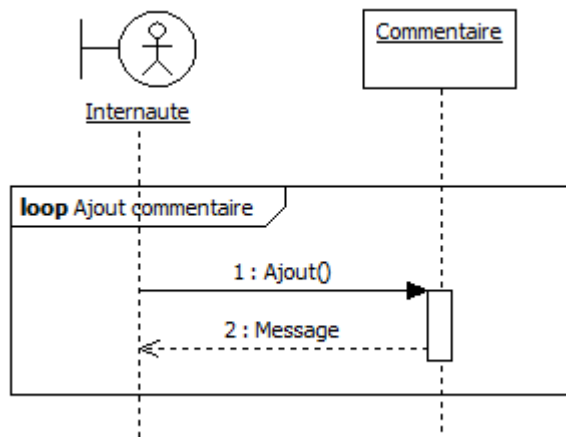
Autre exemple : Une recherche dans un site peut être fructueuse ou infructueuse. Voici sa représentation.



1.5.3 - La boucle (loop)

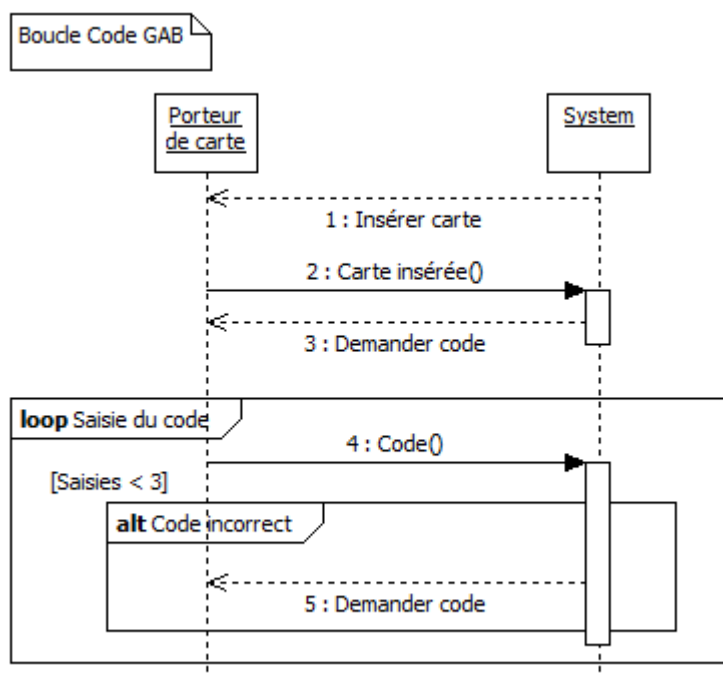
Représentation d'une action répétitive.

Exemple : l'ajout de commentaires.



Note : le message de retour pourrait être représenté avec une alternative (Insertion réussie/Echec insertion) ou par une annotation d'erreur.

Autre exemple avec une garde : saisie du code au GAB

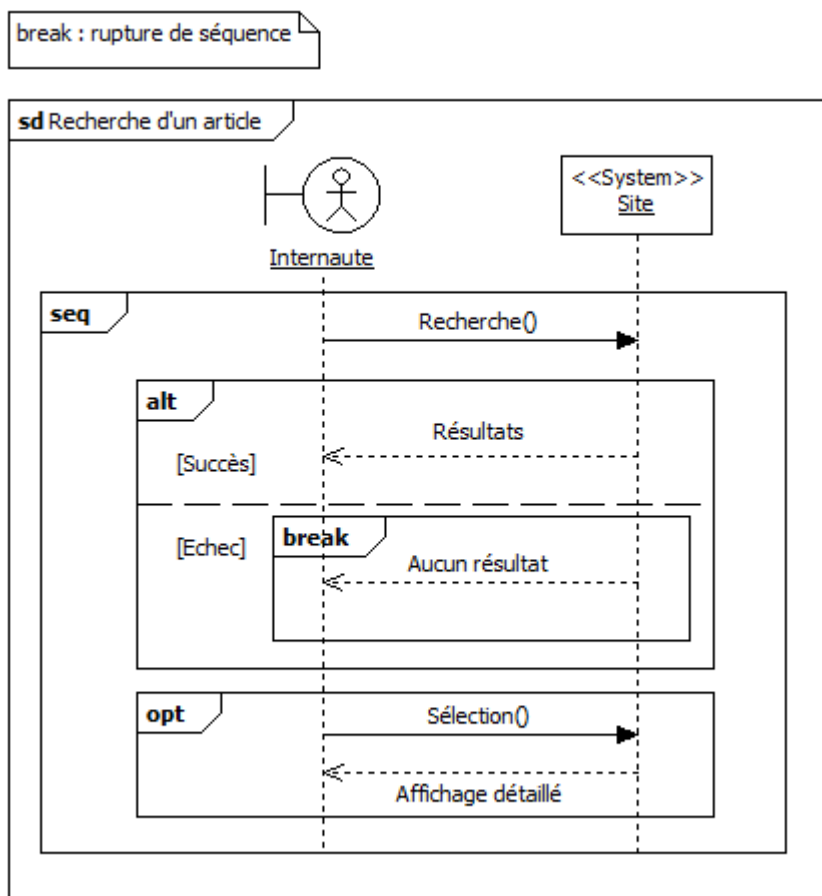


1.5.4 - La rupture (break)

L'opérateur break décrit une rupture de séquence. Il correspond à un scénario d'exception.

Exemple : Recherche fructueuse ou infructueuse.

Dans cet exemple, lors d'une recherche, toute la séquence est interrompue si aucun résultat n'est obtenu.

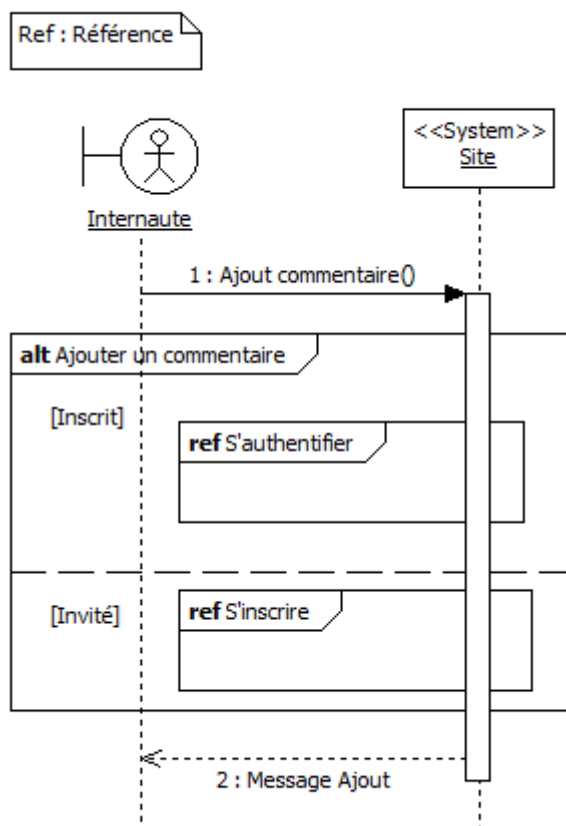


1.5.5 - La référence (ref)

Elle représente un appel à une autre séquence représentée dans un autre diagramme situé dans un autre document.

Exemple : Pour pouvoir ajouter un commentaire dans un site de publication, un internaute doit être inscrit et s'il est déjà inscrit il doit s'authentifier.

Diagramme de séquence [Ajouter un commentaire].

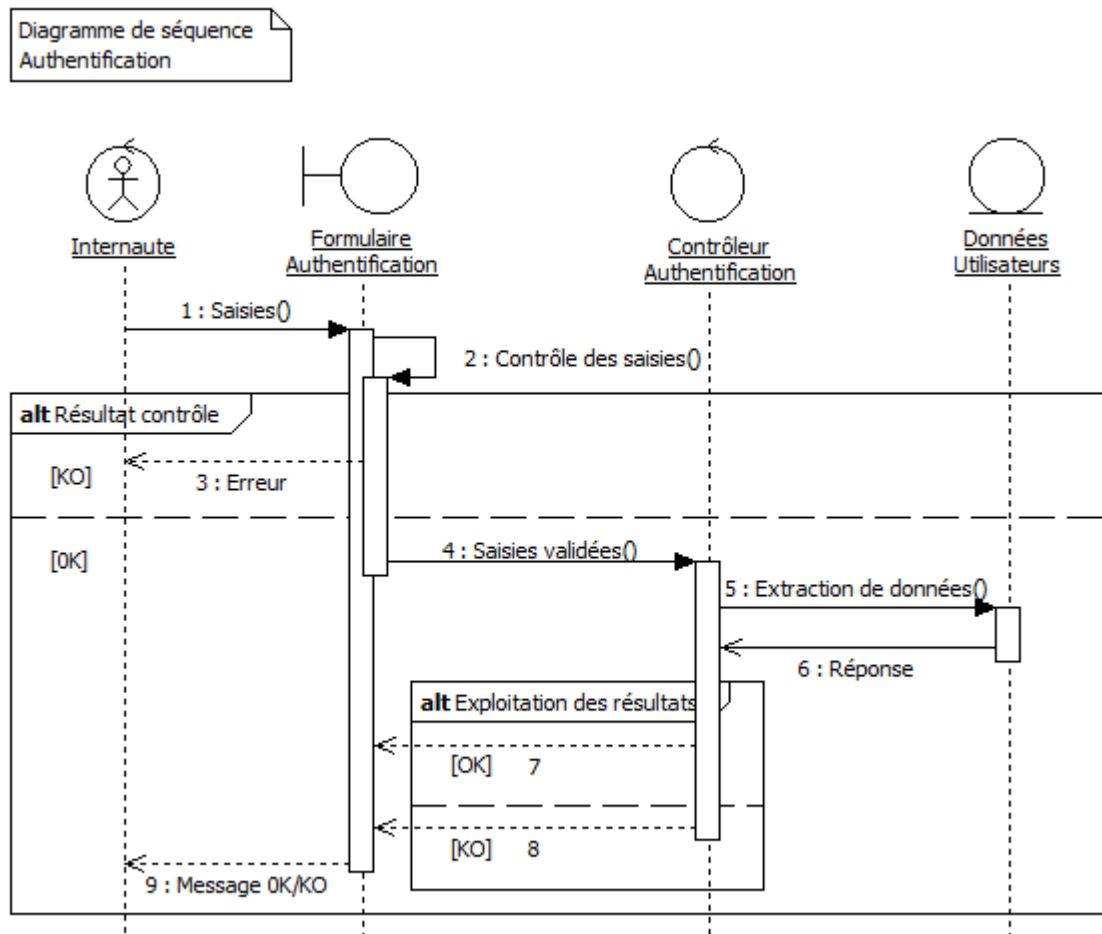


Notes pour StarUML version 1 :

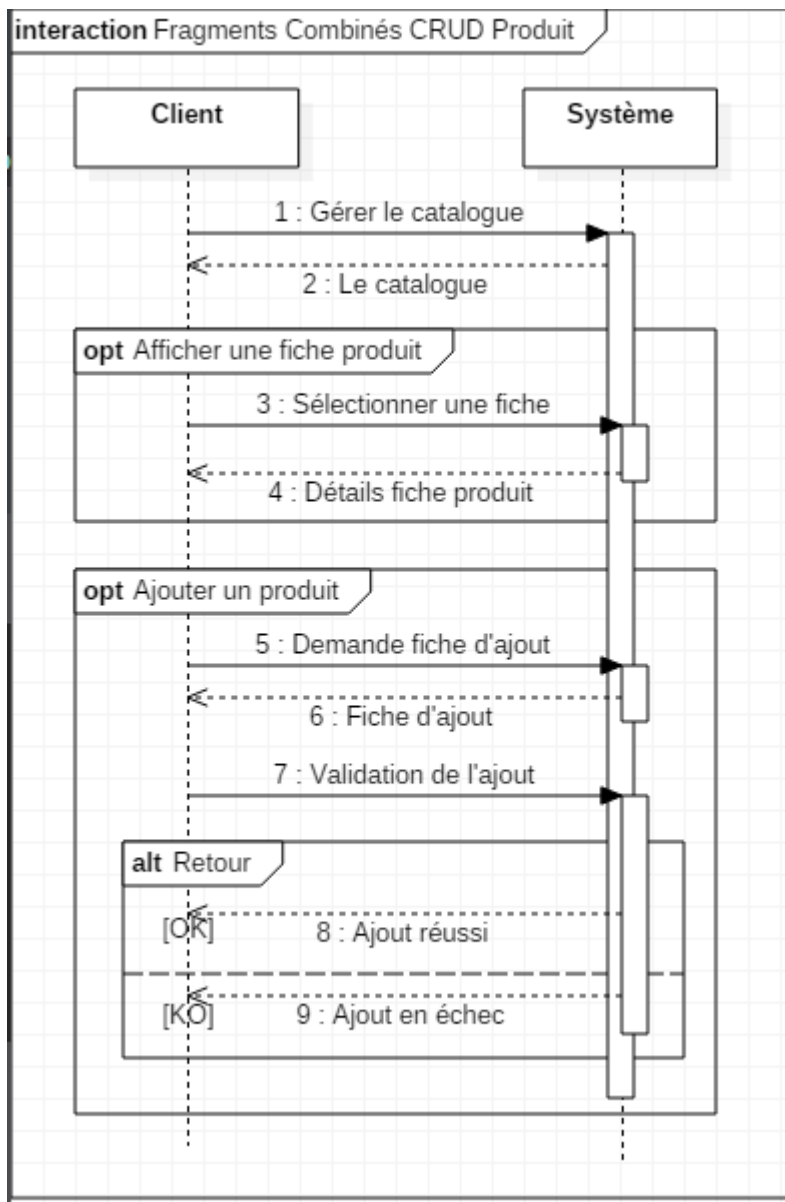
Pour créer une [ref] utilisez une Frame, et pas un Combined Fragment, et saisissez Ref dans la propriété Frame Kind.

[Inscrit] et [Invité] correspondent à des gardes.

Diagramme de séquence [S'authentifier].



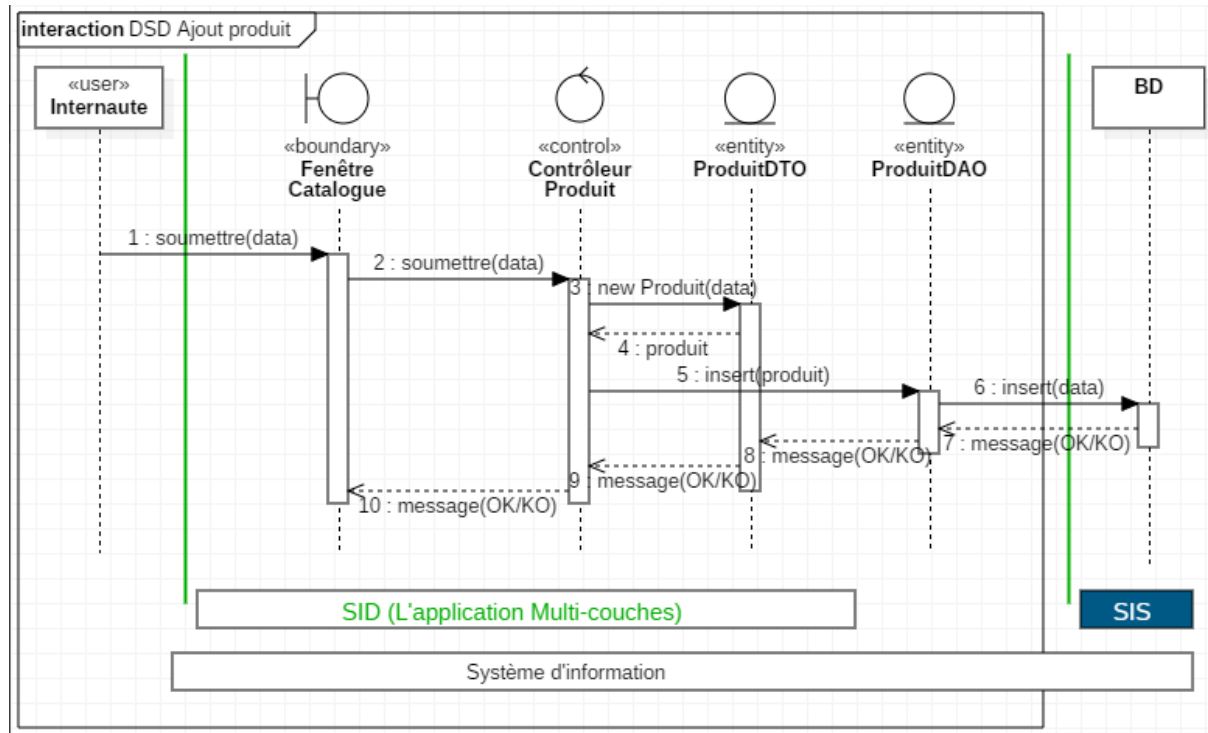
1.5.6 - DSS, Fragments combinés et CRUD Produit



Note : à vous de faire la Suppression et la Modification !

1.6 - DSD AJOUT PRODUIT 4 COUCHES !

1.6.1 - Exemple



1.6.2 - Exercice

DSD selectOne produit.

1.7 - MOTS-CLÉS

Classe ou Objet

Ligne de vie

Activation

Message (synchrone, asynchrone, retour, création, destruction)

Garde, paramètres

Boucle locale

Frame

Fragment combiné