Diagrammes de séquences
Diagrammes de communication

Isabelle BLASQUEZ
@iblasquez

2017

Isabelle BLASQUEZ

@iblasquez

Enseignement : Génie Logiciel

Recherche : Développement logiciel agile

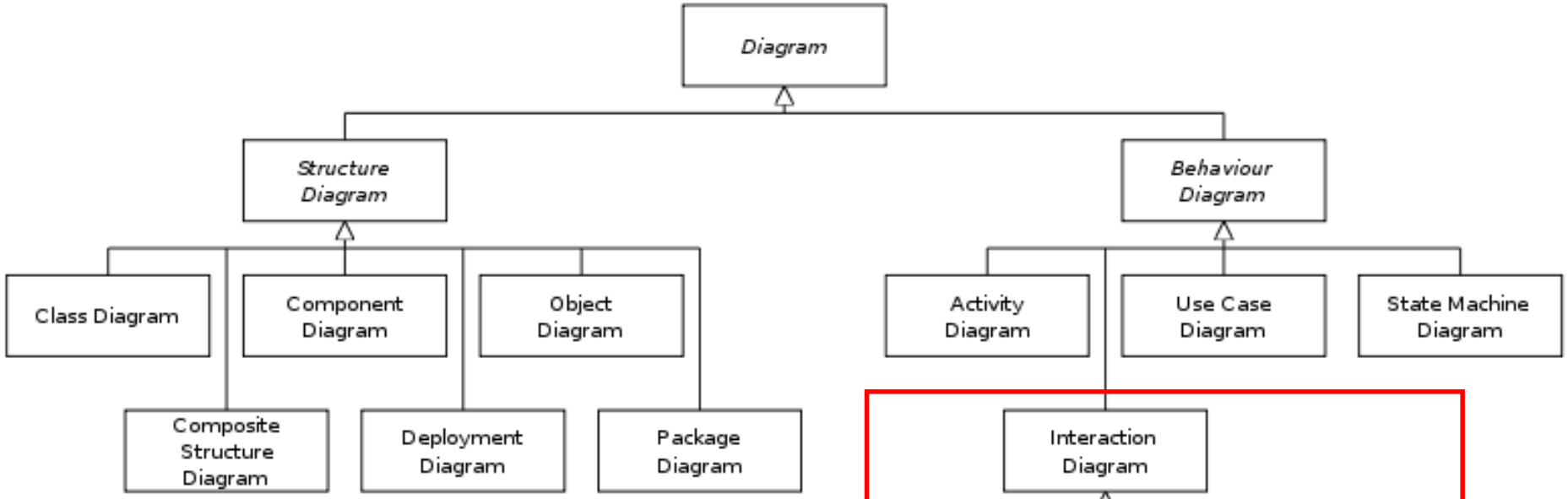
ICSTUG #IUTAgile

CodeWeek.

#Software Craftsmanship

MUSEOMIX LIMOUSIN

De l'interaction avec le diagramme de séquence (et de communication)



représentation **temporelle**

représentation **spatiale**

Représentation graphique

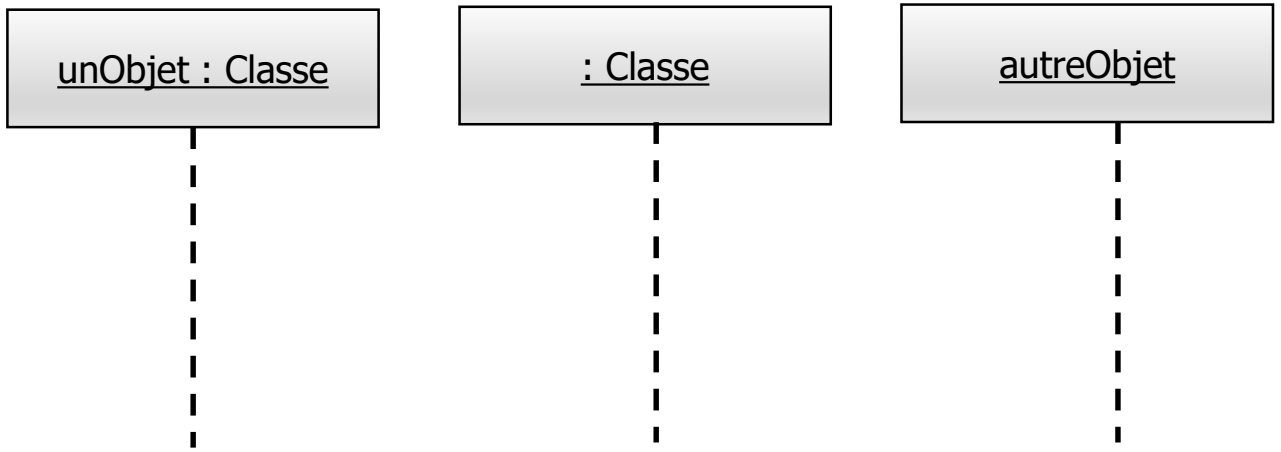
Isabelle BLASQUEZ

Introduction au Diagramme de Séquence (DS)

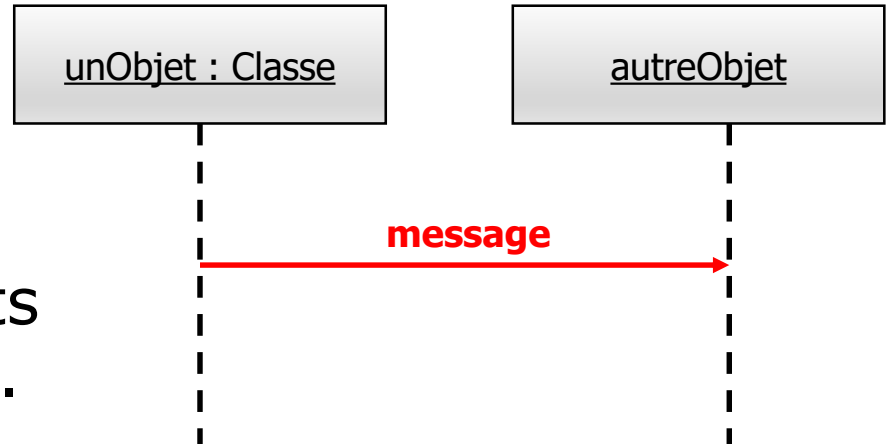
Le diagramme de séquence est un **diagramme d'interaction**.

Un objet est représenté par un **rectangle**

Sa ligne de vie est représentée par une **ligne en pointillée**



Une interaction modélise un **comportement dynamique** entre objets qui se traduit par un envoi de **message**.



Notion de message

Message : **communication unidirectionnelle entre objets** qui transporte l'information avec l'intention de déclencher une activité chez le récepteur.

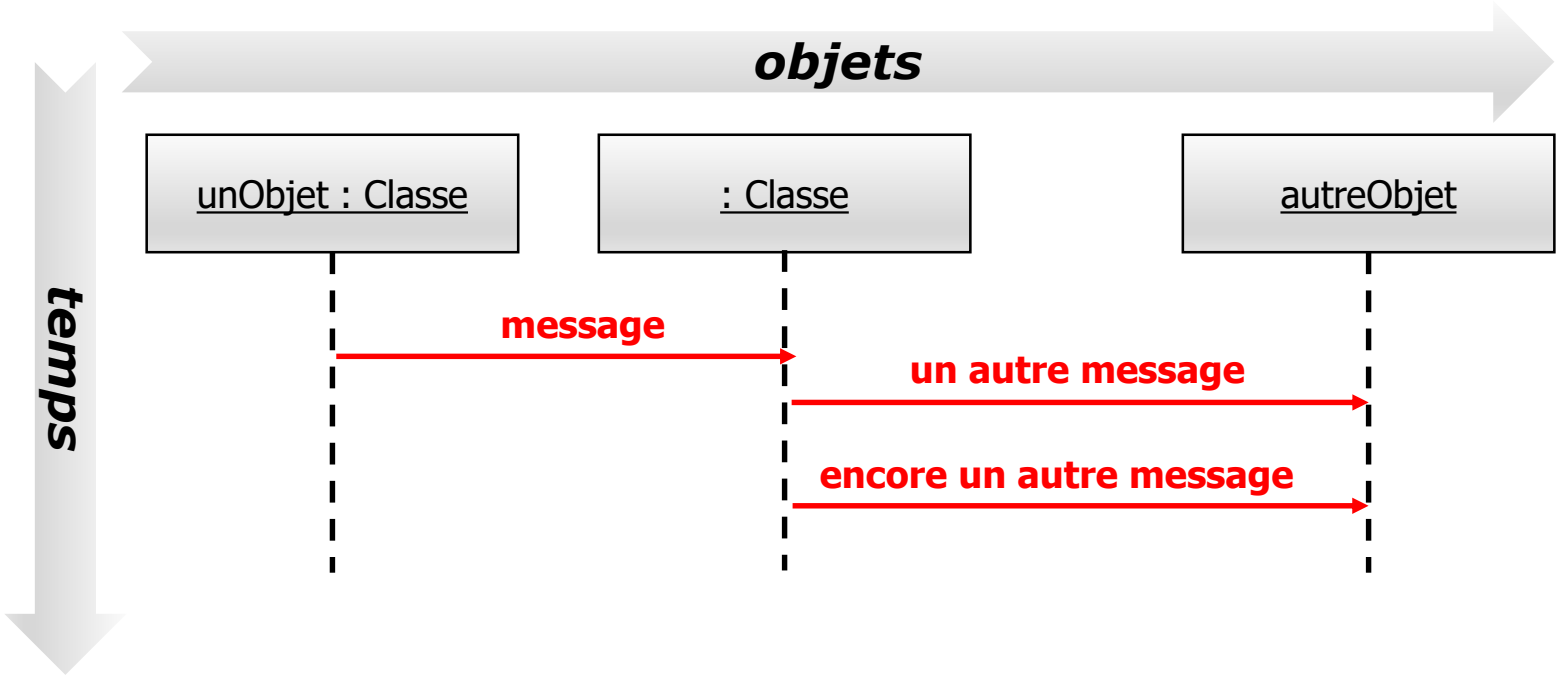
Un **message simple** :

Expression syntaxique complète d'un **message** :

Paramètre de retour := numéro de séquence : nom du message (Liste des paramètres)

Dimension temporelle du DS

Un diagramme de séquence est constitué d'une **séquence d'interactions** respectant un **point de vue temporel.**

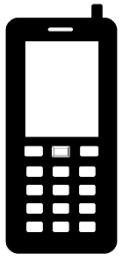


Un diagramme de séquence a **deux dimensions**

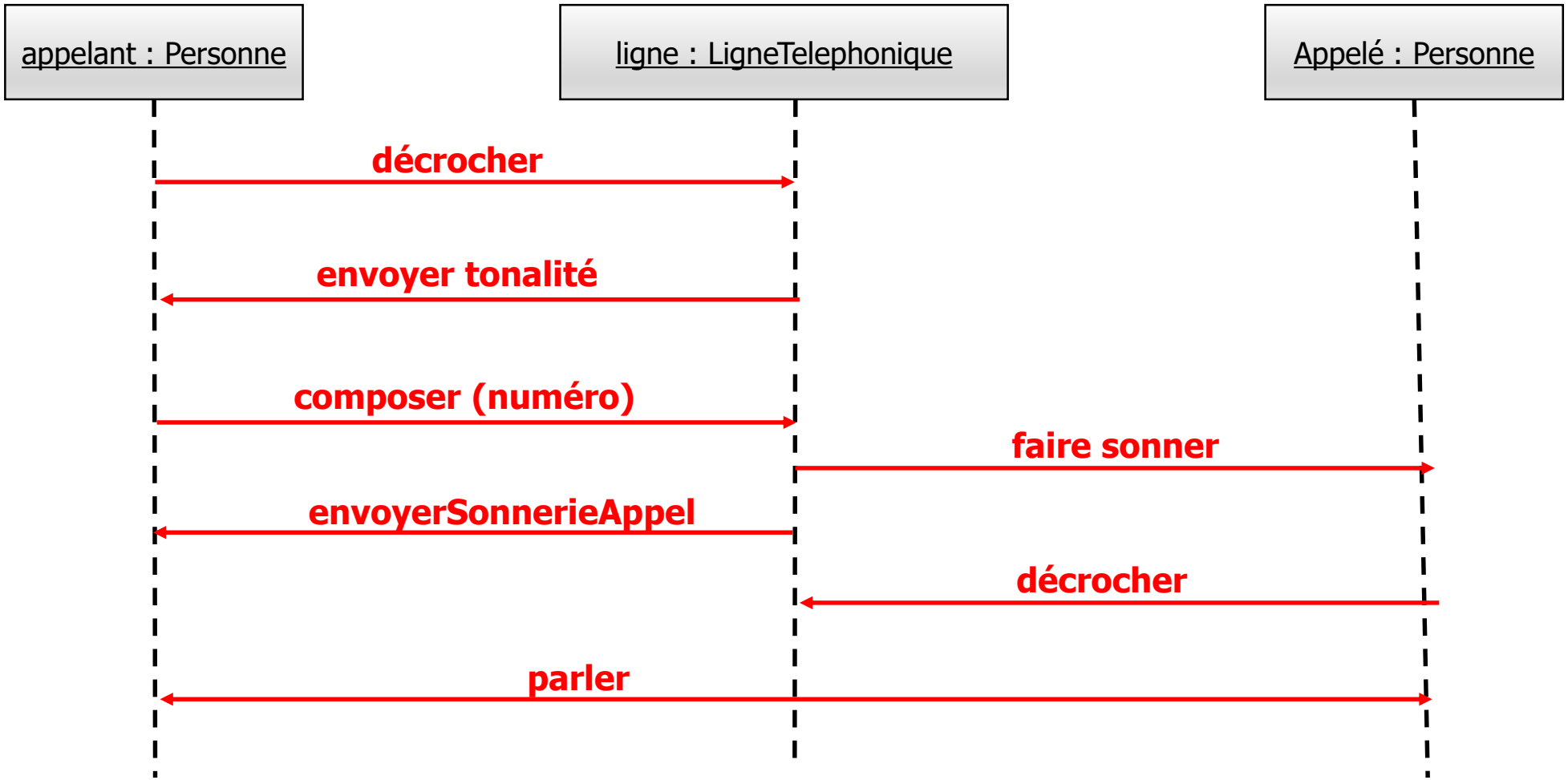
→ Dimension verticale : **le temps** (ordre indique ordre d'envoi des messages)

→ Dimension horizontale : **les objets** (peu importe l'ordre)

Exemple de diagramme de séquence



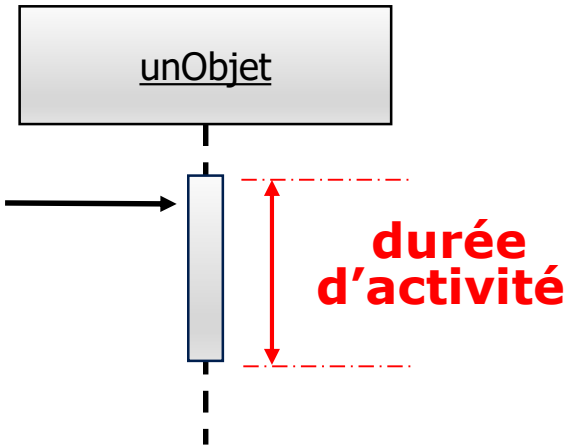
Exemple d'une communication téléphonique (flot nominal) :



Les activations (ou Focus of Control)

Une période d'activité est représentée par une bande rectangulaire placée sur la ligne de vie.

Elle correspond au temps pendant lequel un objet effectue une action.



La période d'activité est également appelée « activation » ou « focus of control ».

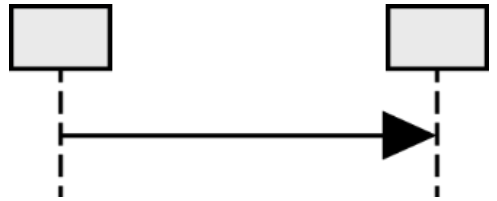
(période sans interruption durant laquelle les différents messages s'enchaînent)

La représentation du focus est optionnelle.

Les messages (1/3) : les classiques …

Message synchrone (CALL)

→ bloque l'émetteur qui attend jusqu'à la réception d'une réponse (Exemple : appel d'une *opération*) (le plus couramment utilisé)



Message asynchrone (SEND)

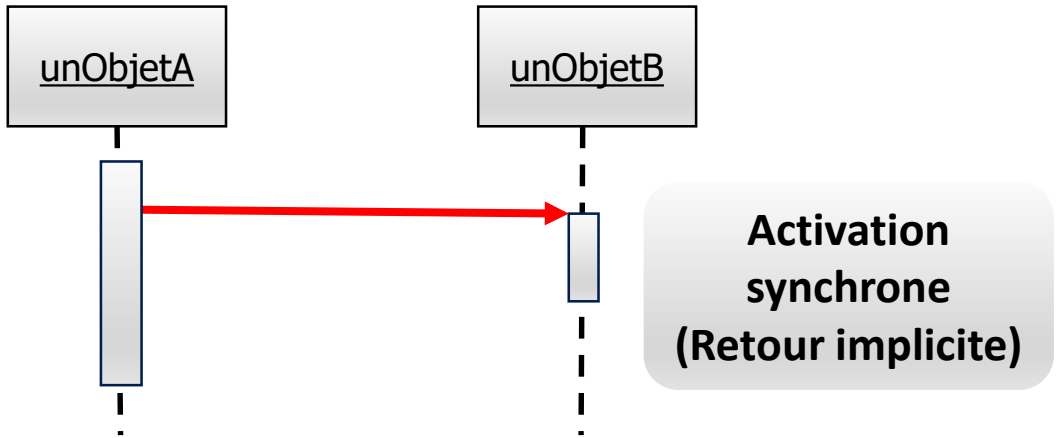
→ l'émetteur continue sans attendre la réponse du récepteur (émetteur non bloqué). C'est un *signal*

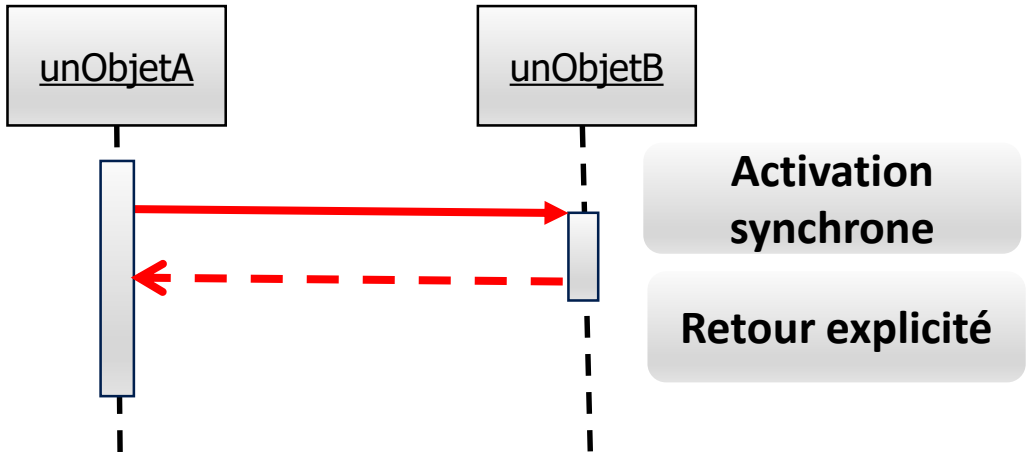
Message de retour (RETURN)

→ pas forcément représenté si le retour est explicite (comme par exemple pour message synchrone)

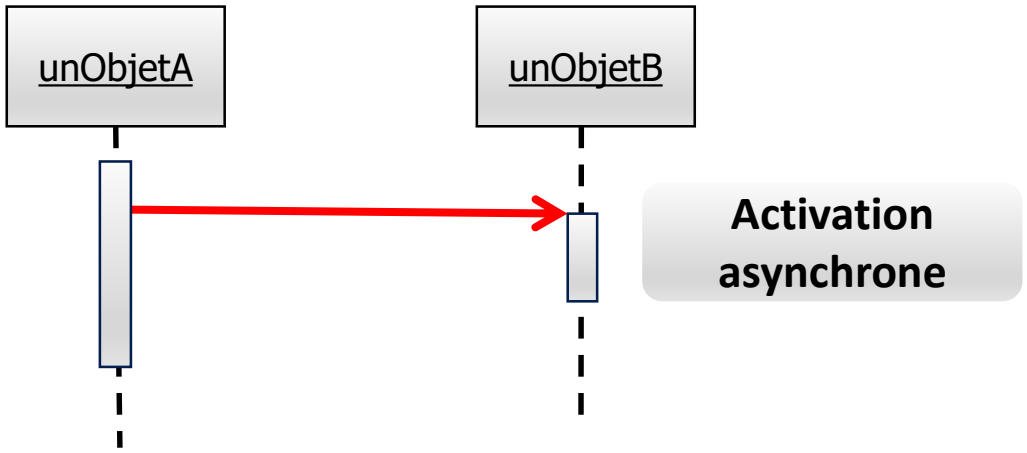
Le retour dans les messages synchrone & asynchrone

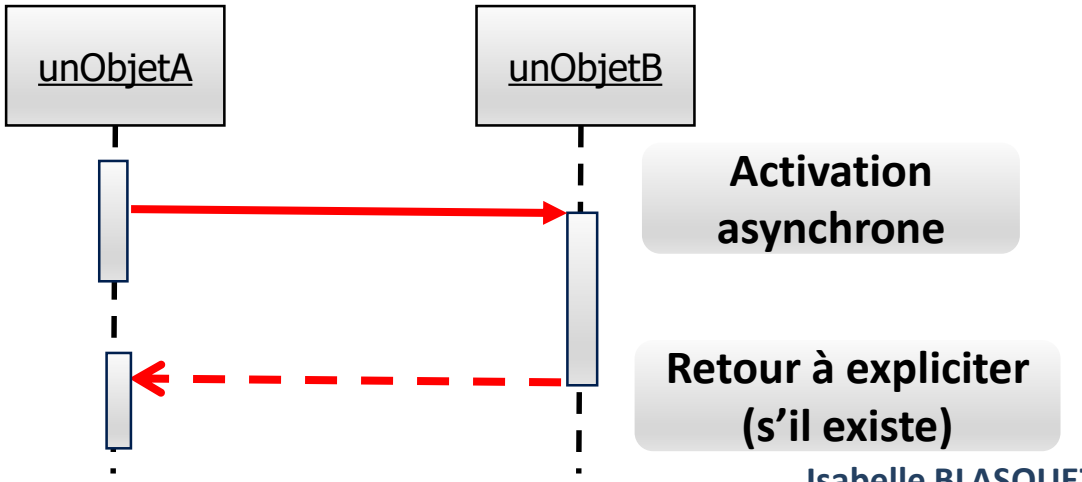
Message synchrone (CALL)





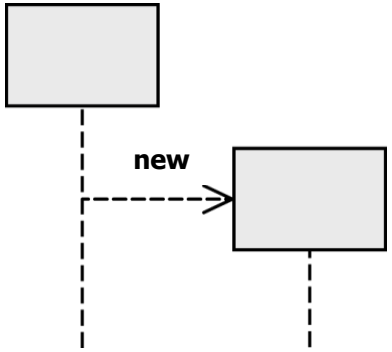
Message asynchrone (SEND)

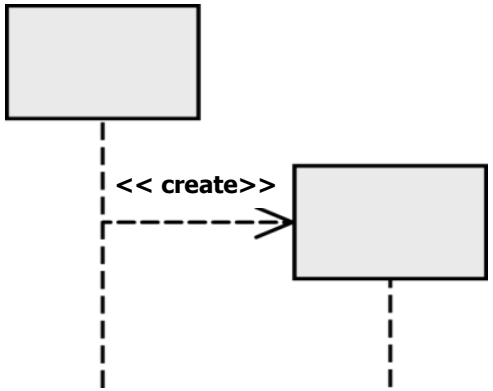




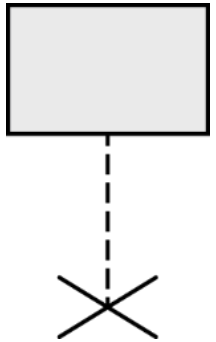
Les messages (2/3) : création et destruction d'objet

Création d'objet





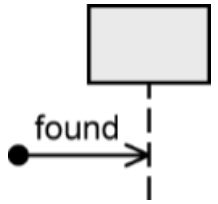
Destruction d'objet



Les messages (3/3) : moins couramment utilisés

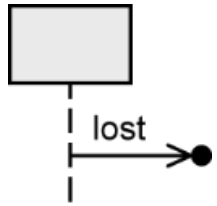
Message trouvé (*found*)

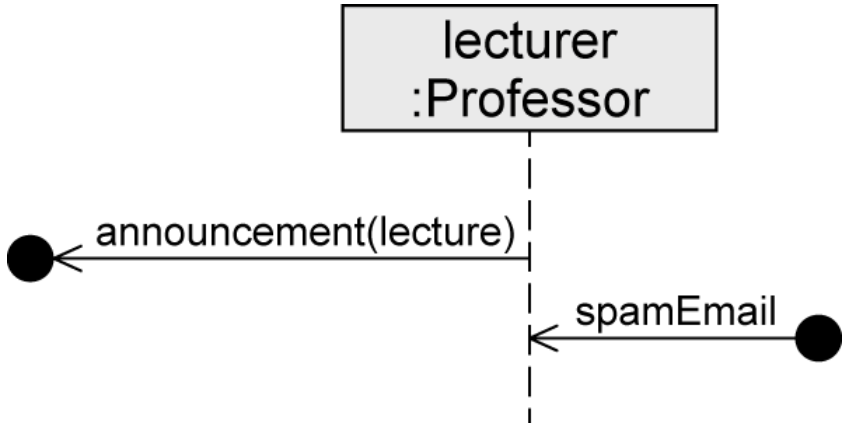
→ l'émetteur du message n'est pas connu



Message perdu (lost)

→ le récepteur du message n'est pas connu



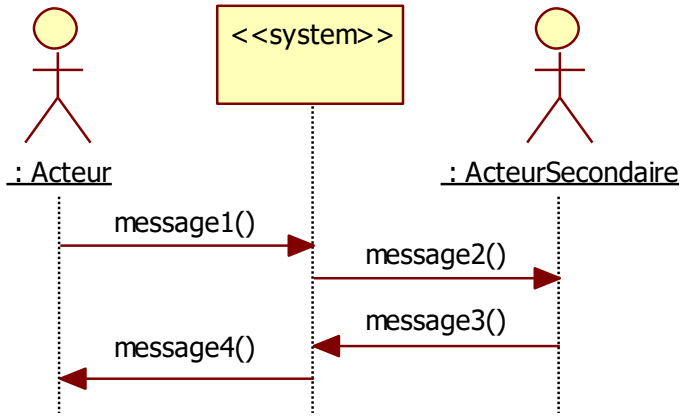


Le diagramme de Séquence Système (DSS)

Isabelle BLASQUEZ

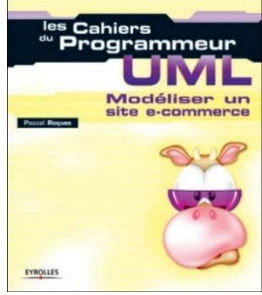
Diagrammes de Séquence Système (DSS)

Proposé par Larman, les **diagramme de Séquence Système (DSS)** montrent les **acteurs** qui interagissent directement avec le système.

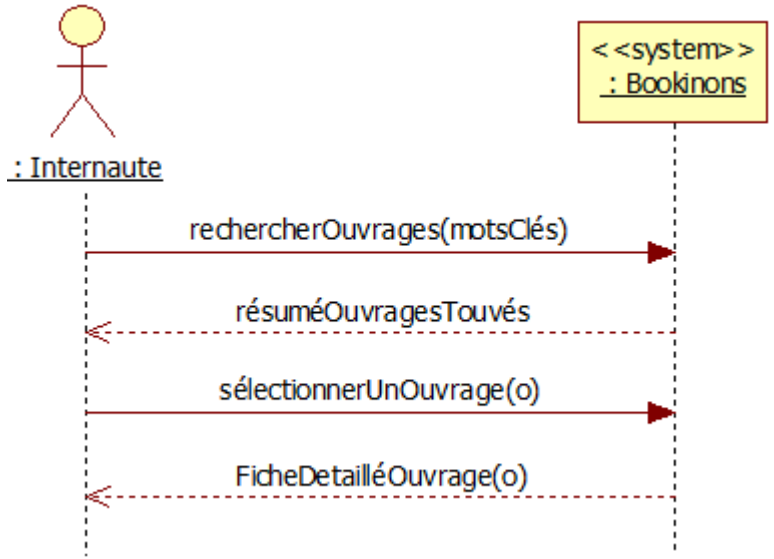


Le système informatique est vu comme une ***boîte noire***
car son comportement est décrit de l'extérieur
(peut-être utilisé durant une phase d'analyse ...
la boîte pourra ensuite être ouverte en phase de conception !)

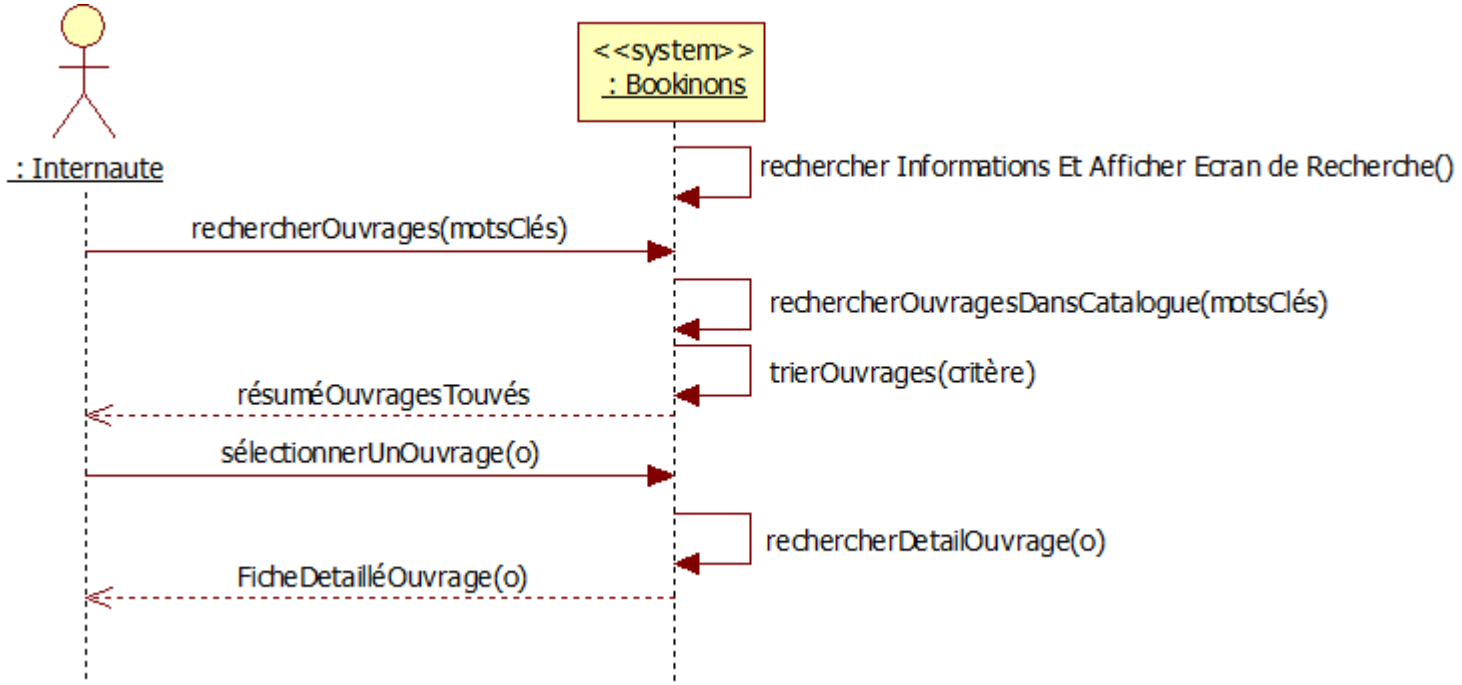
DSS Rechercher un ouvrage



Sans action interne

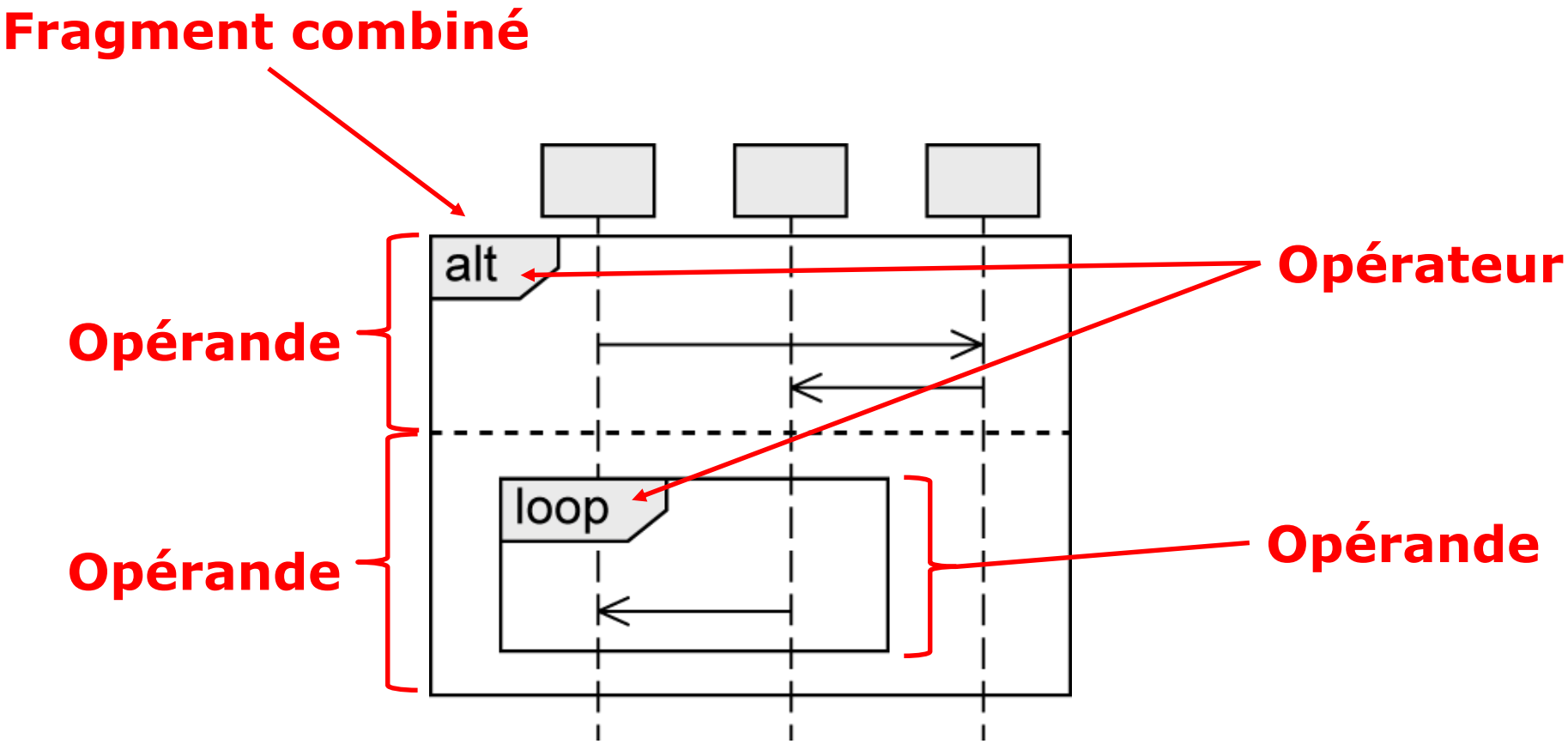


Avec action interne (message réflexif)



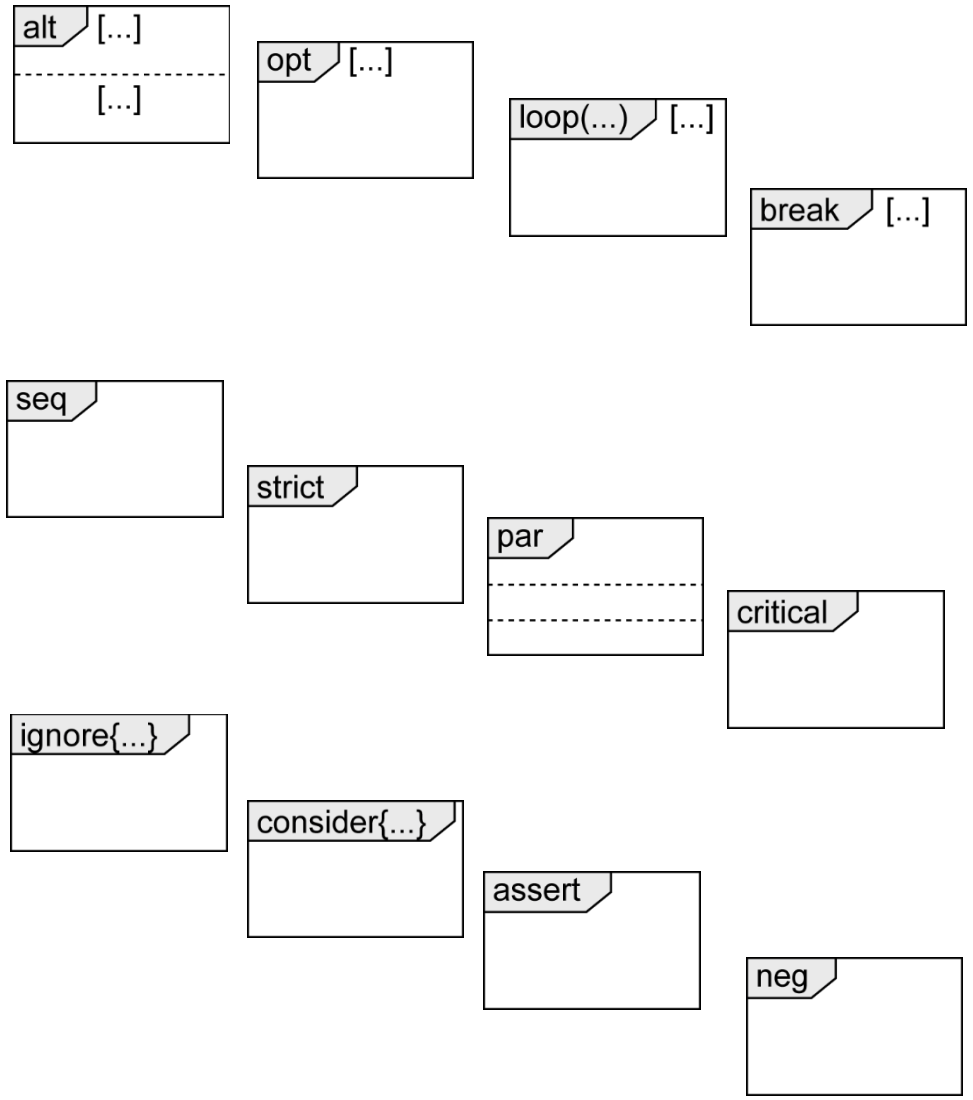
Fragment combiné

Fragment combiné : Présentation



Liste des 12 opérateurs

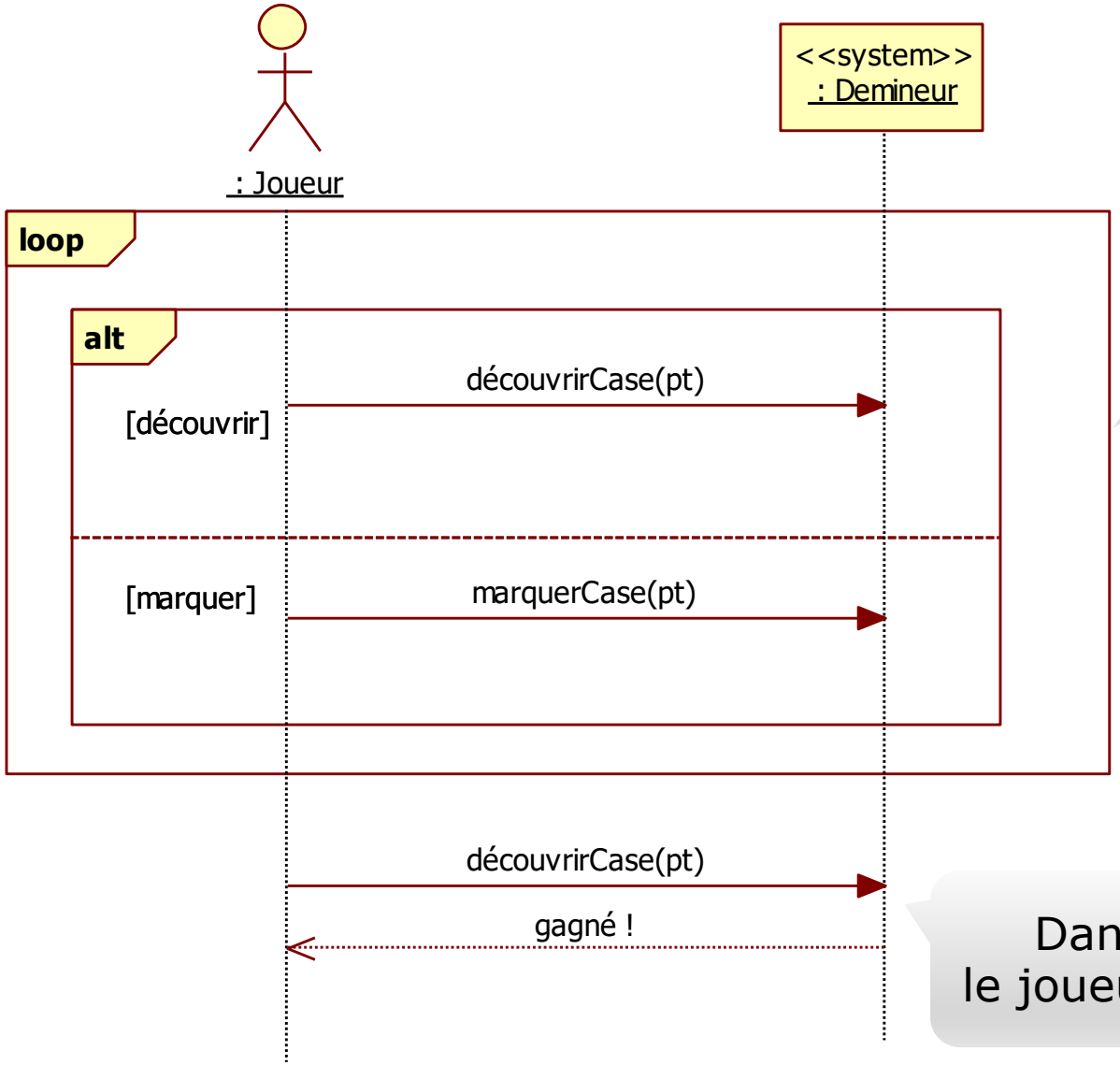
	Opérateur	
Choix et boucles	`alt`	Alternative
	`opt`	Option
	`loop`	Boucle (répétition)
	`break`	Exception
Parallélisation et ordre d'envoi	`seq`	*Weak sequencing*
	`strict`	*Strict sequencing*
	`par`	Parallèle (execution concurrente)
	`critical`	Section critique
Contrôle envoi de message	`ignore`	Ignorer (messages non présents dans le fragment combiné)
	`consider`	Considérer (interactions à prendre en compte dans la séquence)
	`assert`	Assertion (le fragment combine est une assertion)
	`neg`	Negative (interaction invalide)

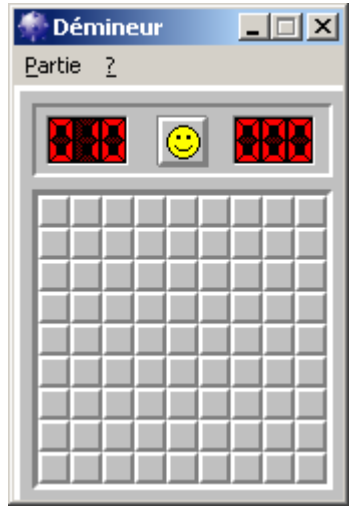


Explication détaillée des opérateurs : voir annexe
Exemples d'utilisation des opérateurs disponibles sur : http://cian.developpez.com/uml2/tutoriel/sequence/

Fragments combinés : exemple

Diagramme séquence système simplifié
pour le flot de base du UC «Jouer une partie de démineur»





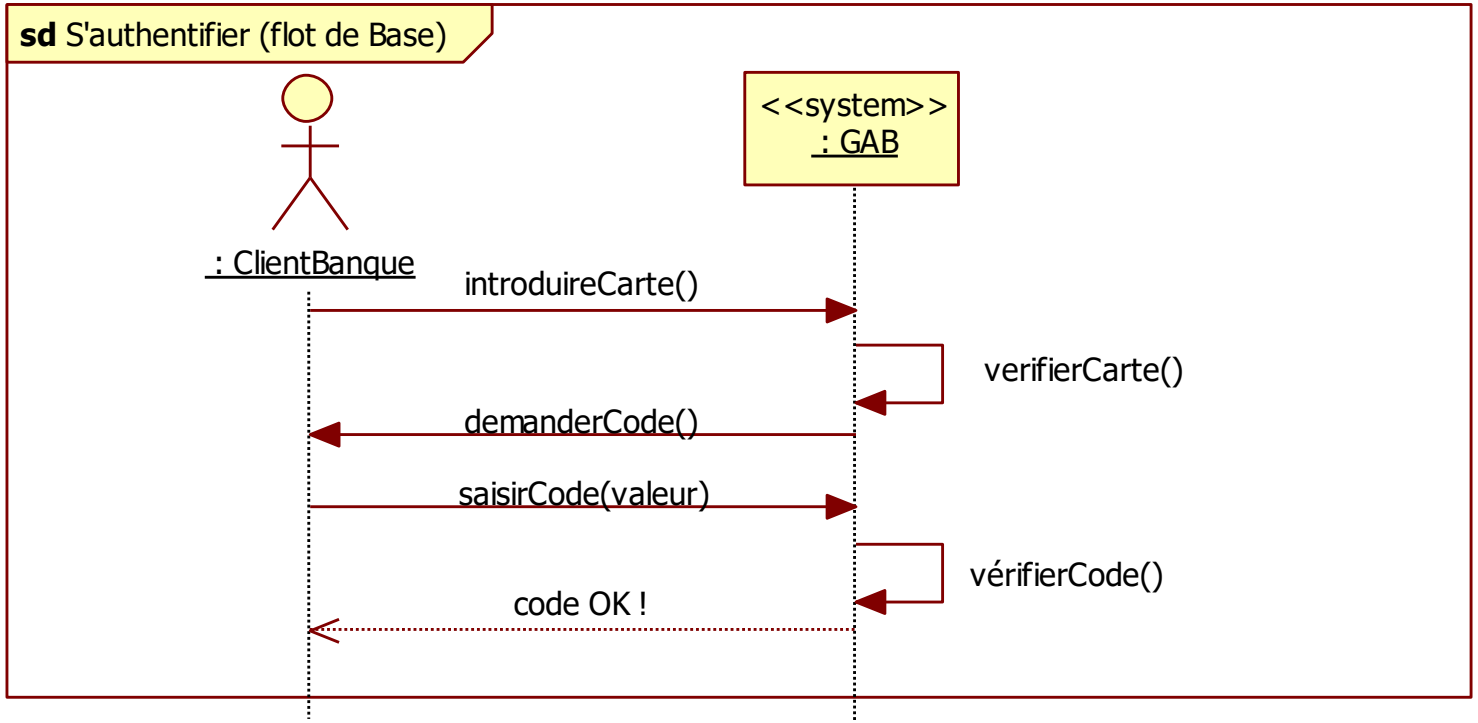
Le joueur passe son temps à découvrir ou à marquer des cases …

Dans le flot de base, le joueur finit par gagner...

Cadre de diagramme & référence

Présentation de la notion de cadre de diagramme

Un diagramme peut être inclus dans un **cadre de diagramme** (opérateur **sd**)

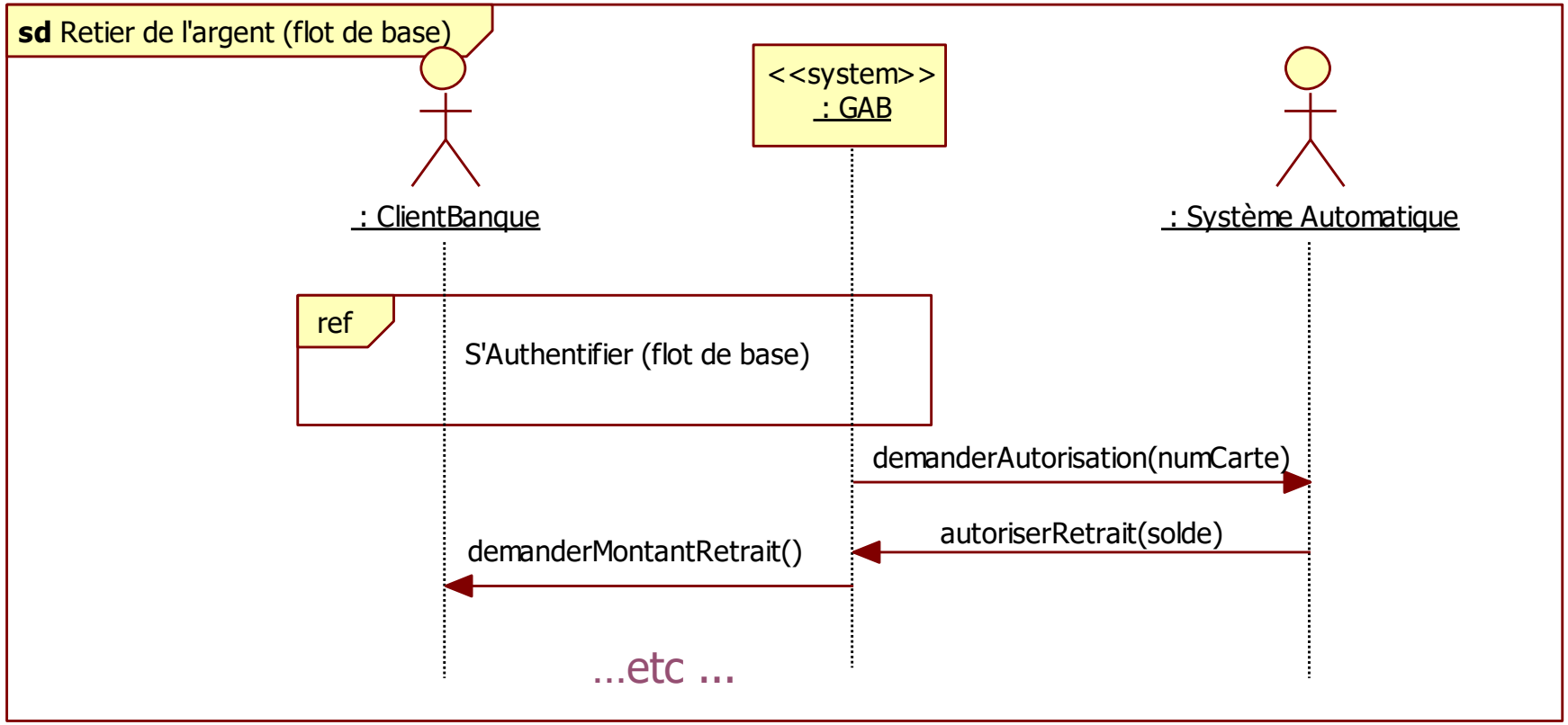


sd est le type du diagramme et signifie **sequence diagram**
Le cadre n'est pas obligatoire lorsque le contexte est clair.

Référencement d'interactions avec l'opérateur `ref`

Il est possible de factoriser des parties de comportement et
d'alléger un diagramme de séquence
en utilisant un cadre de diagramme ayant **ref** comme mot clé

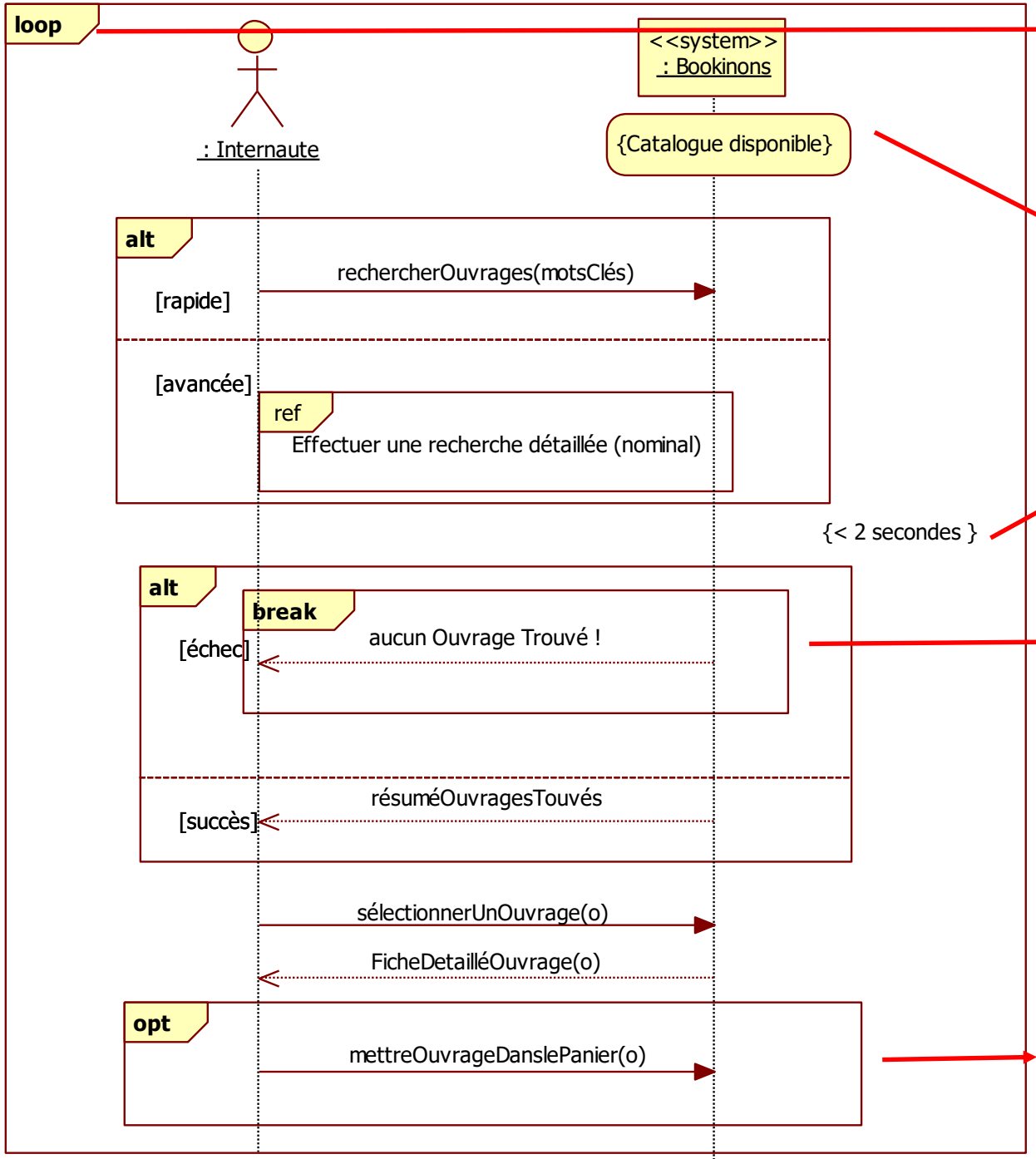
Le diagramme de séquence ***`Retirer de l'argent`*** *fait* ***réf****érence au diagramme de séquence* ***`S'authentifier`***



Une **réf****érence** (*interaction occurrence*) peut être vue comme
un pointeur ou un raccourci vers un autre diagramme de séquence existant

Diagramme de séquence : Exemple (notation UML 2.0)

Isabelle BLASQUEZ



La recherche d'ouvrage est répétable à volonté

Pré-conditon rajoutée sous forme d'une **contrainte d'état**

Contrainte de temps

Exception : un break interrompt la séquence (*sélection et mise dans le panier ne seront pas possible si passage dans le break*)

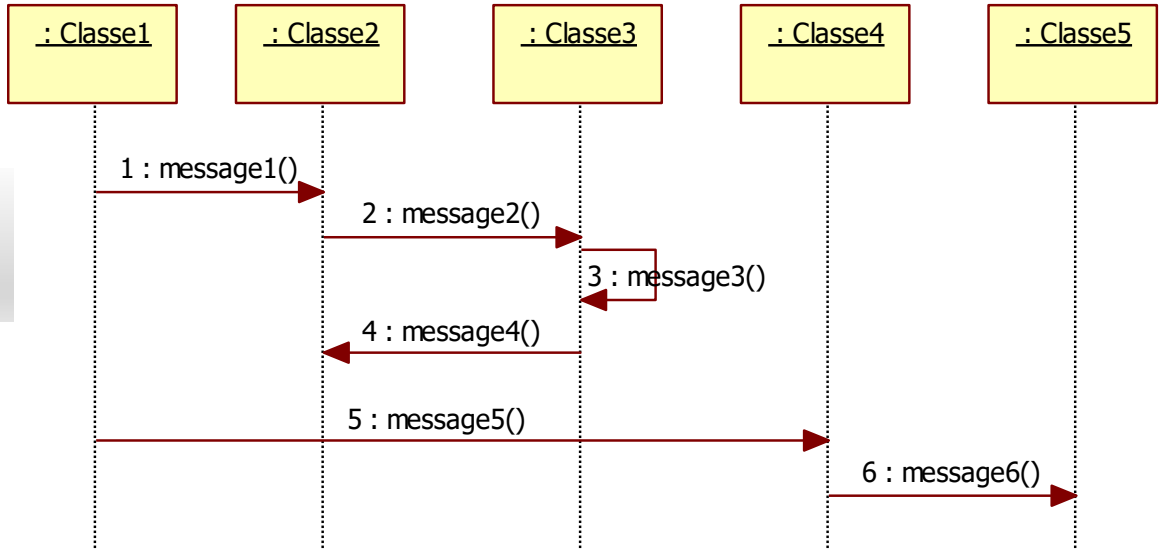
On peut *éventuellement* (*optionnellement*) mettre l'ouvrage trouvé dans le panier

Diagramme de communication

Isabelle BLASQUEZ

Diagramme Séquence vs Diagramme Communication

Diagramme de séquence
représentation **temporelle**



Les AGL permettent de convertir automatiquement un diagramme en un autre

Diagramme de communication
représentation spatiale (structurelle)

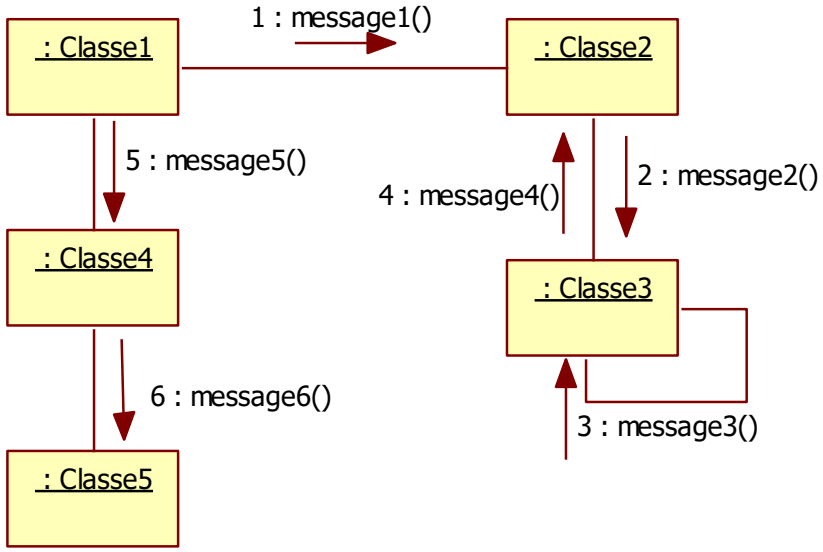


Illustration de la communication téléphonique

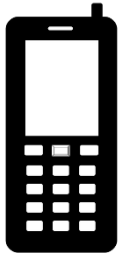


Diagramme de communication
représentation spatiale (structurelle)

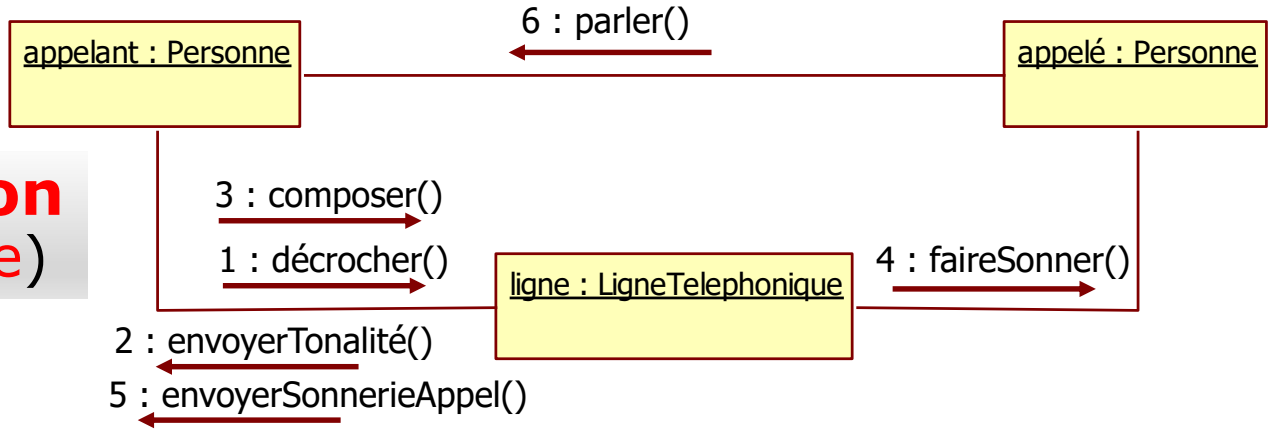
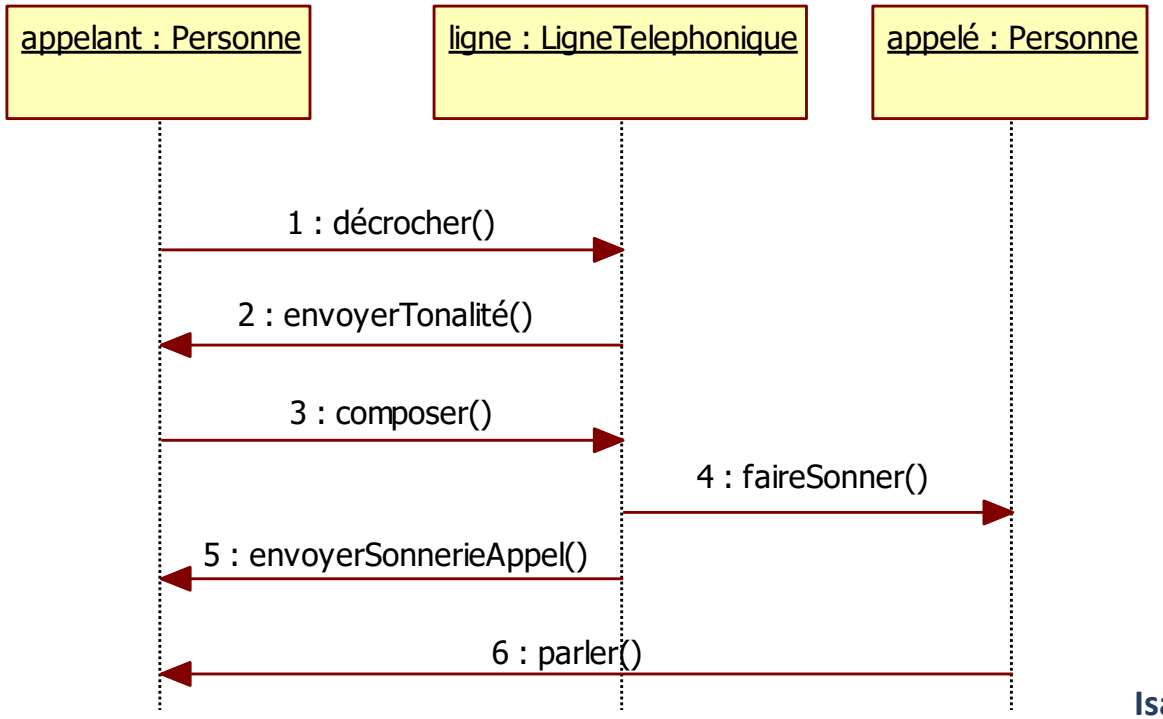


Diagramme de séquence
représentation **temporelle**



Utilisation des diagrammes de séquence

Isabelle BLASQUEZ

Diagramme de séquence en tant qu'outil de documentation ...

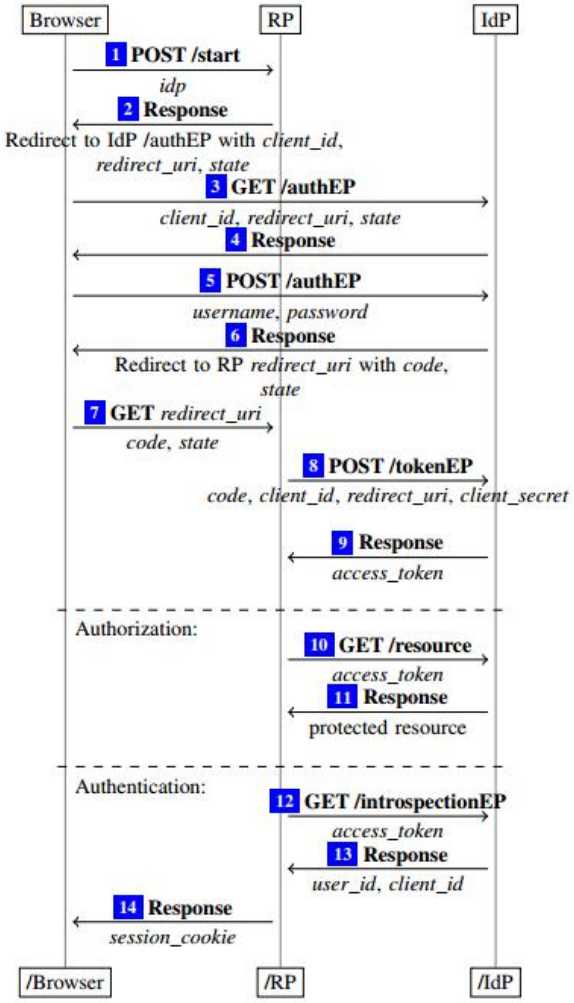


Figure 1: OAuth 2.0 authorization code mode. Note that data depicted below the arrows is either transferred in URI parameters, HTTP headers, or POST bodies.

Expliquer
un traitement, un protocole...

Extrait : https://blog.acolyer.org/2016/11/07/a-comprehensive-formal-security-analysis-of-oauth-2-0/

DS en tant qu'outil de documentation : Exemple

L'idée : déclencher un paiement par la voix :
What if a merchant could **accept Apple Pay transactions using only their voice**?

voir la video ici

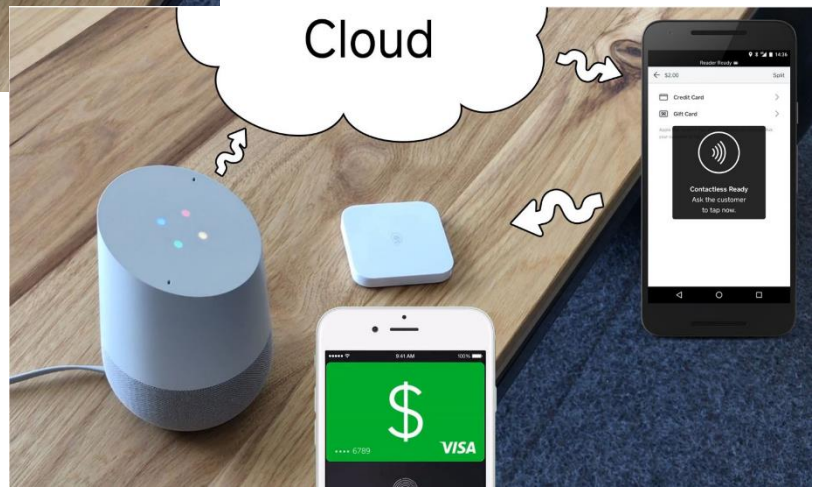
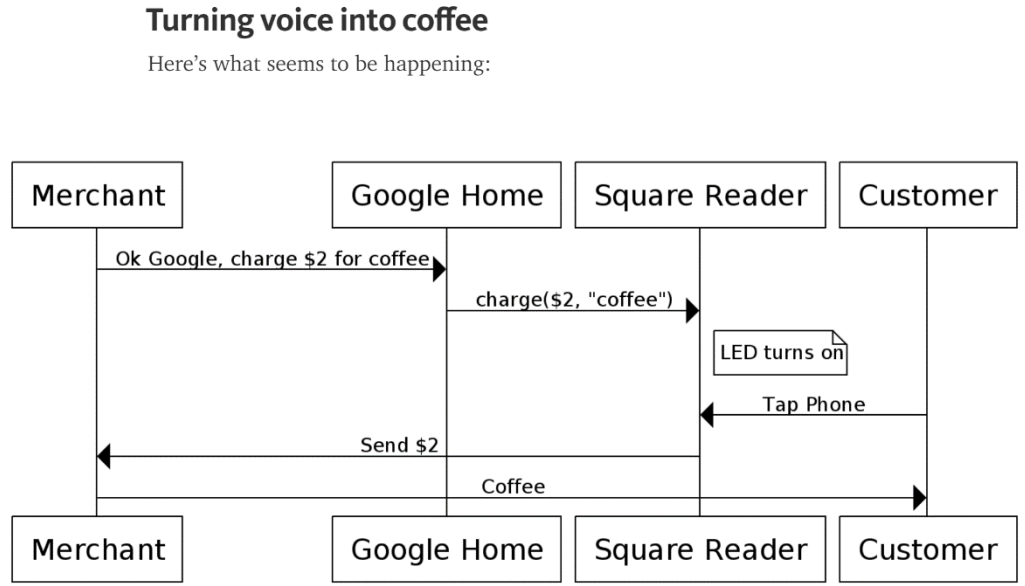


Diagramme de séquence système pour expliquer le principe de fonctionnement (gros grain)



Google Home to activate the Square Contactless Reader and take a real Apple Pay transaction backed by a Square Cash virtual card (only public APIs)

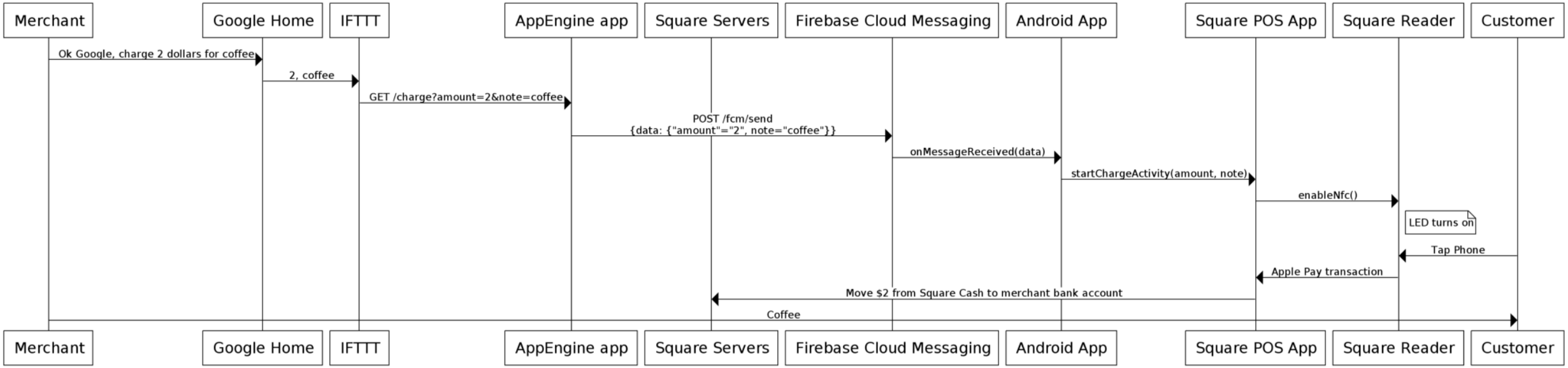
Extrait : https://medium.com/square-corner-blog/ok-google-charge-2-dollars-for-coffee-4d7fdbacd6ef#.auf0vbdn7

DS en tant qu'outil de documentation : Exemple

Diagramme de séquence pour détailler le traitement

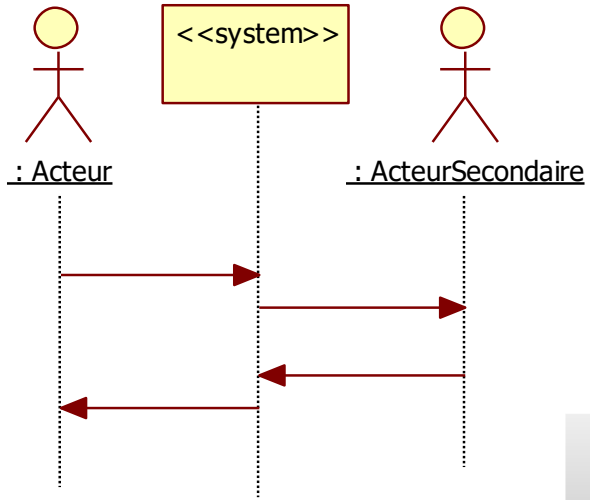
The Full Picture

Now that we know the steps needed to turn voice into coffee, we can update our initial sequence diagram to include all the interactions taking place:

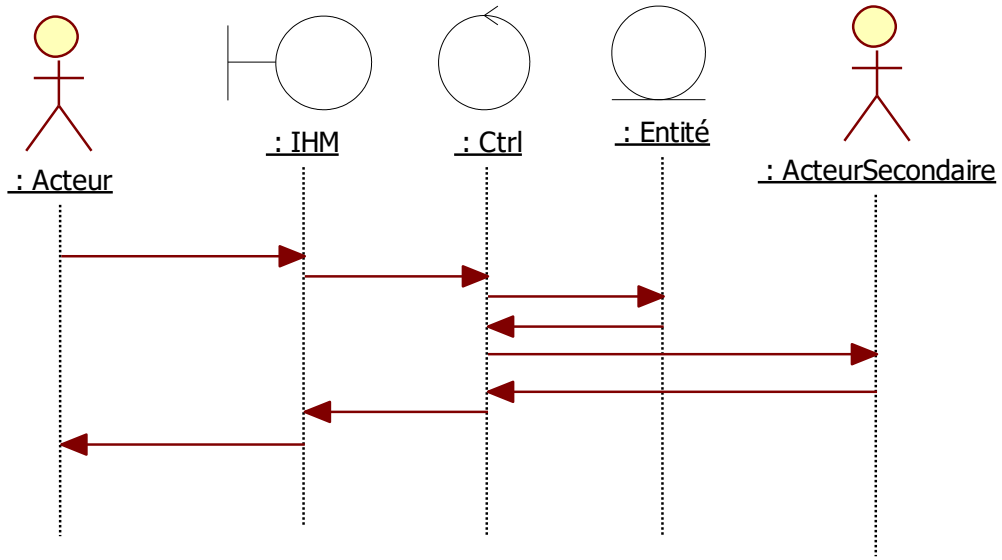


Pour tout savoir, tout comprendre et recoder ce projet de [Pierre-Yves Ricau](#) par vous-même, rendez-vous sur :
https://medium.com/square-corner-blog/ok-google-charge-2-dollars-for-coffee-4d7fdbacd6ef#.auf0vbdn7

Diagramme de séquence en tant qu'outil de modélisation pour les phases d'Analyse et de Conception



Ouvrons la boîte !



Analyse
système = boîte noire

Vue externe du système
(aspect fonctionnel : QUOI)

Conception préliminaire :
Apparition de composants logiciels

Vue interne du système
(aspect dynamique : QUAND)

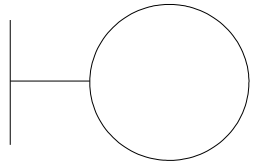
Isabelle BLASQUEZ

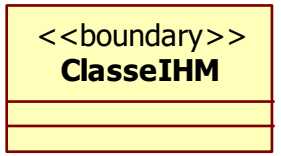
Stéréotypes de Jacobson

Isabelle BLASQUEZ

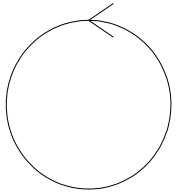
Typologie des stéréotypes de Jacobson

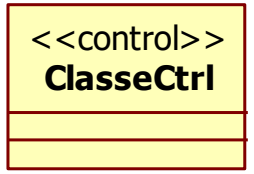
Les dialogues (<<boundary>>) sont les classes qui permettent l'interaction entre l'application et ses utilisateurs. Il s'agit des écrans proposés à l'utilisateur



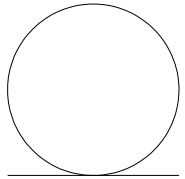


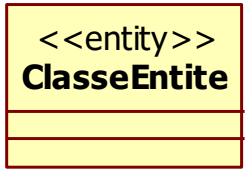
Les contrôles (<<control>> ou contrôleur) sont les classes qui contiennent la dynamique de l'application.





Les **entités** (<<entity>>) sont les classes qui représentent les concepts métiers. Elles sont souvent persistantes.





Utilisation des stéréotypes de Jacobson pour la modélisation d'un scénario via le pattern MVC

(**M**odèle **V**ue **C**ontrôleur)

Isabelle BLASQUEZ

Maquette pour un cas d'usage qui consiste à

Rechercher un Ouvrage à partir de mots clés

Isabelle BLASQUEZ

Scénario à modéliser (flot de base)

Rechercher un Ouvrage à partir de mots clés

1. Le système recherche les informatives relatives à la recherche (Nouveautés, Meilleures ventes,…) et affiche l'écran de recherche …

2. L'Internaute saisit un ou plusieurs mots-clés (un thème, un titre, un auteur, un nom d'auteur) et valide.

3. Le système recherche dans le catalogue les ouvrages pouvant correspondre à la demande de l'utilisateur.

4. Le système trie les ouvrages dans l'ordre souhaité.

5. Le système affiche dans une page de résultat un résumé des ouvrages trouvés.

6. L'Internaute sélectionne un ouvrage.

7. Le système recherche le détail de l'ouvrage.

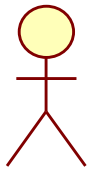
8. Le système affiche une fiche détaillée de l 'ouvrage qui contient :
   - une image de l 'ouvrage, le titre, l'auteur
   - l'éditeur, l'isbn, la langue, la date de parution
   - le prix et la disponibilité.

9. L'Internaute choisit de quitter.

10. Le système ferme le Use Case.

Mise en place du diagramme de séquence (pattern MVC)

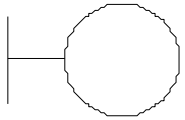
L'**acteur** principal du scénario



: Internaute

L'acteur enverra des messages mais n'en recevra jamais

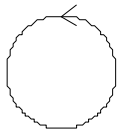
Vue permet à l'acteur d'interagir avec le système via une IHM



: RechercherOuvrageIHM

<<boundary>>
Par convention :
nom du UC suivi de IHM

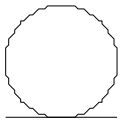
Contrôleur : contrôle et orchestre l'application



: RechercherOuvrageCtrl

<<control>>
Par convention :
nom du UC suivi de Ctrl

Métier :classes métiers que le diagramme de séquence va permettre de découvrir ...



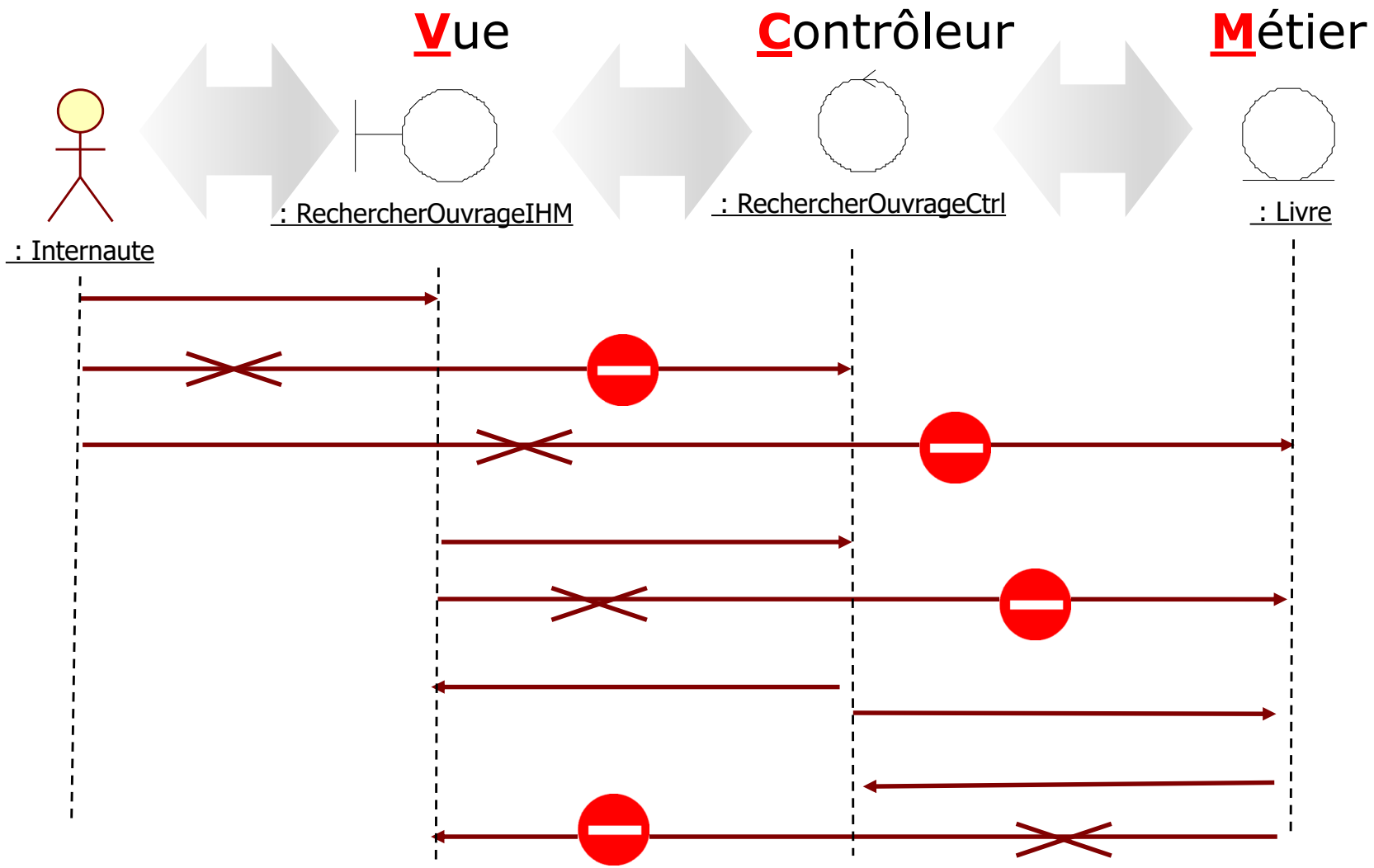
: Livre

<<entity>>

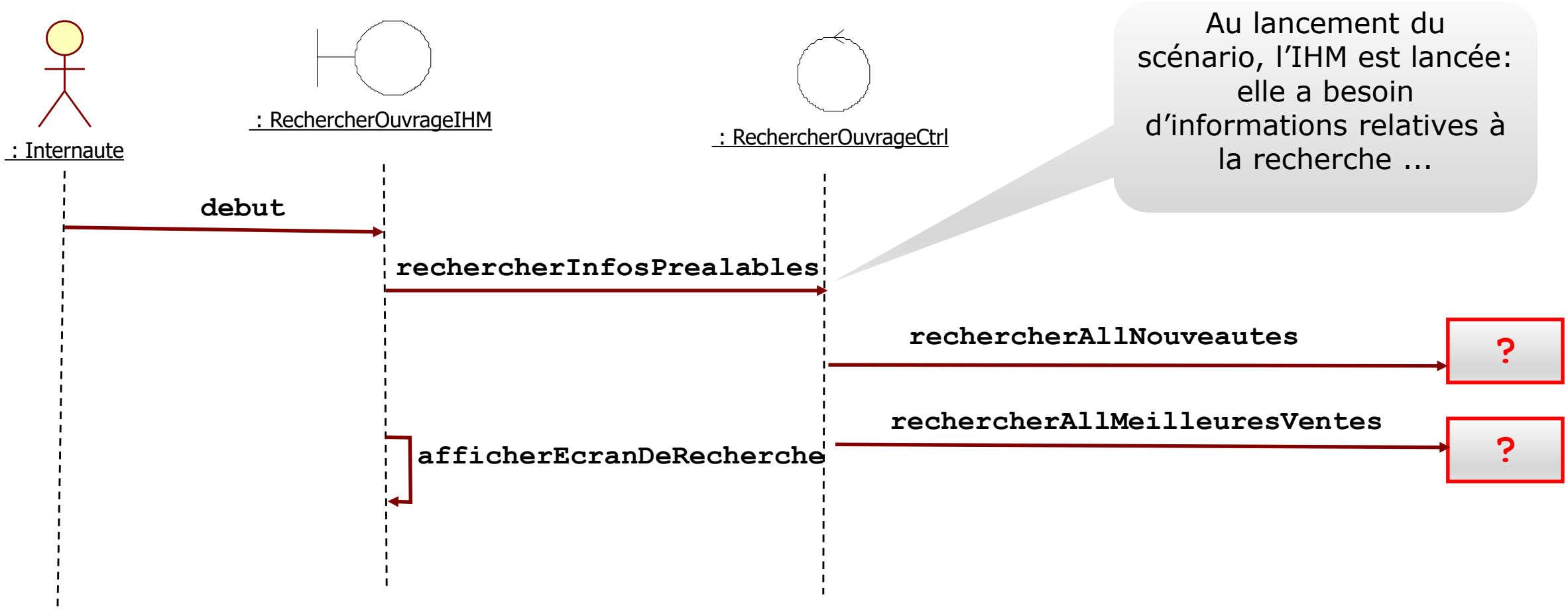
(… mais aussi des <<interface>> et autres <<control>>)

Rappel des règles de construction d'un pattern d'analyse MVC

A retenir : Les messages ne devront pas "sauter" de couche !!!

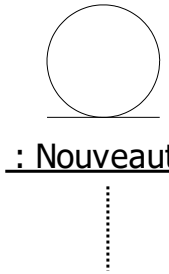


1. Le système recherche les informatives relatives à la recherche (Nouveautés, Meilleures ventes,…) et affiche l'écran de recherche



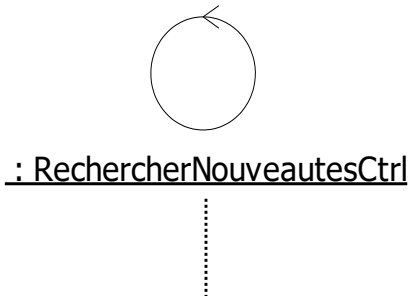
Comment récupérer les nouveautés ?

Lorsqu'un nouvel objet est identifié, 3 possibilités sont envisageables



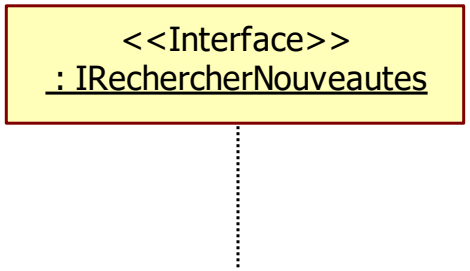
Une **classe métier** (<<entity >>)

→ classe de « notre responsabilité »
→ appel à un service à implémenter au sein du UC



Un **contrôleur** « secondaire » (<<control>>)

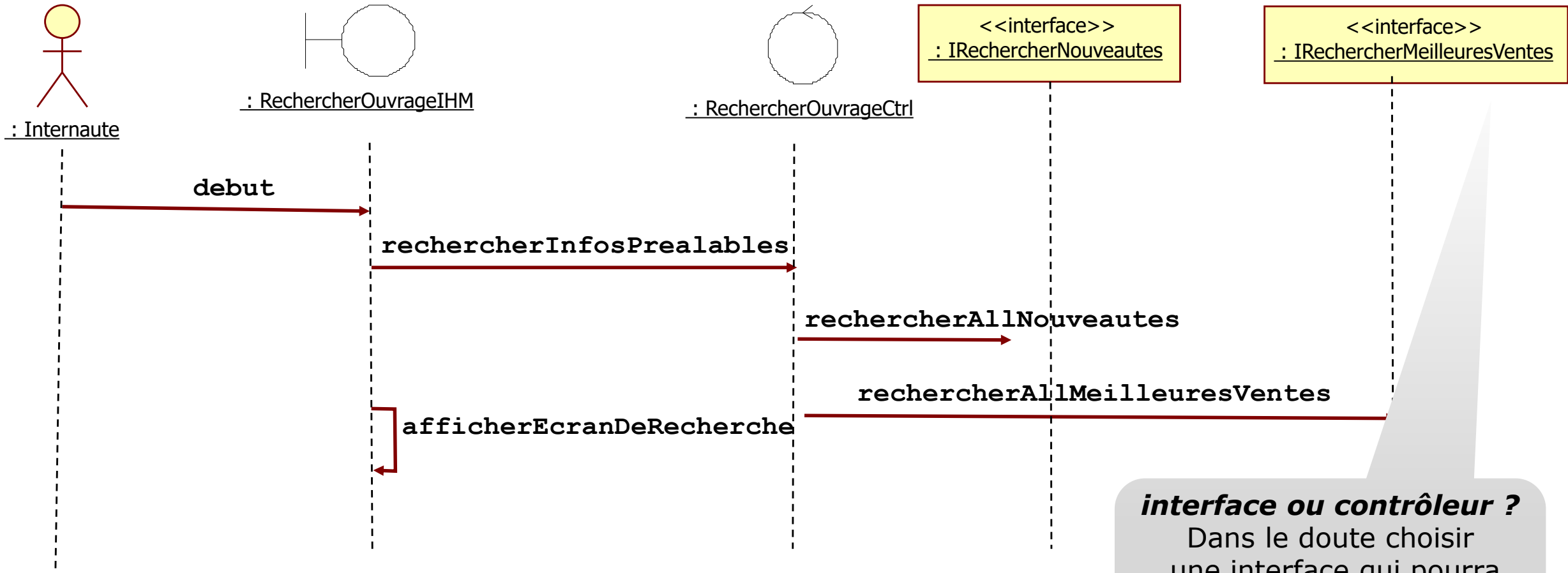
→ référentiel INTERNE
→ appel à un **service interne** réalisée par un autre UC de l'application.



Une **interface** (<<interface >>)

→ référentiel EXTERNE
→ appel à un **service externe** (hors périmètre)

1. Le système recherche les informatives relatives à la recherche (Nouveautés, Meilleures ventes,…) et affiche l'écran de recherche

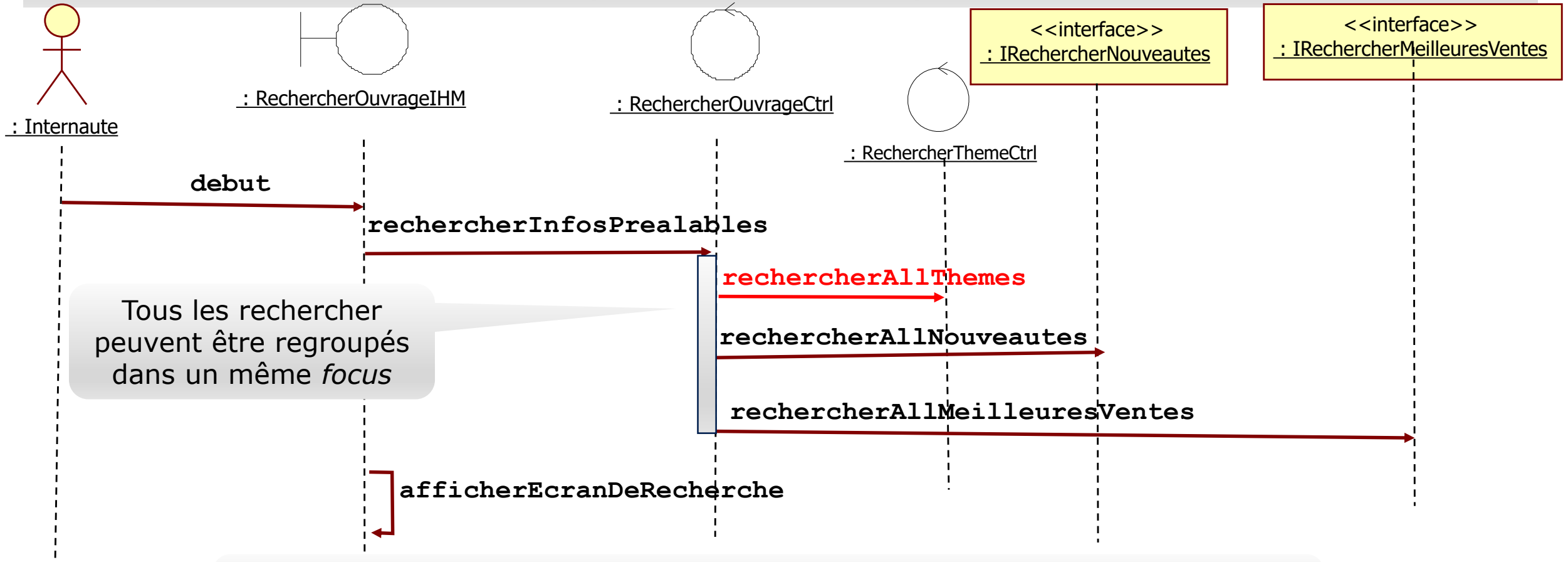


interface ou contrôleur ?
Dans le doute choisir une interface qui pourra ensuite être affinée en contrôleur

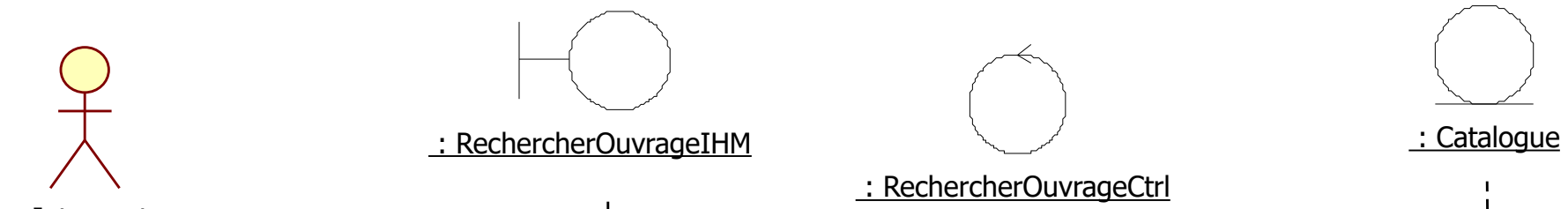
Pas de message de retour pour ne pas alourdir le diagramme :
Rappel: Le retour est implicite avec un appel synchrone !

Chargement de toutes les données préalables avant l'affichage d'un écran …

Avant d'afficher l'écran de recherche, il faut ***s'assurer d'avoir chargé toutes les données qui vont apparaître sur cet écran***
(notamment les données qui viendront « remplir » les listes déroulantes…)



interface ou contrôleur ?
Choix (subjectif) d'un contrôleur : on suppose que `Theme` donnera lieu au sein de cette même application (référentiel Interne) à un UC de type CRUD



2. L'Internaute saisit un ou plusieurs mots-clés (un thème, un titre, un auteur, un nom d'auteur) et valide.

`saisirMotsCles(chaine)`

3. Le système recherche dans le catalogue les ouvrages pouvant correspondre à la demande de l'utilisateur.

`rechercherLivresParMotsCles(chaine)`

`rechercherLivresParMotsCles(chaine)`

L 'IHM ***délègue*** la recherche au contrôleur qui va ensuite lui-même ***déléguer*** la recherche à une entité. Laquelle ?...

… Le `catalogue` qui semble le mieux placé pour effectuer une recherche parmi une *collection d'ouvrages* qu'il aura construit dynamiquement

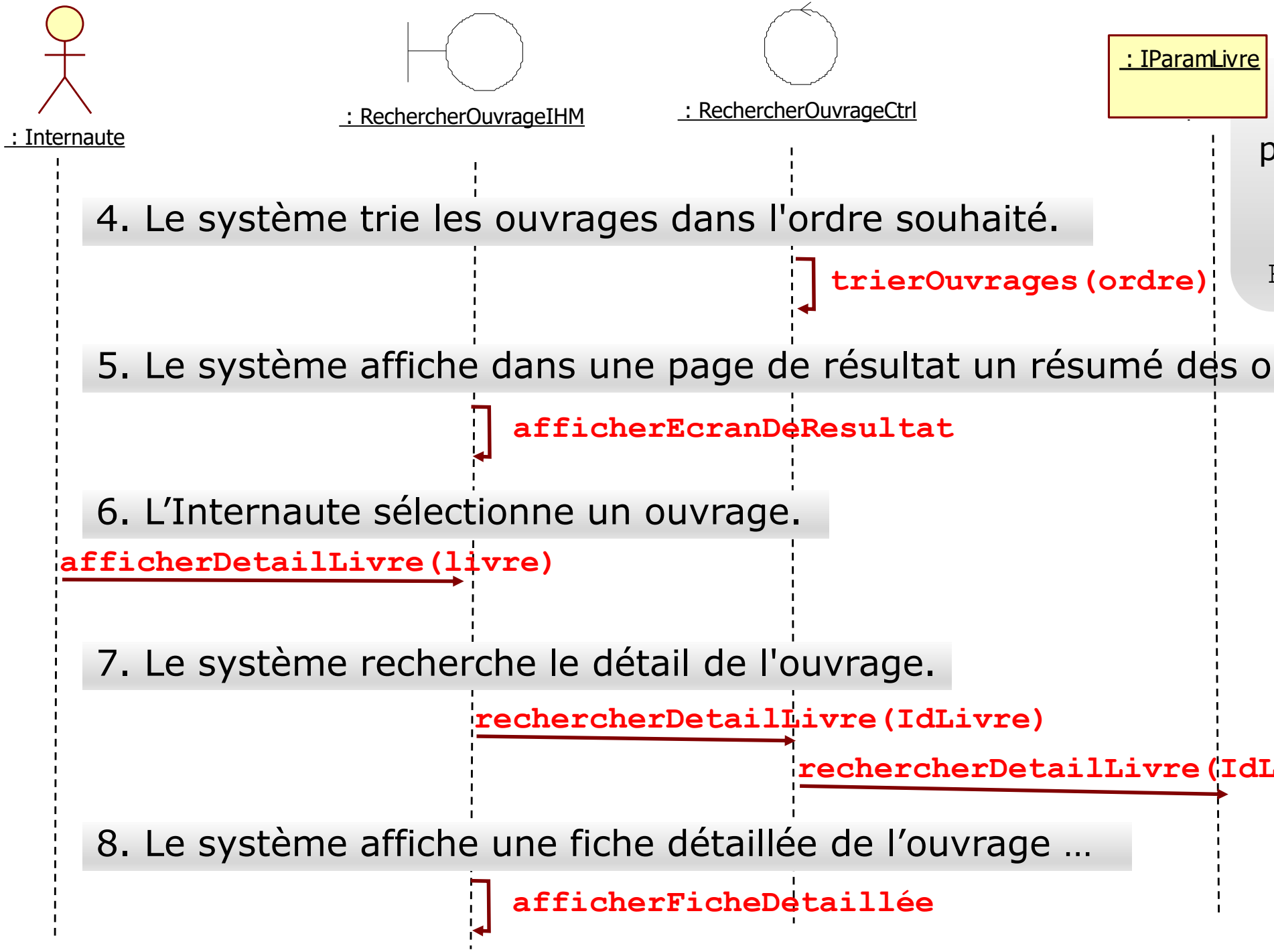
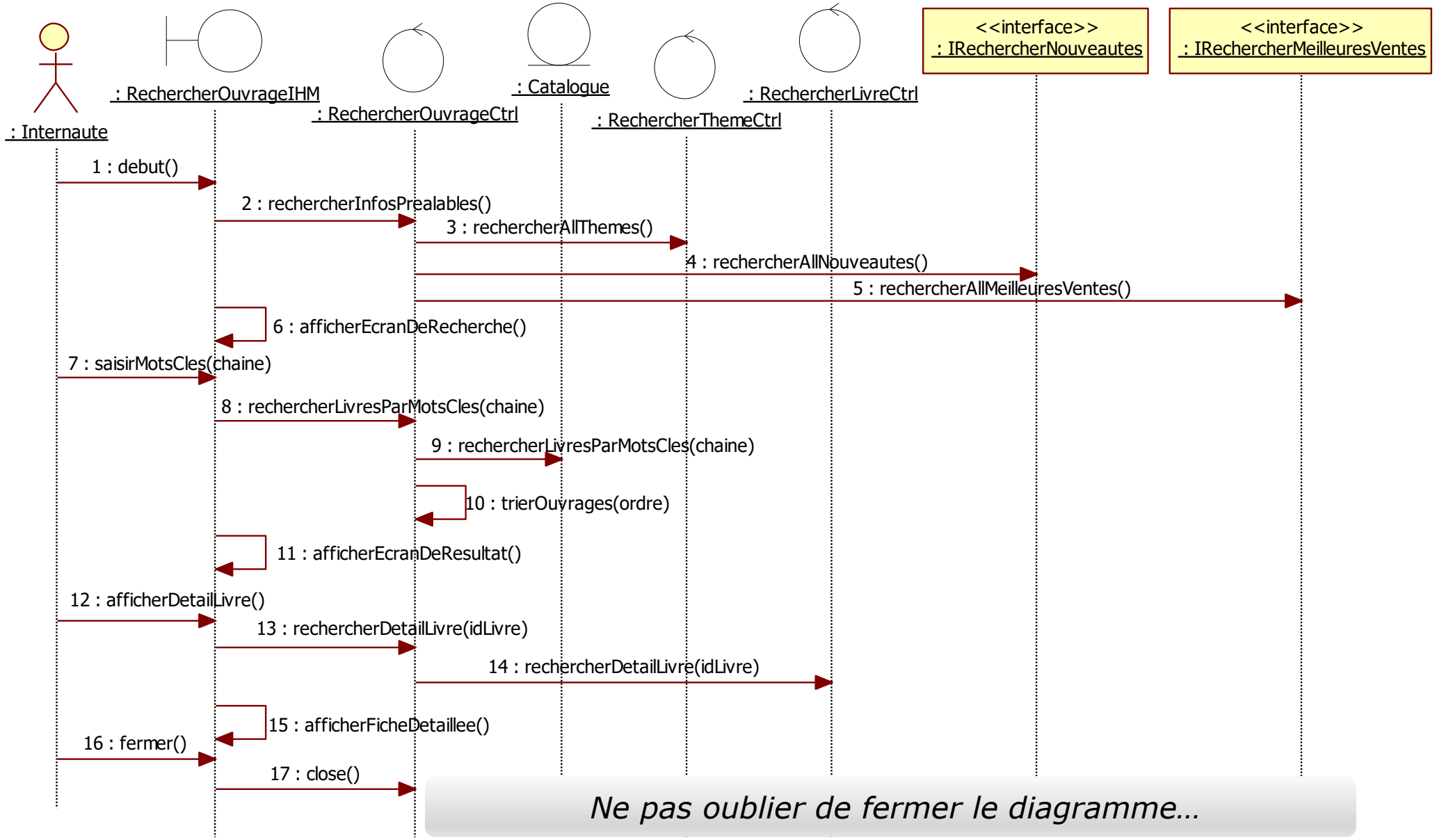

: Internaute
: RechercherOuvrageIHM
: RechercherOuvrageCtrl
: IParamLivre
Choix d'une interface IParamLivre pour une 1ère approche, qui sera certainement transformée plus tard en un contrôleur RechercherLivreCtrl
4. Le système trie les ouvrages dans l'ordre souhaité.
trierOuvrages(ordre)
5. Le système affiche dans une page de résultat un résumé des ouvrages trouvés.
afficherEcranDeResultat
6. L'Internaute sélectionne un ouvrage.
afficherDetailLivre(livre)
7. Le système recherche le détail de l'ouvrage.
rechercherDetailLivre(IdLivre)
rechercherDetailLivre(IdLivre)
8. Le système affiche une fiche détaillée de l'ouvrage …
afficherFicheDetaillée

Diagramme de séquence du Scénario (flot de base)

Rechercher un Ouvrage à partir de mots clés



Ne pas oublier de fermer le diagramme…

Intérêt de construire un diagramme de séquence pendant la phase de conception

Du point de vue de la **conception objet**,
lancer un message synchrone à un objet revient à
provoquer **l'exécution d'une opération**
définie dans la classe de cet objet.

Le diagramme de séquence permet
d'enrichir le diagramme de classes
en identifiant de nouvelles **opérations**
et de nouvelles **classes**

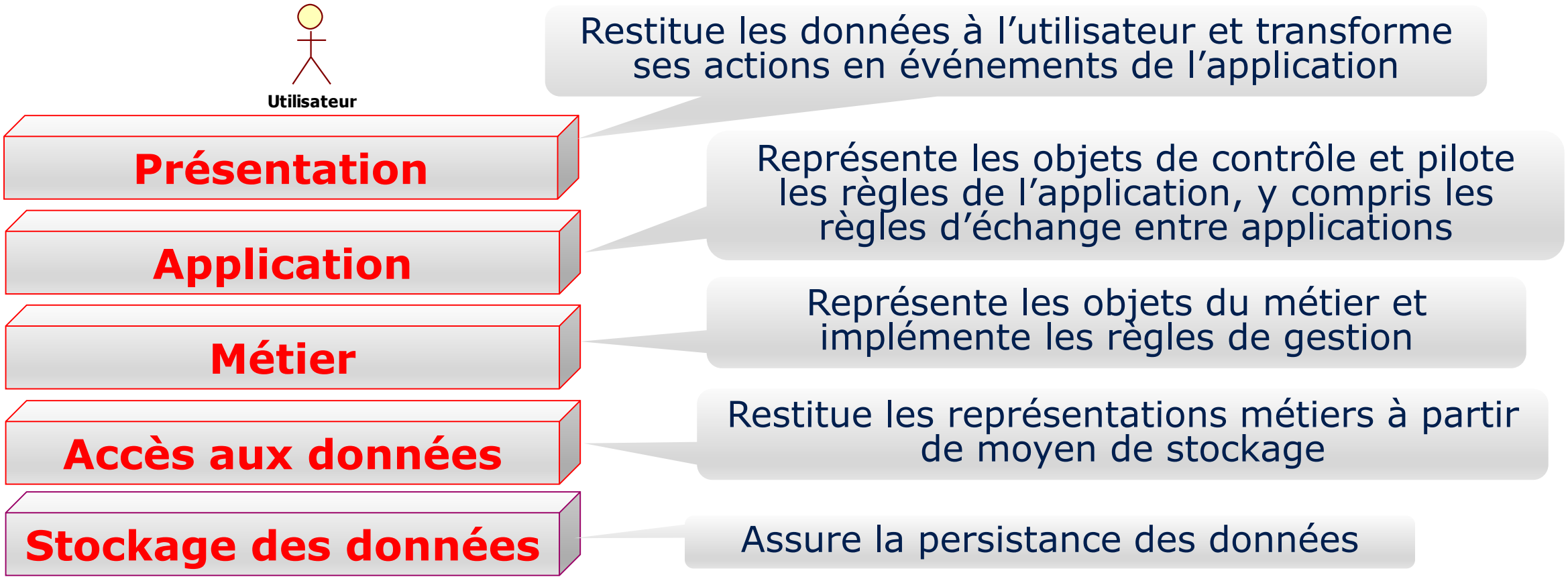
Notion d’architecture logicielle

Isabelle BLASQUEZ

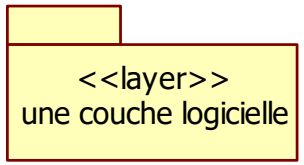
Notion de couches logicielles

Une **couche logicielle** représente un ensemble de spécifications ou de réalisations qui expriment ou mettent en œuvre des responsabilités techniques et homogènes pour un système logiciel

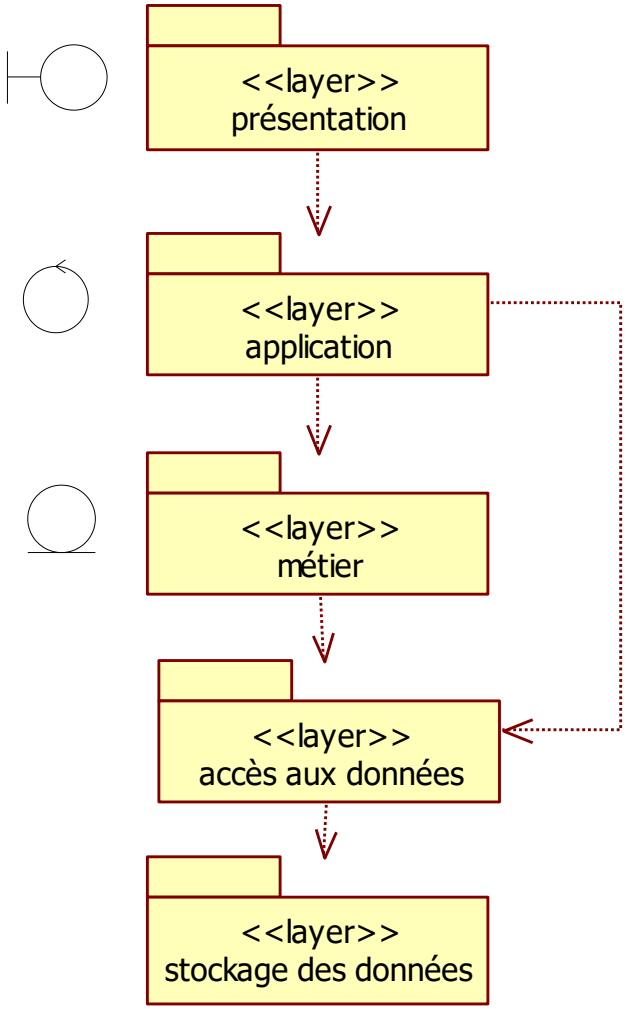
Architecture en 5 couches : 1 responsabilité / couche



Les couches logicielles dans un modèle UML



Le **package** (élément UML permettant de regrouper d'autres éléments UML) associé au stéréotype **<<layer>>** permet de modéliser une couche



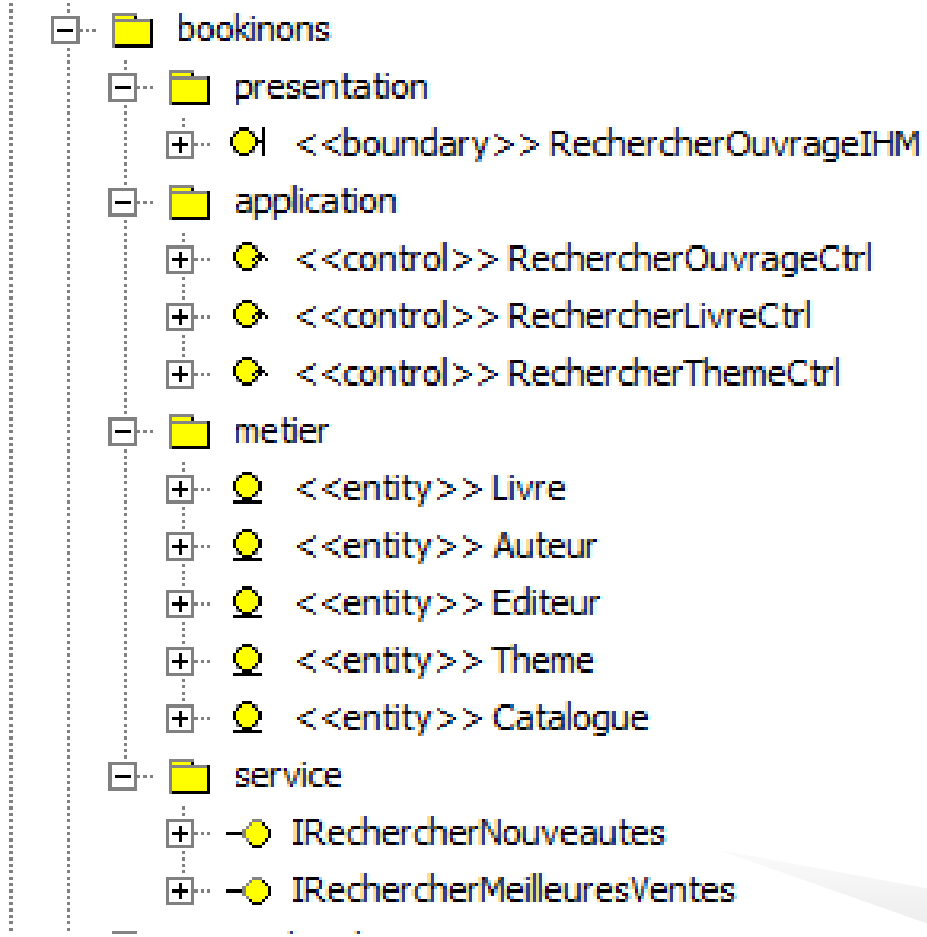
Un **diagramme de packages** permet de montrer les **couches** et leurs **dépendances**.

Les **dépendances** indiquent les interactions (*messages*) qui peuvent être envoyées d'une couche à l'autre

L'intérêt d'une architecture en couche est de pouvoir répondre à un **critère d'évolutivité**

⇒ les dépendances entre couches doivent être minimisées afin de **favoriser un faible couplage**

Les couches logicielles dans Bookinons



Les objets stéréotypés **<<interface>>** ont été regroupés dans une couche appelée **service**.

Annexes

Opérateurs pour un fragment combiné d'interaction

Choix et boucles

→ Alternatif (**alt**) : plusieurs fragments possibles. Seul celui dont la condition est vraie s'exécute
→ Optionnel (**opt**) : ne s'exécute que si la condition est vraie
→ Exception (**break**) : la fin de ce fragment interrompt la séquence entière
→ Itération (**loop**) : le fragment peut s'exécuter plusieurs fois selon les conditions de la garde

Parallélisation

→ Parallèle (**par**) : chaque fragment est exécuté en parallèle
→ Critique (**critical**) : le fragment ne peut avoir qu'un thread qui s'exécute à la fois

Contrôle de l'envoi de messages

→ Insignifiant (**ignore**) : les messages du fragment sont considérés comme insignifiants
→ Signifiant (**consider**) : seuls les messages du fragment sont considérés comme signifiants
→ Assertion (**assert**) : seul l'interaction du fragment est considérée comme valide
→ Invalide (**negative**) : le fragment représente une interaction invalide

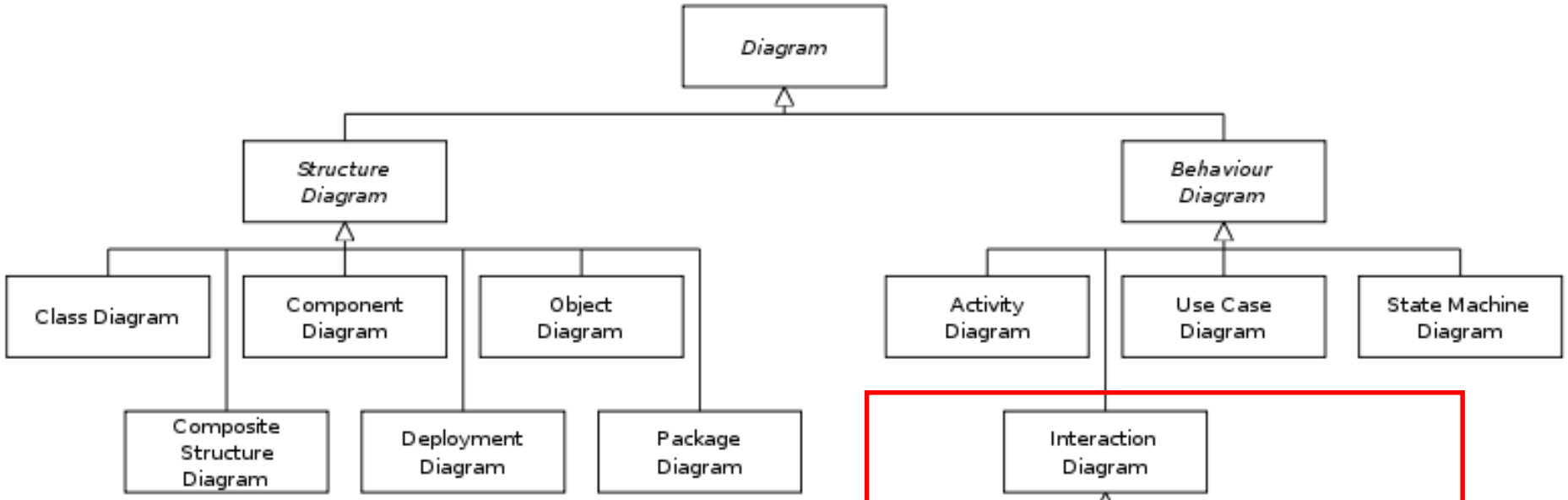
Fixe l'ordre d'envoi des messages

→ Séquencement faible (**seq**) : les sous-fragment s'exécutent dans un ordre quelconque
→ Séquencement fort (**strict**) : les sous-fragments s'exécutent selon l'ordre d'apparition.

Référence

→ Référence (**ref**) : référencement d'une interaction
→ Diagramme de séquence (**sd**) : référencement d'un diagramme de séquence

Les 4 diagrammes d'interaction

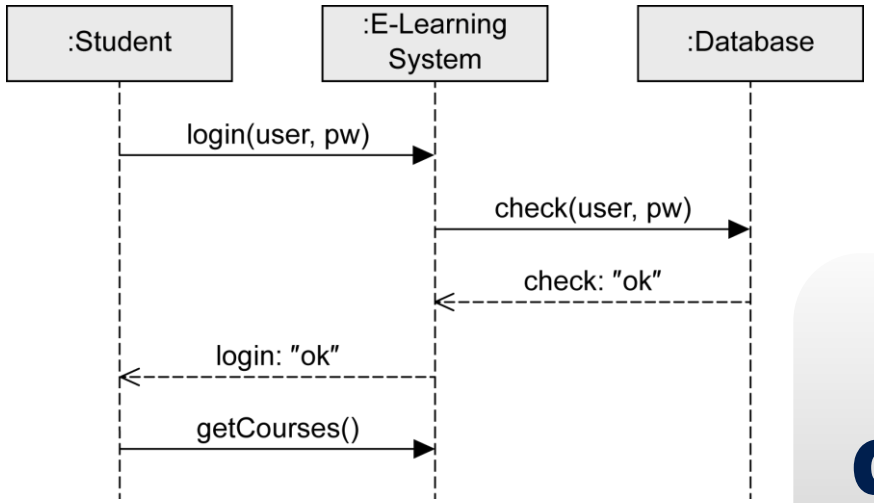


représentation **temporelle**

représentation **spatiale**

Les 4 diagrammes d'interaction

Diagrammes de séquence



Diagrammes de communication

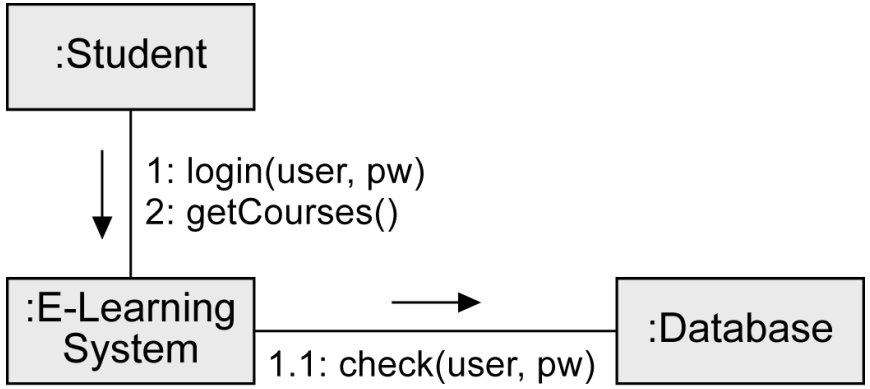


Diagramme global d'interactions

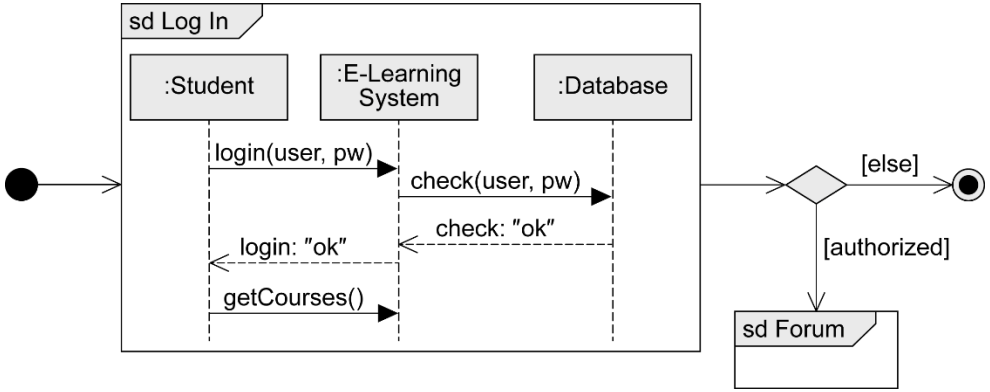
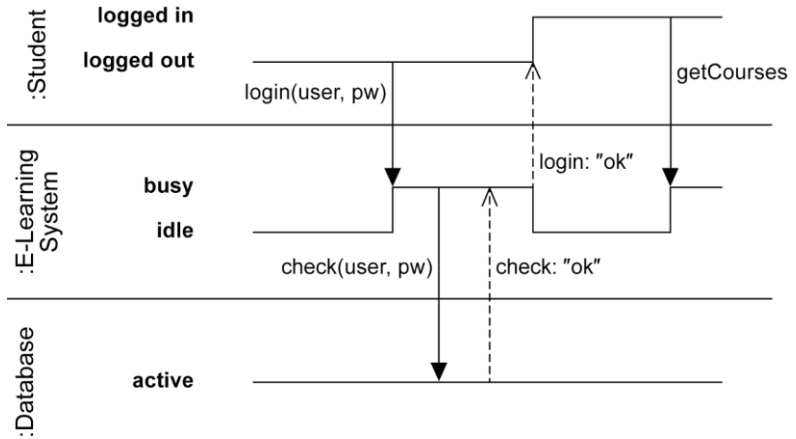
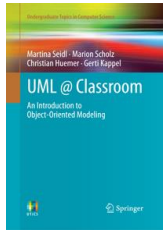


Diagramme de temps



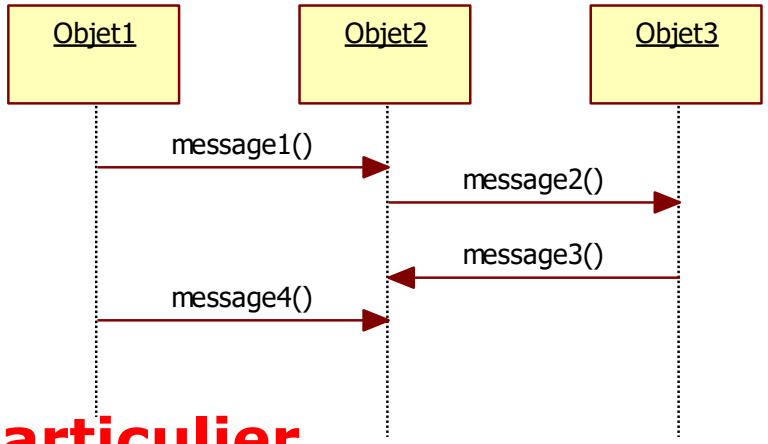


Diagrammes extraits de http://www.uml.ac.at/en/lernen

Isabelle BLASQUEZ

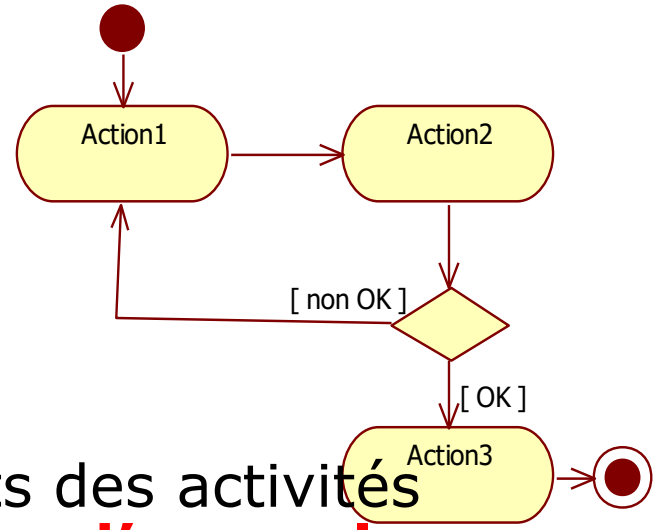
Diagramme de séquence vs Diagramme d'activité

Le **Diagramme de séquence** propose
une représentation temporelle (séquentielle)
du déroulement des traitements
et des interactions entre
les éléments du système et/ou de ses acteurs



⇨ Il permet d'illustrer graphiquement **un scénario en particulier.**

Le **Diagramme d'activité** représente
les règles d'enchaînements des actions et
des décisions au sein d'une activité :
c'est un graphe orienté d'actions et de transitions



⇨ Il permet de documenter graphiquement les enchaînements des activités au sein d'un cas d 'utilisation puisqu'il est possible **d'identifier d'un seul coup d'œil la famille de tous les scénarios du cas d'utilisation** et d'envisager ainsi toutes les possibilités d'exécution offertes par ce Use Case.

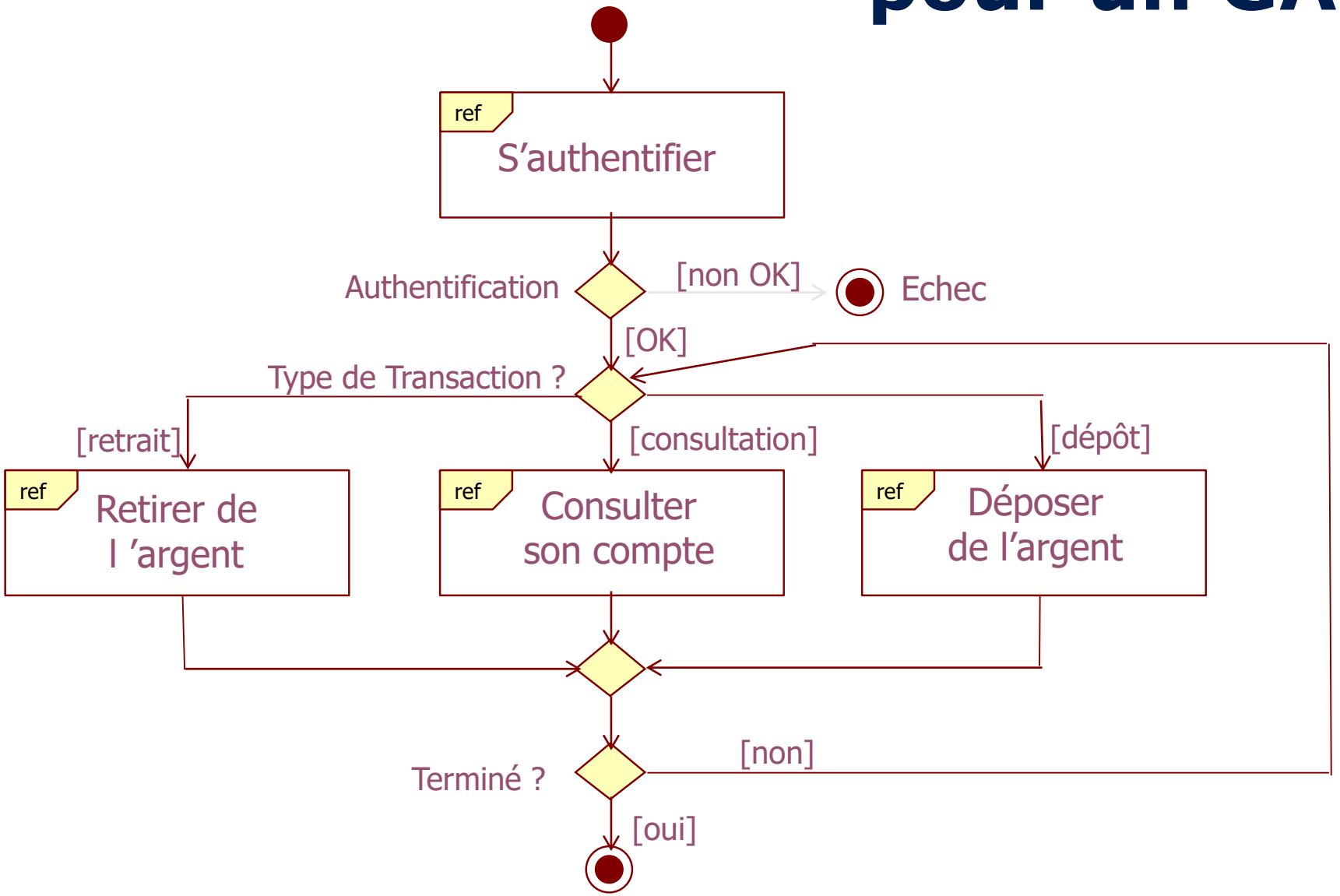
Interaction Overview Diagram

UML 2 a proposé un nouveau type de diagramme, issu de la *fusion* des notations du diagramme d'**activité** et du diagramme de **séquence**, qui est appelé : **Interaction Overview Diagram** (ou diagramme global d'interactions)

L'**Interaction Overview Diagram** permet d'**organiser des interactions** (représentées par exemple par des diagrammes de séquence) **au moyen de noeuds de contrôle** (présents habituellement dans un diagramme d'activités).

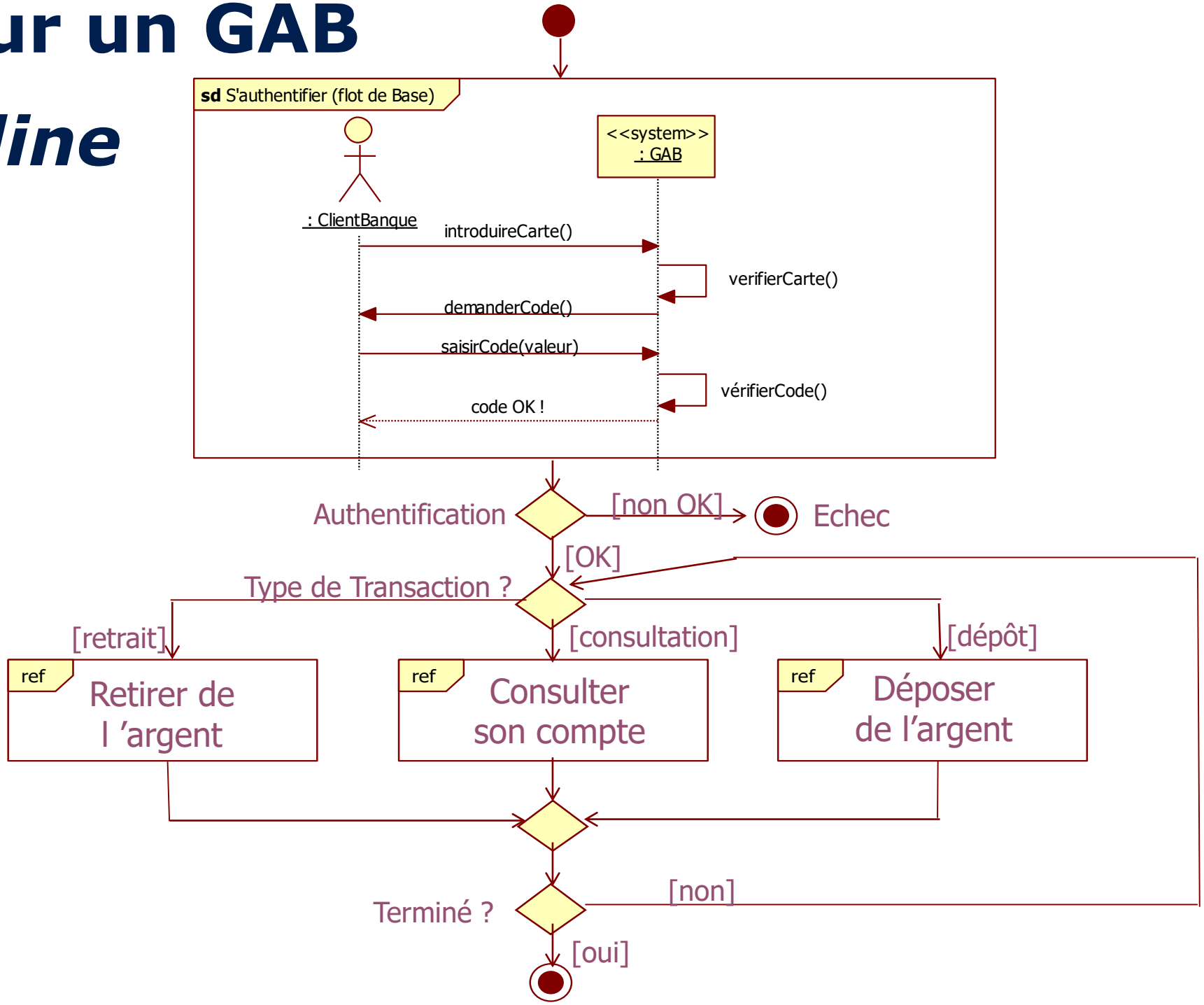
L'**Interaction Overview Diagram** est une sorte de diagramme d'activités où les actions sont remplacées par des interactions.

Interaction Overview Diagram pour un GAB



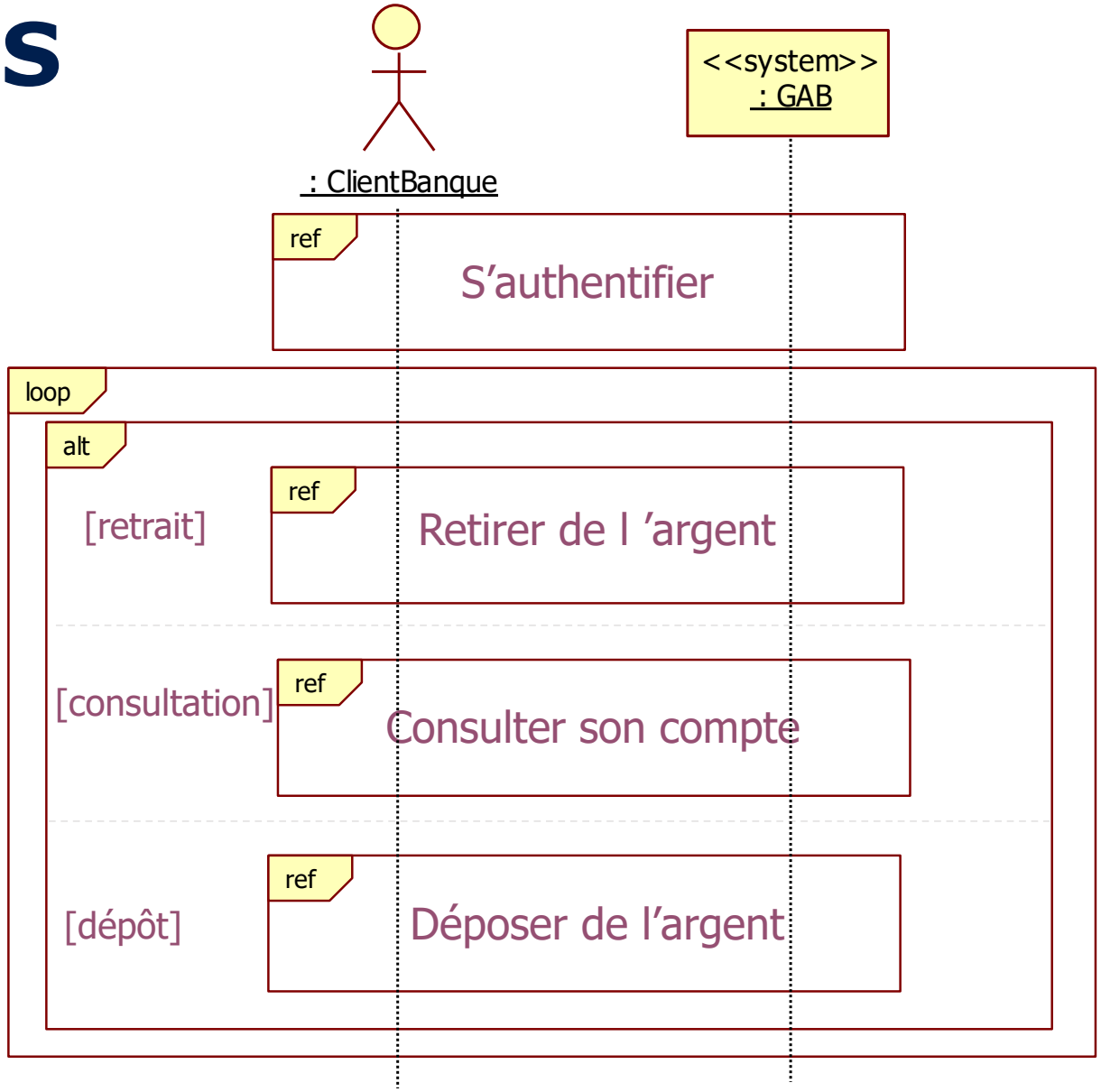
Remarque : il est également possible de remplacer chaque référence (**ref**) par un diagramme de séquence *in line*

IOD pour un GAB avec DSS *in line*



Isabelle BLASQUEZ

IOD vs DSS



La valeur ajoutée de l'Interaction Overview Diagram par rapport au Diagramme de Séquence n'est pas vraiment évidente sur cet exemple. IOD pourra par exemple être utilisé pour la description d'une méthode complexe