Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Mila

Département Math & Informatique

Génie logiciel II

Processus simplifié pour les applications Web (Pascal Roques)

HEDJAZ Sabrine

2019-2020

Processus simplifié

1. Processus simplifié:

- L'auteur propose un processus pour le développement d'applications web, ce processus se situe à mi-chemin entre **UP (Unified Process),** et les méthodes agiles en vogue actuellement, telles que **XP (eXtreme Programming).**

UP (Unified Process)

Agile Modeling

Processus simplifié

2. Le s méthodes Agiles

- **Une méthode Agile** est une approche **itérative** et **collaborative**, capable de prendre en compte les **besoins initiaux** du client et ceux **liés aux évolutions**.
- La méthode Agile se base sur un cycle de développement qui porte **le client au centre**. Le client est impliqué dans la réalisation du **début à la fin** du projet.
- L'implication du client dans le processus permet à l'équipe d'obtenir un feedback régulier afin d'appliquer directement les changements nécessaires. **Cette méthode vise à accélérer le développement d'un logiciel**.

Processus simplifié

3. Le s méthodes UP

- **Le Processus Unifié** est un processus de développement moderne, itératif, efficace sur des projets informatiques de toutes tailles. **Très complet**, il couvre l'ensemble des activités, depuis **la conception** du projet jusqu'à la **livraison de la solution**.
- Intégrant une organisation de projet type, une méthodologie utilisant **UML** et un ensemble de bonnes pratiques cohérentes entre elles, il permet de circonvenir aux problèmes récurrents que rencontrent nombre de réalisations (délais, besoins des utilisateurs, ...)
- UP peut se décliner en fonction de **l'ampleur d'un projet**, de **l'expérience de l'équipe** qui l'assume, de **la nature de la solution à construire**.

Processus simplifié

4. Le s principes des deux méthodes

UP (Unified Process)

- Interactif et incrémental
- Centré sur l'architecture
- Piloté par les risques
- Conduit par les cas d'utilisation

Agile Modeling

- Personne et interaction plus tôt que processus et outil.
- Logiciel fonctionnel plus tôt que documentation.
- Collaboration avec client plus tôt que négociation du contrat.
- Réagir au changement plus tôt que suivi un plan.

Processus simplifié pour le développement des application Web

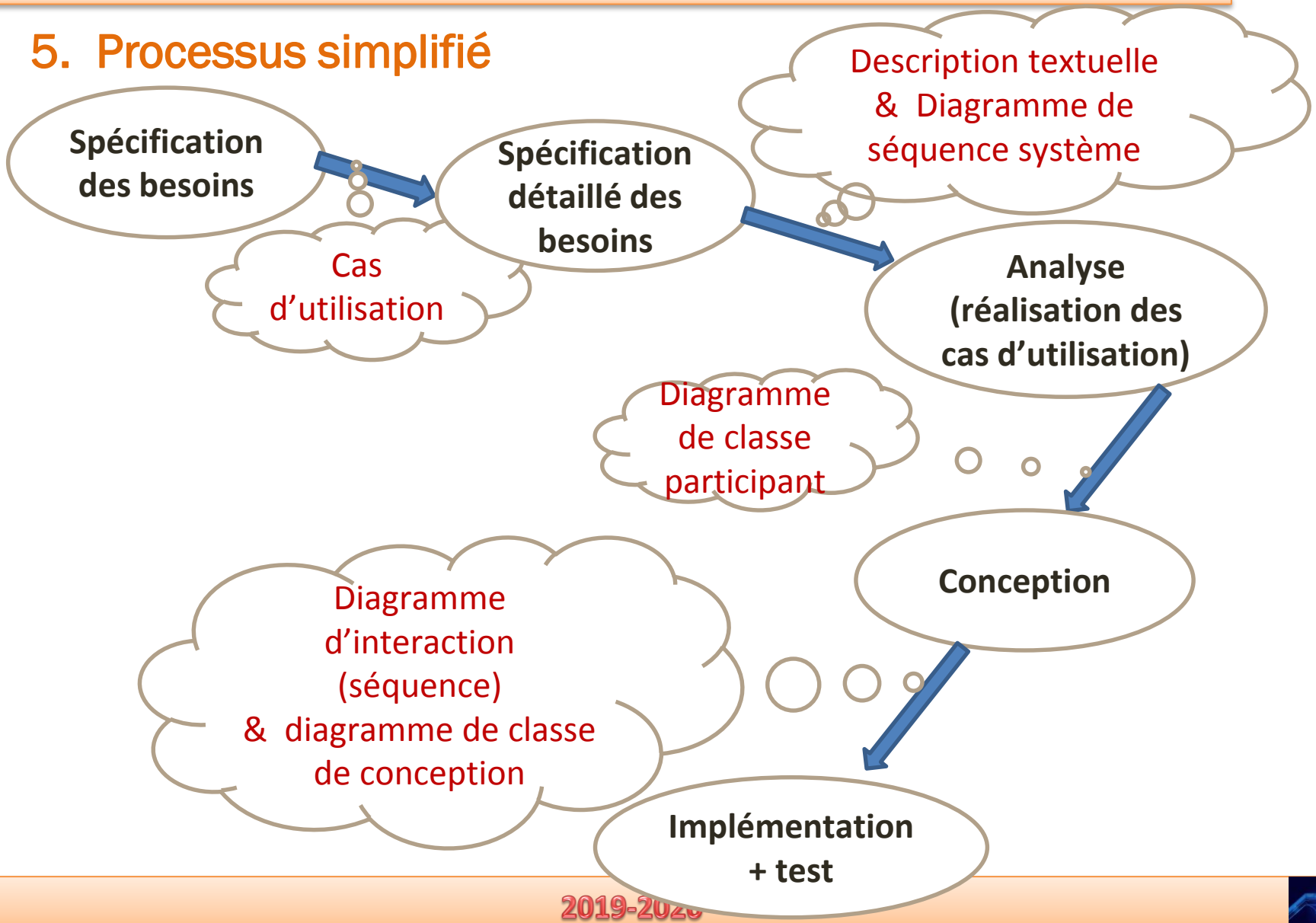
Processus simplifié

5. Processus simplifié

- Conduite par les cas d'utilisation comme les processus UP, mais beaucoup plus simple
- Relativement léger et restreint comme les méthodes agile, mais sans négliger les activités de modélisation en analyse et conception.
- Fondé sur l'utilisation d'un ensemble nécessaire et suffisant du langage UML.
- L'objectif ici est de trouver le meilleur rapport (Qualité/Prix).

Processus simplifié

5. Processus simplifié



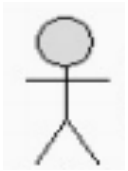
Processus simplifié

5. 1. Spécification des besoins:

5.1.1. Identification des acteurs:

Un acteur représente un rôle joué par une entité externe (**utilisateur humain, dispositif matériel ou autre système**) qui interagit directement avec le système étudié.

Nous appelons **acteur principal** celui pour qui le cas d'utilisation produit un résultat observable. Par opposition, nous qualifions **d'acteurs secondaires** les autres participants du cas d'utilisation.



Acteur principale

Acteur secondaire

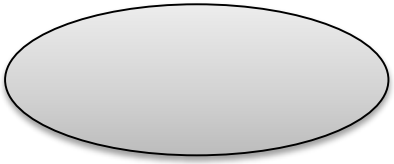
Processus simplifié

5. 1. Spécification des besoins:

5.1.2. Identification des cas d'utilisation

Un cas d'utilisation (use case) représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier.

Nommez les **cas d'utilisation** par un **verbe à l'infinitif suivi d'un complément, du point de vue de l'acteur** (et non pas du point de vue du système).



Cas d'utilisation

Processus simplifié

5. 1. Spécification des besoins:

5.1.3. Identification des relations entre cas d'utilisation

Pour affiner le diagramme de cas d'utilisation, UML définit trois types de relations standardisées entre cas d'utilisation.

- **Une relation d'inclusion: <<include>>** : le cas d'utilisation de base en incorpore explicitement un autre, **de façon obligatoire.**
- **Une relation d'extension: <<extend>>** : le cas d'utilisation de base en incorpore implicitement un autre, **de façon optionnelle.**
- **Une relation de généralisation/spécialisation :** les cas d'utilisation descendants héritent de la description de leur parent commun.

5.1.4. Elaboration du diagramme de cas d'utilisation final

Processus simplifié

5. 2. Spécification détaillée des besoins:

Nous allons maintenant décrire de façon détaillée les cas d'utilisation que nous avons identifiés au chapitre dans la phase précédente. Nous apprendrons ainsi à remplir une **fiche-type** pour chaque cas d'utilisation. Nous complèterons cette **description textuelle** par une **représentation graphique UML** très utile : **le diagramme de séquence « système »**

5.2.1. Description textuelle des cas d'utilisation:

La fiche de description textuelle d'un cas d'utilisation n'est pas normalisée par UML, Les auteurs donc en proposer la fiche type suivante:

Processus simplifié

5. 2. Spécification détaillée des besoins:

5.2.2. Fiche de description textuelle:

Cas d'utilisation	Nom de cas d'utilisation.
Acteur principale	.Les acteurs principales.
Acteur secondaire	Les acteurs secondaires.
Objectif	L'objectif du cas d'utilisation.
Pré-condition	Définissent ce qui doit être vrai en amont du cas d'utilisation pour que celui-ci puisse démarrer.
Post-conduction	Les post-conditions définissent ce qui doit être vrai lorsque le cas d'utilisation se termine.
Scenario nominal	Celui qui satisfait les objectifs des acteurs par le chemin le plus direct de succès.
Scenario alternative	Comprennent tous les autres scénarios, de succès (fin normale) ou d'échec (erreur).
Exigences supplémentaires	Il s'agira de **performance**, de **sécurité** ou **d'ergonomie**. On complètera par exemple la description des scénarios par des copies d'écran de la maquette

Processus simplifié

5. 2. Spécification détaillée des besoins:

5.2.2. Fiche de description textuelle:

Exemple: Ajouter commande

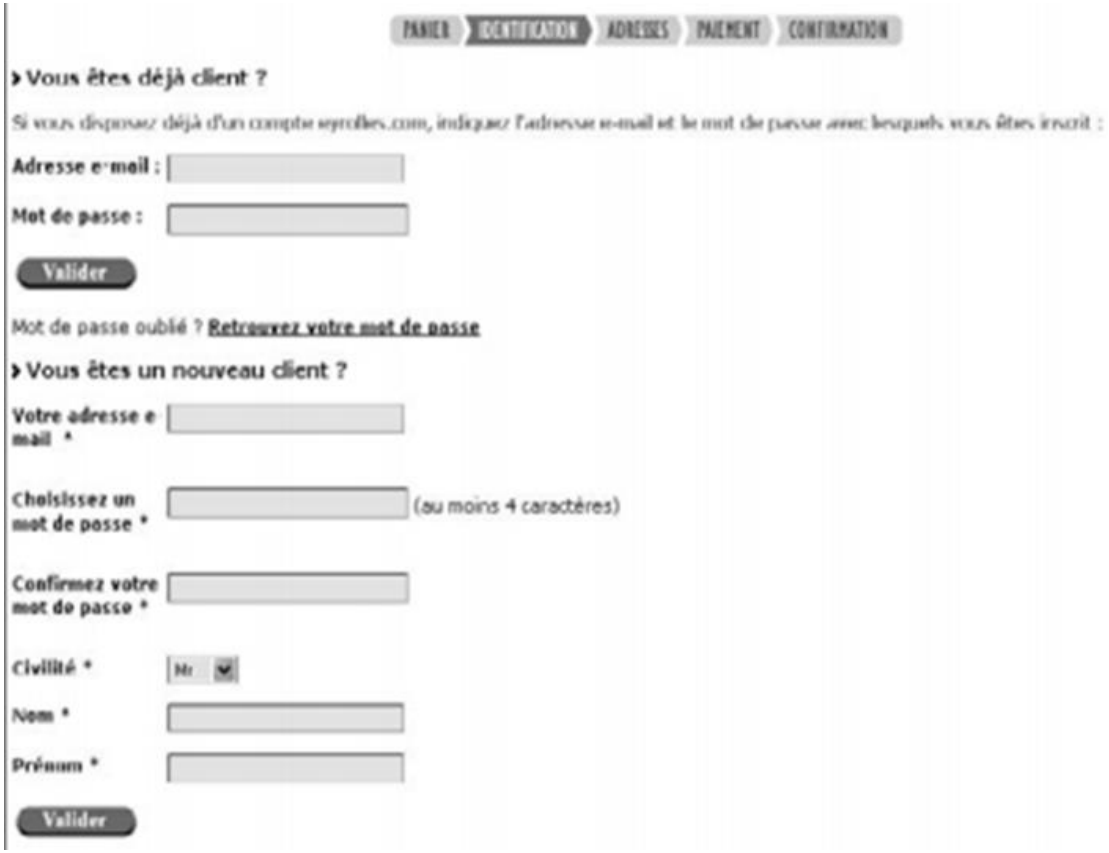
Acteur	Client.
Objectif	Ajouter une commande.
Pré condition	le cas d'utilisation commence lorsque l'utilisateur est s'authentifié.
Post condition	Le client passe une commande
Scénario nominal	1. Le client demander de remplir le formulaire de commande 2. Le système affiche le formulaire. 3. le client saisie ces informations nécessaire (nom, prénom, adresse, …). 4. le système enregistre les informations , il renvoie un message de réussite.
Scénario d'erreur (alternative)	4-a- Le client n'a pas remplit un champ obligatoire ou a saisie des informations non valides. Le système indique au client « le champ non accepté avec une couleur rouge et le scénario reprend à partir de 2 ».
Exigences supplémentaires	/

Processus simplifié

5. 2. Spécification détaillée des besoins:

5.2.2. Fiche de description textuelle:

Ecran IHM:



Processus simplifié

5. 2. Spécification détaillée des besoins:

5.2.3 Description graphique des cas d'utilisation:

Les cas d'utilisation décrivent **les interactions des acteurs** avec le **système** que nous voulons spécifier et concevoir. Lors de ces interactions, **les acteurs produisent des messages** qui **affectent le système informatique** et appellent généralement une réponse de celui-ci.

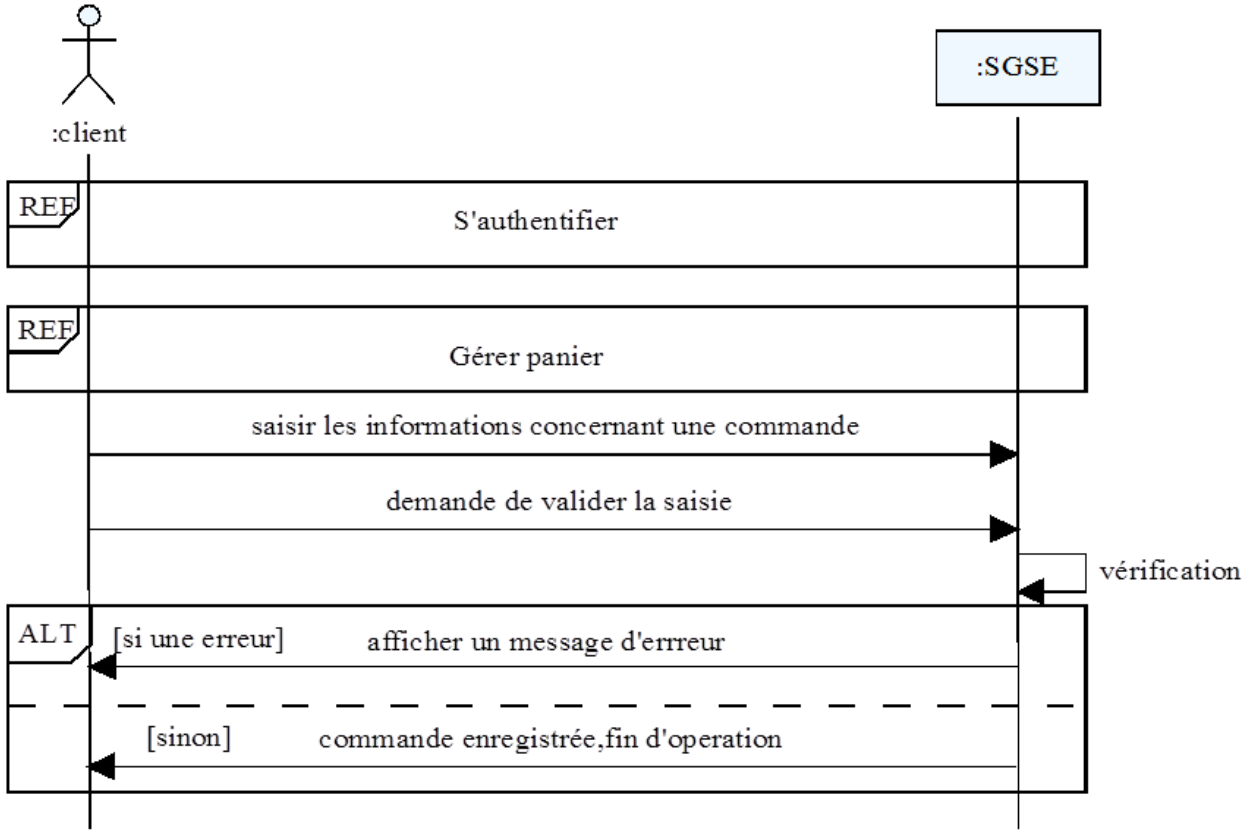
Nous utilisons le terme de diagramme **de séquence « système »** pour souligner le fait que nous considérons le système informatique comme une **boîte noire**. Le comportement du système est décrit vu **de l'extérieur**, sans préjuger de comment il le réalisera.

Processus simplifié

5. 2. Spécification détaillée des besoins:

5.2.3 Description graphique des cas d'utilisation:

Exemple: passer une commande



Processus simplifié

5. 3. Analyse (Réalisation des cas d'utilisation):

Nous apprendrons à identifier les concepts du domaine à partir de l'expression initiale des besoins de notre étude de cas. la conception objet demande principalement une description **structurelle**, **statique**, du système à réaliser sous forme d'un ensemble de **classes logicielles**

5.3.1. Démarche:

- **Identification des concepts du domaine**: Classes, objets, …
- **Ajouter les attributs et les associations**: Elaborer le diagramme de classe métier,
- **Typologie des classes d'analyse**: Classes métiers, classes dialogues, et classes contrôles.

Processus simplifié

5. 3. Analyse (Réalisation des cas d'utilisation):

Les classes d'analyses:

1- Classe dialogue:

Les dialogues vont posséder **des attributs** et **des opérations**. Les **attributs représenteront des champs de saisie** ou **des résultats**. Les résultats seront distingués en utilisant la notation de. **l'attribut dérivé(/)**. **Les opérations représenteront des actions de l'utilisateur sur l'IHM**.

Les dialogues ne peuvent être reliés **qu'aux contrôles** ou à **d'autres dialogues**, mais pas directement aux entités.

Processus simplifié

5. 3. Analyse (Réalisation des cas d'utilisation):

Les classes d'analyses:

1- Classe contrôle:

Les contrôles vont seulement posséder **des opérations**. Ces opérations montrent la logique de l'application, les règles transverses à plusieurs entités, bref les **comportements du système informatique**.

Les contrôles ont **accès à tous les types de classes**, y compris d'autres contrôles

Processus simplifié

5. 3. Analyse (Réalisation des cas d'utilisation):

Les classes d'analyses:

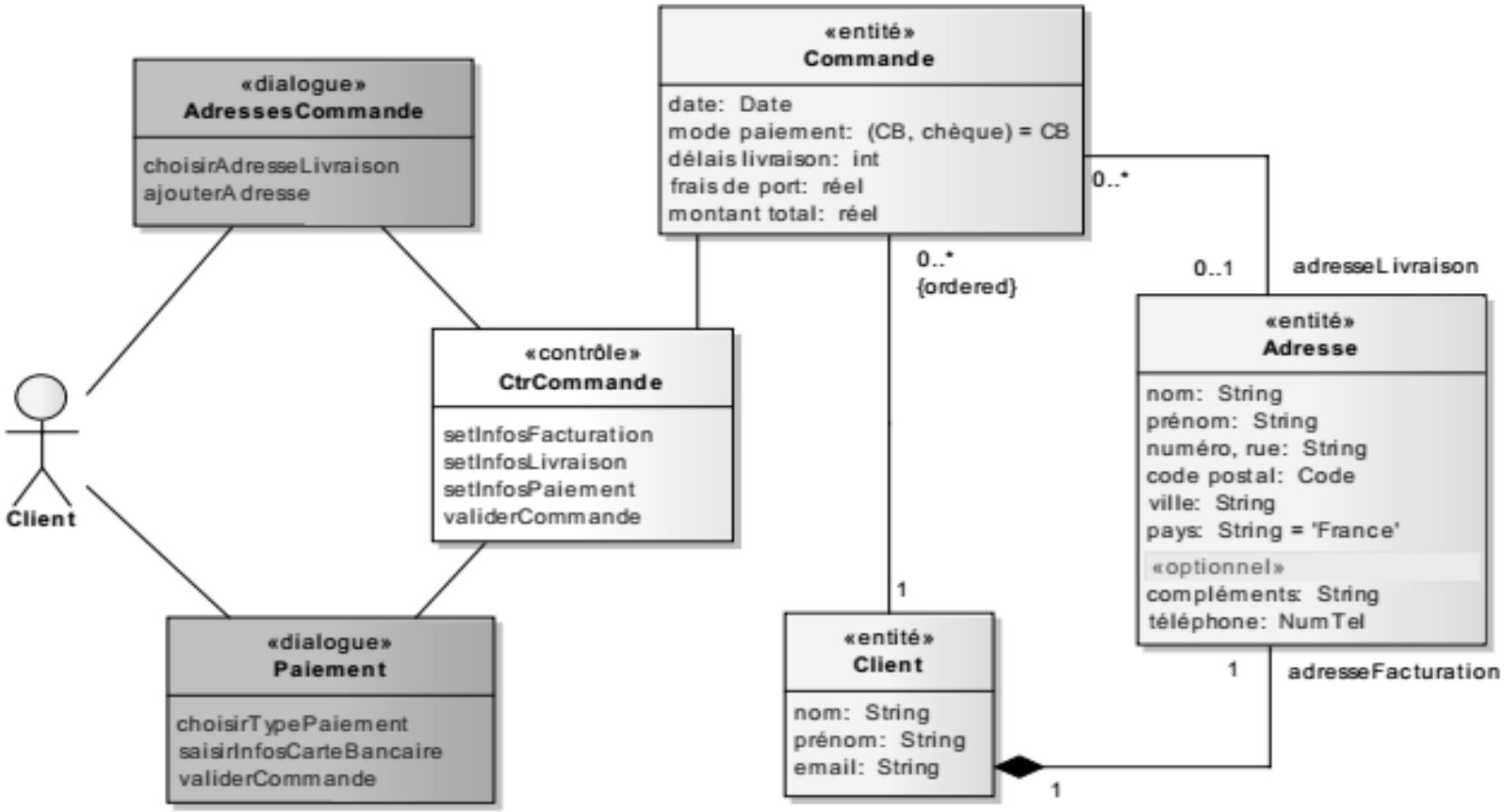
1- Classe métier (Entité):

Les entités vont seulement posséder **des attributs.** Ces attributs représentent en général des informations **persistantes de l'application.** **Les entités ne peuvent être reliées** qu'aux **contrôles** ou **à d'autres entités.**

Processus simplifié

5. 3. Analyse (Réalisation des cas d'utilisation):

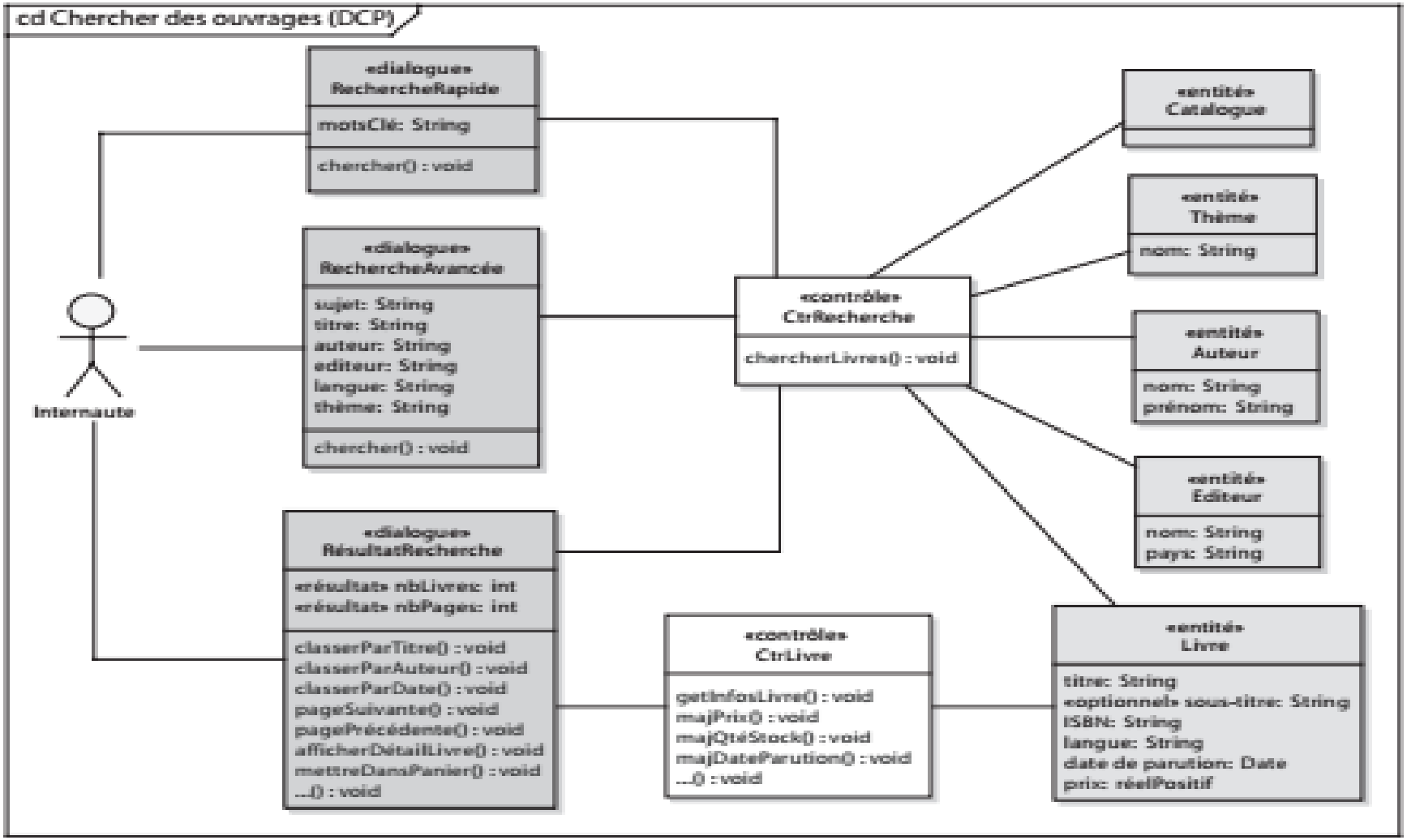
Exemple 1: Effectuer commande



Processus simplifié

5. 3. Analyse (Réalisation des cas d'utilisation):

Exemple 2 : Rechercher ouvrage



Processus simplifié

5. 4. : Conception:

Pour passer maintenant vraiment à la conception, nous allons **répartir tout le comportement** du système entre **ces classes d'analyse** et nous **décrirons les interactions correspondantes.**

L'attribution des **bonnes responsabilités** aux **bonnes classes** est l'un des problèmes les plus délicats de la **conception orientée objet**. Pour chaque service ou fonction, il faut **décider** quelle est la **classe qui va la contenir**. Les diagrammes d'interactions (**séquence ou communication**) sont particulièrement utiles au concepteur pour **représenter graphiquement ses décisions d'allocation de responsabilités.**

Processus simplifié

5. 4. 1.: Diagramme de séquence:

Dans ce genre de diagramme, **les objets** communiquent **en s'envoyant des messages** qui invoquent **des opérations (ou méthodes)** sur les objets récepteurs **(Figure1).** Il est ainsi possible de suivre visuellement les interactions dynamiques entre objets, et les traitements réalisés par chacun.

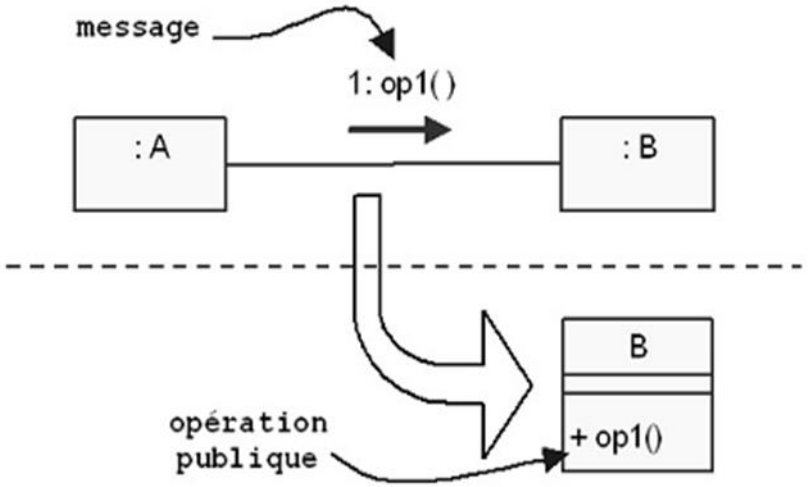


Figure1

Processus simplifié

5. 4. 1.: Diagramme de séquence:

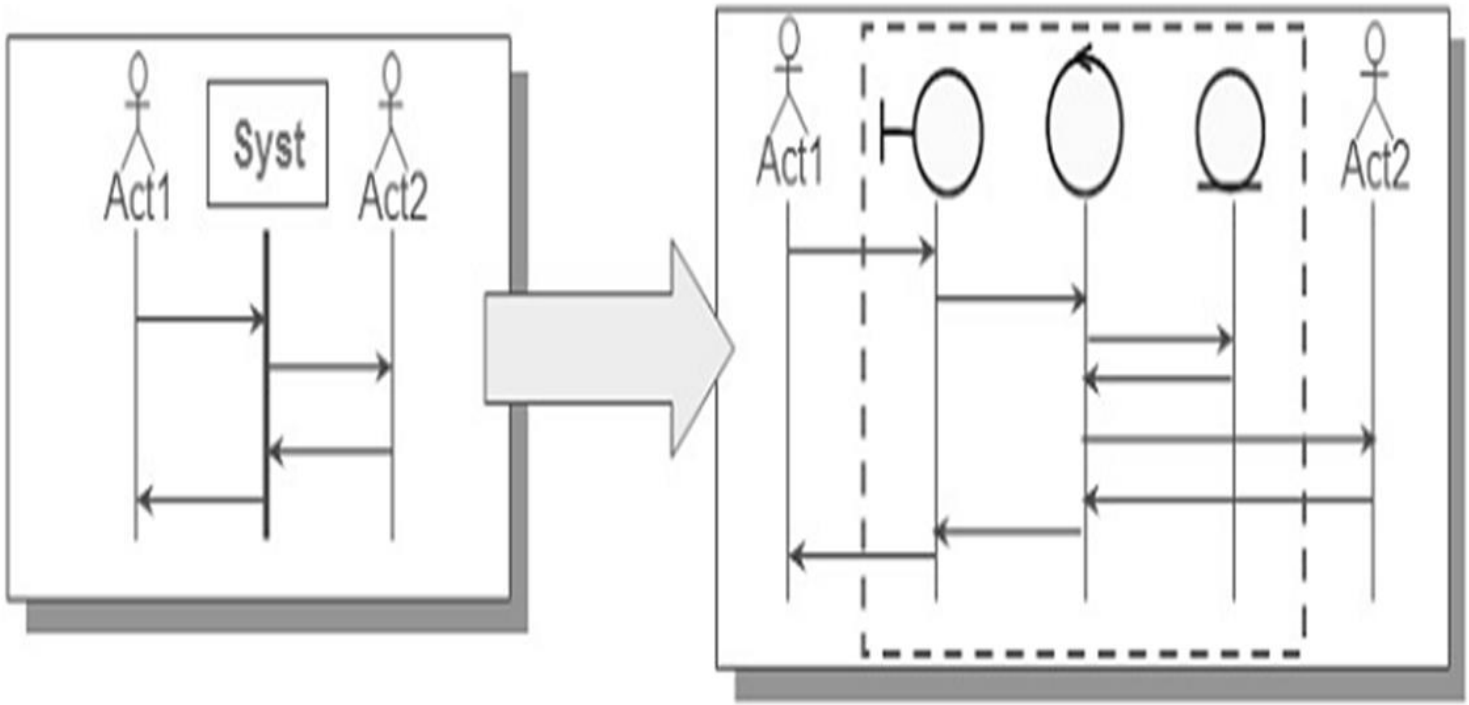
Par rapport aux diagrammes de **séquence système**, nous allons **remplacer** le système vu comme une **boîte noire** par un ensemble **d'objets en interaction**. Pour cela, nous utiliserons encore dans ce chapitre les trois types de classes d'analyse.

- **Les acteurs ne** peuvent interagir (envoyer des messages) **qu'avec les dialogues.**
- **Les dialogues** peuvent **interagir** avec **les contrôles.**
- **Les contrôles** peuvent **interagir** avec **les dialogues**, **les entités**, ou d'autres **contrôles**.
- **Les entités ne** peuvent interagir **qu'entre elles .**

Processus simplifié

5. 4. 1.: Diagramme de séquence:

Exemple 1:



Passage de l'analyse à la conception préliminaire

Processus simplifié

5. 4. 1.: Diagramme de séquence:

Notation détaillée des diagrammes de séquence:

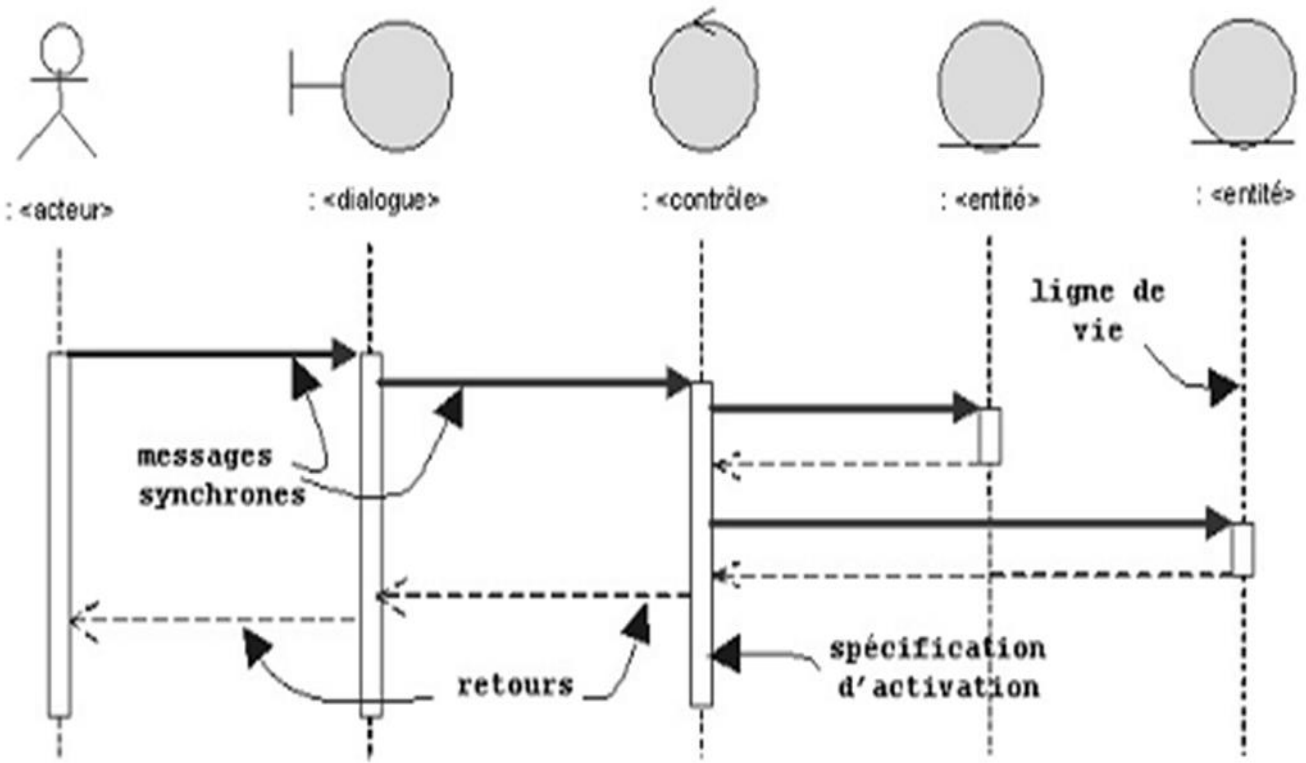
Dans le diagramme de séquence **on représente la ligne de vie** de chaque objet par un trait pointillé vertical. Cette ligne de vie sert de **point de départ** ou **d'arrivée** à **des messages** représentés eux-mêmes par **des flèches horizontales**.

Par convention, **le temps** coule **de haut en bas**. Il indique ainsi visuellement la séquence relative des envois et réceptions de messages.

Processus simplifié

5. 4. 1.: Diagramme de séquence:

Exemple 2:



Exemple: notation diagramme de séquence

Processus simplifié

5. 4. 1.: Diagramme de séquence:

Notation détaillée des diagrammes de séquence:

Le message de création a une représentation particulière **(flèche pointillée ouverte)** sur le diagramme de séquence.

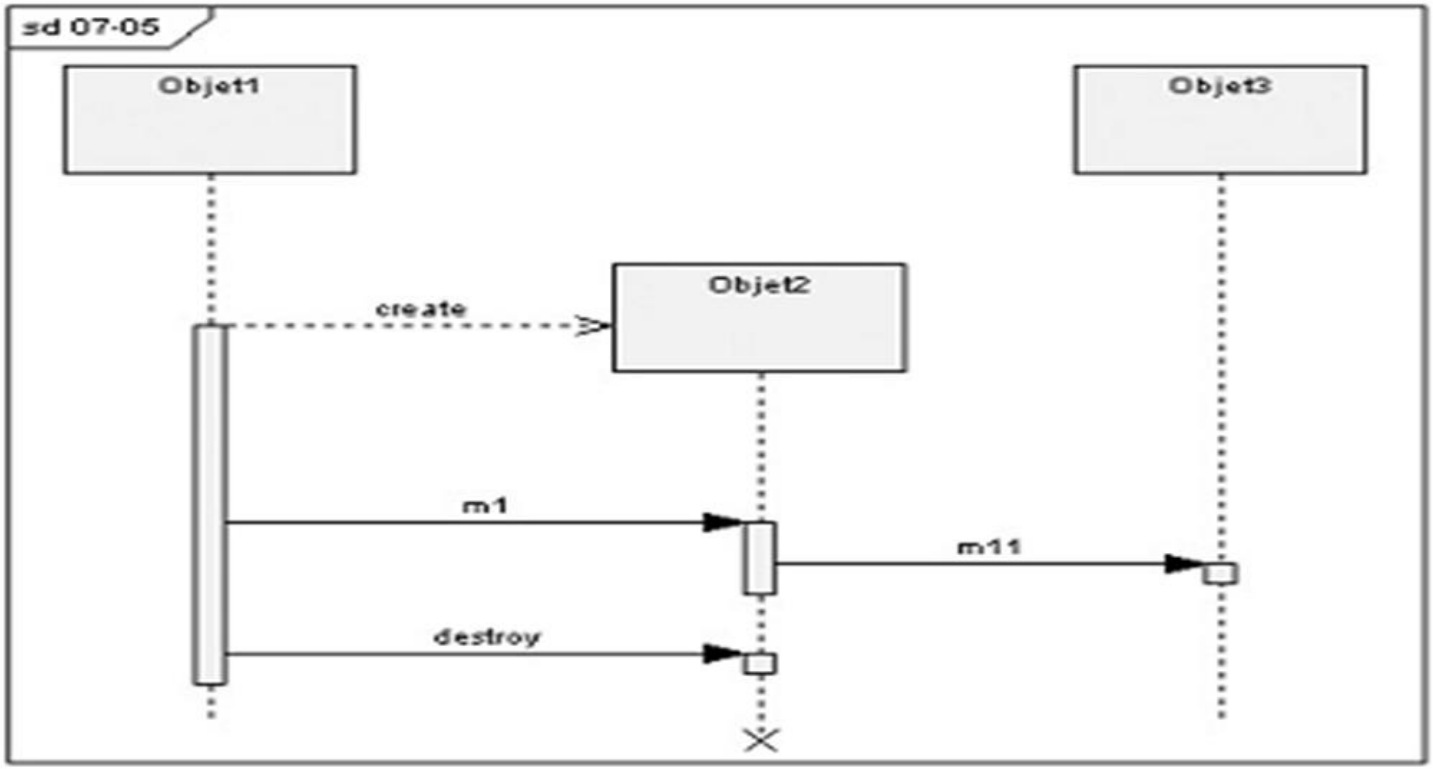
L'événement de **destruction** est pour sa part représenté par **une croix noire** qui termine la ligne de vie détruite.

Dans l'exemple ci-dessous, **l'objet 2** est créé et détruit durant le scénario, contrairement aux **objets 1 et 3** qui préexistent et survivent au scénario concerné.

Processus simplifié

5. 4. 1.: Diagramme de séquence:

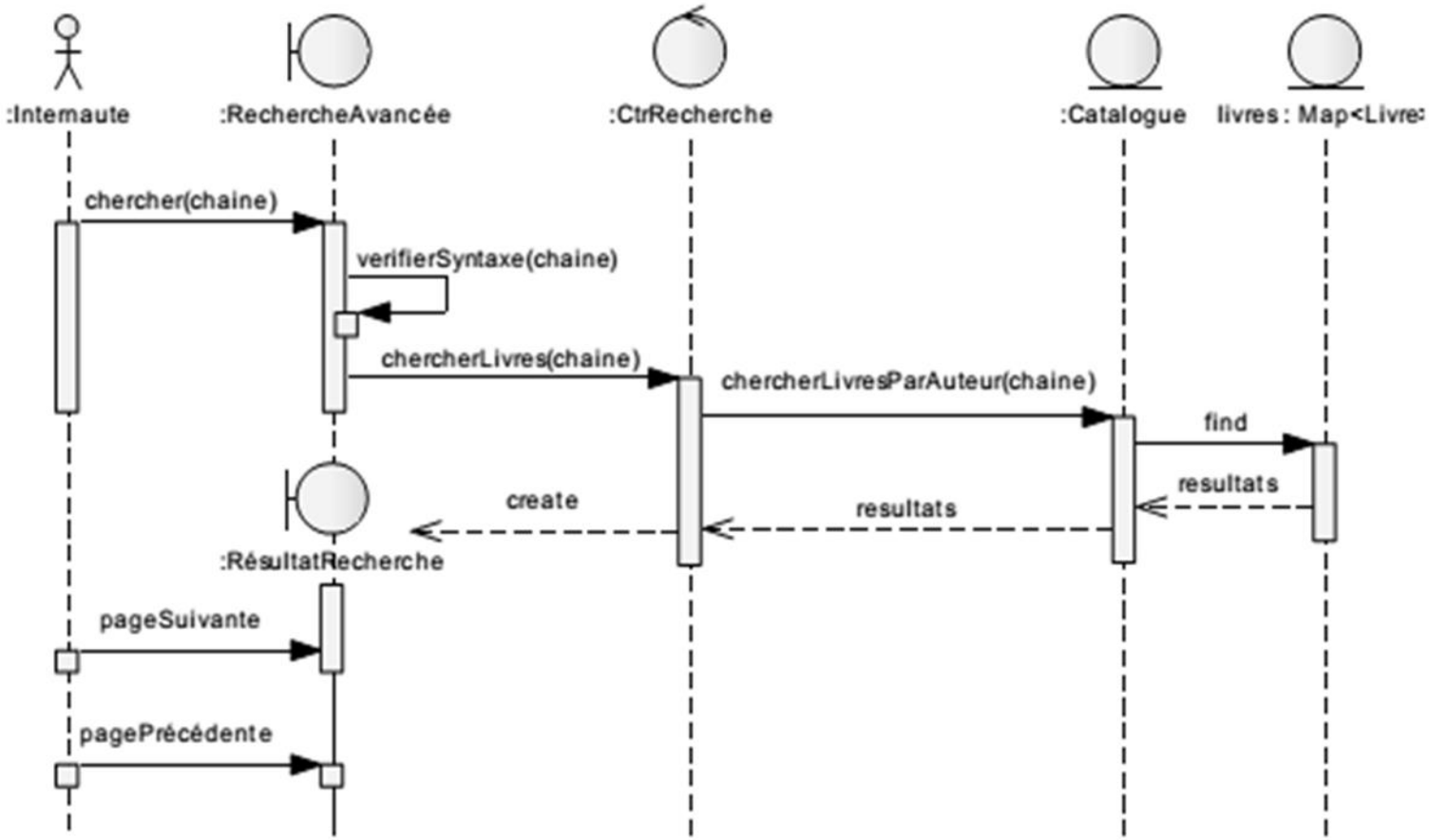
Exemple 3:



Processus simplifié

5. 4. 1.: Diagramme de séquence:

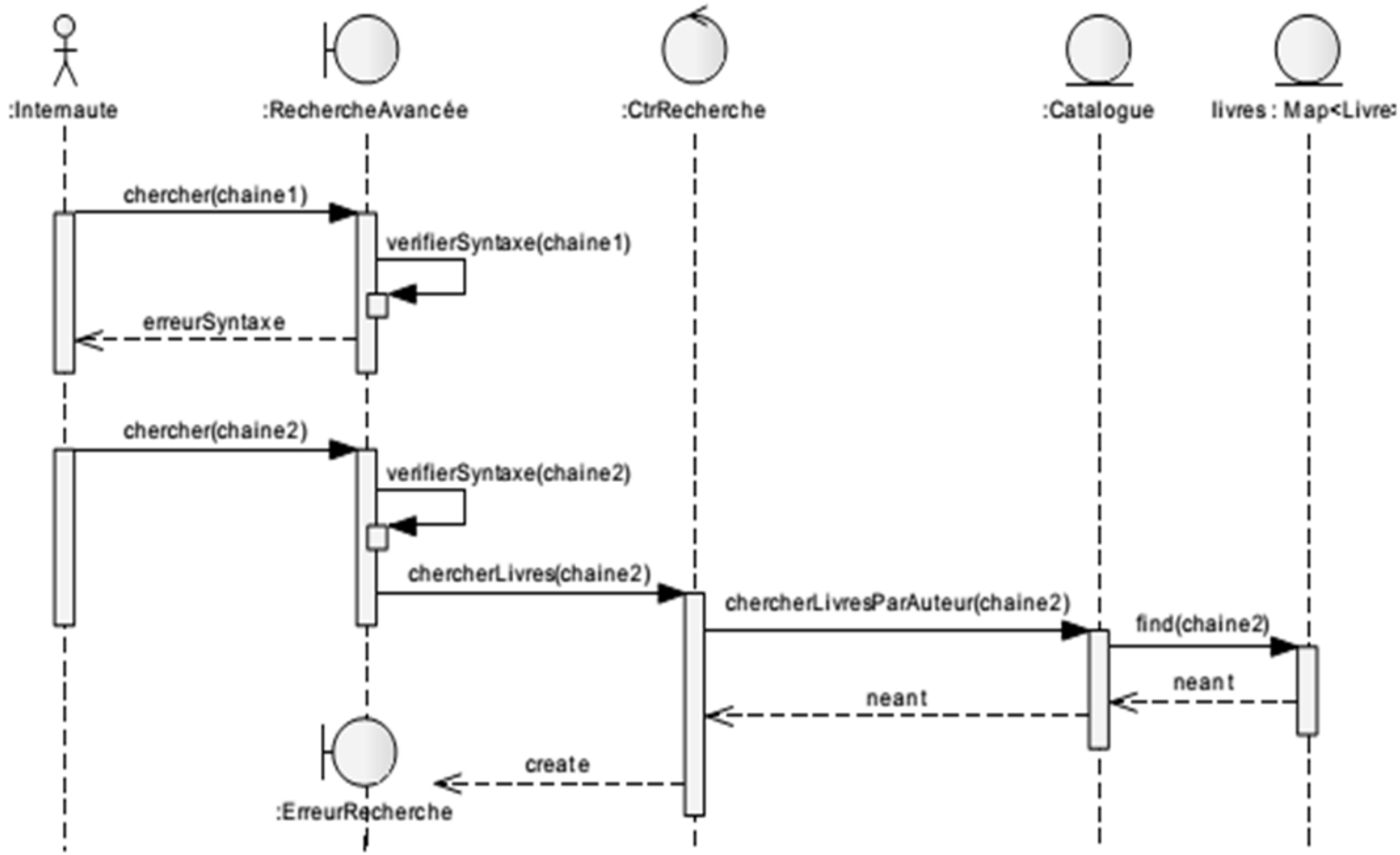
Exemple 4: (Cas d'utilisation « recherche avancé scénario nominal »):



Processus simplifié

5. 4. 1.: Diagramme de séquence:

Exemple 4: (Cas d'utilisation « recherche avancé scénario d'erreur»):



Processus simplifié

5. 4. 2.: Diagramme de classe de conception:

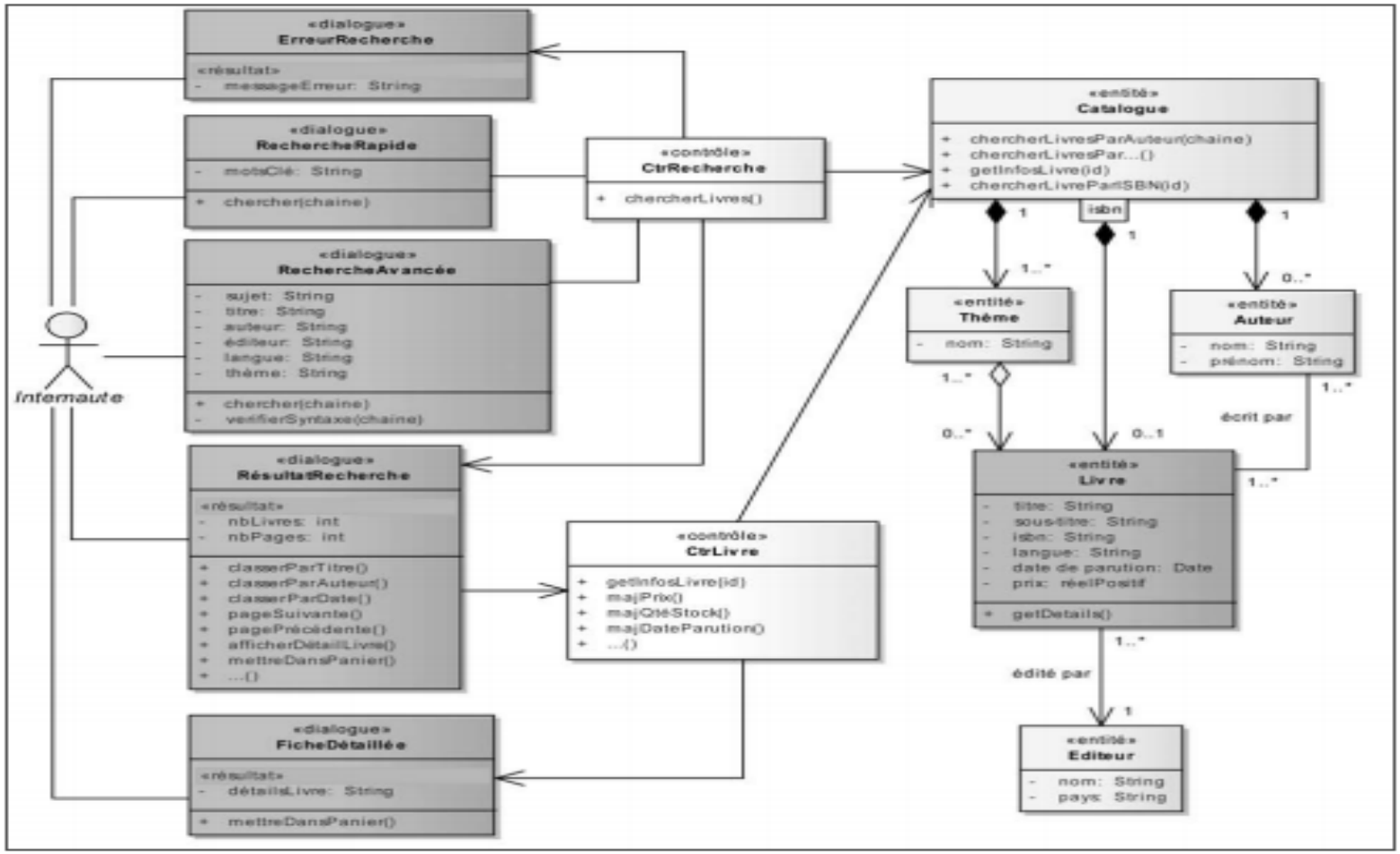
En partant **du modèle d'analyse**, nous allons **affiner et compléter** les **diagrammes de classes participantes** obtenus précédemment. Pour cela nous utiliserons **les diagrammes de séquence** que nous venons de réaliser pour:

- **Ajouter ou préciser** les opérations **dans les classes** (un message ne peut être reçu par un objet que si sa classe a déclaré l'opération publique correspondante).
- **Ajouter des types aux attributs** et aux **paramètres** et **retours des opérations.**
- **Affiner** les **relations entre classes : associations (avec indication de navigabilité), généralisations ou dépendances**

Processus simplifié

5. 4. 2.: Diagramme de classe de conception:

Exemple 1:



Processus simplifié

Les métriques et les testes logiciels

Chapitre 4