Analyse : de l'exigence à la spécification

CM4 — Modéliser le métier : structure et comportement

Ressource R3.03

BUT2 Informatique

September 23, 2026

Où en sommes-nous

CM1–CM3	le besoin, les exigences, les stories, les cas d'utilisation avec leurs exceptions
TP1–TP3	un outil qui sauvegarde, qui n'écrit que ce qui a changé, et qui est testé
Aujourd'hui	**passer du monde de l'utilisateur au monde de la solution — sans y perdre le langage du client**

La méthode est déroulée aujourd'hui sur un autre domaine, une bibliothèque. Vous l'appliquerez à BioSeq au TD : les réponses de votre dossier ne sont pas dans ces diapositives.

Déroulé de la séance

#	Partie	Durée
1	Le modèle du domaine : un dictionnaire visuel, pas un plan	18 min
2	Le construire : méthode, notation, et les deux constructions qui comptent	30 min
3	Les processus métier enrichissent le modèle	12 min
4	L'automate à états, et ce qu'il interdit	15 min

Le piège de cette étape

Penser immédiatement en termes de code,
de bases de données et d'interfaces techniques.

Avant de décider *comment* coder une `Facture`...

...il faut s'accorder sur **ce qu'est une** `Facture` dans le contexte métier du client.

Le diagramme de classes du domaine n'est pas le plan d'une maison.
C'est un **dictionnaire visuel validé par le client**.

Le piège de cette étape

Penser immédiatement en termes de code,
de bases de données et d'interfaces techniques.

Avant de décider *comment* coder une `Facture`...

...il faut s'accorder sur **ce qu'est une** `Facture` dans le contexte métier du client.

Le diagramme de classes du domaine n'est pas le plan d'une maison.
C'est un **dictionnaire visuel validé par le client**.

Le piège de cette étape

Penser immédiatement en termes de code,
de bases de données et d'interfaces techniques.

Avant de décider *comment* coder une `Facture`...

...il faut s'accorder sur **ce qu'est une** `Facture` dans le contexte métier du client.

Le diagramme de classes du domaine n'est pas le plan d'une maison.
C'est un **dictionnaire visuel validé par le client**.

Ce qu'il contient, ce qu'il exclut

Il contient

Classes	les « choses » importantes du métier : `Client`, `Commande`, `Facture`
Attributs	leurs propriétés *pertinentes* : un `Produit` a un `nom`, un `prix`
Associations	les relations, avec leurs **multiplicités**

Il exclut volontairement

- les **méthodes** : `calculerTotal()`, `sauvegarder()`
- les **types techniques** : on écrit « prix », pas `BigDecimal`
- les **classes de solution** : `DatabaseManager`, `PDFController`, `FileWalker`

À ce stade, on répond au **quoi**. Le **comment** viendra après.

Ce qu'il contient, ce qu'il exclut

Il contient

Classes	les « choses » importantes du métier : `Client`, `Commande`, `Facture`
Attributs	leurs propriétés *pertinentes* : un `Produit` a un `nom`, un `prix`
Associations	les relations, avec leurs **multiplicités**

Il exclut volontairement

- les **méthodes** : `calculerTotal()`, `sauvegarder()`
- les **types techniques** : on écrit « prix », pas `BigDecimal`
- les **classes de solution** : `DatabaseManager`, `PDFController`, `FileWalker`

À ce stade, on répond au **quoi**. Le **comment** viendra après.

Analyse ou conception ? Le tableau à garder en tête

Classe d'analyse (du domaine)	Classe de conception
répond au « quoi » (le métier)	répond au « comment » (la solution)
`Facture`, `Client`, `Produit`	`FactureController`, `FactureDAO`
attributs simples : « prix »	types techniques : `BigDecimal prix`
aucune méthode	`calculerTVA()`, `save()`
validée avec le client	validée par l'équipe technique

Le piège numéro un des débutants

Faire figurer une `FactureDAO` dans un diagramme du domaine, c'est **polluer l'analyse avec des choix techniques prématurés**. À ce stade, on parle le langage du client, pas celui de la machine.

Analyse ou conception ? Le tableau à garder en tête

Classe d'analyse (du domaine)	Classe de conception
répond au « quoi » (le métier)	répond au « comment » (la solution)
`Facture`, `Client`, `Produit`	`FactureController`, `FactureDAO`
attributs simples : « prix »	types techniques : `BigDecimal prix`
aucune méthode	`calculerTVA()`, `save()`
validée avec le client	validée par l'équipe technique

Le piège numéro un des débutants

Faire figurer une `FactureDAO` dans un diagramme du domaine, c'est **polluer l'analyse avec des choix techniques prématurés**. À ce stade, on parle le langage du client, pas celui de la machine.

Le langage ubiquitaire (Eric Evans, *Domain-Driven Design*, 2003)

Un vocabulaire unique, partagé

par les experts métier, les analystes et les développeurs.

Si le client parle d'« **Adhérent** » et que le code manipule un « **User** », les malentendus sont inévitables — et ils se paieront à chaque réunion, pendant des années.

➪ Fil rouge — `snapguard` : le glossaire de BioSeq

Relisez la transcription de l'entretien : Karim y emploie « *backup* » pour désigner tantôt le support, tantôt une sauvegarde particulière. **Ce mot-là devra disparaître de votre glossaire**, remplacé par les termes précis que vous aurez arrêtés avec lui — c'est le premier travail du TD.

Le langage ubiquitaire (Eric Evans, *Domain-Driven Design*, 2003)

Un vocabulaire unique, partagé

par les experts métier, les analystes et les développeurs.

Si le client parle d'« **Adhérent** » et que le code manipule un « **User** », les malentendus sont inévitables — et ils se paieront à chaque réunion, pendant des années.

➪ Fil rouge — `snapguard` : le glossaire de BioSeq

Relisez la transcription de l'entretien : Karim y emploie « *backup* » pour désigner tantôt le support, tantôt une sauvegarde particulière. **Ce mot-là devra disparaître de votre glossaire**, remplacé par les termes précis que vous aurez arrêtés avec lui — c'est le premier travail du TD.

Le langage ubiquitaire (Eric Evans, *Domain-Driven Design*, 2003)

Un vocabulaire unique, partagé

par les experts métier, les analystes et les développeurs.

Si le client parle d'« **Adhérent** » et que le code manipule un « **User** », les malentendus sont inévitables — et ils se paieront à chaque réunion, pendant des années.

➪ Fil rouge — `snapguard` : le glossaire de BioSeq

Relisez la transcription de l'entretien : Karim y emploie « *backup* » pour désigner tantôt le support, tantôt une sauvegarde particulière. **Ce mot-là devra disparaître de votre glossaire**, remplacé par les termes précis que vous aurez arrêtés avec lui — c'est le premier travail du TD.

La méthode en trois étapes

1. **Identifier les classes candidates** — cherchez les **noms** et groupes nominaux dans vos exigences et vos cas d'utilisation.
2. **Identifier les attributs** — les propriétés qui décrivent une classe : le « titre » d'une « Idée », le « prix » d'un « Produit ».
3. **Identifier les associations** — cherchez les **verbes** : un « Employé » *soumet* une « Idée ».

Puis le raffinement, qui est le vrai travail

Éliminer les doublons, fusionner, rejeter, et surtout **déterminer les multiplicités** en posant des questions simples au client.

La méthode en trois étapes

1. **Identifier les classes candidates** — cherchez les **noms** et groupes nominaux dans vos exigences et vos cas d'utilisation.
2. **Identifier les attributs** — les propriétés qui décrivent une classe : le « titre » d'une « Idée », le « prix » d'un « Produit ».
3. **Identifier les associations** — cherchez les **verbes** : un « Employé » *soumet* une « Idée ».

Puis le raffinement, qui est le vrai travail

Éliminer les doublons, fusionner, rejeter, et surtout **déterminer les multiplicités** en posant des questions simples au client.

Savoir lire un diagramme avant d'en écrire un

```
+------------------+       Une CLASSE : nom au singulier, majuscule.
|    Exemplaire    |       Les attributs en dessous, sans type technique.
+------------------+
| codeBarres       |
| etat             |
+------------------+

Ouvrage "1" ------- "1..*" Exemplaire       ASSOCIATION + MULTIPLICITES
   ^                  ^
   |                  +-- un Ouvrage a 1..* Exemplaires
   +-- un Exemplaire appartient a 1 Ouvrage (la multiplicite se lit
                                              EN FACE, jamais a cote)
      Adherent                              HERITAGE
          ^                                 Le triangle vide pointe
          |                                 vers le PARENT.
      Enseignant                            « Enseignant EST UN
                                              Adherent. »
Adherent "0..5" ---------- "0..*" Exemplaire  CLASSE-ASSOCIATION
                  :                         Trait POINTILLE vers une
               Emprunt                      classe qui porte les
            { dateEmprunt }                 attributs de la relation.
```

Savoir lire un diagramme avant d'en écrire un

```
+------------------+       Une CLASSE : nom au singulier, majuscule.
|    Exemplaire    |       Les attributs en dessous, sans type technique.
+------------------+
| codeBarres       |
| etat             |
+------------------+

Ouvrage "1" ------- "1..*" Exemplaire       ASSOCIATION + MULTIPLICITES
   ^                  ^
   |                  +-- un Ouvrage a 1..* Exemplaires
   +-- un Exemplaire appartient a 1 Ouvrage (la multiplicite se lit
                                              EN FACE, jamais a cote)
      Adherent                              HERITAGE
          ^                                 Le triangle vide pointe
          |                                 vers le PARENT.
      Enseignant                            « Enseignant EST UN
                                              Adherent. »
Adherent "0..5" ---------- "0..*" Exemplaire  CLASSE-ASSOCIATION
                  :                         Trait POINTILLE vers une
               Emprunt                      classe qui porte les
            { dateEmprunt }                 attributs de la relation.
```

À vous — le cas qui servira de fil à ce cours

Le besoin, tel que la responsable de la bibliothèque l'a formulé

« Nos adhérents empruntent des ouvrages, au plus cinq à la fois, pour trois semaines. Certains ouvrages existent en plusieurs exemplaires. Je veux savoir qui a quoi, depuis quand, et pouvoir prolonger un emprunt une fois. »

Parmi ces candidats, lesquels sont des **classes**, des **attributs**, ou à **rejeter** ?

`Adhérent` `Ouvrage` `Exemplaire` `Emprunt` `DateRetour` `CodeBarres` `RequêteSQL` `Prolongation`

Et surtout : **où mettez-vous la date d'emprunt** — sur l'`Adhérent`, sur l'`Exemplaire`, ou ailleurs ?

À vous — le cas qui servira de fil à ce cours

Le besoin, tel que la responsable de la bibliothèque l'a formulé

« Nos adhérents empruntent des ouvrages, au plus cinq à la fois, pour trois semaines. Certains ouvrages existent en plusieurs exemplaires. Je veux savoir qui a quoi, depuis quand, et pouvoir prolonger un emprunt une fois. »

Parmi ces candidats, lesquels sont des **classes**, des **attributs**, ou à **rejeter** ?

`Adhérent` `Ouvrage` `Exemplaire` `Emprunt` `DateRetour` `CodeBarres` `RequêteSQL` `Prolongation`

Et surtout : **où mettez-vous la date d'emprunt** — sur l'`Adhérent`, sur l'`Exemplaire`, ou ailleurs ?

Le tri : quatre décisions possibles

Candidat	Décision	Justification
`Adhérent`, `Exemplaire`	**classe**	concepts autonomes, dont la responsable parle spontanément
`Ouvrage`	**classe**	un titre, distinct de ses exemplaires physiques
`DateRetour`, `CodeBarres`	**attribut**	ne vivent pas indépendamment de leur porteur
`Prolongation`	**fusion**	« une fois » : un compteur sur l'emprunt suffit
`Emprunt`	**à suivre**	ni tout à fait classe, ni attribut — on y revient dans trois diapositives
`RequêteSQL`	**rejet**	solution technique, pas concept métier

Le rejet est la décision la plus difficile

Un candidat technique se glisse toujours dans la liste, et c'est en général celui dont l'équipe est la plus fière. **Le test :** la responsable de la bibliothèque l'emploierait-elle spontanément ? `Exemplaire`, oui. `RequêteSQL`, jamais.

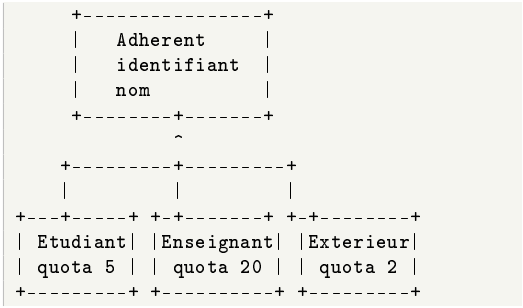
Le tri : quatre décisions possibles

Candidat	Décision	Justification
`Adhérent`, `Exemplaire`	**classe**	concepts autonomes, dont la responsable parle spontanément
`Ouvrage`	**classe**	un titre, distinct de ses exemplaires physiques
`DateRetour`, `CodeBarres`	**attribut**	ne vivent pas indépendamment de leur porteur
`Prolongation`	**fusion**	« une fois » : un compteur sur l'emprunt suffit
`Emprunt`	**à suivre**	ni tout à fait classe, ni attribut — on y revient dans trois diapositives
`RequêteSQL`	**rejet**	solution technique, pas concept métier

Le rejet est la décision la plus difficile

Un candidat technique se glisse toujours dans la liste, et c'est en général celui dont l'équipe est la plus fière. **Le test :** la responsable de la bibliothèque l'emploierait-elle spontanément ? `Exemplaire`, oui. `RequêteSQL`, jamais.

L'héritage : une seule question à se poser



Le test : « EST-UN »

« Un enseignant **est un** adhérent » se dit sans effort.

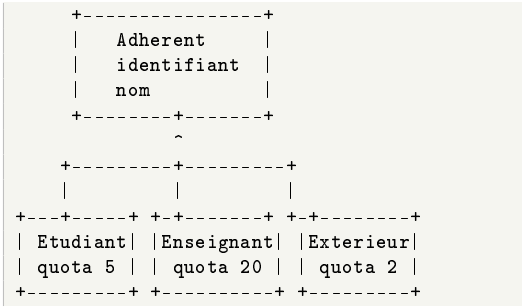
« Un exemplaire *est un* ouvrage » ne se dit pas : c'est une composition, pas une spécialisation.

On spécialise quand les sous-classes **partagent des attributs** et **diffèrent par leur comportement ou leurs droits**. Ici : même identité, trois quotas et trois durées de prêt.

Le piège : l'héritage de commodité

« Ces deux classes ont trois attributs en commun, je fais hériter. » Non : la ressemblance n'est pas la spécialisation. Si vous ne pouvez pas dire « *est un* » à voix haute sans grimacer, c'est une association.

L'héritage : une seule question à se poser



Le test : « EST-UN »

« Un enseignant **est un** adhérent » se dit sans effort.

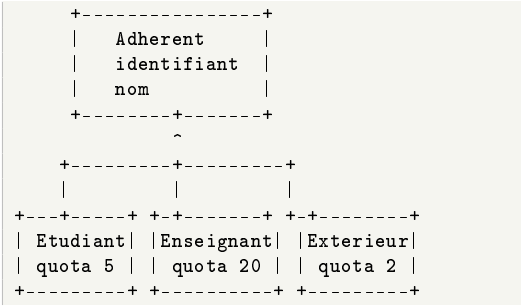
« Un exemplaire *est un* ouvrage » ne se dit pas : c'est une composition, pas une spécialisation.

On spécialise quand les sous-classes **partagent des attributs** et **diffèrent par leur comportement ou leurs droits**. Ici : même identité, trois quotas et trois durées de prêt.

Le piège : l'héritage de commodité

« Ces deux classes ont trois attributs en commun, je fais hériter. » Non : la ressemblance n'est pas la spécialisation. Si vous ne pouvez pas dire « *est un* » à voix haute sans grimacer, c'est une association.

L'héritage : une seule question à se poser



Le test : « EST-UN »

« Un enseignant **est un** adhérent » se dit sans effort.

« Un exemplaire *est un* ouvrage » ne se dit pas : c'est une composition, pas une spécialisation.

On spécialise quand les sous-classes **partagent des attributs** et **diffèrent par leur comportement ou leurs droits**. Ici : même identité, trois quotas et trois durées de prêt.

Le piège : l'héritage de commodité

« Ces deux classes ont trois attributs en commun, je fais hériter. » Non : la ressemblance n'est pas la spécialisation. Si vous ne pouvez pas dire « *est un* » à voix haute sans grimacer, c'est une association.

Les multiplicités se déterminent en interrogeant le client

- *Un ouvrage peut-il avoir plusieurs exemplaires ?*
  « Oui, on achète trois copies des manuels. » → `Ouvrage 1 - 1..* Exemplaire`
- *Un exemplaire peut-il appartenir à deux ouvrages ?*
  « Évidemment non. » → la multiplicité 1 du côté ouvrage
- *Un adhérent peut-il emprunter plusieurs exemplaires ?*
  « Cinq au plus, en même temps. » → `0..5` — et la contrainte est notée

La question qui fait tout basculer

Un même exemplaire peut-il être emprunté plusieurs fois ?
« Bien sûr, c'est même le but — et je veux garder l'historique. »

Nous avons donc une relation $n \dots m$ **et** une date d'emprunt à ranger quelque part.

Les multiplicités se déterminent en interrogeant le client

- *Un ouvrage peut-il avoir plusieurs exemplaires ?*
  « Oui, on achète trois copies des manuels. » → `Ouvrage 1 - 1..* Exemplaire`
- *Un exemplaire peut-il appartenir à deux ouvrages ?*
  « Évidemment non. » → la multiplicité 1 du côté ouvrage
- *Un adhérent peut-il emprunter plusieurs exemplaires ?*
  « Cinq au plus, en même temps. » → `0..5` — et la contrainte est notée

La question qui fait tout basculer

Un même exemplaire peut-il être emprunté plusieurs fois ?
« Bien sûr, c'est même le but — et je veux garder l'historique. »

Nous avons donc une relation $n \dots m$ **et** une date d'emprunt à ranger quelque part.

La classe-association : la date d'emprunt n'appartient à personne

Ni à l'adhérent (il a emprunté plusieurs fois), ni à l'exemplaire (il a été emprunté par plusieurs personnes).

Elle appartient à la relation entre les deux.

```
Adherent  0..5  ------------------  0..*  Exemplaire
                        :
                    Emprunt
          { dateEmprunt, dateRetourPrevue, nbProlongations }
```

La règle, et le pont vers le relationnel

Dès qu'une association *n*…*m* doit porter ses propres attributs, c'est une **classe-association**. Elle deviendra une **table de jonction** à clé composite — celle qui garantira, par exemple, qu'on ne saisit pas deux fois le même emprunt.

Un choix d'analyse prépare directement une décision technique.

La classe-association : la date d'emprunt n'appartient à personne

Ni à l'adhérent (il a emprunté plusieurs fois), ni à l'exemplaire (il a été emprunté par plusieurs personnes).

Elle appartient à la relation entre les deux.

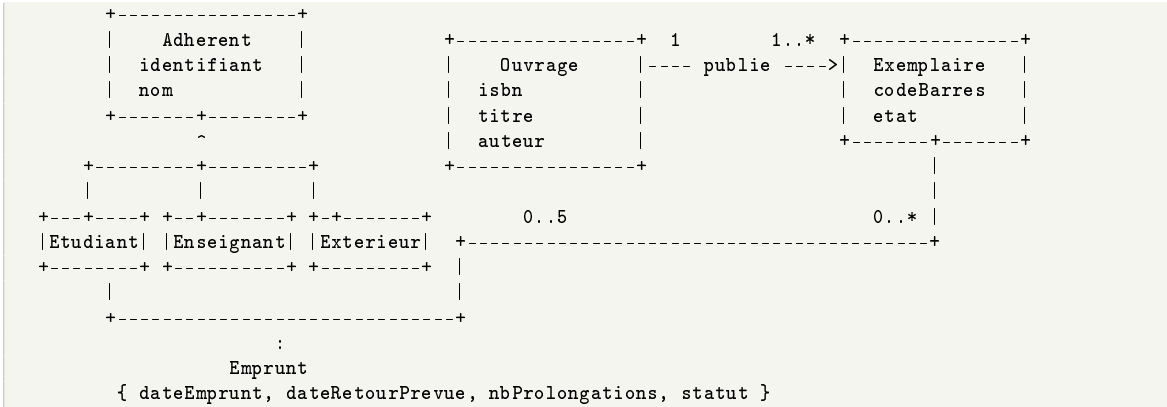
```
Adherent  0..5  ------------------  0..*  Exemplaire
                        :
                    Emprunt
            { dateEmprunt, dateRetourPrevue, nbProlongations }
```

La règle, et le pont vers le relationnel

Dès qu'une association $n \dots m$ doit porter ses propres attributs, c'est une **classe-association**. Elle deviendra une **table de jonction** à clé composite — celle qui garantira, par exemple, qu'on ne saisit pas deux fois le même emprunt.

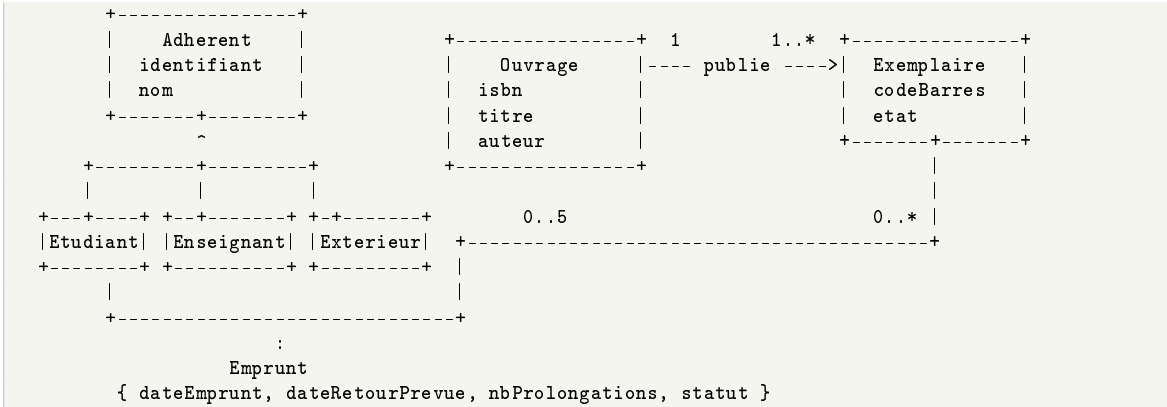
Un choix d'analyse prépare directement une décision technique.

Le résultat : le modèle du domaine de la bibliothèque



Trois constructions à repérer : l'**héritage** des trois catégories d'adhérents, la **classe-association** `Emprunt` (en pointillés), et une association ordinaire `Ouvrage` – `Exemplaire`.

Le résultat : le modèle du domaine de la bibliothèque



Trois constructions à repérer : l'**héritage** des trois catégories d'adhérents, la **classe-association** `Emprunt` (en pointillés), et une association ordinaire `Ouvrage` – `Exemplaire`.

Ce modèle ne contient rien de technique. Vérifions.

Absent	Pourquoi
`RequêteSQL`, `IndexISBN`, `CacheRecherche`	vocabulaire de la machine, pas du client
`EmpruntDAO`, `AdherentController`	artefacts de solution : ils viendront à la conception
`LocalDate`, `varchar(13)`	types techniques : « date d'emprunt » suffit à ce stade
`calculerRetard()`, `save()`	des opérations : on décrit le **quoi**, pas le **comment**

Le test qui tranche, si vous hésitez sur une classe

Le client la comprendrait-il sans explication ?

`Exemplaire`, `Emprunt`, `Prolongation` : oui, il les emploie lui-même. `EmpruntDAO` : non. Ce test départage plus vite que n'importe quelle règle.

Ce modèle ne contient rien de technique. Vérifions.

Absent	Pourquoi
`RequêteSQL`, `IndexISBN`, `CacheRecherche`	vocabulaire de la machine, pas du client
`EmpruntDAO`, `AdherentController`	artefacts de solution : ils viendront à la conception
`LocalDate`, `varchar(13)`	types techniques : « date d'emprunt » suffit à ce stade
`calculerRetard()`, `save()`	des opérations : on décrit le **quoi**, pas le **comment**

Le test qui tranche, si vous hésitez sur une classe

Le client la comprendrait-il sans explication ?

`Exemplaire`, `Emprunt`, `Prolongation` : oui, il les emploie lui-même. `EmpruntDAO` : non. Ce test départage plus vite que n'importe quelle règle.

Le modèle de classes est une vue statique

Il dit ce que *sont* un `Emprunt` et un `Exemplaire`, et qu'ils sont liés.

Il ne dit pas comment un emprunt passe de « en cours » à « rendu », ni qui intervient et dans quel ordre.

Un processus métier

une séquence d'activités qui, exécutées dans un ordre donné, accomplissent un objectif métier. C'est la **vue dynamique** du système.

On les décrit d'abord en **récit textuel structuré** : une liste numérotée, avec l'acteur de chaque étape. Simple, rapide, discutable avec le métier — et suffisant dans 90 % des cas. (Diagramme de flux ou BPMN si le processus est critique ou complexe.)

Le modèle de classes est une vue statique

Il dit ce que *sont* un `Emprunt` et un `Exemplaire`, et qu'ils sont liés.

Il ne dit pas comment un emprunt passe de « en cours » à « rendu », ni qui intervient et dans quel ordre.

Un processus métier

une séquence d'activités qui, exécutées dans un ordre donné, accomplissent un objectif métier. C'est la **vue dynamique** du système.

On les décrit d'abord en **récit textuel structuré** : une liste numérotée, avec l'acteur de chaque étape. Simple, rapide, discutable avec le métier — et suffisant dans 90 % des cas. (Diagramme de flux ou BPMN si le processus est critique ou complexe.)

Le modèle de classes est une vue statique

Il dit ce que *sont* un `Emprunt` et un `Exemplaire`, et qu'ils sont liés.

Il ne dit pas comment un emprunt passe de « en cours » à « rendu », ni qui intervient et dans quel ordre.

Un processus métier

une séquence d'activités qui, exécutées dans un ordre donné, accomplissent un objectif métier. C'est la **vue dynamique** du système.

On les décrit d'abord en **récit textuel structuré** : une liste numérotée, avec l'acteur de chaque étape. Simple, rapide, discutable avec le métier — et suffisant dans 90 % des cas. (Diagramme de flux ou BPMN si le processus est critique ou complexe.)

La synergie : le processus met le modèle à l'épreuve

Processus « Rendre un ouvrage »

1. **[Adhérent]** présente l'exemplaire à la banque de prêt.
2. **[Système]** retrouve l'emprunt en cours pour cet exemplaire.
3. **[Système]** compare la date du jour à la date de retour prévue.
4. **[Système]** enregistre le retour et clôt l'emprunt.
5. **[Système]** si l'exemplaire est attendu par un autre adhérent, le met de côté et prévient celui-ci.

À chaque étape, une seule question :
notre modèle de classes permet-il de faire cela ?

La synergie : le processus met le modèle à l'épreuve

Processus « Rendre un ouvrage »

1. **[Adhérent]** présente l'exemplaire à la banque de prêt.
2. **[Système]** retrouve l'emprunt en cours pour cet exemplaire.
3. **[Système]** compare la date du jour à la date de retour prévue.
4. **[Système]** enregistre le retour et clôt l'emprunt.
5. **[Système]** si l'exemplaire est attendu par un autre adhérent, le met de côté et prévient celui-ci.

À chaque étape, une seule question :
notre modèle de classes permet-il de faire cela ?

Quatre manques révélés par une simple description

Découverte	Étape
il faut pouvoir **retrouver l'emprunt en cours** : l'emprunt a donc un **statut**	2 et 4
la **date de retour réelle** manque — on n'avait que la date prévue	4
un adhérent peut **attendre** un exemplaire déjà prêté : il manque une classe `Réservation`	5
le système **prévient** quelqu'un : par quel moyen ? il manque un contact sur `Adhérent`	5

La boucle vertueuse

Modélisation statique et modélisation dynamique ne sont pas deux activités séparées : **elles se valident mutuellement**. Décrire un processus, c'est « exécuter » son diagramme de classes à la main — et cela révèle systématiquement des manques.

Quatre manques révélés par une simple description

Découverte	Étape
il faut pouvoir **retrouver l'emprunt en cours** : l'emprunt a donc un **statut**	2 et 4
la **date de retour réelle** manque — on n'avait que la date prévue	4
un adhérent peut **attendre** un exemplaire déjà prêté : il manque une classe `Réservation`	5
le système **prévient** quelqu'un : par quel moyen ? il manque un contact sur `Adhérent`	5

La boucle vertueuse

Modélisation statique et modélisation dynamique ne sont pas deux activités séparées : **elles se valident mutuellement**. Décrire un processus, c'est « exécuter » son diagramme de classes à la main — et cela révèle systématiquement des manques.

Une exigence métier déguisée en détail technique

Étape 8 : notifier *si et seulement si* le statut n'est pas complet

L'étudiant écrit spontanément « le système notifie l'administrateur ».

Karim : *« Si je reçois un mail toutes les nuits, dans un mois je les filtre, et le jour où ça plante je ne le vois pas. »*

Notifier uniquement les échecs est une exigence métier, pas une paresse d'implémentation.

C'est le pendant exact du « bouton PDF » du CM1 : la bonne solution est parfois de *ne rien faire* la plupart du temps.

Une exigence métier déguisée en détail technique

Étape 8 : notifier *si et seulement si* le statut n'est pas complet

L'étudiant écrit spontanément « le système notifie l'administrateur ».

Karim : *« Si je reçois un mail toutes les nuits, dans un mois je les filtre, et le jour où ça plante je ne le vois pas. »*

Notifier uniquement les échecs est une exigence métier, pas une paresse d'implémentation.

C'est le pendant exact du « bouton PDF » du CM1 : la bonne solution est parfois de *ne rien faire* la plupart du temps.

Une exigence métier déguisée en détail technique

Étape 8 : notifier *si et seulement si* le statut n'est pas complet

L'étudiant écrit spontanément « le système notifie l'administrateur ».

Karim : *« Si je reçois un mail toutes les nuits, dans un mois je les filtre, et le jour où ça plante je ne le vois pas. »*

Notifier uniquement les échecs est une exigence métier,
pas une paresse d'implémentation.

C'est le pendant exact du « bouton PDF » du CM1 : la bonne solution est parfois de *ne rien faire* la plupart du temps.

Un attribut « statut » cache toujours un automate

Quand un attribut ne peut prendre qu'un nombre **fini** de valeurs
Soumise → *En évaluation* → *Approuvée* / *Rejetée*

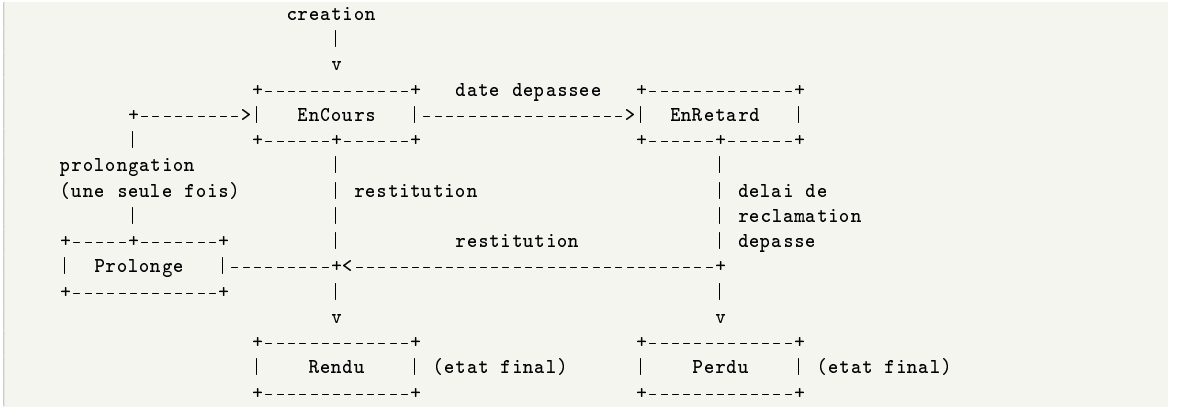
vous décrivez, sans le nommer, un **automate à états finis**.

Il se définit par

- un ensemble fini d'**états** ;
- un état **initial**, un ou des états **finaux** ;
- des **transitions** autorisées, chacune déclenchée par un événement.

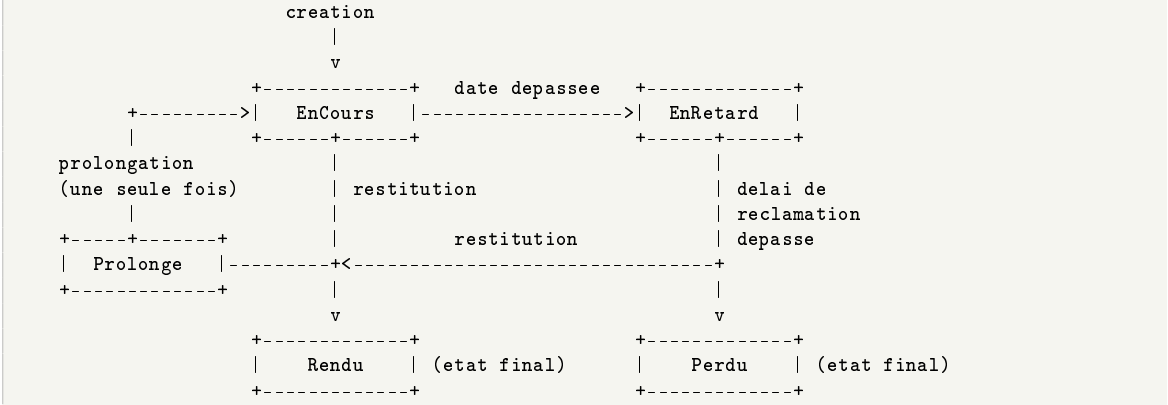
Notion centrale de l'informatique, que vous croisez aussi en théorie des langages et en conception de protocoles réseau. UML lui consacre un diagramme dédié.

L'automate de l'Emprunt



Cinq états, un état initial, **deux** états finaux. Chaque transition porte l'événement qui la déclenche.

L'automate de l'Emprunt



Cinq états, un état initial, **deux** états finaux. Chaque transition porte l'événement qui la déclenche.

L'intérêt de l'automate est ailleurs

Il ne sert pas à lister ce qui est permis.

Il rend explicite ce qui est interdit.

- *Rendu* → *EnCours* : un emprunt clos ne se rouvre pas. Si l'adhérent reprend le livre, c'est un **nouvel** emprunt.
- *Prolongé* → *Prolongé* : une seule prolongation. La règle métier devient une transition absente.
- *EnRetard* → *Prolongé* : on ne prolonge pas ce qui est déjà en retard.
- *Perdu* → quoi que ce soit : état final.

Ces quatre règles sont invisibles sur le diagramme de classes

Elles n'existent que parce qu'on a dessiné l'automate. **Repérer un attribut « statut » est le signal qu'un diagramme d'états mérite d'être dessiné.**

L'intérêt de l'automate est ailleurs

Il ne sert pas à lister ce qui est permis.

Il rend explicite ce qui est interdit.

- *Rendu* → *EnCours* : un emprunt clos ne se rouvre pas. Si l'adhérent reprend le livre, c'est un **nouvel** emprunt.
- *Prolongé* → *Prolongé* : une seule prolongation. La règle métier devient une transition absente.
- *EnRetard* → *Prolongé* : on ne prolonge pas ce qui est déjà en retard.
- *Perdu* → quoi que ce soit : état final.

Ces quatre règles sont invisibles sur le diagramme de classes

Elles n'existent que parce qu'on a dessiné l'automate. **Repérer un attribut « statut » est le signal qu'un diagramme d'états mérite d'être dessiné.**

L'intérêt de l'automate est ailleurs

Il ne sert pas à lister ce qui est permis.

Il rend explicite ce qui est interdit.

- *Rendu* → *EnCours* : un emprunt clos ne se rouvre pas. Si l'adhérent reprend le livre, c'est un **nouvel** emprunt.
- *Prolongé* → *Prolongé* : une seule prolongation. La règle métier devient une transition absente.
- *EnRetard* → *Prolongé* : on ne prolonge pas ce qui est déjà en retard.
- *Perdu* → quoi que ce soit : état final.

Ces quatre règles sont invisibles sur le diagramme de classes

Elles n'existent que parce qu'on a dessiné l'automate. **Repérer un attribut « statut » est le signal qu'un diagramme d'états mérite d'être dessiné.**

Quand une contrainte de modélisation vous rend service

La responsable revient avec une demande

« Si on retrouve un livre déclaré perdu — ça arrive — je veux pouvoir annuler la déclaration. »

Cela introduirait une transition *Perdu* → *Rendu*… qui contredit le caractère final de l'état.

La solution que la contrainte suggère

L'emprunt reste *Perdu* : c'est un fait, il a bien été perdu à cette date. L'exemplaire retrouvé fait l'objet d'une **régularisation** — un événement distinct, daté, qui le remet au catalogue.

On ne réécrit pas le passé, on enregistre ce qui arrive ensuite.

Une bonne contrainte de modélisation ne bloque pas : **elle oriente vers une meilleure solution** — et, ici, vers une donnée que le client n'avait pas pensé à demander.

Quand une contrainte de modélisation vous rend service

La responsable revient avec une demande

« Si on retrouve un livre déclaré perdu — ça arrive — je veux pouvoir annuler la déclaration. »

Cela introduirait une transition *Perdu* → *Rendu*… qui contredit le caractère final de l'état.

La solution que la contrainte suggère

L'emprunt reste *Perdu* : c'est un fait, il a bien été perdu à cette date. L'exemplaire retrouvé fait l'objet d'une **régularisation** — un événement distinct, daté, qui le remet au catalogue.

On ne réécrit pas le passé, on enregistre ce qui arrive ensuite.

Une bonne contrainte de modélisation ne bloque pas : **elle oriente vers une meilleure solution** — et, ici, vers une donnée que le client n'avait pas pensé à demander.

Quand une contrainte de modélisation vous rend service

La responsable revient avec une demande

« Si on retrouve un livre déclaré perdu — ça arrive — je veux pouvoir annuler la déclaration. »

Cela introduirait une transition *Perdu* → *Rendu*… qui contredit le caractère final de l'état.

La solution que la contrainte suggère

L'emprunt reste *Perdu* : c'est un fait, il a bien été perdu à cette date. L'exemplaire retrouvé fait l'objet d'une **régularisation** — un événement distinct, daté, qui le remet au catalogue.

On ne réécrit pas le passé, on enregistre ce qui arrive ensuite.

Une bonne contrainte de modélisation ne bloque pas : **elle oriente vers une meilleure solution** — et, ici, vers une donnée que le client n'avait pas pensé à demander.

Quand une contrainte de modélisation vous rend service

La responsable revient avec une demande

« Si on retrouve un livre déclaré perdu — ça arrive — je veux pouvoir annuler la déclaration. »

Cela introduirait une transition *Perdu* → *Rendu*… qui contredit le caractère final de l'état.

La solution que la contrainte suggère

L'emprunt reste *Perdu* : c'est un fait, il a bien été perdu à cette date. L'exemplaire retrouvé fait l'objet d'une **régularisation** — un événement distinct, daté, qui le remet au catalogue.

On ne réécrit pas le passé, on enregistre ce qui arrive ensuite.

Une bonne contrainte de modélisation ne bloque pas : **elle oriente vers une meilleure solution** — et, ici, vers une donnée que le client n'avait pas pensé à demander.

Et chez BioSeq ? Ce que vous aurez à trancher en TD

La méthode que vous venez de voir dérouler sur la bibliothèque, vous allez l'appliquer à votre propre dossier. **Les réponses ne sont pas dans ce cours.**

Le tri	Karim et vos TP ont fait naître une quinzaine de candidats. Lequel est une solution technique déguisée en concept métier ?
Les multiplicités	Une même source peut-elle relever de deux politiques de sauvegarde ? La réponse de Karim change tout.
Le processus	Lisez « Sauvegarde nocturne » étape par étape. Combien de manques votre modèle laisse-t-il apparaître ?
L'automate	Quel objet de votre modèle porte un statut ? Quels sont ses états, et surtout : **quelles transitions devez-vous interdire ?**

Cette dernière question vaut le déplacement au TP4 : vous y vérifierez si votre code respecte l'automate que vous aurez dessiné.

Et chez BioSeq ? Ce que vous aurez à trancher en TD

La méthode que vous venez de voir dérouler sur la bibliothèque, vous allez l'appliquer à votre propre dossier. **Les réponses ne sont pas dans ce cours.**

Le tri	Karim et vos TP ont fait naître une quinzaine de candidats. Lequel est une solution technique déguisée en concept métier ?
Les multiplicités	Une même source peut-elle relever de deux politiques de sauvegarde ? La réponse de Karim change tout.
Le processus	Lisez « Sauvegarde nocturne » étape par étape. Combien de manques votre modèle laisse-t-il apparaître ?
L'automate	Quel objet de votre modèle porte un statut ? Quels sont ses états, et surtout : **quelles transitions devez-vous interdire ?**

Cette dernière question vaut le déplacement au TP4 : vous y vérifierez si votre code respecte l'automate que vous aurez dessiné.

Ce qui vous attend

☞ TD4 — Modéliser BioSeq

Tri des concepts (et débat sur « lien dur »), multiplicités, classe-association, lecture du processus nocturne, puis construction de l'**automate à états** et de ses transitions interdites.

☞ TP4 — Votre code respecte-t-il votre automate ?

Vous venez d'affirmer qu'un instantané interrompu ne peut pas être *Complet*.

Au TP, vous lancerez une sauvegarde et la tuerez d'un `kill -9` en plein parcours.

Puis vous regarderez ce que votre outil a laissé sur le disque.

Suivront : la correction par écriture atomique, la revue croisée entre groupes, et les soutenances individuelles.

Ce qui vous attend

☞ TD4 — Modéliser BioSeq

Tri des concepts (et débat sur « lien dur »), multiplicités, classe-association, lecture du processus nocturne, puis construction de l'**automate à états** et de ses transitions interdites.

☞ TP4 — Votre code respecte-t-il votre automate ?

Vous venez d'affirmer qu'un instantané interrompu ne peut pas être *Complet*.

Au TP, vous lancerez une sauvegarde et la tuerez d'un `kill -9` en plein parcours.

Puis vous regarderez ce que votre outil a laissé sur le disque.

Suivront : la correction par écriture atomique, la revue croisée entre groupes, et les soutenances individuelles.

Ce qui vous attend

☞ TD4 — Modéliser BioSeq

Tri des concepts (et débat sur « lien dur »), multiplicités, classe-association, lecture du processus nocturne, puis construction de l'**automate à états** et de ses transitions interdites.

☞ TP4 — Votre code respecte-t-il votre automate ?

Vous venez d'affirmer qu'un instantané interrompu ne peut pas être *Complet*.

Au TP, vous lancerez une sauvegarde et la tuerez d'un `kill -9` en plein parcours.

Puis vous regarderez ce que votre outil a laissé sur le disque.

Suivront : la correction par écriture atomique, la revue croisée entre groupes, et les soutenances individuelles.

Le mot de la fin : ce que ce cours aura montré

CM1	un besoin mal compris coûte cent fois plus cher en production
CM2	la formalisation préserve la liberté de conception
CM3	la valeur est dans les cas d'exception, et une doublure les rend testables
CM4	le modèle parle le langage du client ; le processus le met à l'épreuve

Passer du temps sur l'analyse n'est jamais du temps perdu.

C'est ce qui garantit que l'effort de développement, toujours coûteux,
est dirigé vers le bon objectif.

Le mot de la fin : ce que ce cours aura montré

CM1	un besoin mal compris coûte cent fois plus cher en production
CM2	la formalisation préserve la liberté de conception
CM3	la valeur est dans les cas d'exception, et une doublure les rend testables
CM4	le modèle parle le langage du client ; le processus le met à l'épreuve

Passer du temps sur l'analyse n'est jamais du temps perdu.

C'est ce qui garantit que l'effort de développement, toujours coûteux,
est dirigé vers le bon objectif.

À retenir

1. Le diagramme de classes du domaine est un **dictionnaire visuel validé par le client** : pas de méthodes, pas de types techniques, pas de classes de solution.
2. Le **langage ubiquitaire** : un seul vocabulaire pour le métier, l'analyse et le code.
3. Méthode : les noms donnent les classes, les verbes donnent les associations, puis on raffine et on fixe les multiplicités *avec le client*.
4. En cas de doute sur une classe : **le client la comprendrait-il sans explication ?**
5. On **spécialise** quand « B *est un* A » se dit sans grimacer — jamais par simple ressemblance d'attributs.
6. Un attribut qui n'appartient ni à l'une ni à l'autre des classes appartient à **leur relation** : c'est une **classe-association**, et elle deviendra une table de jonction.
7. Décrire un **processus métier**, c'est exécuter son modèle à la main : cela révèle systématiquement des manques.
8. Un attribut « statut » cache un **automate à états** ; son intérêt est de rendre explicites les **transitions interdites**.