Analyse du domaine

Gerson Sunyé
gerson.sunye@univ-nantes.fr

TODO

- How to Avoid Use-Case Pitfalls:
- http://www.ddj.com/architect/184414560
- parler de CRUD [DONE]

Plan

- Définition d’analyse du domaine
- Objets et actions
- Processus d’analyse
- Cas d’utilisation
- Étude de cas
- Derniers conseils

Analyse du domaine

- Modélisation du métier, et non de la solution proposée
- Utilisée pour comprendre le métier
- Peut servir pour différents projets

Objets et actions

Concepts de base

- Objet: information + comportement.
- Action: quelque chose qui se passe.
  - Les actions sont (ou peuvent être) indépendantes des objets.
  - Elles peuvent être liées aux objets plus tard (lors de la conception).
  - Les actions modifient les objets.

Concepts de base

- Les actions sont au même niveaux que les objets.
- Un modèle bien découpé demande une réflexion précise sur les actions et ce qu’elles réalisent.

Paradigme des fractales

- Une image fractale a la même apparence à tous les niveaux
- Objets: départements d’affaires, machines, composants logiciels, concepts de langage de programmation.
- Actions représentent les interactions entre objets: accords d’affaires, appels téléphoniques, tours en vélo, etc.

Actions (1/2)

- Une action représentes les collaborations entre objets.
- Les collaboration sont utilisées pour modéliser:
  - ce qu'il se passe à l'intérieur d'un composant logiciel.
  - interaction entre composants.
  - analyse du domaine: comment les objets du monde réel interagissent?

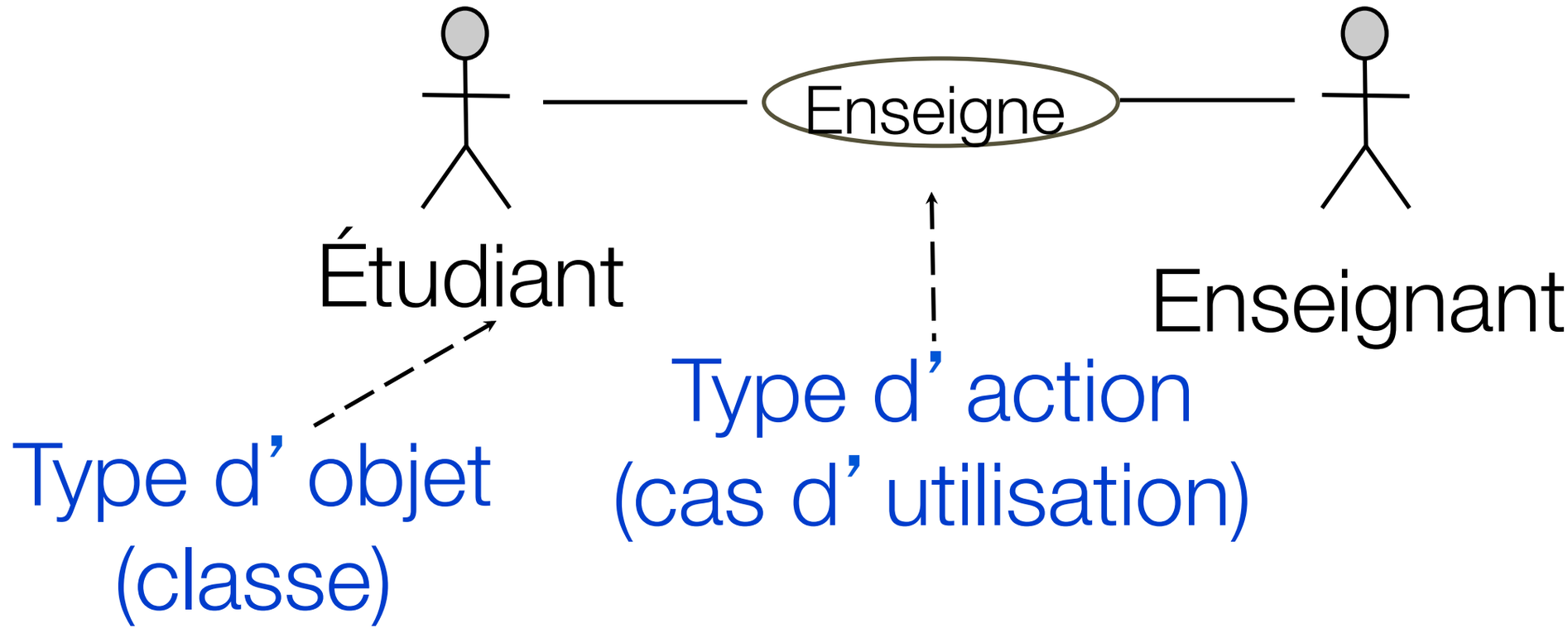
Actions (2/2)

- Les Actions ont des participants (objets)
  - Combien d'objets sont nécessaires? Autant qu'il faut pour décrire les actions.
  - Les associations fournissent un vocabulaire qui sert à décrire les effets d'une action

Deux sortes d’actions

- Non orientée
  - dite aussi «action conjointe».
  - symétrique, à n participants.
  - En UML: cas d’utilisation.
- **Orientée** (un émetteur, un destinataire):
  - dite aussi «action localisée».
  - En UML: opération.

Les objets participent aux actions



- Un diagramme seul n’est pas suffisant. Il doit être accompagné par une explication.

Concepts de base

- Les responsabilités ne sont pas assignées aux objets dès le début.
- Étapes:
  - Ce qu'il se passe.
  - Quel objet est responsable du déclenchement de ce qu'il se passe.

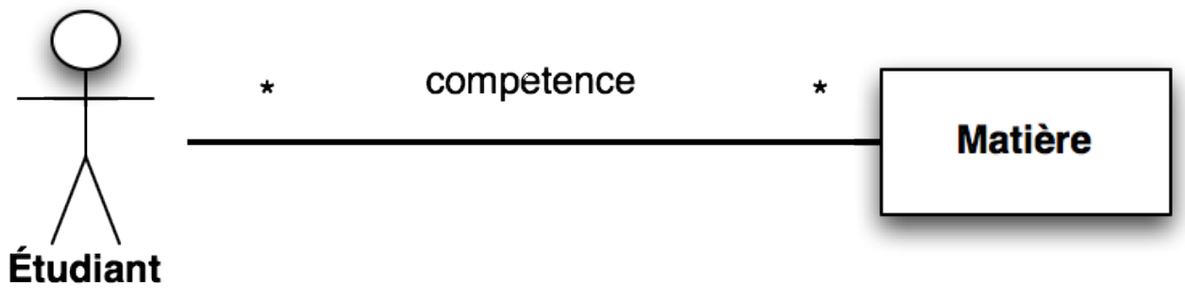
Les Actions affectent les Objets

- Les objets sont utilisés pour décrire l'effet des action sur les participants.
- Les actions sont caractérisées par ce qu'elles accomplissent.
- De nouveaux termes sont introduits pour décrire l'effet (la post-condition).

Les Actions affectent les Objets

action(étudiant, enseignant)::enseigne(matière)
post -- cette matière a été ajoutée aux *compétences* de l’ étudiant

Les Actions affectent les Objets

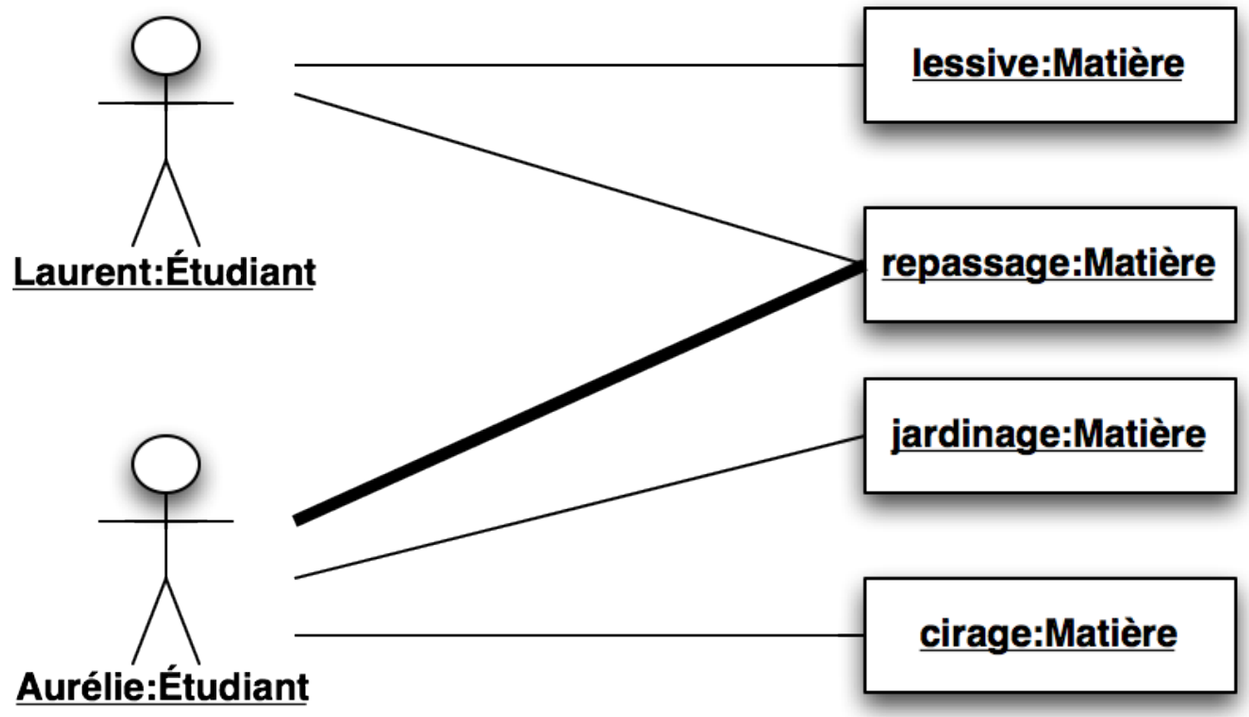


- De nouveaux termes sont introduits pour décrire l'effet (ou la post-condition)
- Les étoiles indiquent qu'un étudiant peut avoir plusieurs compétences
- La figure est une indication (optionnelle) qu'un étudiant est un rôle

Les Actions affectent les Objets

- Utilisation d'instantanés (snapshots) pour visualiser une action

- Les lignes épaisses sont utilisées pour représenter une création de lien



action(Aurélie, unEnseignant)::enseigne(repassage)

Les Actions affectent les Objets

- Généralement, les modifications sont présentées en gras (ou en couleurs)
- Les associations représentent des relations du monde réel ou d'un logiciel (Laurent dans un SGBD)
- L'utilisation d'associations fournit:
  - le modèle statique (vocabulaire pour les actions)
  - le modèle dynamique (un guide clair des objets requis)

Spécification précise

- **action**(étudiant,enseignant)::enseigne (matière)

- **post** étudiants.compétences = étudiant.compétences@pre.including(matière)

Processus d’analyse

Processus d’analyse (1/3)

- Questions initiales:
  - Que faites vous?
  - Avec qui vous traitez?
- Chaque verbe correspond à un comportement (ou action).

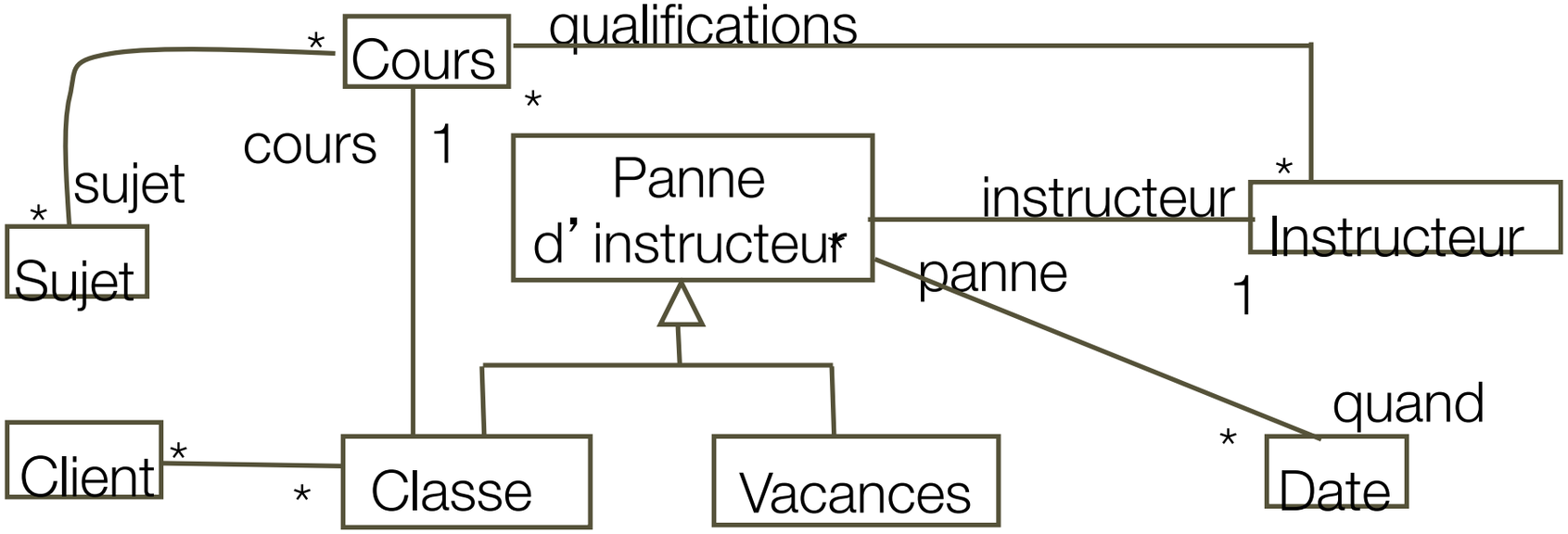
Processus d'analyse (2/3)

- Ajouter à la liste de questions: «qui participe à l'action?» et «quel est l'impact de l'action sur leur état?»
- Les réponses (à ces questions) conduisent aux classes (types d'objets) et aux post-conditions des actions.
- Les types d'objets et les post-conditions conduisent aux associations et aux attributs qui illustrent les effets.

Processus d’analyse (3/3)

- Nouvelle classe: ajouter à la liste de questions:
  - «Quelles actions l’affectent?»
  - «Quelles actions sont affectées par cette classe?»

Modèle du domaine



- Modèles statiques et leurs invariants:
  - Montrent les relations statiques et les attributs

Analyse du domaine

- Un invariant, comme une post-condition, est écrit en suivant les liens à partir d'un point de départ.
- Ils sont décrits de manière informelle et écrits dans un langage de conditions booléennes (OCL).

Analyse du domaine

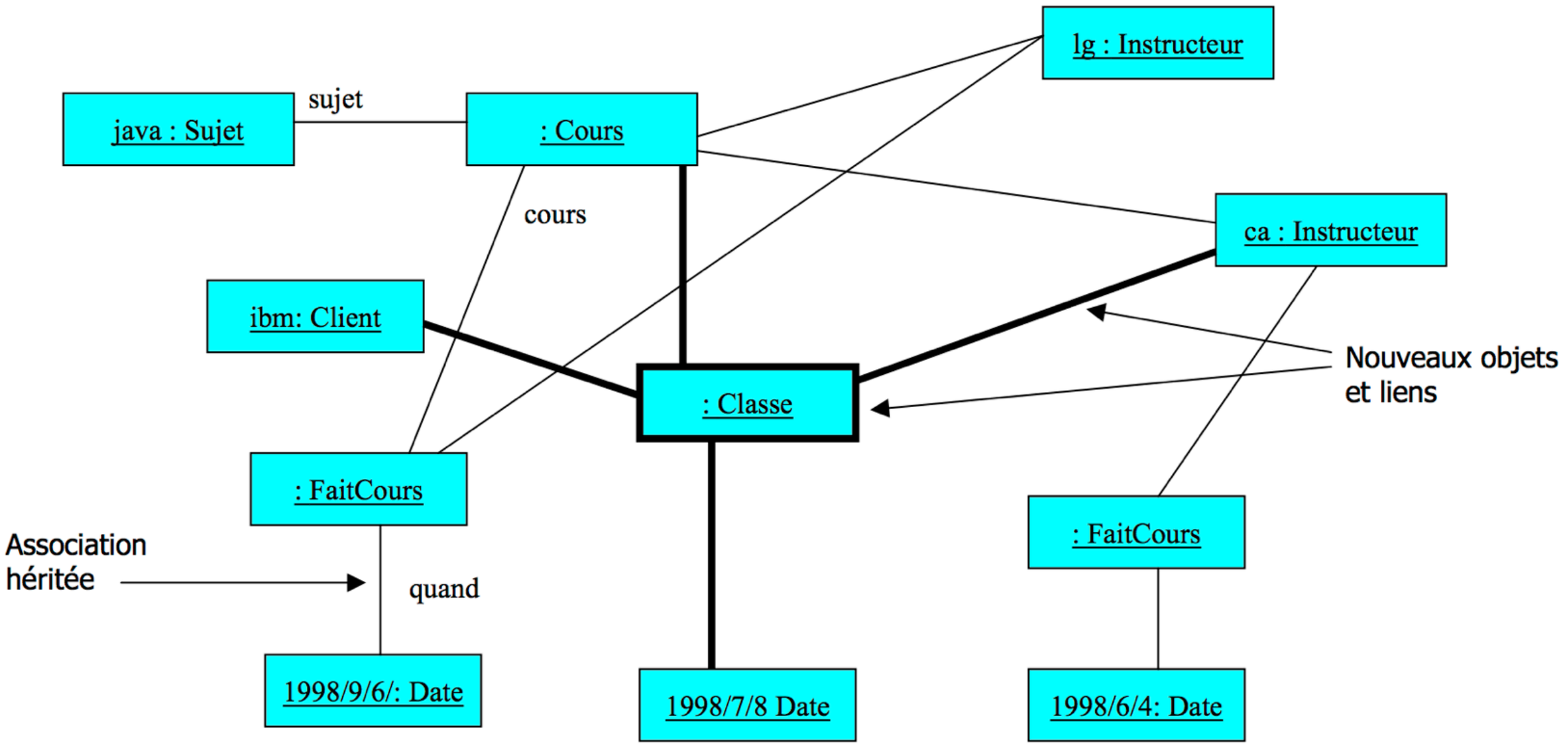
- L'analyste a décidé que les vacances et les classes un cours sont des sortes de panne d'instructeur (indisponibilité).

```
inv -- pour toute Classe, les qualifications de l' instructeur doivent inclure le cours:
Classe::instructeur.qualifications->includes(cours)

inv -- pour tout Instructeur et pour toute Date, le nombre de panne
        d' instructeurs ne peut pas dépasser 1
Instructeur:: d:Date :: panne[quand = d] <= 1
```

Analyse du domaine

- Instantanés:



Analyse du domaine

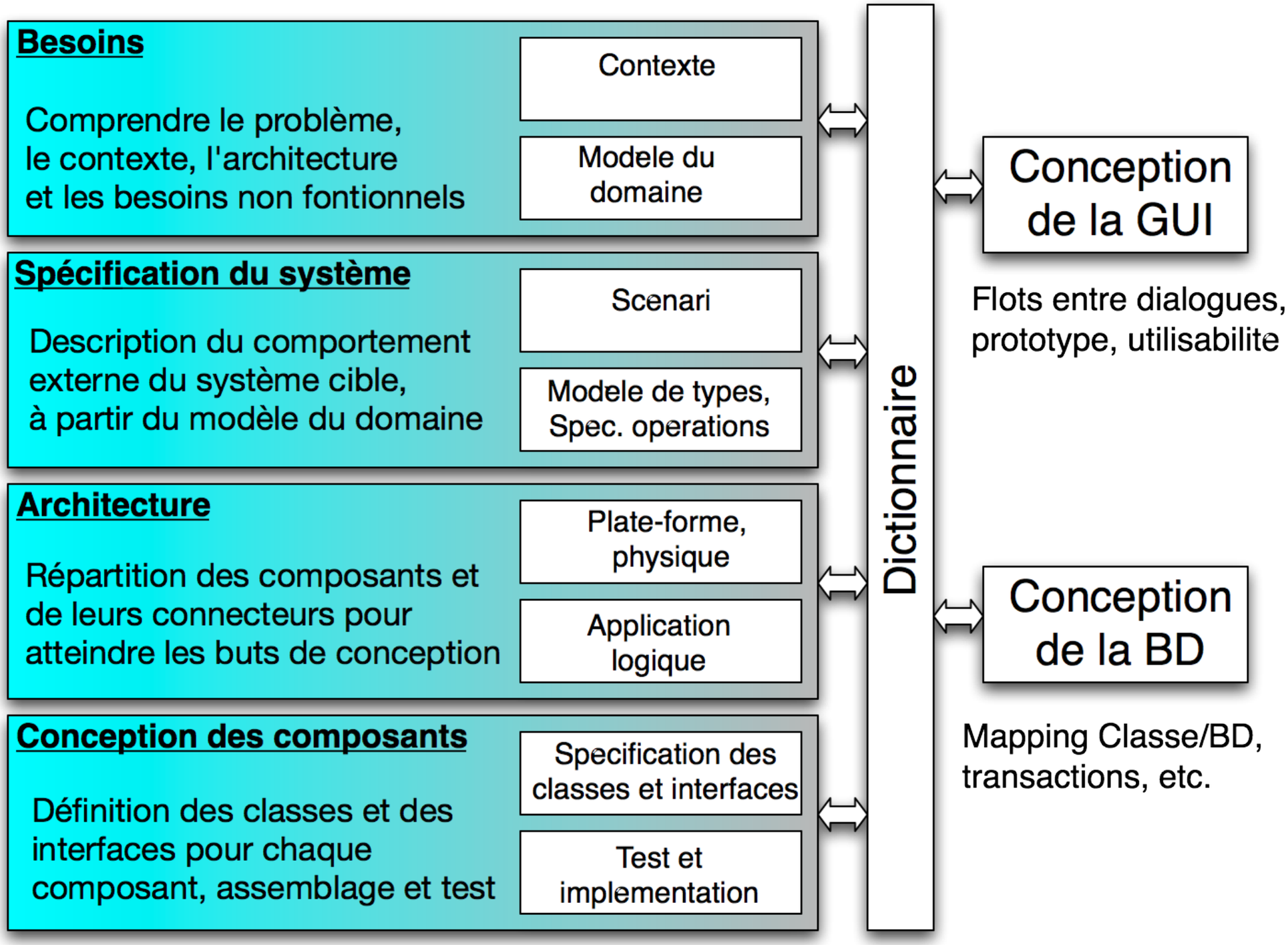
- Les post-conditions des actions sont écrites en termes d'associations statiques.

```
action (c:Client, s:CompagnieSéminaires) :: programme (t: Cours, d: Date)
post-- une nouvelle Classe est créée avec une date, un client et un cours
        r: Classe.new [date = d, client = c, cours = t,
                -- Un instructeur, qui était libre et qualifié pour le cours, est affecté
            instructeur.panne @ pre[quand = d] = 0 and
            instructeur.qualifications -> includes( t )]
```

Aperçu du processus

- Modéliser, concevoir, implémenter et tester, périodiquement
- Différent chemins possibles à l’intérieur de la méthode
- Le processus est non-linéaire, itératif et parallèle

Processus


Besoins
Comprendre le problème, le contexte, l'architecture et les besoins non fontionnels
Contexte
Modele du domaine
Spécification du système
Description du comportement externe du système cible, à partir du modèle du domaine
Scenari
Modele de types, Spec. operations
Architecture
Répartition des composants et de leurs connecteurs pour atteindre les buts de conception
Plate-forme, physique
Application logique
Conception des composants
Définition des classes et des interfaces pour chaque composant, assemblage et test
Specification des classes et interfaces
Test et implementation
Dictionnaire
Conception de la GUI
Flots entre dialogues, prototype, utilisabilite
Conception de la BD
Mapping Classe/BD, transactions, etc.

Modélisation du domaine

Techniques de modélisation

- Description textuelle.
- Dictionnaire.
- Carte heuristique (Concept Map).
- Langage du domaine.
- UML.

Dictionnaire (1/2)

- Permet de figer la terminologie du domaine d'application.
- Constitue le point d'entrée et le référentiel initial du système.
- Outil de dialogue informel, évolutif et simple à réaliser.

Dictionnaire (2/2)

Exemple pour un simulateur de vol :

Notion	Définition	Traduit en ...	Nom informatique
Pilotage	Action de piloter un avion en enchaînant des manoeuvres élémentaires	Package	Pilotage
Instrument	Organe d'interaction entre le pilote et l'avion ou entre l'avion et le pilote	Classe abstraite	Instrument
Manette des gaz	Instrument qui permet d'agir sur la quantité de carburant injectée dans le moteur	Classe	Manette_gaz
Action	**Définition**	**Traduit en ...**	**Nom informatique**
Mettre les gaz à fond	Action qui permet d'injecter le maximum de carburant pour atteindre la vitesse maximale	Opération	Mettre_a_fond

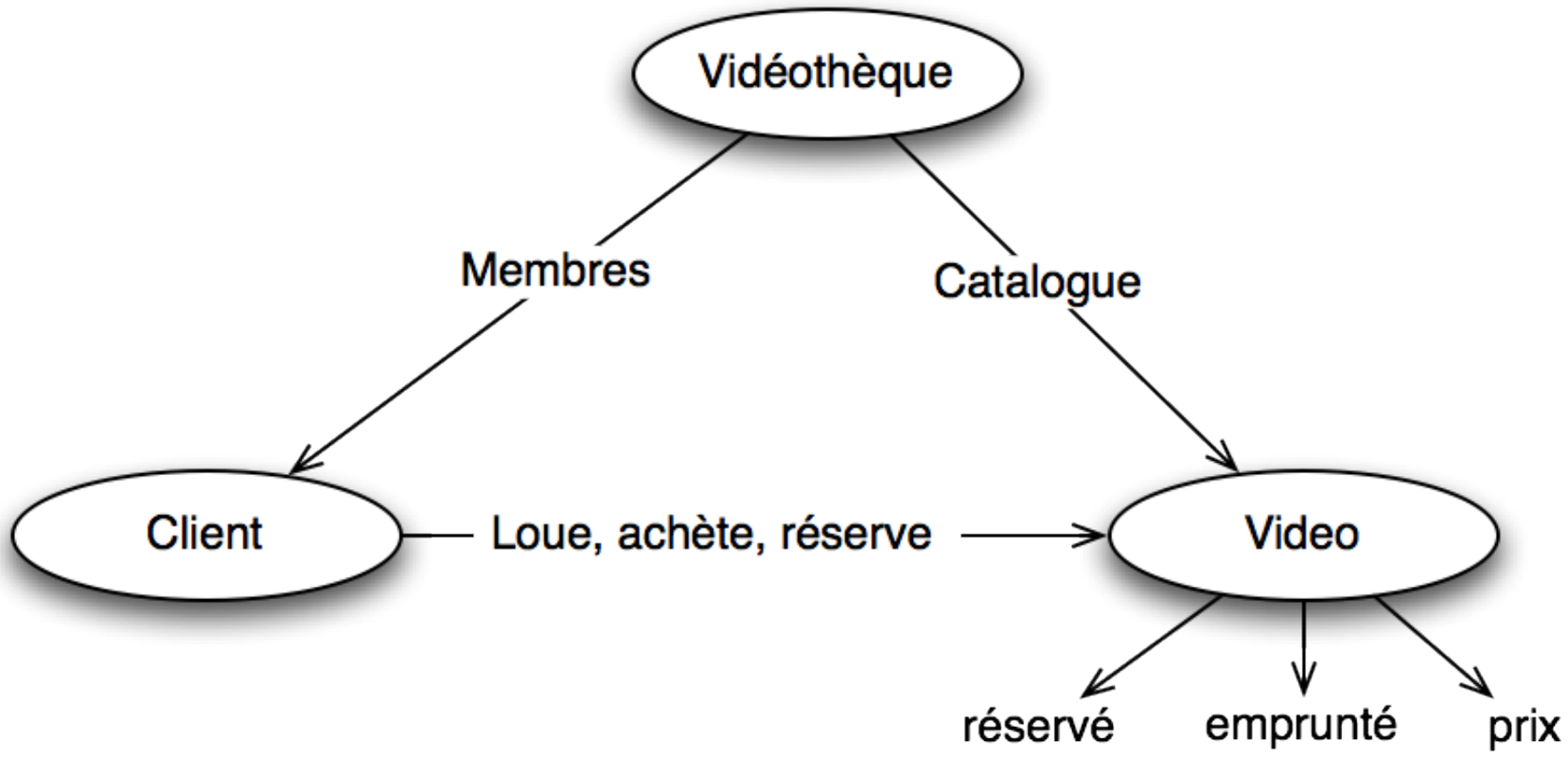
Carte heuristique (1/3)
[Novak 84]

- Une carte heuristique est une représentation graphique d'une base de connaissances déclaratives qui possède une organisation hiérarchique.

Carte heuristique (2/3)

- Représentation informelle (bien que structurée) des termes en relation d’un domaine.

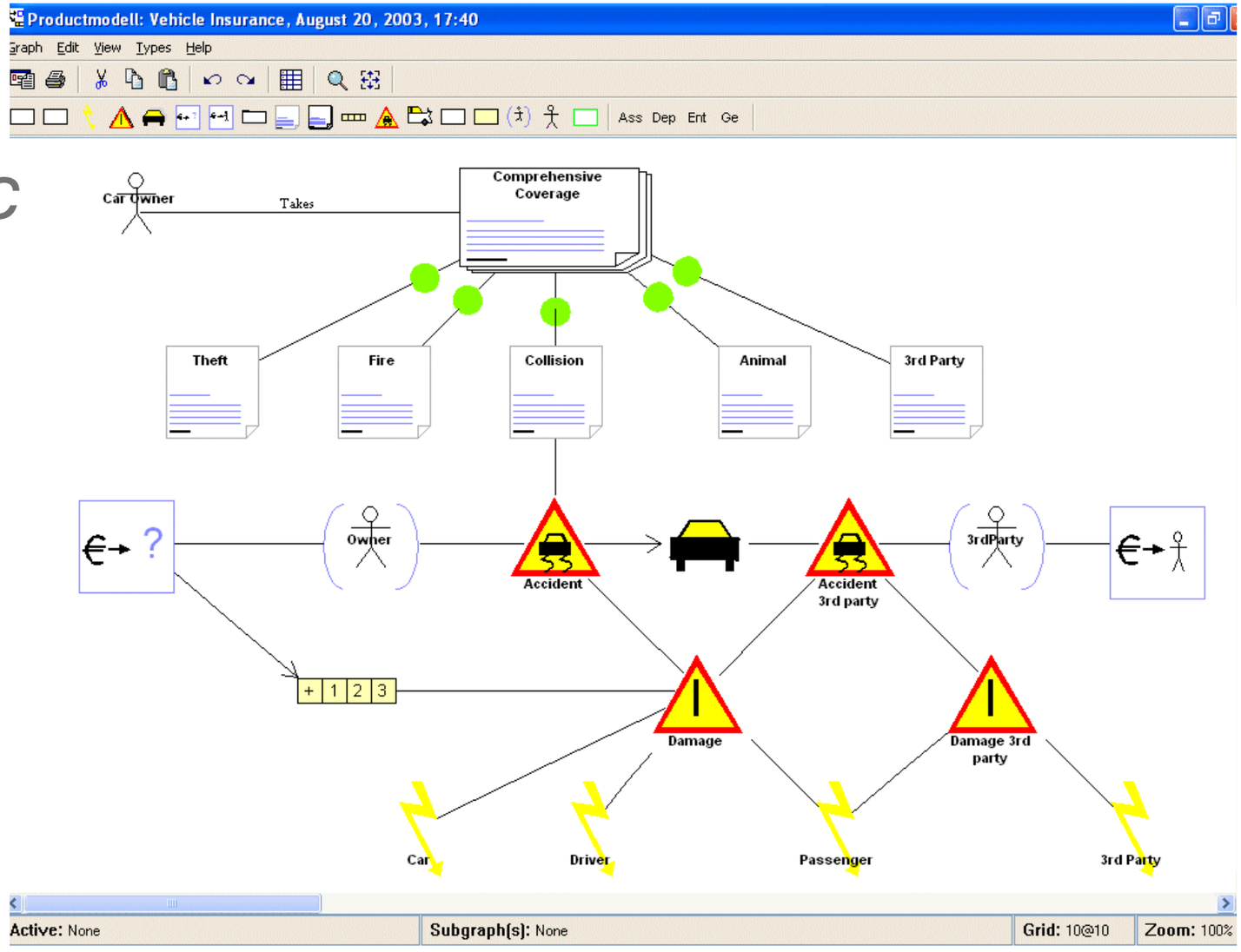
Carte heuristique (3/3)



Langage du domaine

- *DSL - Domain Specific Language*.

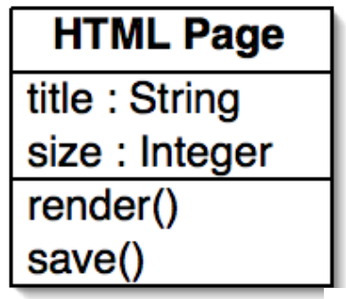
- Spécification d'un langage à partir du vocabulaire du domaine.



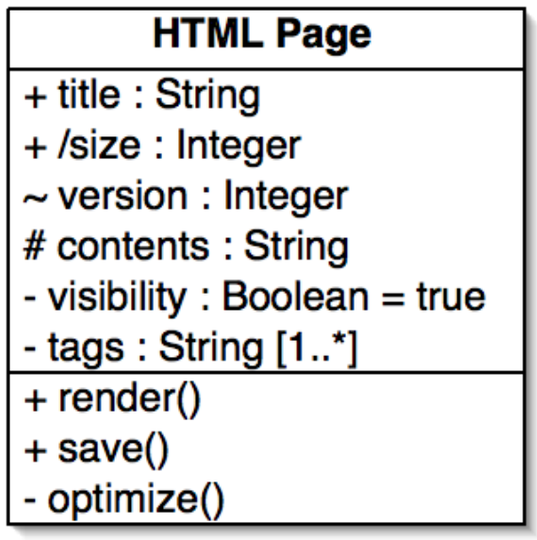
UML - Diagramme de classes

- Sans détails

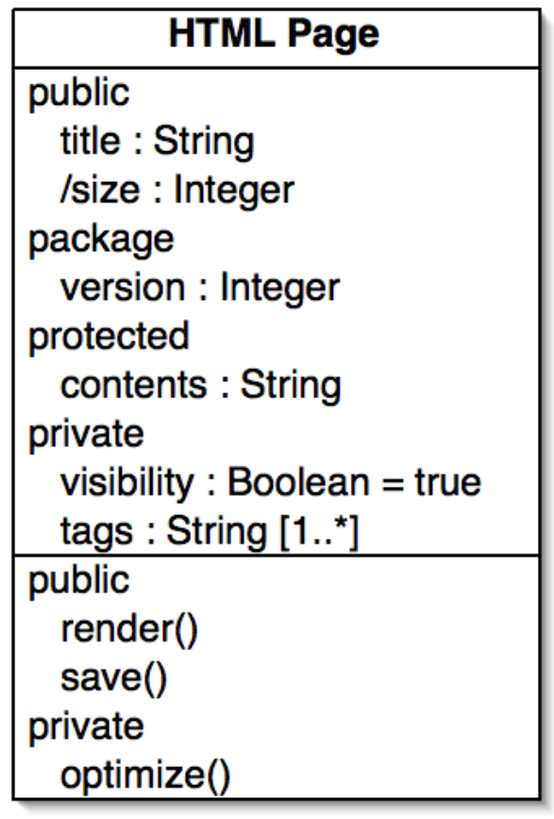
- Analyse



- Conception

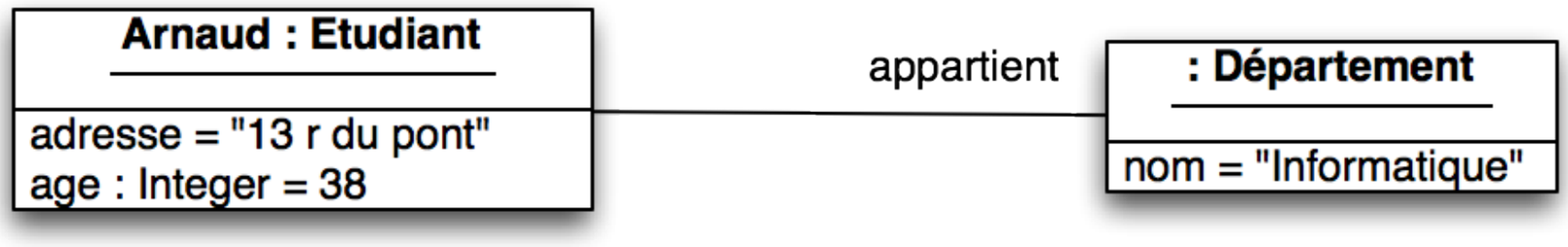


- Propriétés groupées selon leur visibilité



UML - Diagramme d’instances

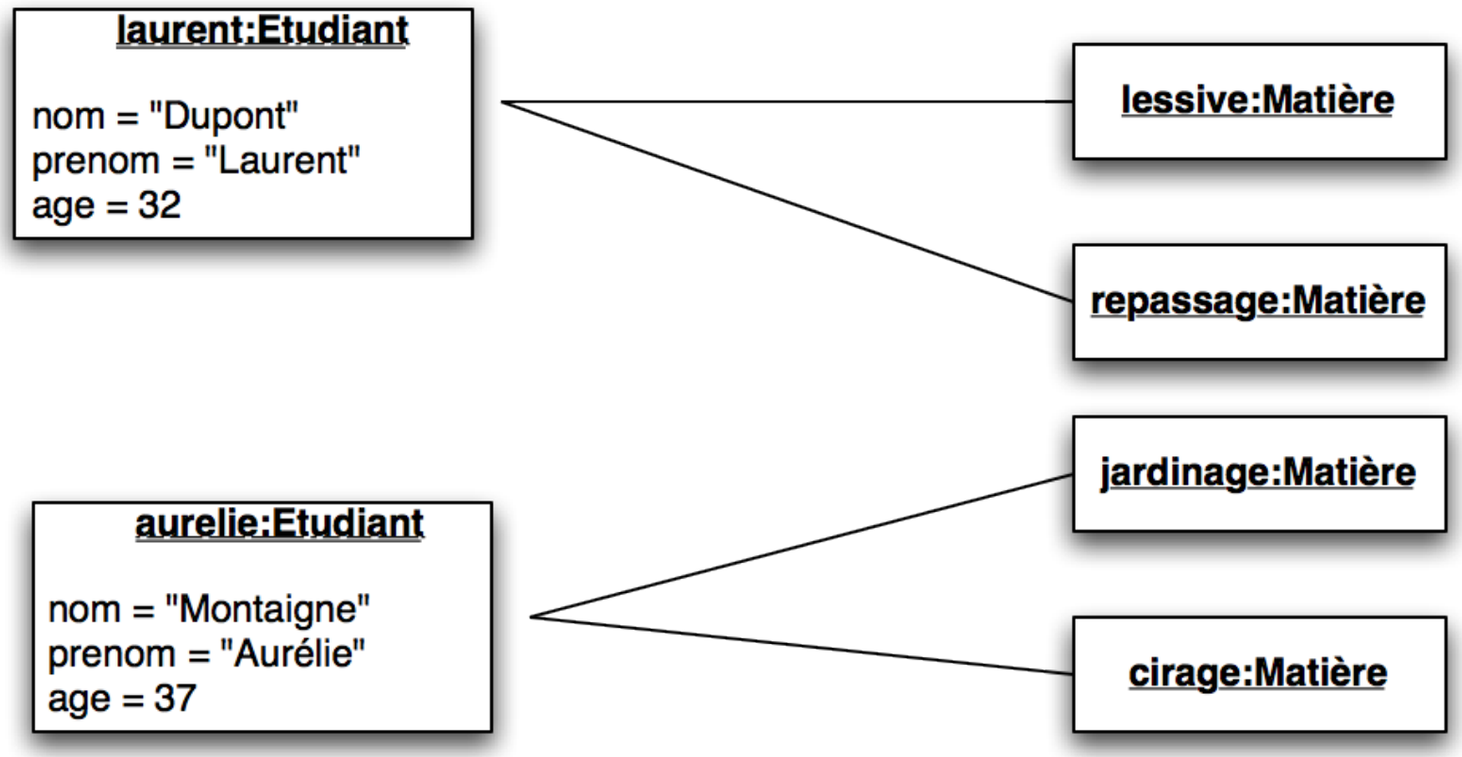
- Similaire à celle d’une classe
- Le nom de l’instance et celui de la classe sont soulignés et séparés par “:”
- Les deux noms sont facultatifs



Instantanés (1/2)

- Un diagramme d'instances représentant l'état d'un système à un moment donné.
- Représentation du problème (métier), non obligatoirement un logiciel:
  - Les objets et liens peuvent représenter aussi bien des fiches en papier que des tuples d'une base de données.

Instantanés (2/2)

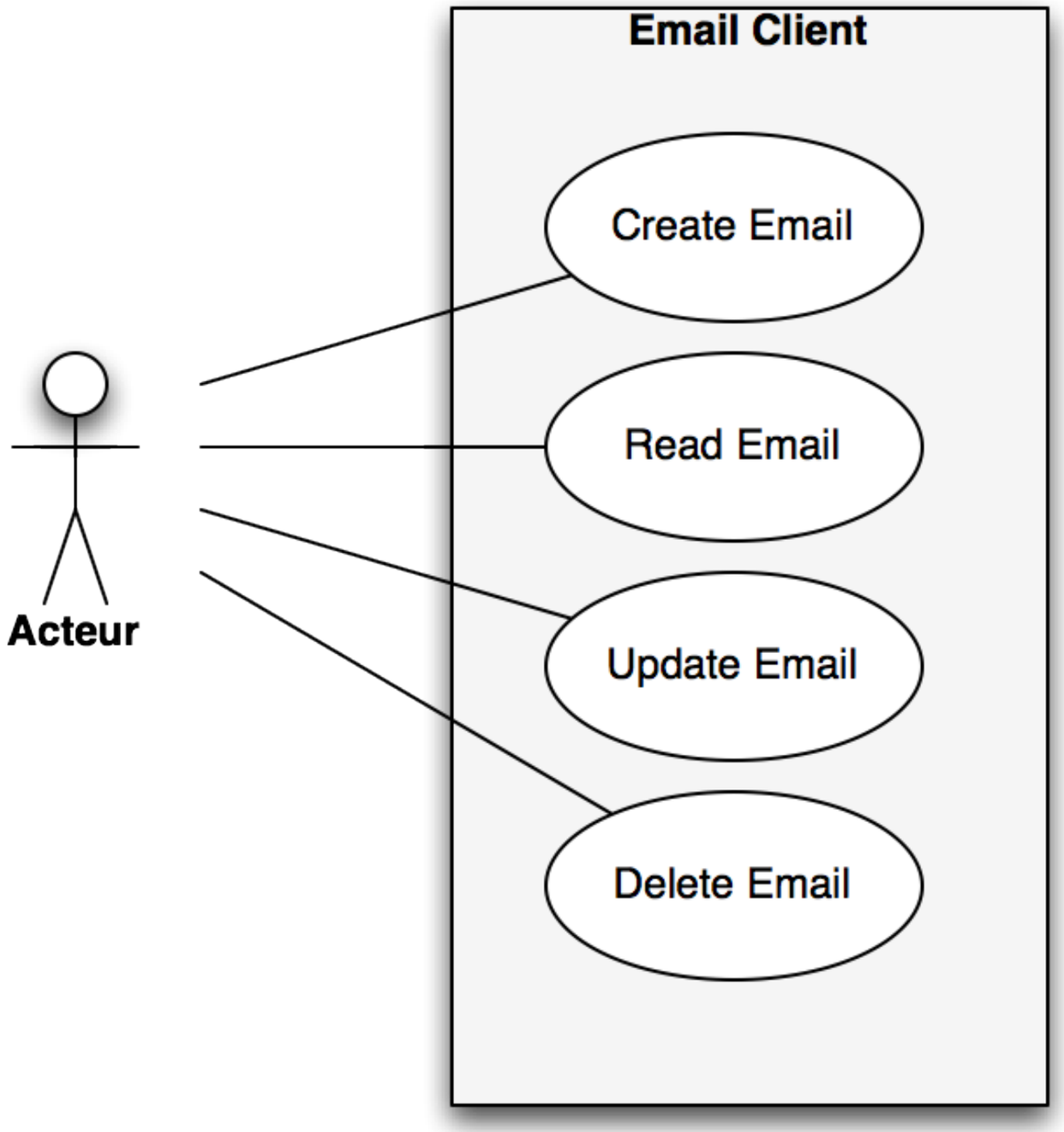


Cas d’utilisation

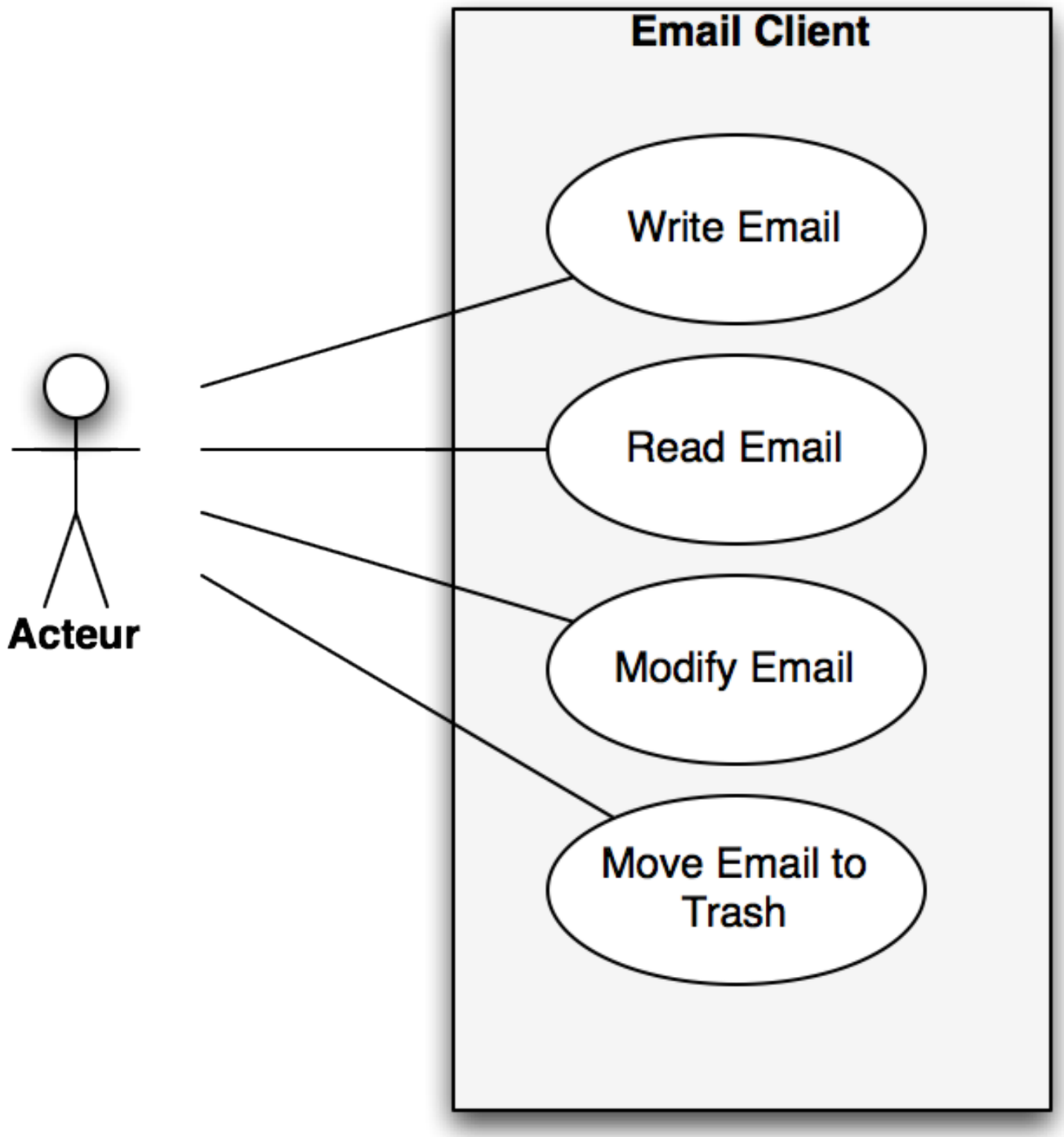
Les cas d'utilisation (use-cases)

- Un cas d'utilisation est une manière particulière d'utiliser le système:
  - séquence d'interactions entre le système et un ou plusieurs acteurs.
  - Ils s’expriment par des diagrammes de séquences (scénarios).

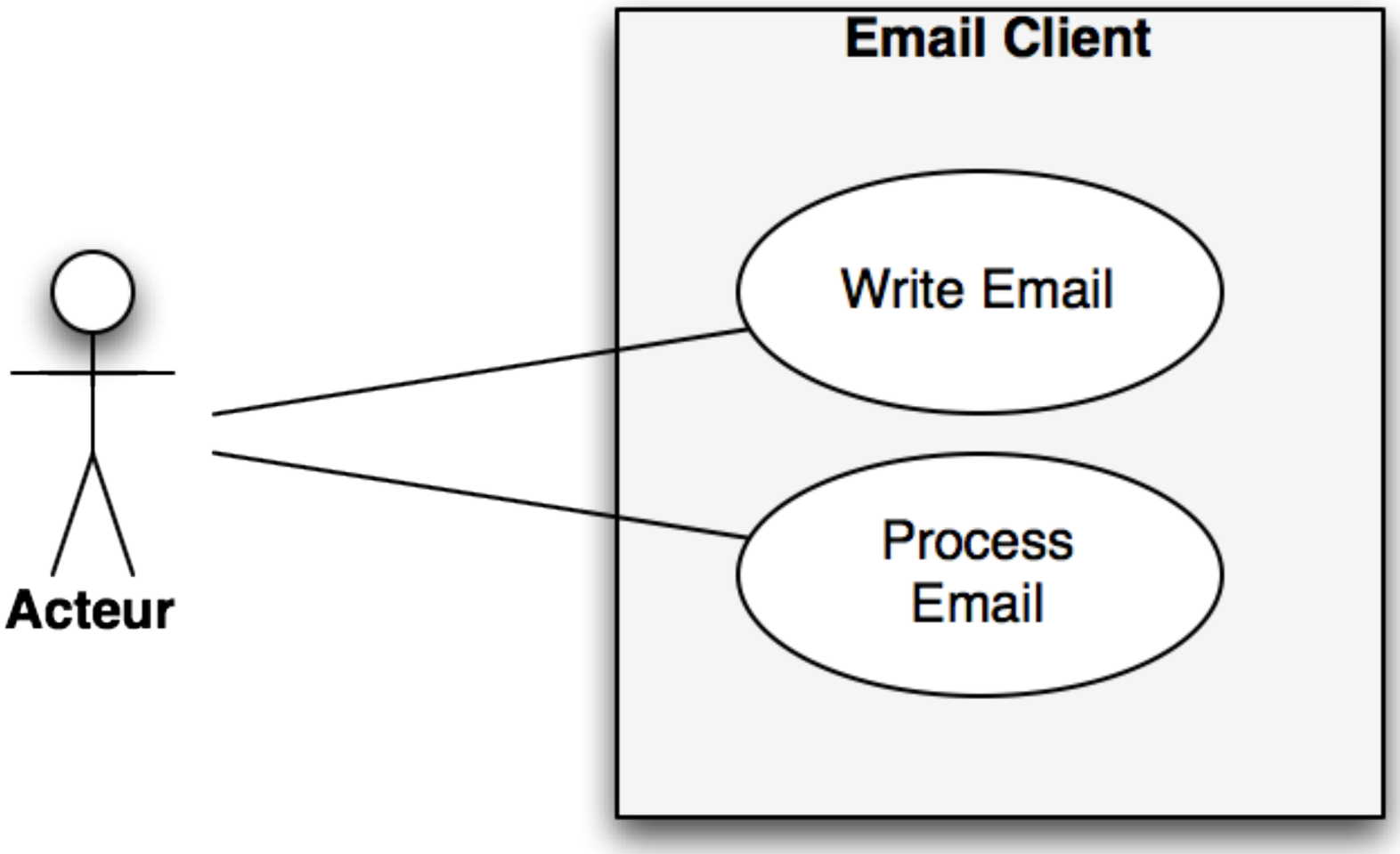
Les cas d’utilisation



Les cas d'utilisation



Les cas d’utilisation



Les cas d'utilisation

- La compilation des cas d'utilisation décrit de manière informelle le service rendu par le système
    - fournissent une expression "fonctionnelle" du besoin
    - peuvent piloter la progression d’un cycle en spirale
- Les cas d'utilisation sont nommés en utilisant la terminologie décrite dans le dictionnaire

Cas d’utilisation

- Un cas d’utilisation est un texte.
- Écrire des bons cas d’utilisation est surtout une question de style.

Canevas de description [Cockburn]

- **Use Case**: <number> <the name should be the goal as a short active verb phrase>
- **Goal in Context**: <a longer statement of the goal, if needed>
- **Scope**: <what system is being considered black-box under design>
- **Level**: <one of: Summary, Primary task, Subfunction>
- **Primary Actor**: <a role name for the primary actor, or description>
- **Priority**: <how critical to your system / organization>
- **Frequency**: <how often it is expected to happen>

Canevas de description (suite)

- Pre-conditions
- Post-conditions
- Main success scenario :
  1.<description de l'action>
  2.<description de l'action>
  3....
  4.<description de la dernière action avant la fin du cas>

Canevas de description (suite)

- Extensions
  - <#> : <condition> : <action or use-case>
  - <#> : <condition> : <action or use-case>
- Variations
  - <#> : <action or use-case>
- Superordinate Use Case: <optional, name of use case that includes this one>
- Subordinate Use Cases: <optional, depending on tools, links to sub.use cases>

Canevas de description (suite)

- Performance Target: <the amount of time this use case should take>
- Open Issues
- Schedule

Canevas de description
(fin)

- [Contraintes]
- [Annexes]

Exemple

- Use Case: 5 Buy Goods
- CHARACTERISTIC INFORMATION
- Goal in Context: Buyer issues request directly to our company, expects goods shipped and to be billed.
- Scope: Company
- Level: Summary
- Preconditions: We know Buyer, their address, etc.

- **Success End Condition**: Buyer has goods, we have money for the goods.

- **Failed End Condition**: We have not sent the goods, Buyer has not spent the money.

- **Primary Actor**: Buyer, any agent (or computer) acting for the customer

- **Trigger**: purchase request comes in.

- Main Sucess Scenario

1. Buyer calls in with a purchase request.

2. Company captures buyer's name, address, requested goods, etc.

3. Company gives buyer information on goods, prices, delivery dates, etc.

4. Buyer signs for order.

5. Company creates order, ships order to buyer.

6. Company ships invoice to buyer.

7. Buyers pays invoice

- **Extensions**
  - 3a. Company is out of one of the ordered items:
    - 3a1. Renegotiate order.
  - 4a. Buyer pays directly with credit card:
    - 4a1. Take payment by credit card (use case 44)
  - 7a. Buyer returns goods:
    - 7a. Handle returned goods (use case 105)

- **Sub-variations**
  - 1. Buyer may use phone in, fax in, use web order form, electronic interchange
  - 7. Buyer may pay by cash or money order check

- **RELATED INFORMATION**
- **Priority**: top
- **Performance Target**: 5 minutes for order, 45 days until paid
- **Frequency**: 200/day
- **Superordinate Use Case**: Manage customer relationship (use case 2)

- **Subordinate Use Cases**:
  - Create order (use case 15)
  - Take payment by credit card (use case 44)
  - Handle returned goods (use case 105)
- **Channel to primary actor**: may be phone, file or interactice
- **Secondary Actors**: credit card company, bank, shipping service
- **Channels to Secondary Actors**:

- **Open Issues**
  - What happens if we have part of the order?
  - What happens if credit card is stolen?
- **Schedule**
  - **Due Date**: release 1.0

Conseils

- Ne pas mélanger l’étude de l’IHM et l’étude fonctionnelle pour un cas d’utilisation.

- Les cas d'utilisation ne sont pas de simples fonctions.

- Ne pas chercher à détailler les cas d’utilisation en une seule fois.

- Ne pas hésiter à détailler le cas d’utilisation à l’aide de diagrammes UML.

Recommandations d'écriture [Cockburn, 2003]

- Partir du sommet (les grandes fonctions) et se maintenir le plus possible au niveau utilisateur.
- Centrer son attention sur le cas nominal (un scénario typique de succès).
- Préciser toujours les parties prenantes et leurs intérêts.

- Utiliser un verbe au présent de l'indicatif à chaque étape.

- Utiliser la voix active pour décrire les sous-objectifs en cours de satisfaction.

- Le sujet doit être clairement localisable (en début de phrase généralement).

- Rester concis et pertinent (éviter les longs documents).

- Eviter les si, et placer les comportements alternatifs dans les extensions .

- Signaler les sous-cas d'utilisation. Ils sont toujours représentés par la relation d'inclusion « include» d'UML.

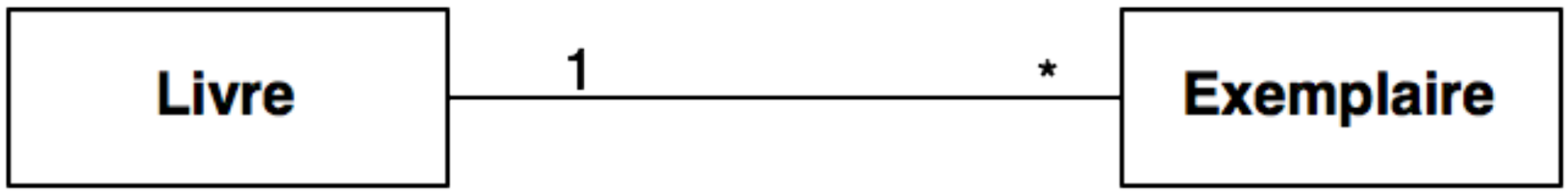
- Identifier le bon objectif.

- Signaler la portée.

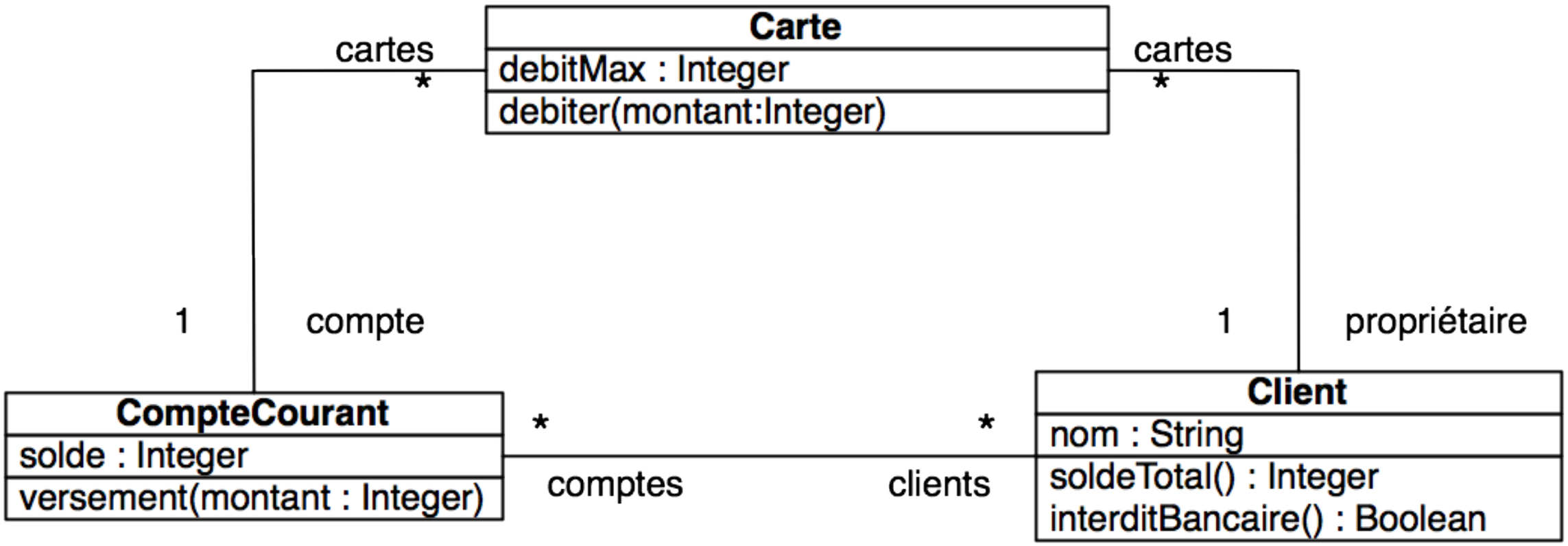
- Laisser de côté l'interface utilisateur.

Patrons d’analyse

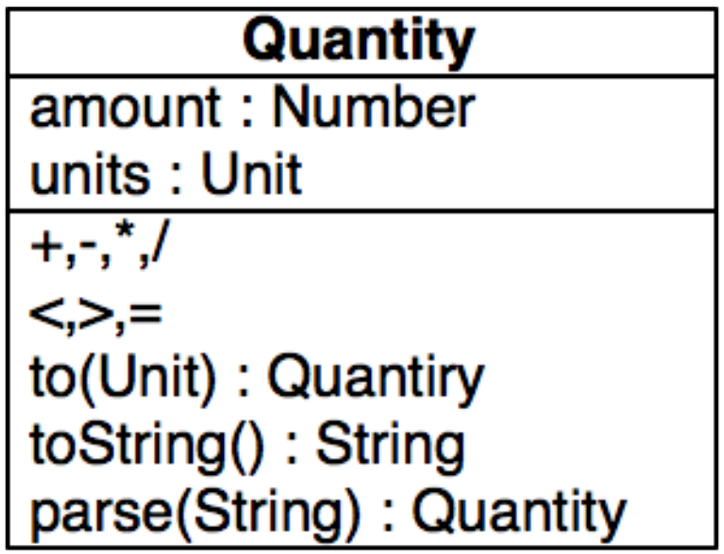
Articles et descripteurs [Catalysis]



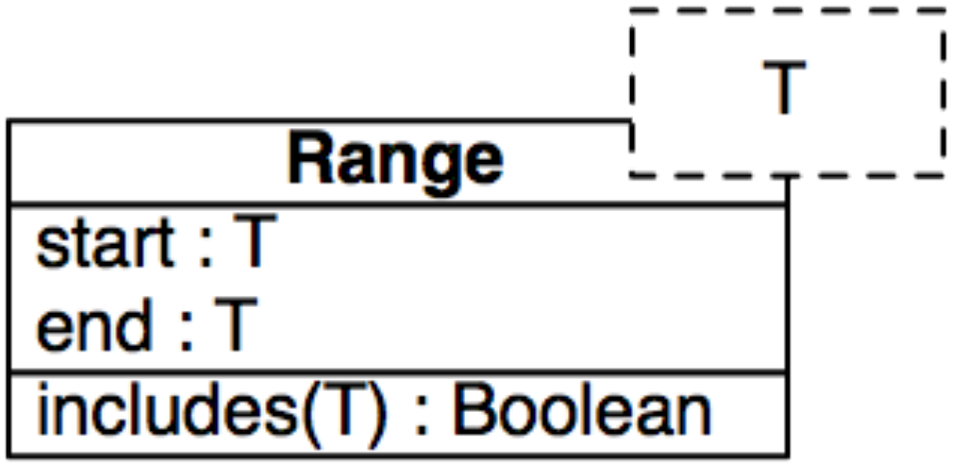
Invariants à partir de boucles d'associations [Catalysis]



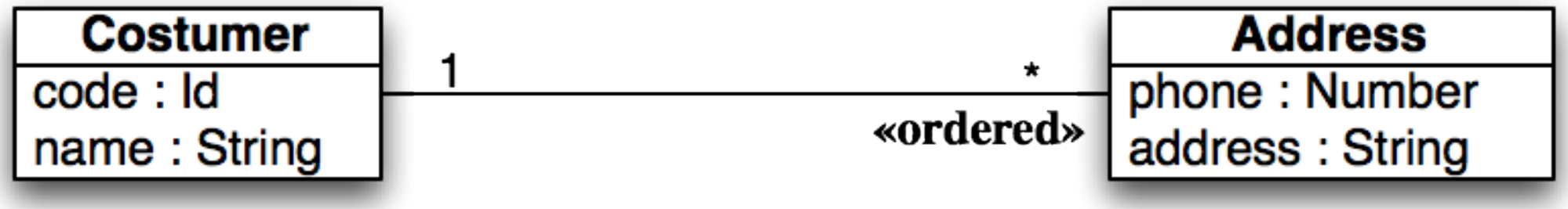
Quantités [Fowler]



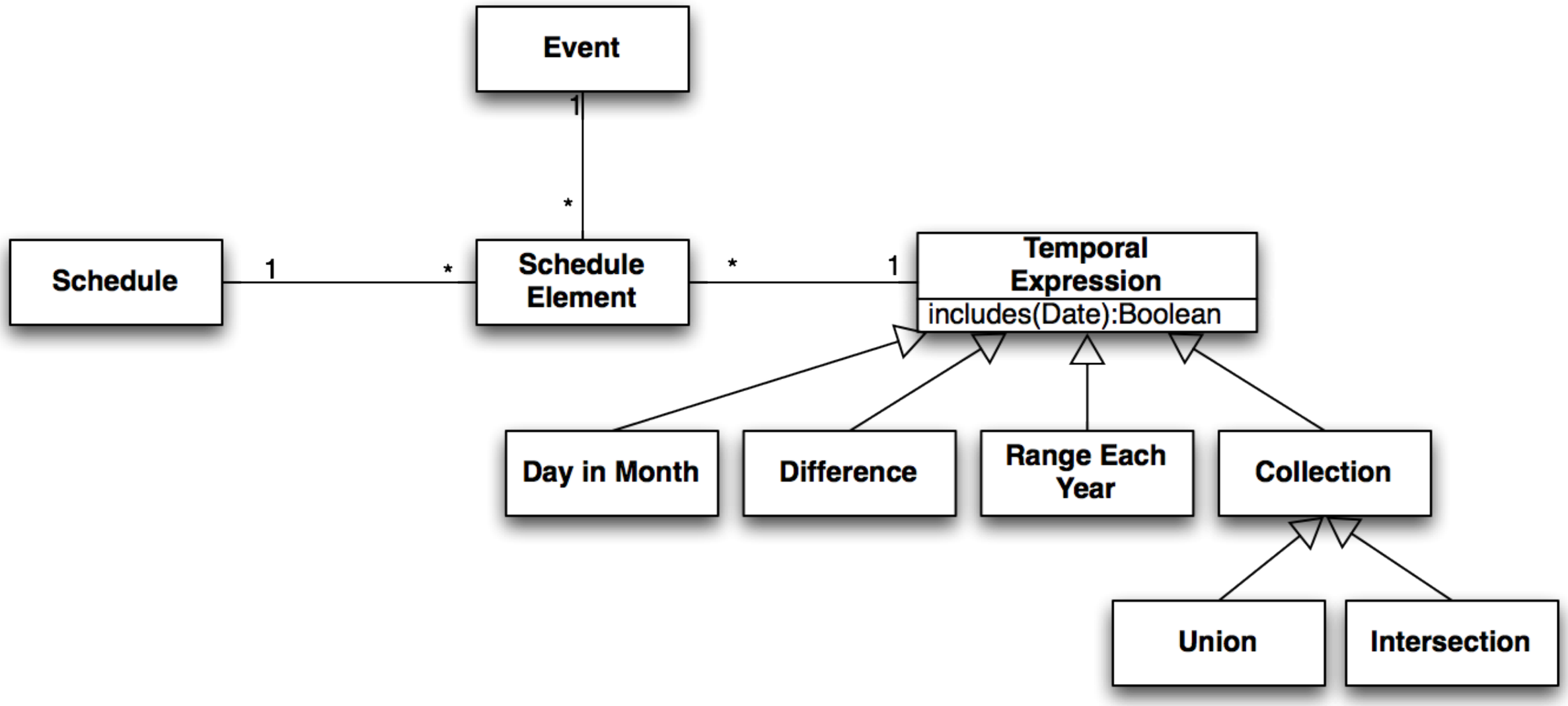
Intervalle [Fowler]



Propriétés temporelles



Événements récurrents [Fowler]



Etude de cas

Etude de cas

- Un serveur de réunions virtuelles
  - Adaptation du concept d'IRC à un contexte de réunions de travail au sein d'une entreprise géographiquement dispersée
- Cette étude de cas va nous servir de fil conducteur pour décrire un processus d'analyse et de conception avec UML

Cahier des charges (1/2)

- Il s'agit de réaliser la partie serveur d'une application client-serveur permettant de faire des réunions virtuelles multimédia sur Internet. L'objectif de cette application est de permettre d'imiter le plus possible le déroulement de réunions de travail classiques. Cependant, dans la première version de ce projet, les interventions des participants se feront en mode mono-média seulement (i.e. échanges en forme textuelle).
- Le serveur devra permettre de planifier et de gérer le déroulement de plusieurs réunions simultanées. Des programmes clients existeront dans l'avenir pour plusieurs plate-formes (Mac, Windows, Unix) afin de permettre à des personnes désirant organiser des réunions virtuelles ou y participer de dialoguer avec le serveur en utilisant un protocole ad hoc développé au dessus de IP.

Cahier des charges (2/2)

- Après s'être connecté au serveur (à l'aide d'un nom de login et d'un mot de passe mémorisé par le système), une personne a la possibilité de planifier des réunions virtuelles (choix d'un nom, définition du sujet, date de début et durée prévue, ordre du jour), de consulter les détails d'organisation d'une réunion, de les modifier (seulement l'organisateur), d'ouvrir et de clôturer une réunion (seulement l'animateur), d'entrer (virtuellement) dans une réunion précédemment ouverte, et d'en sortir. En cours de réunion, un participant peut demander à prendre la parole. Quand elle lui est accordée, il peut entrer le texte d'une intervention qui sera transmise en ``temps-réel'' par le serveur à tous les participants de la réunion.

- Plusieurs sortes de réunions doivent pouvoir être organisables :

  - Réunions standards, avec un organisateur qui se charge de la planification de la réunion et désigne un animateur chargé de choisir les intervenants successifs parmi ceux qui demandent la parole.

  - Réunions privés, qui sont des réunions standards dont l'accès est réservé à un groupe de personnes défini par l'organisateur

  - Réunions démocratiques, qui sont planifiées comme des réunions standards, mais où les intervenants successifs sont choisis automatiquement par le serveur sur la base d'une politique premier demandeur-premier servi.

Démarche de modélisation avec UML

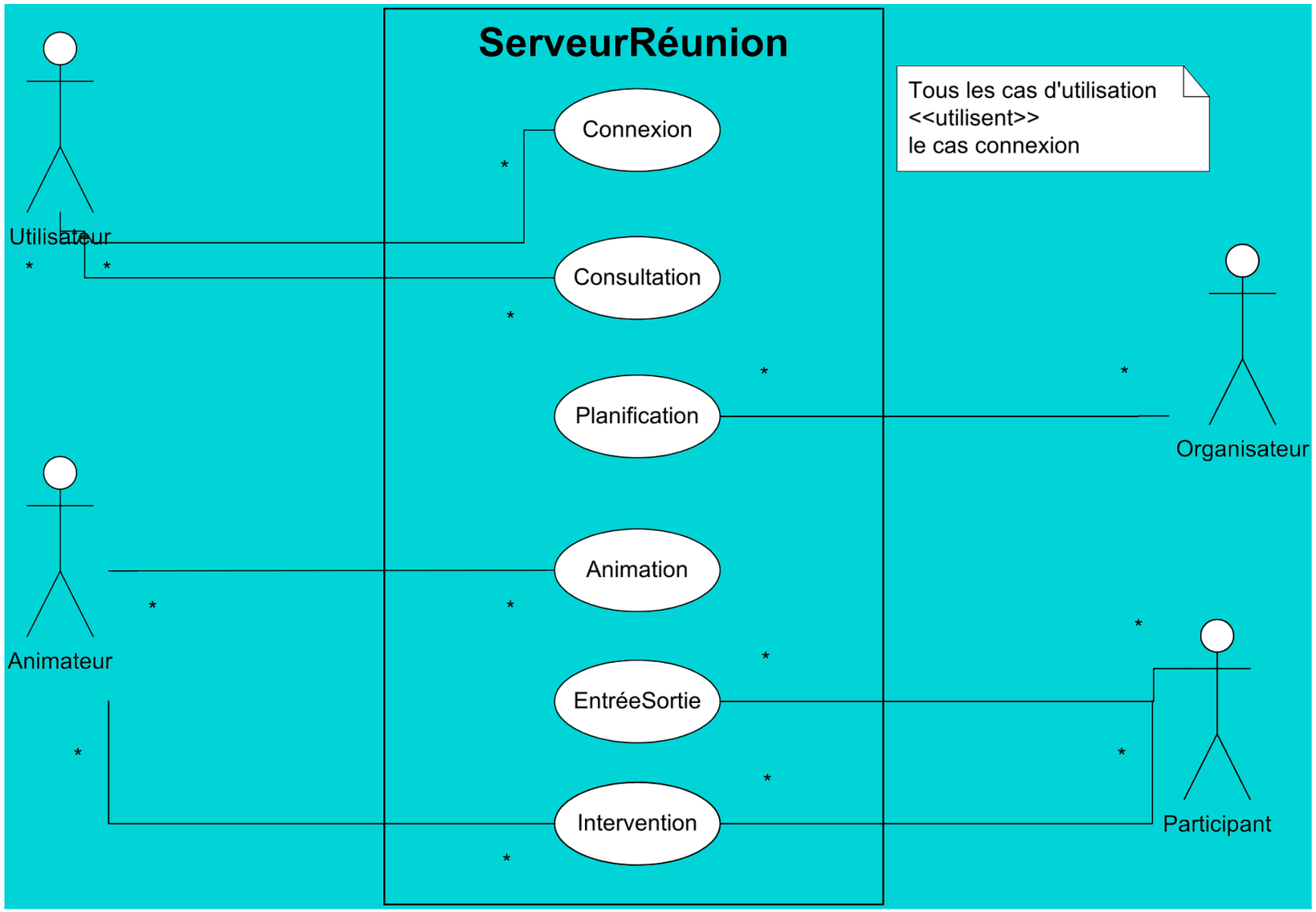
- Construction du diagramme de cas d'utilisations
  - Grande découpe fonctionnelle (10% de l'effort)
- Construction du diagramme de classes
  - à partir des noms des données du problème (30%)
- Construction de diagrammes de séquences et de collaborations
  - instances des cas d'utilisation (25%)
- Généralisation à l'aide de diagrammes d'états-transitions
  - à partir des diag. Séquences (15%)
- Affiner et préciser la solution (20%)

Cas d’utilisation

- Grande découpe fonctionnelle
  - pilote les incréments dans la spirale
  - chaque incrément correspond à la réalisation d’un cas d’utilisation

- 1 diagramme global + texte 10 lignes / cas

Diagramme de cas d'utilisations



Exemple de cas d'utilisation : Planification

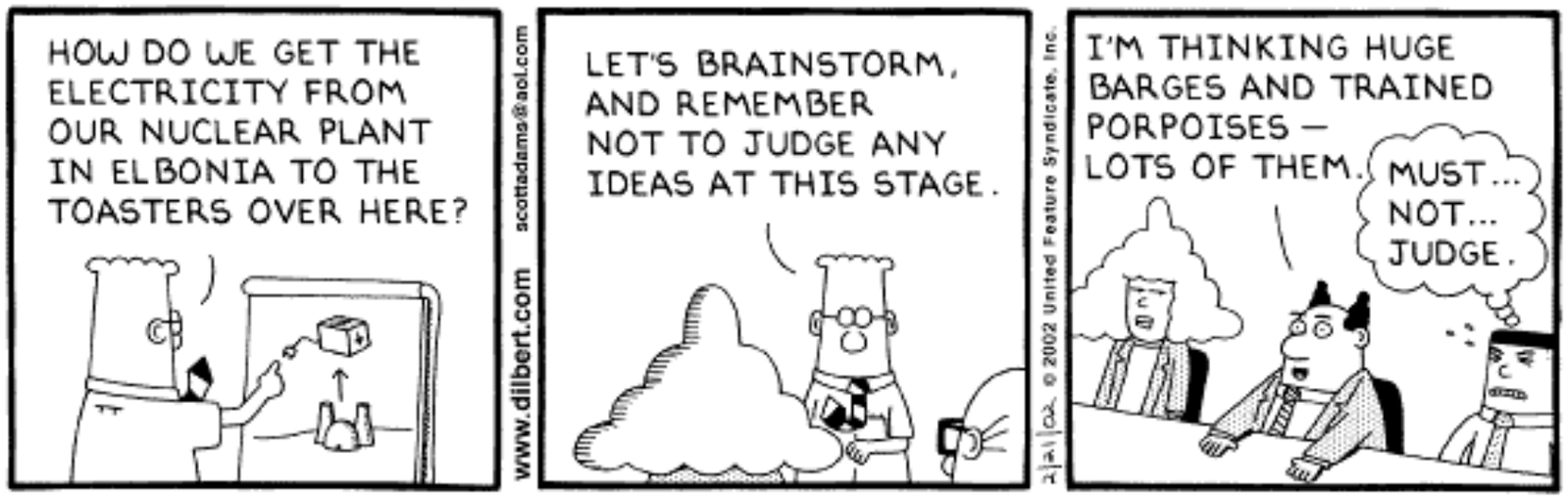
- La **planification** d'une réunion virtuelle est effectuée par une personne jouant le rôle d'organisateur pour cette réunion. Ceci consiste à créer une nouvelle réunion dans le système (ou à la mettre à jour si elle existe déjà) en faisant le choix d'un nom, la définition du sujet, de la date de début et la durée prévue, ainsi que l'ordre du jour.

Processus de construction du diagramme de classes

- Identifier les classes d'objets
  - Garder les bonnes classes
  - constitution du dictionnaire de données
- Identifier les associations.
  - Garder les bonnes associations
- Identifier les attributs.
  - Garder les bons attributs
- Raffiner au moyen de l'héritage
  - Généralisations et raffinages
- Itérer la modélisation
- Grouper les classes en modules

Identification des classes

- A l'aide des noms des données du problème
  - processus de remue-méninges (brainstorming)



- Eliminer les classes :
  - redondantes (noms synonymes)
  - non pertinentes (vs. le modèle)
  - trop vagues (non réifiable facilement)
  - attributs, opérations, rôles de relations
  - constructions liées à l'implantation

Souligner les noms dans le cahier des charges

- Il s'agit de réaliser la partie serveur d'une application client-serveur permettant de faire des réunions virtuelles multimédia sur Internet. L'objectif de cette application est de permettre d'imiter le plus possible le déroulement de réunions de travail classiques. Cependant, dans la première version de ce projet, les interventions des participants se feront en mode mono-média seulement (i.e. échanges en forme textuelle).

- Le serveur devra permettre de planifier et de gérer le déroulement de plusieurs réunions simultanées. Des programmes clients existeront dans l'avenir pour plusieurs plate-formes (Mac, Windows, Unix) afin de permettre à des personnes désirant organiser des réunions virtuelles ou y participer de dialoguer avec le serveur en utilisant un protocole ad hoc développé au dessus de IP.

Classes potentielles

Serveur	Mise en oeuvre
Application	Redondant serveur
Réunion	OK
Internet	Mise en oeuvre
Objectif	Non pertinent
Déroulement	Action
Version	Mise en oeuvre
Projet	Non pertinent
Intervention	OK (?)
Participant	Rôle relation Personne-Réunion
Mode	Mise en oeuvre
Programme	Mise en oeuvre
Plate-formes	Mise en oeuvre
Personne	OK
Protocole	Mise en oeuvre

Identification des relations entre classes

- Recherche des phrases verbales
- Eliminer les relations :
  - entre classes éliminées
  - non pertinentes ou liées à l'implantation
  - qui sont en fait des actions
  - pouvant être dérivées d'autres relations
  - Raffiner la sémantique des relations
  - ajouter les rôles
  - qualifier les relations (sélecteur)
  - spécifier la multiplicité

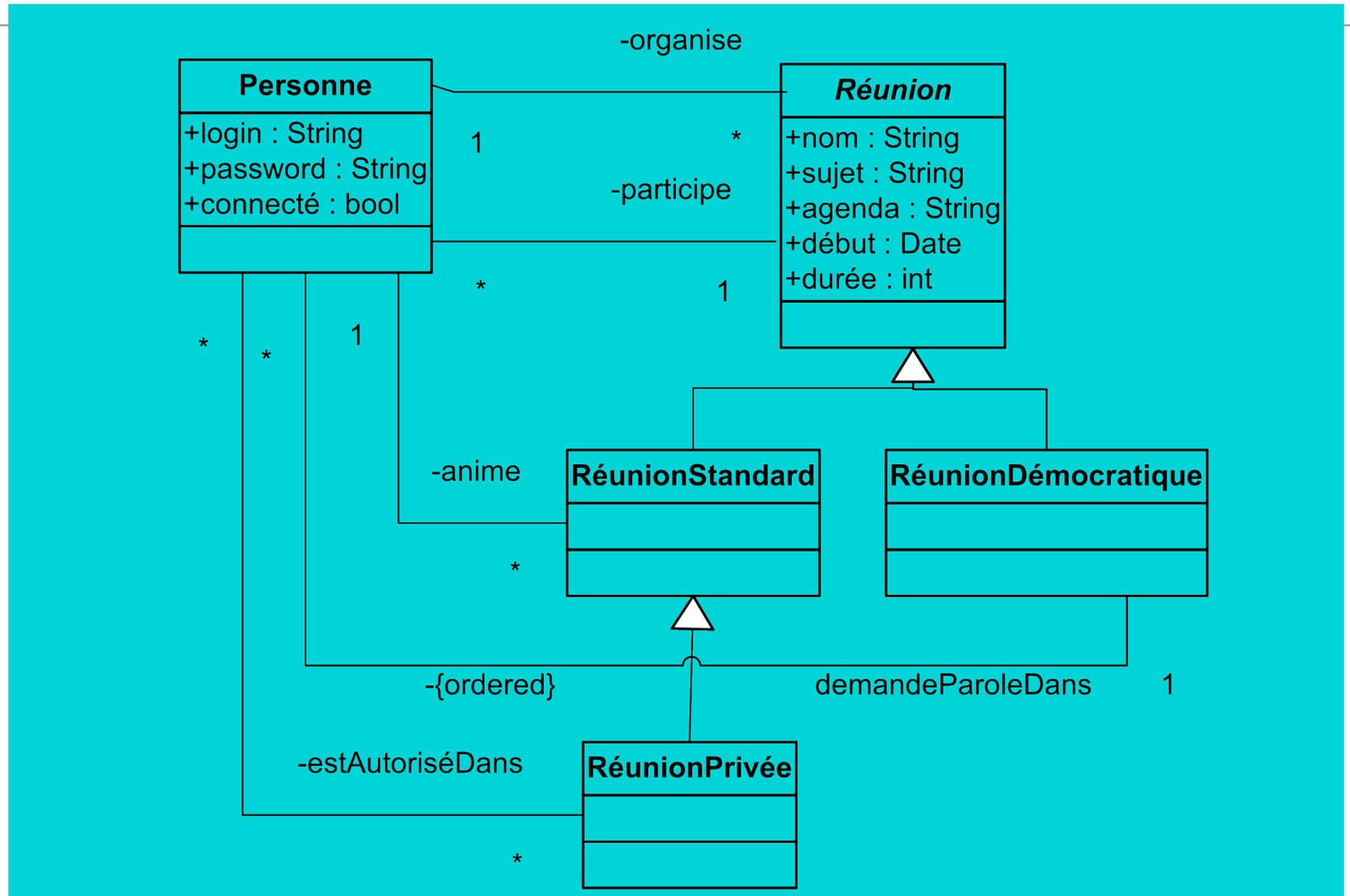
Identification des attributs

- Propriétés d'objets individuels
  - à rechercher à l'aide des adjectifs (couleur, poids...) ou propositions substantives du problème
- Eliminer les attributs non nécessaires ou incorrects
  - s'ils sont en fait des objets
  - sélecteurs de relations
  - identificateurs (clef de BD)
  - attributs de relations
  - valeurs internes ou détails d'implantation

Raffiner au moyen de l'héritage

- Généralisation à l'aide de super-classes
  - Recherche de classes avec des attributs, relations ou opérations similaires
- Spécialisation à l'aide de sous-classes
  - Différentes variantes d'une même classe
    - Réunion privée, démocratique …
  - sous-classe ou attribut pour distinguer ?

Diagramme de classes d'analyse



Itérer la modélisation

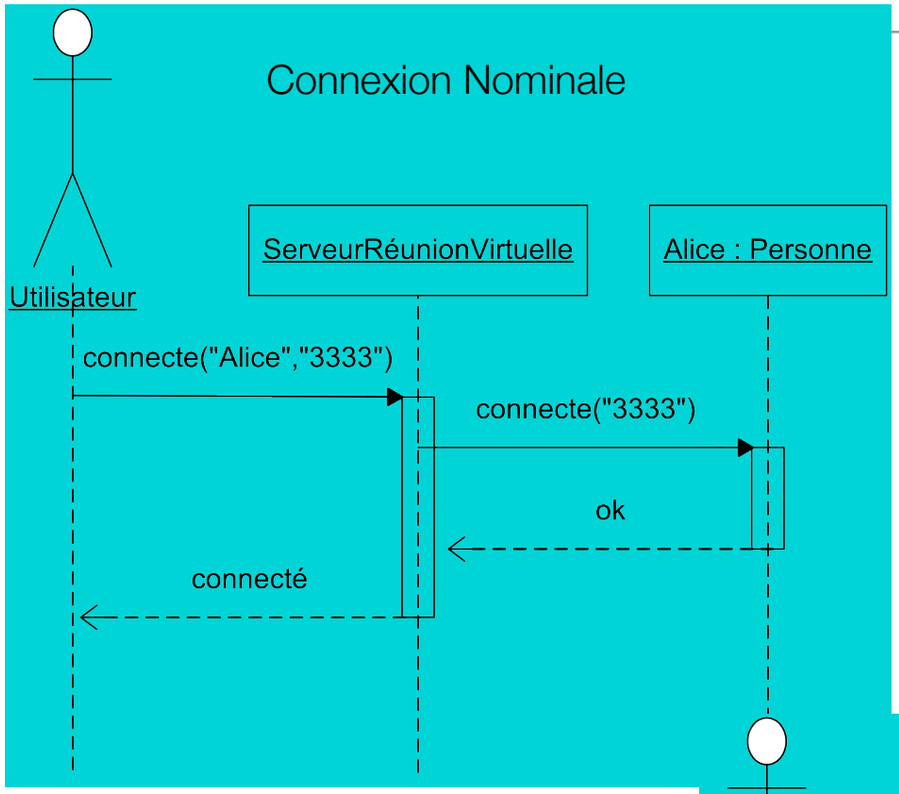
- Classes manquantes ou en trop
  - asymétries : ajout de classes par analogies
  - scinder les classes disparates en classes plus élémentaires
  - A quoi sert une classe si pas d'attributs ou d'opérations?
- Relations manquantes, en trop ou mal placées
  - si aucune opération ne traverse une relation...
- Attributs
  - accéder à un objet par un attribut -> relation qualifiée
- Il faut parfois attendre d'avoir fait les autres vues pour pouvoir itérer

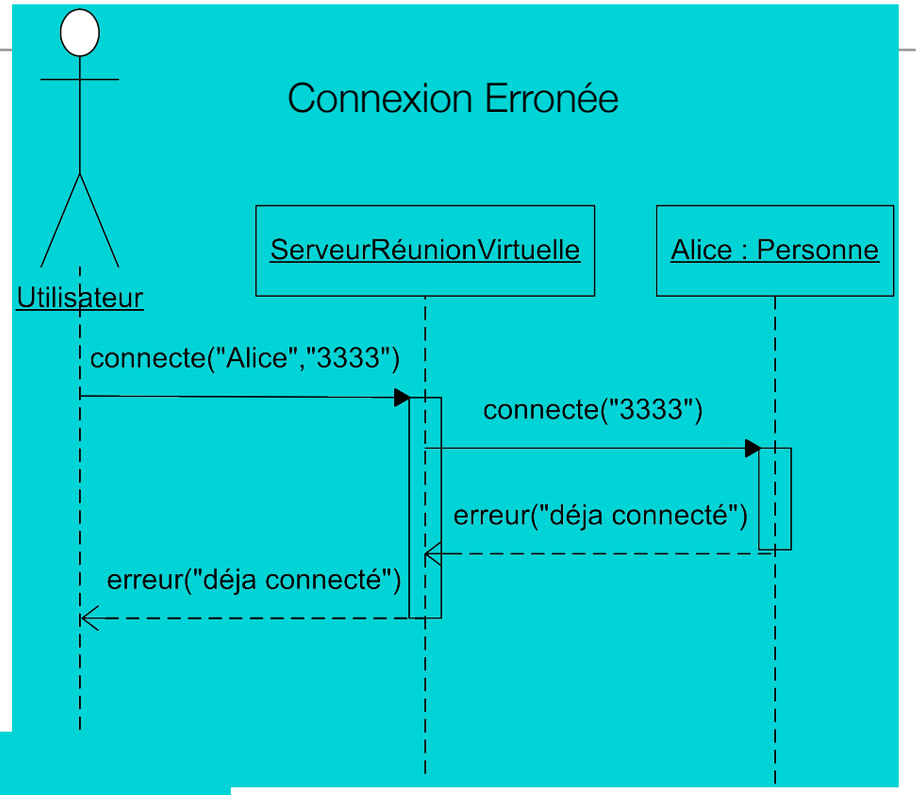
Diagrammes dynamiques

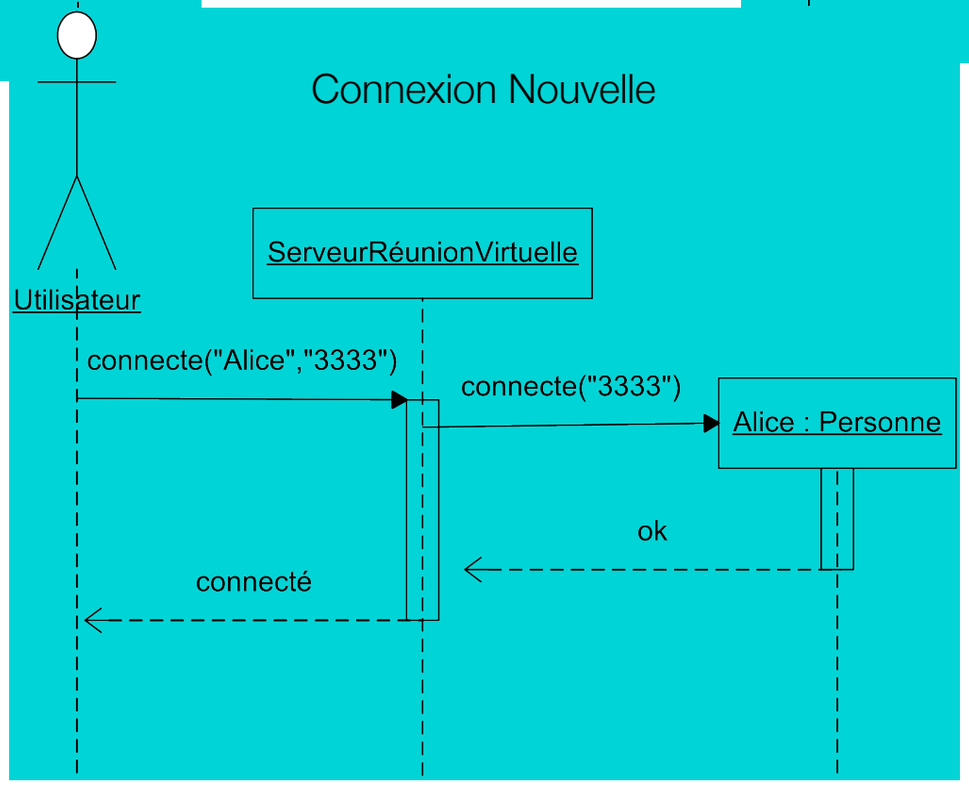
- Pour chaque cas d ’utilisation
  - un scénario nominal (diagramme de séquence)
    - tout se passe bien
  - quelques scénarios exceptionnels
    - montrent des variations sur le scénario optimal

- Alternativement (Catalysis)
  - description par pre/post de l ’état du système avant/après chaque occurrence d ’événement

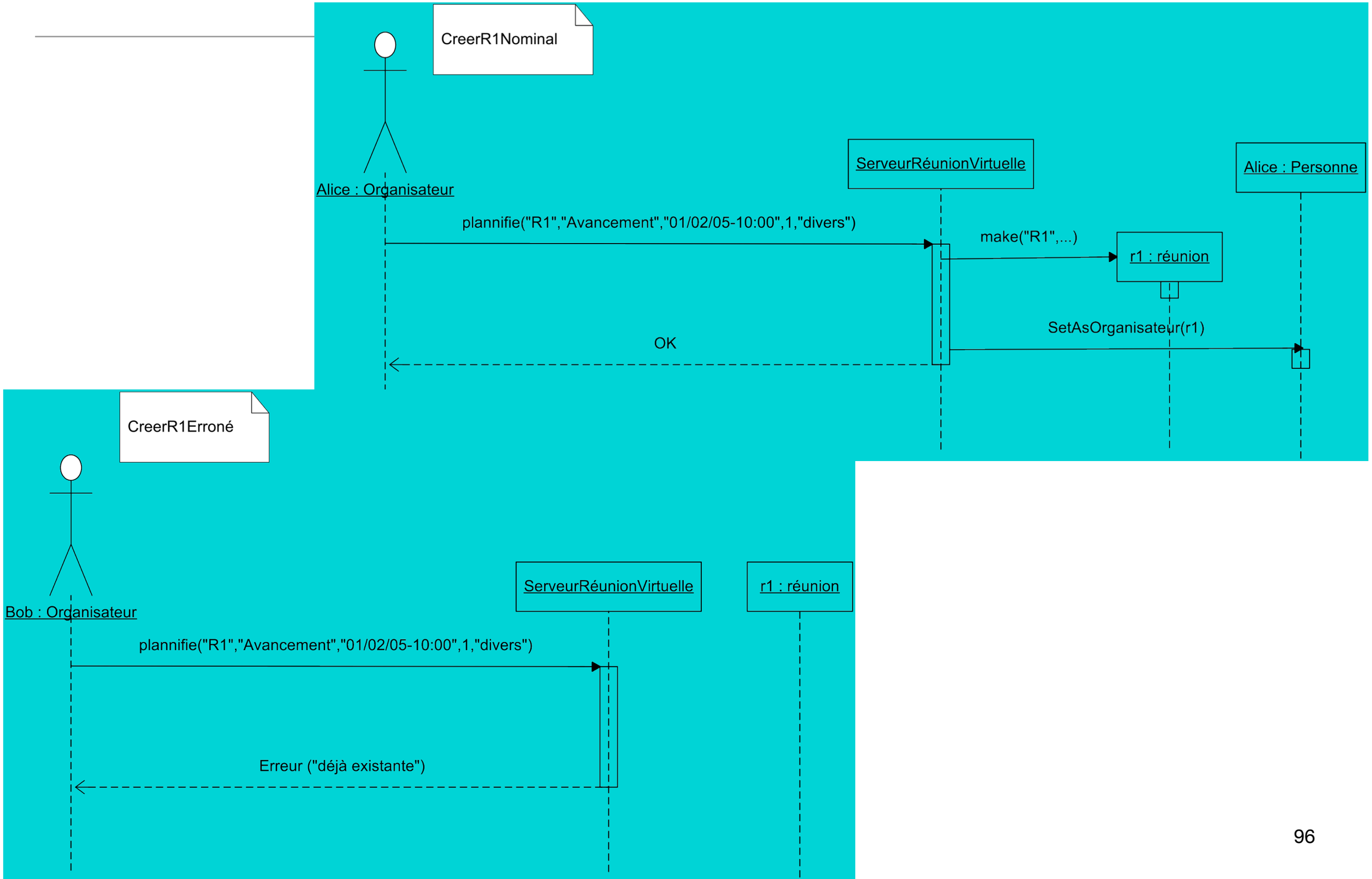
Exemples de scénarios :
Cas d'utilisation Connexion



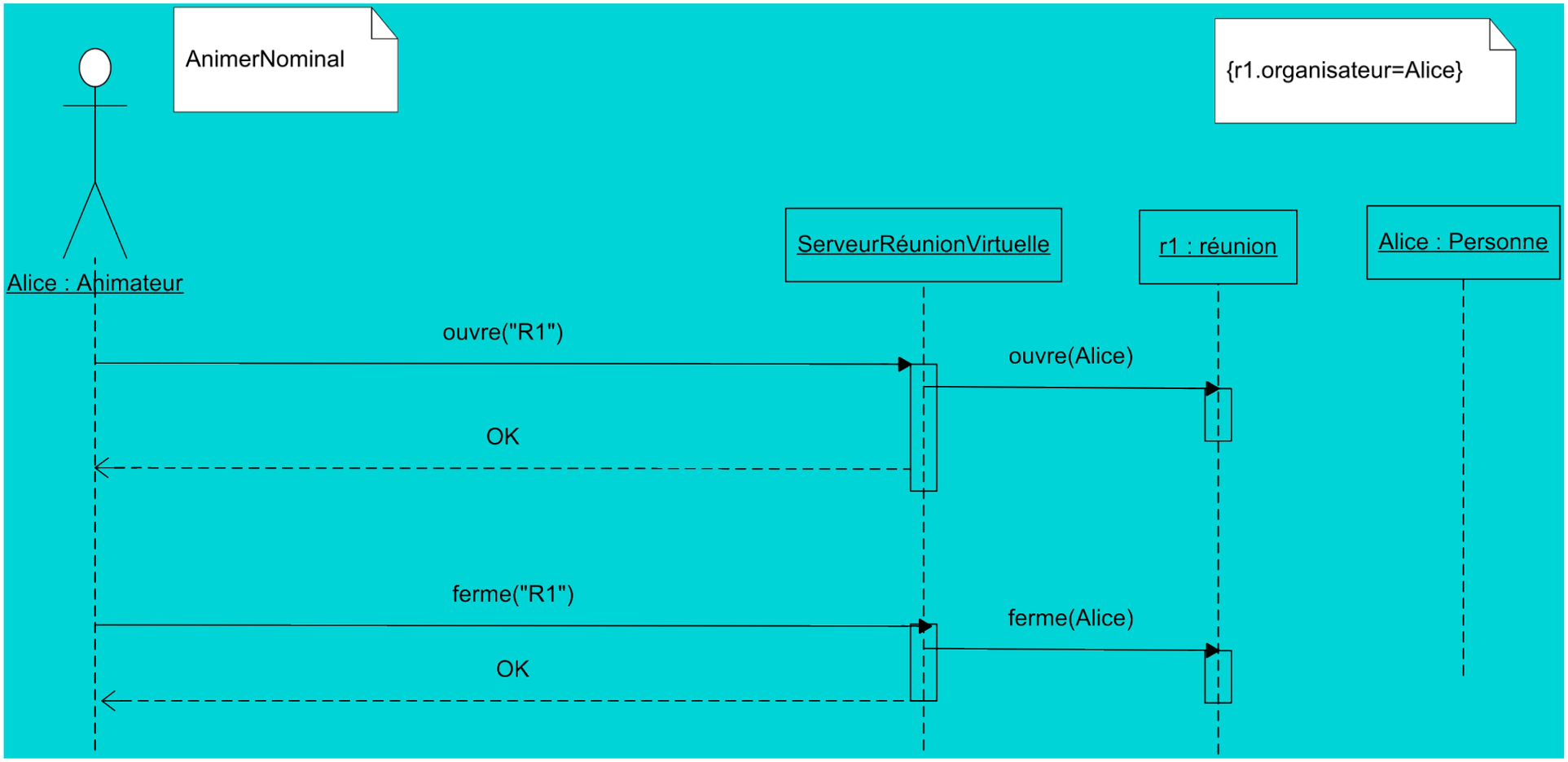




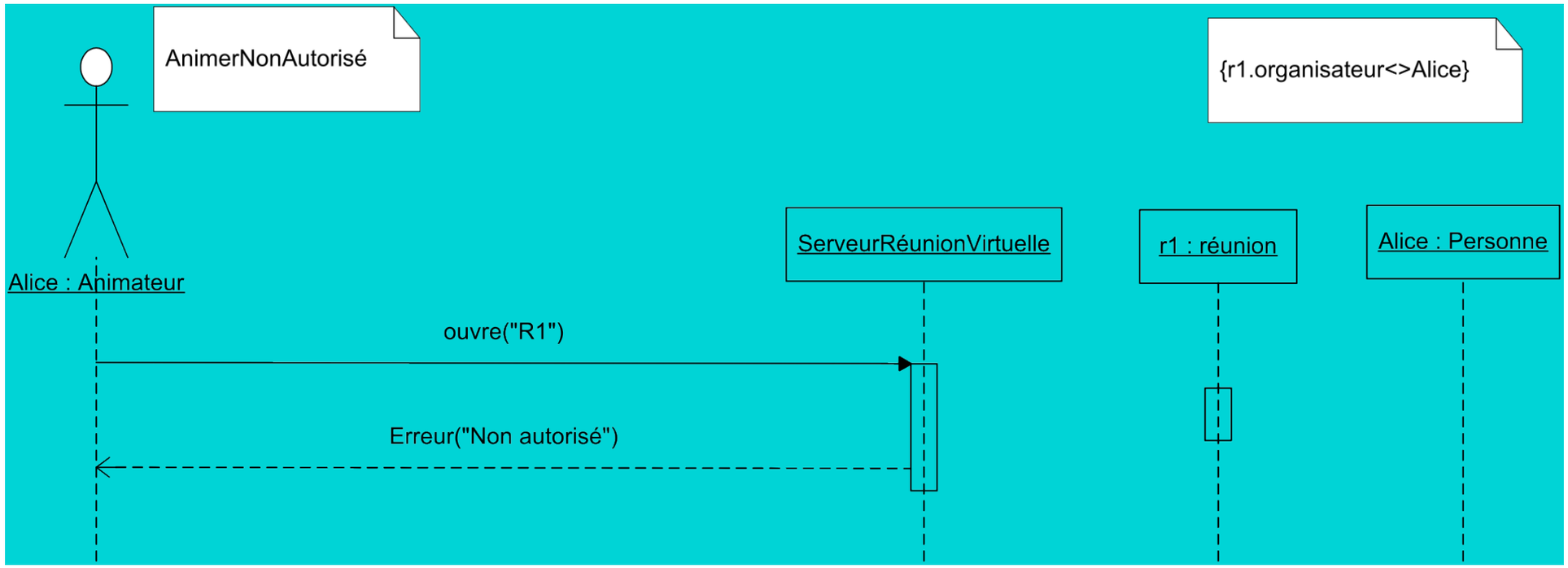
Exemples de scénarios :
Cas d'utilisation Planification



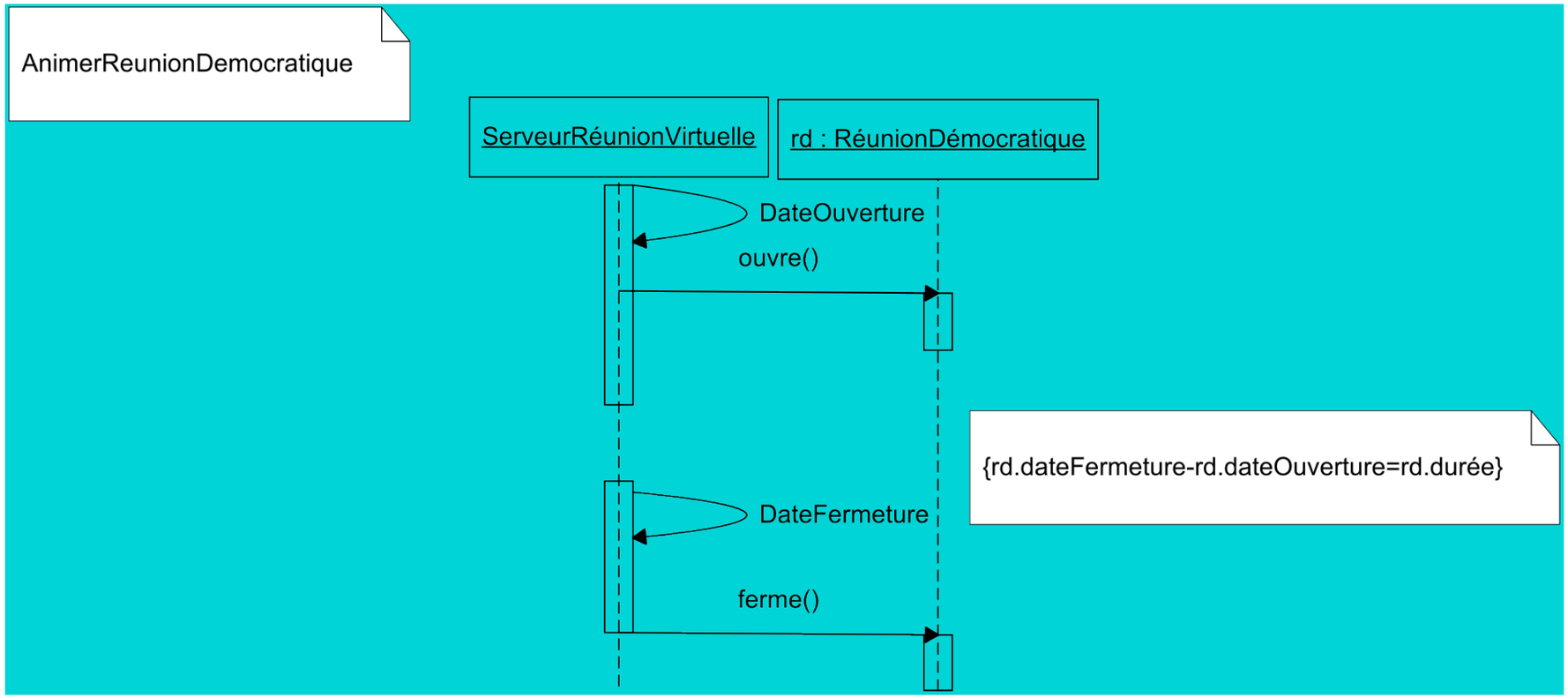
Exemples de scénarios :
Cas d'utilisation Animation 1/3



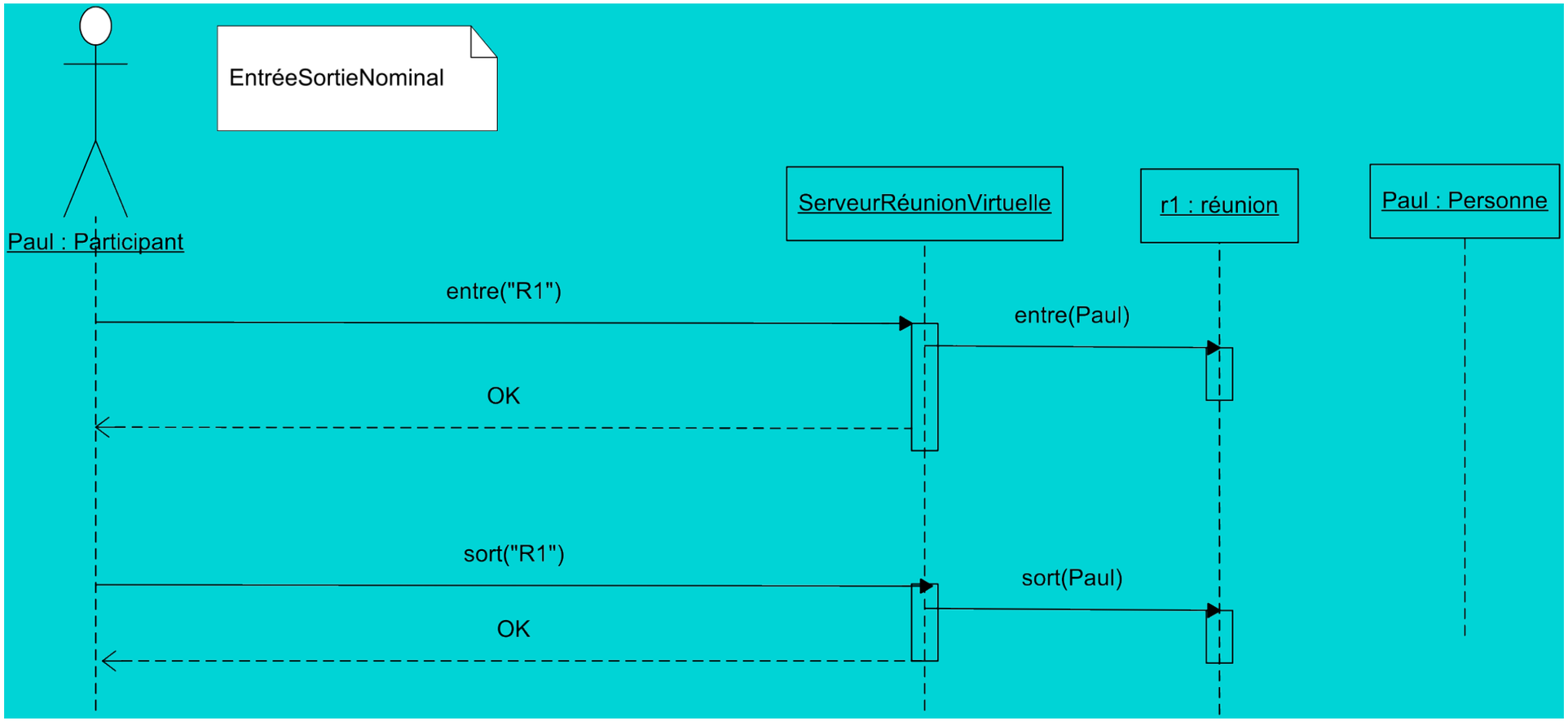
Exemples de scénarios :
Cas d'utilisation Animation 2/3



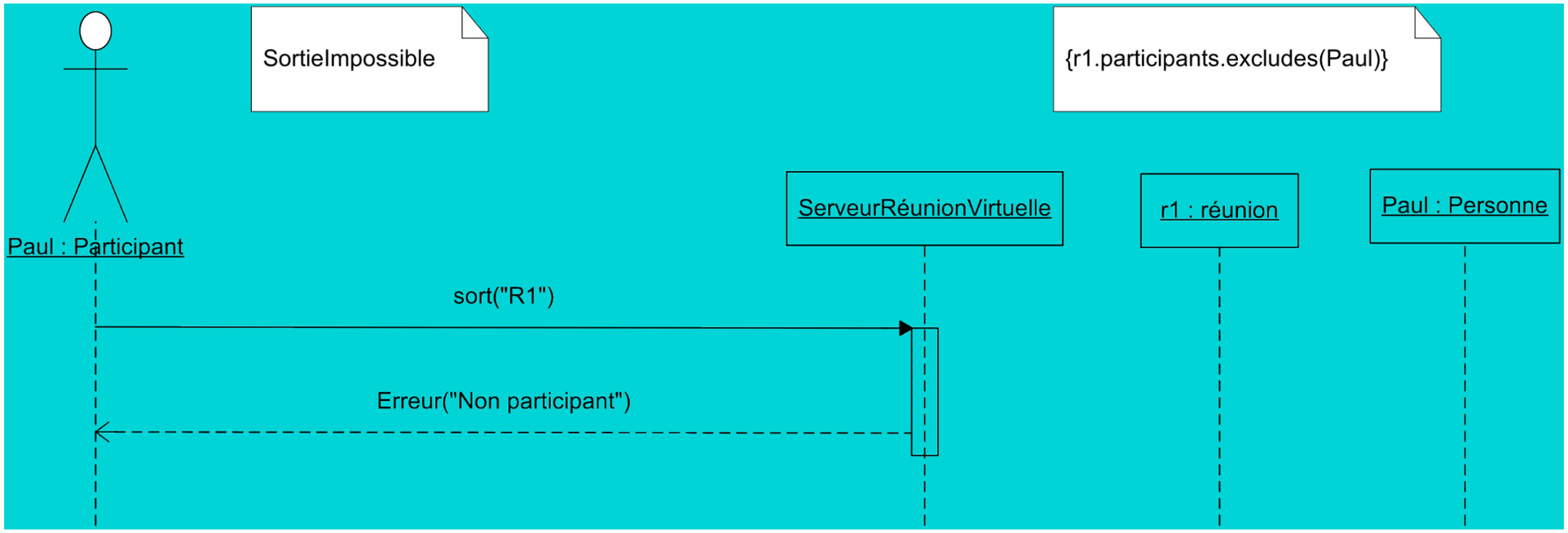
Exemples de scénarios :
Cas d'utilisation Animation 3/3



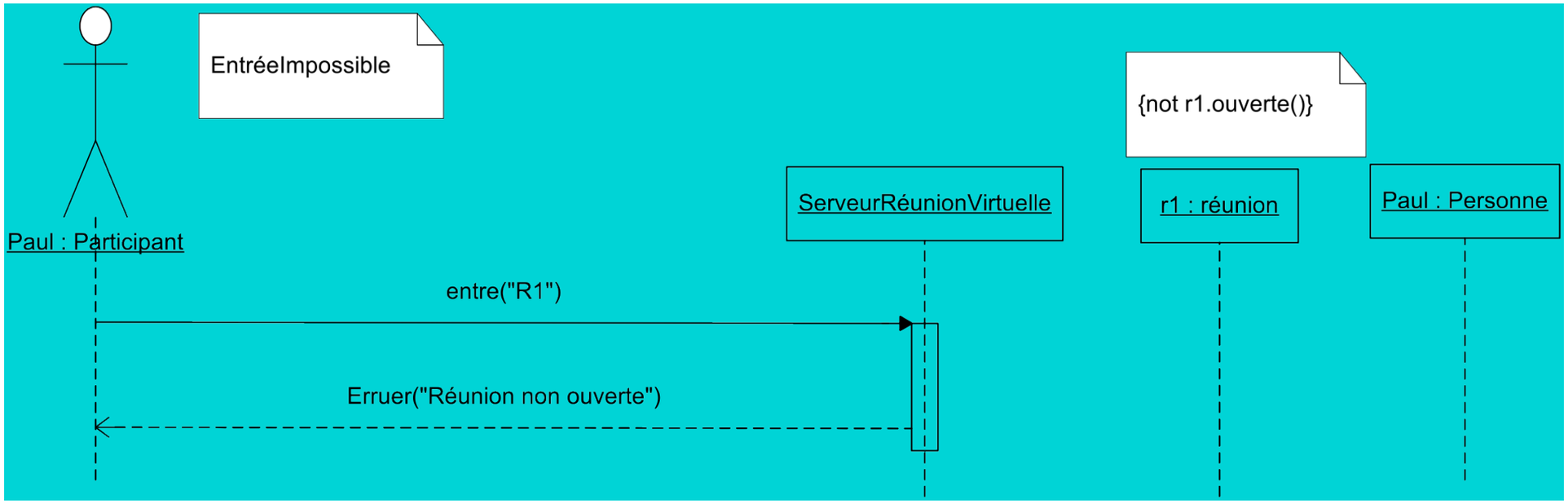
Exemples de scénarios :
Cas d'utilisation EntréeSortie 1/4



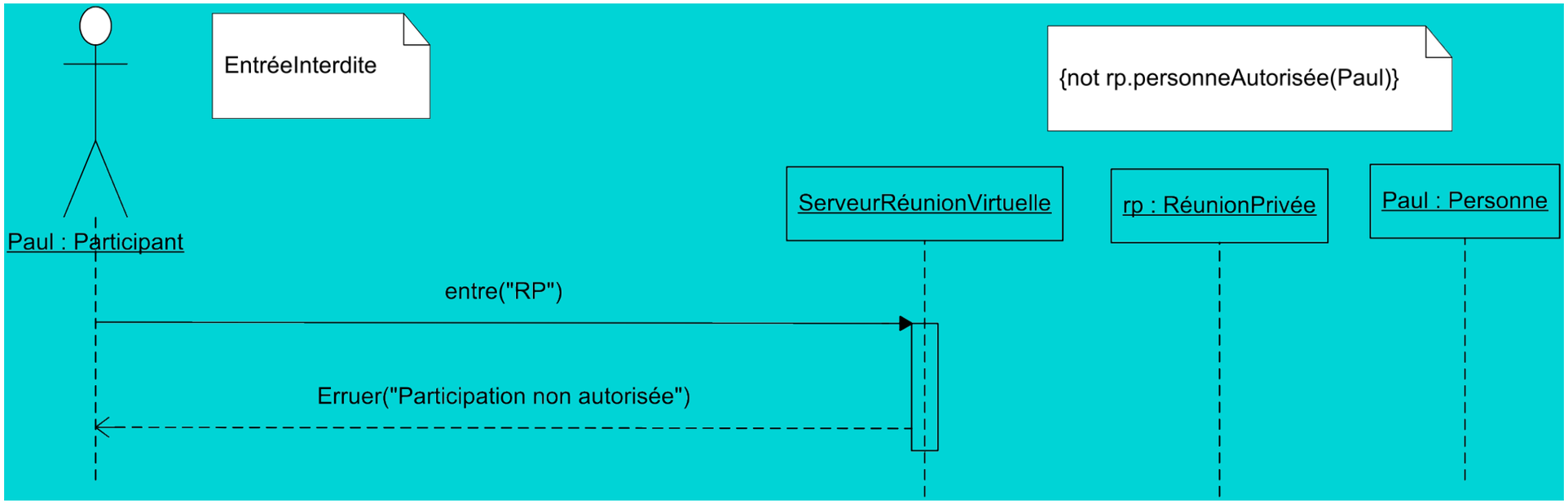
Exemples de scénarios :
Cas d'utilisation EntréeSortie 2/4



Exemples de scénarios :
Cas d'utilisation EntréeSortie 3/4



Exemples de scénarios :
Cas d'utilisation EntréeSortie 4/4



Construction des diagrammes d’états

- Généralisation pour une classe donnée de l’ensemble des scénarios qui mettent en jeu ses instances
  - en suivant les transition de l’automate, on doit pouvoir retrouver tous les scénarios

Diagramme d'états : Réunion

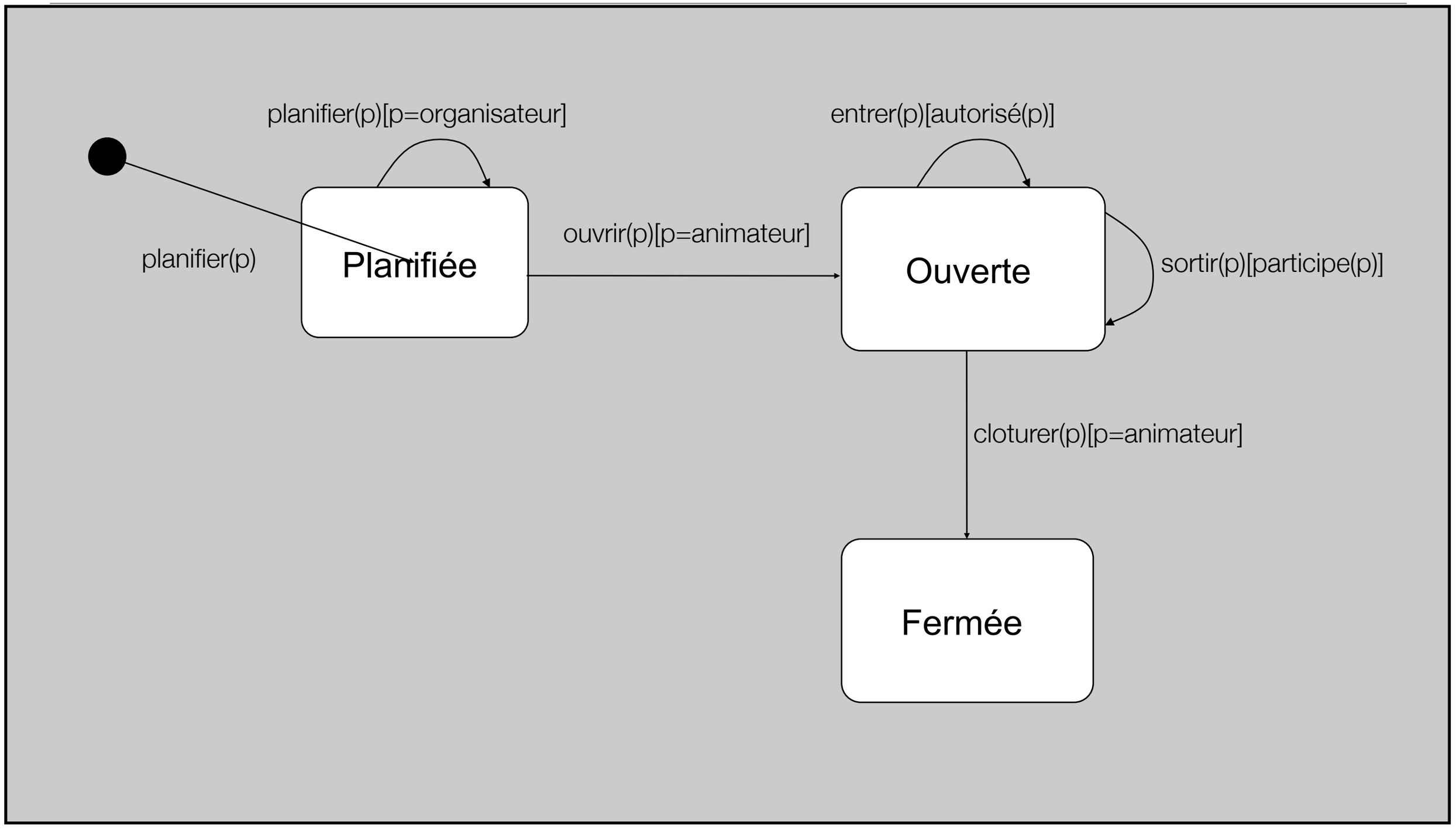


Diagramme d'états :
Réunion démocratique

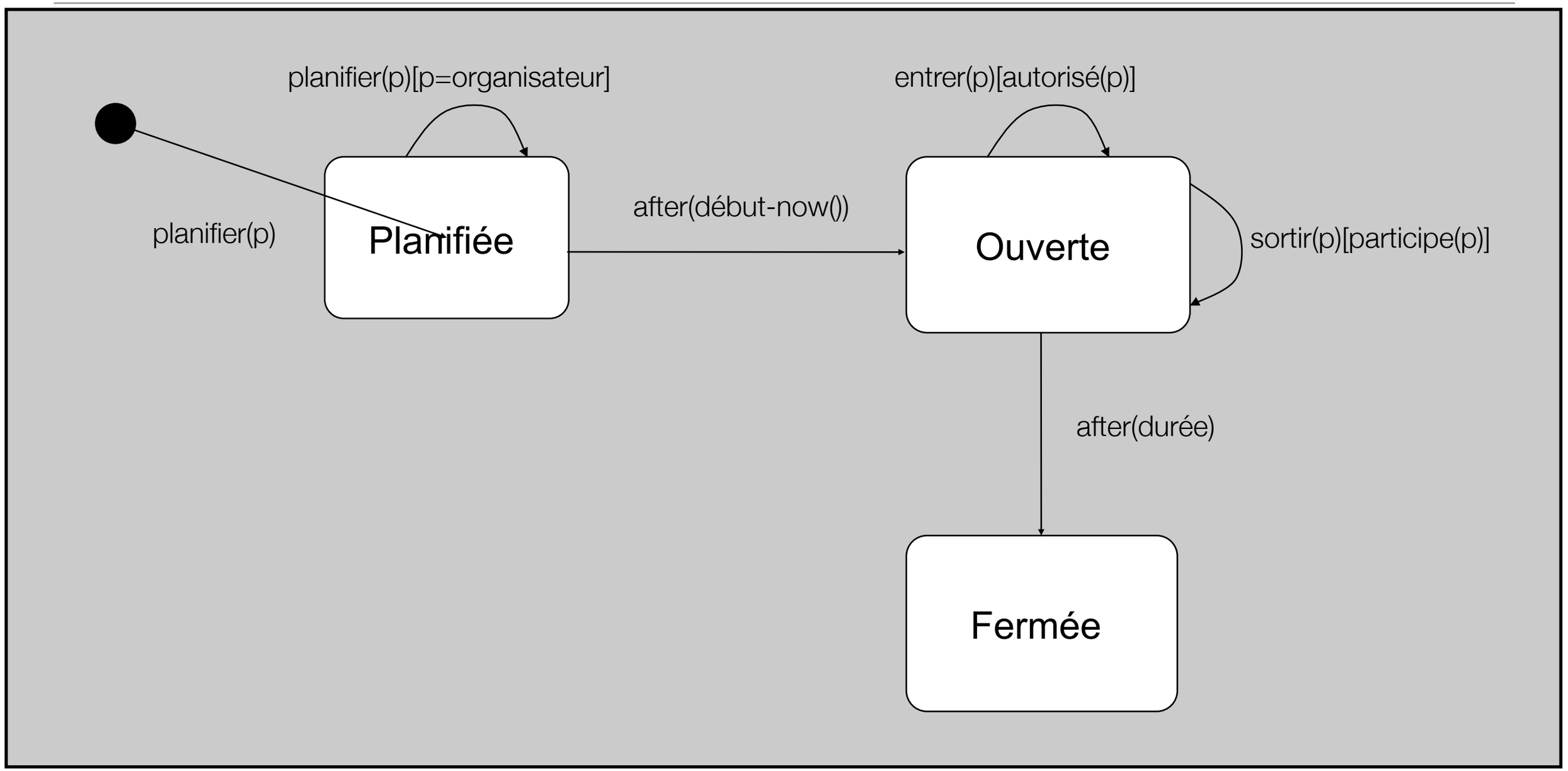
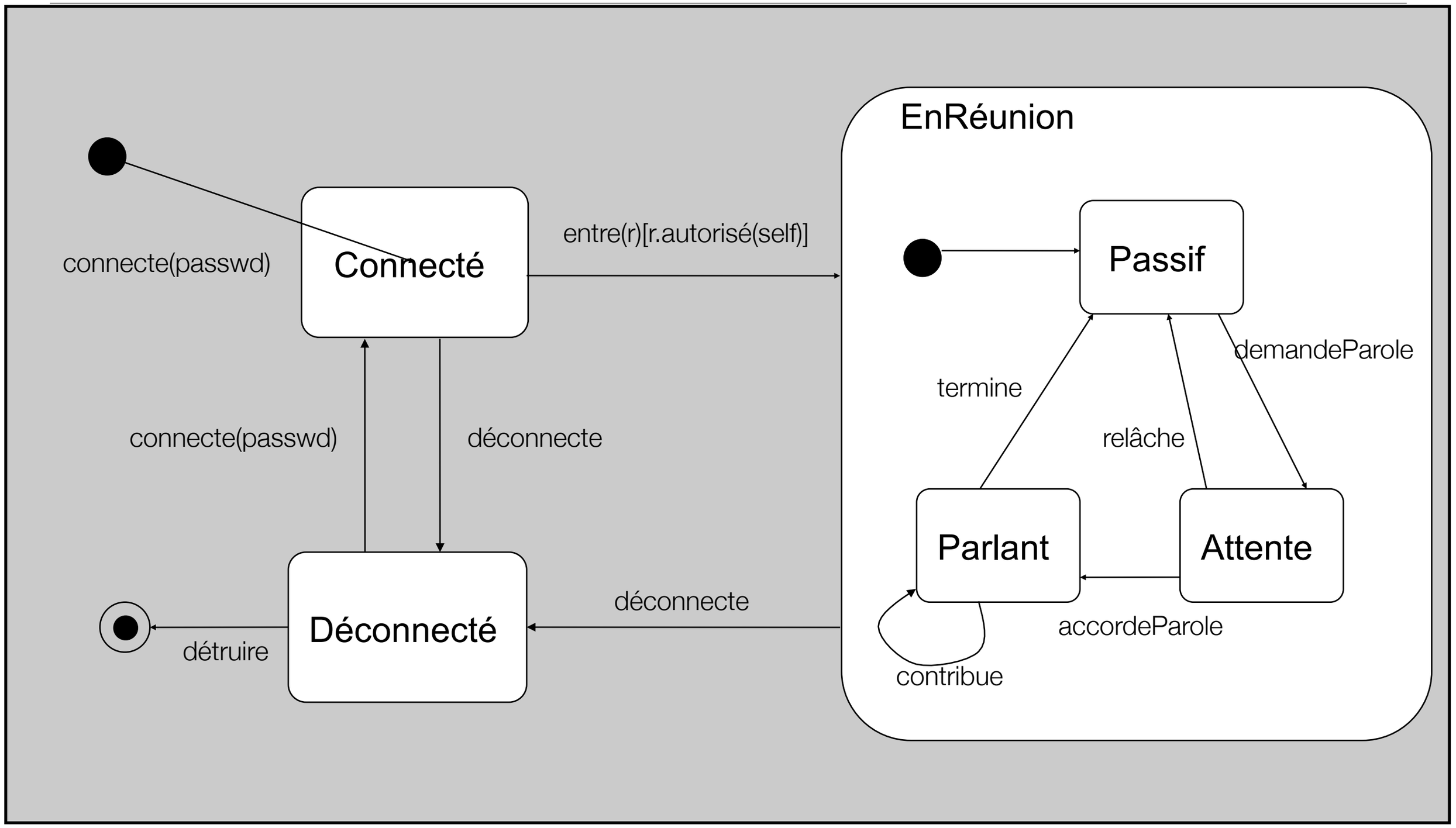


Diagramme d'états : Personne



Derniers conseils

Conseils pratiques

- Réfléchir au problème avant de commencer
- Éviter toute complication nuisible
- Se dégager de l'implémentation : raisonner objets, classes, messages, relations, attributs, opérations
- Ne pas s'inquiéter si les possibilités de la notation ne sont pas toutes exploitées

Conseils pratiques (suite)

- Suivre une approche incrémentale:
    - Itérer
    - Confronter ses modèles aux autres
- Savoir s'arrêter avant d’atteindre la perfection:
    - prise en compte qualité (niveau de précision), coûts, délais.
    - asservissement au processus de développement.

Diagramme de classes

- Soigner le nommage, insister sur le nommage des relations et des rôles.
- Éviter les relations ternaires, quaternaires (trop complexes).
- Oublier les visibilités.
- Utiliser les types de données (Datatypes).

Cas d’utilisation

- Environ 6 cas, nommage correct des acteurs (noms) et des cas (actions)
- Complétude vis à vis du Cahier des charges

Critères de qualité d'un bon diagramme de classes

- Nommage correct des classes (noms)
- Définitions présentes dans un dictionnaire
- Nommage de toutes les associations, présence des cardinalités
- Attributs seulement de types simples
- Non duplication attributs/relations (utilisation héritage)

Critères de qualité d'un diagramme de séquence

- Toujours attachés à un cas d'utilisation
- Pour chaque cas d'utilisation:
  - présence d'un scénario nominal
  - quelques scénarios exceptionnels.
- Chaque message (reçu par un objet) est défini par une opération dans la classe correspondante du diagramme statique.

Critères de qualité d'un bon diagramme d'états

- Chaque automate est attaché à une classe.
- Chaque attribut/opération utilisée sur l'automate doit être défini dans la classe englobante.
- En suivant les transition de l'automate, on doit pouvoir retrouver tous les scénarios.

«Things must be as simple as possible, but no simpler».
[A. Einstein]

«Un bon modèle n’est pas un modèle où l’on ne peut plus rien ajouter, mais un modèle où on ne peut plus rien enlever.»
[A. de St-Exupéry]

Un modèle incomplet n’est pas un modèle simple.

Bibliographie

- http://www.drdobbs.com/how-to-avoid-use-case-pitfalls/184414560
- Writing Effective Use Cases. Alistair Cockburn.
- UML Distilled: A Brief Guide to the Standard Object Modeling Language. Martin Fowler.