
Licence Sciences, Technologies, Santé
Mention SPI, parcours Informatique

Introduction au Génie Logiciel
L3 / 175EN002

# C4 – Besoins et contraintes non-fonctionnels
# Séparation traitement/données
# Synthèse sur analyse
# Notion de modèle

Thierry Lemeunier
thierry.lemeunier@univ-lemans.fr
www-lium.univ-lemans.fr/~lemeunie

# Plan du cours

- Études de cas
  - Système de contrôle d'accès à un bâtiment – Partie III
  - Système de vente – Partie II
- Besoins et contraintes non fonctionnels
  - Contraintes techniques
  - Besoins d'IHM
- Étude de cas
  - Système de contrôle d'accès à un bâtiment – Partie IV
- Traitements et données
  - Séparation entre traitements et données
  - Stockage des données
- Analyse
  - Démarche d'analyse
  - Outils pour l'analyse
- Modèle et modélisation
  - Notion
  - Exemples

# Système de contrôle d'accès (1/2)

- Lisons le document Annexe C4-1
- Quels sont les exigences du client ?
  - Utilisation de lecteurs de badges existant

  ➔Etudier les caractéristiques des lecteurs
    - Caractéristiques logiciels : mémorisation de 4000 badges avec chacun 8 plages horaires, opérations de contrôle intégrées avec  mémorisation des 100 derniers événements, différents types de messages, lecteur esclave, etc.
    - Caractéristiques matériels : de la mémoire vive, une gâche électrique avec une commande électronique, une adresse réseau, un port série, une trame réseau, etc.

  ➔Discuter des conditions de conception du système de contrôle selon ces caractéristiques

# Système de contrôle d'accès (2/2)

- Comment le système va-t-il être utilisé ?
  - Interfaces logicielles
    - Pour le superviseur : interfaces graphiques d'accès à ses fonctionnalités métier
    - Pour le gardien : idem

    ➔Proposer des maquettes de ces interfaces
  - Interfaces matérielles
    - Pour le porteur de badge : utilise les lecteurs de badges
    - Liaison par réseau série pour les lecteurs de badges

# Système de vente (1/2)

- Lisons le document Annexe C4-2
- Quels sont les exigences du client ?
  - Au niveau des fonctionnalités :
    - Journalisation et traitement des erreurs
    - Intégration de la possibilité de définir des règles des gestion
    - Traitement des paiements par débit et par crédit en différé (la nuit)
    - Prendre en compte différents identificateurs d'article (UPC, EAN, SKU)
  - Au niveau du fonctionnement :
    - Texte de l'afficheur lisible à un mètre et utiliser des signaux sonores plutôt que visuels
    - Réaliser la vente même en cas de perte des connections aux acteurs non humains
    - Demander et obtenir la réponse du système d'autorisation bancaire en moins d'une minute dans 90% des cas
  - Au niveau de la conception/implémentation
    - Prendre en charge différentes règles de gestion (adaptabilité)
    - Configurabilité élevée de l'architecture logicielle
    - Utiliser des composants libres si possibles
  - Etc.

# Système de vente (2/2)

- Comment le système va-t-il être utilisé ?
  - Interfaces logicielles
    - Pour l'administrateur système : interfaces graphiques d'accès à ses fonctionnalités métier (gestion droit d'accès des caissiers)
    - Pour le caissier : écran tactile avec interfaces graphiques
    - Pour les acteurs non humains (système de comptabilité, système des ventes, système ressources humaines, système des stocks, système de calcul de taxe, système autorisation bancaire) : protocoles de communication (logique d'échange des infos)
  - Interfaces matérielles
    - Liaison par réseau Ethernet pour les acteurs non humains ?
    - Liaison USB avec lecteur de carte bancaire ?
    - Liaison USB avec lecteur de code à barre ?
    - Liaison bus I2C avec l'imprimante de reçu ?
    - Liaison bus I2C avec le grand afficheur à cristaux liquide ?
    - Liaison bus USB avec le clavier alphanumérique ?

# Besoins et contraintes non fonctionnels

- En plus des besoins fonctionnels liés à chaque acteur, un système peut nécessiter d'autres besoins et être contraint
- Besoins d'IHM :
  - Ensemble des interfaces homme machine pour les acteurs humains
  - Demandés par le client ou être une conséquence des besoins fonctionnels
  - Exemple :
    - Acteur : le caissier
    - Besoin en IHM : écran tactile
- Contraintes techniques :
  - Ensemble des nécessités non directement liées aux besoins fonctionnels
  - Demandés par le client ou être une conséquence de l'existant ou provenir d'un acteur non humain
  - Exemples :
    - Prendre en compte les caractéristiques des lecteurs de badges existant
    - Prendre en compte les différents types d'identification d'article
- L'analyse doit recenser l'ensemble des besoins et des contraintes non fonctionnels

# Système de contrôle (1/4)

- Comment sont pris en charge les besoins fonctionnels des différents acteurs par le système ?
  - Fonctionnalités du superviseur :
    - **BF1** : Besoin de s'identifier

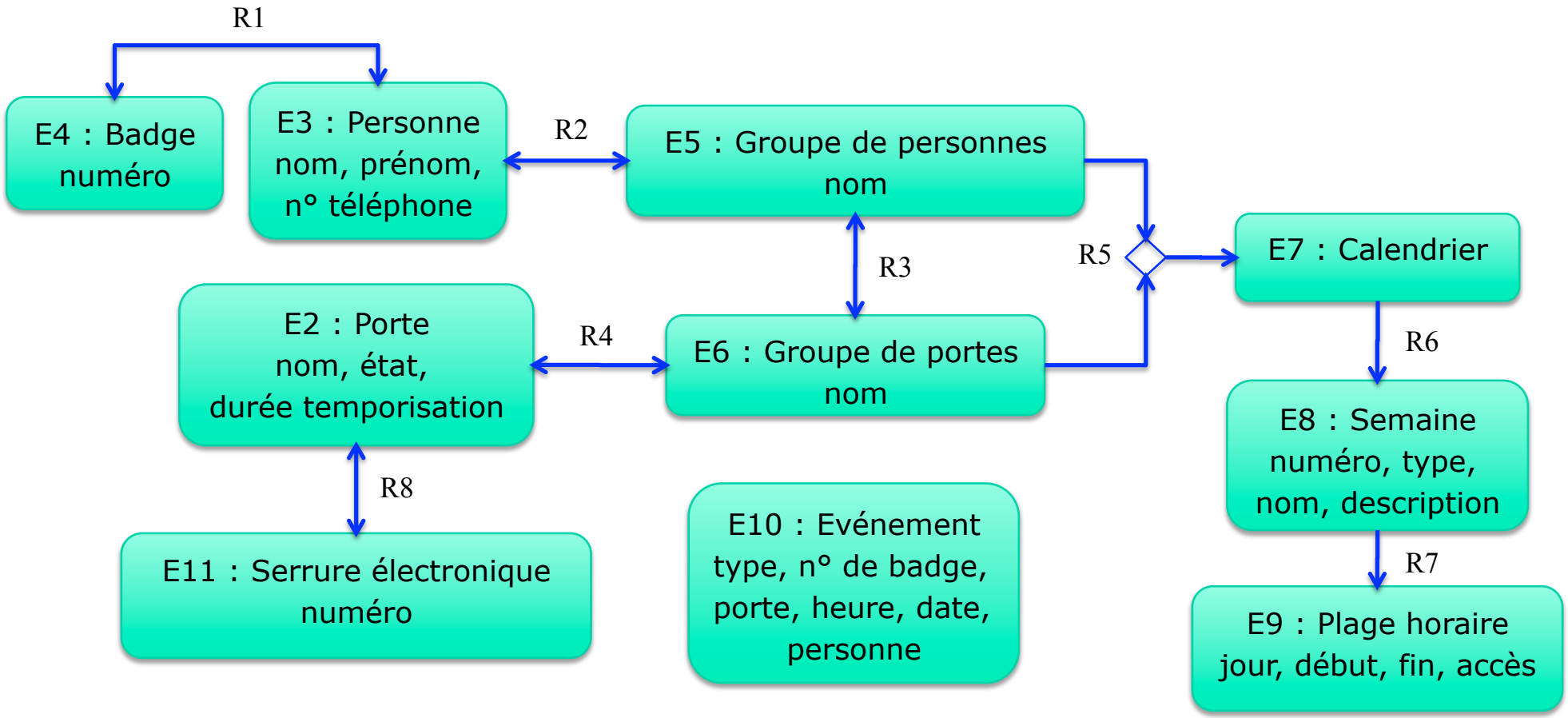
      ➔**O1** : opération traitée sur le poste du superviseur qui consiste à comparer le login et le mot de passe saisis aux login et mot de passe stockés. Le login permettra d'accéder aux opérations du superviseur (**O2** à **O12**)
    - **BF2** : Besoin de modifier les informations relatives à une porte

      ➔**O2** : opération traitée sur le poste du superviseur qui consiste à accéder à la liste des portes connues puis à modifier les informations stockées (nom, état, temporisation) ou à créer une nouvelle porte
    - Etc.
  - Fonctionnalités du gardien :
    - **BF13 à BF17** : opérations **O13** à **O17** sur le poste du gardien
  - Fonctionnalités du porteur de badge :
    - **BF18 :** Besoin de demander l'accès à une porte

      ➔**O18** : opérations intégrées des lecteurs de badges programmées à distance par le superviseur

# Système de contrôle (2/4)

- Quelles sont les informations manipulées ?
  - Une version informatisée  des occurrences des concepts métiers
  - Les relations doivent être respectées, par exemple : si une porte est supprimée il faut supprimée la serrure électronique lié



# Système de contrôle (3/4)

- Comment et où sont stockées les informations ?
  - Dans l'ordinateur du gardien ou dans celui du superviseur ?

  ➔Un stockage centralisé commun accessible depuis les deux ordinateurs
  - Dans chaque lecteur de badge

  ➔Ces informations sont « volatiles » par rapport aux informations centralisées (non perdues en cas de coupure de courant)

# Système de contrôle (4/4)

- Quel est le volume des informations dans le pire des cas ?

| | Volume unitaire | Taille unitaire | Volume total |
|---|---|---|---|
| **Lecture de badge**<br><br>total ≈ 200 ko<br>de mémoire vive<br>par lecteur | 4000 badges | 2 o | 7,9 ko |
| | 100 événements | 4 o | 400 o |
| | 8 plages-horaires par badge | 48 o | 188 ko |
| | avec 3 sous-plages par plage-horaire | 6 o | 18 o |
| **Stockage centralisé**<br><br>total ≈ 6,6 Go<br>de taille disque | 500 badges | 2 o | 1 ko |
| | 500 personnes | 80 o | 40 ko |
| | 100 portes | 25 o | 2,5 ko |
| | 100 lecteurs de badges | 1 o | 100 o |
| | 500 groupes de personnes | 30 o | 14,7 ko |
| | 100 groupes de portes | 30 o | 3 ko |
| | 50 000 événements/jour gardés 1 semaine | 10 o | 3,33 Go |
| | 500 000 calendriers | 6,8 ko | 3,24 Go |
| | 52 semaines par calendrier | 50 o + 84 o | 6,8 ko |
| | avec 21 plage-horaires par semaine | 4 o | 84 o |

# Traitements et données

- Séparation entre traitements et données
  - Les opérations = les **traitements**
  - Les informations manipulées par les opérations = **données**
    - Créations / suppressions / modifications des données :
      - Effectués par le système
      - Demandés par les acteurs (humains et non humains)
      - Grâce aux interfaces logicielles (IHM pour les acteurs humains)
- Stockage des données
  - Le type de stockage est choisi en fonction :
    - du volume des données
    - et de la « volatilité » des données
  - Deux types principaux :
    - Stockage par fichier pour les données très stables et/ou peu nombreuses
    - Stockage par SGBR (Système de Gestion de Base de données Relationnel) pour les données volatiles et/ou nombreuses
- Remarque :
  - La question du stockage des données est plutôt traitée en conception et non en analyse
  - Une contrainte technique peut porter sur le SGBDR utilisé

# Analyse et démarche d'analyse (1/3)

- L'analyste suit une **démarche rigoureuse reproductible**
- Démarche :
  1. Définir le but général du système
  2. Définir la frontière du système
     1. Identifier les acteurs et leurs rôles métiers
     2. Identifier les processus métier et les besoins fonctionnels
     3. Identifier les parties matérielles et logicielles
  3. Définir chaque processus métier de chaque acteur
     1. Décrire les processus métiers en termes de concepts métiers
     2. Schéma synthétique des concepts métiers
  4. Décrire les besoins et contraintes non fonctionnels
     1. Décrire les besoins en interfaces logicielles
     2. Décrire les besoins techniques
- Les deux dimensions de l'analyse :
  - Dimension statique : définir et identifier les éléments du domaine
    - Acteurs, concepts métiers et leurs relations
  - Dimension dynamique : décrire et expliquer les interactions et les modifications des éléments du domaine
    - Les processus métier en termes de manipulation de concepts métiers

# Analyse et démarche d'analyse (2/3)

- Outils pour l'analyse :
  - Outils informels
    - Documents rédigés en langage naturel
    - Schémas du type patatoïde non normalisés
    - Cycles d'écriture/lecture des documents

    1. Écriture version n par X
    2. Lecture critique version n par Y avec X ≠ Y
    3. Écriture version n+1 par X
    4. Etc.

    - Réunion de travail des analystes avec des experts du domaine et/ou des utilisateurs finals
      - Discussion des fonctionnalités
      - Discussion des maquettes d'IHM, etc.
    - Audit chez le client
      - Des analystes/ergonomes observent les acteurs humains en milieu écologique
      - Des analystes/ergonomes font des interviews semi dirigés

# Analyse et démarche d'analyse (3/3)

- Outils pour l'analyse (suite) :
  - Outils semi formels
    - Langage normalisé basé sur une sémantique « symbolique »
    - Transformation automatique entre langages (notamment vers des langages de programmation)
    - Exemple : UML (*Unified Modeling Language*)
  - Outils formels
    - Langage normalisé basé sur les mathématiques et la logique
    - Etablir des preuves du respect des spécifications formelles
    - Transformations automatiques possibles
    - Exemples : langage B, OCL (*Object Constraint Language*)
- Les langages semi formels ou formels permettent de définir des **modèles**
- La spécification par **modélisation** est la manière la plus sûre et efficace de faire de l'analyse

# Modèle et modélisation (1/3)

- Notion de modèle
  - Qu'est-ce que c'est ?
    - Un modèle est une représentation idéale (abstraite et simplifiée) d'un phénomène ou d'objets réels ou imaginaires
    - ➔ C'est une formalisation : c'est l'association d'un énoncé formel à une « réalité empirique »
  - A quoi ça sert ?
    - A étudier, à expliquer, à comprendre, à prédire, à communiquer sur une réalité idéalisée
    - ➔ Un objet O est un modèle d'une réalité R si O permet des répondre aux questions que l'on se pose sur R
  - Pourquoi tant de modèles ?
    - Il y a théoriquement autant de modèles que de modélisateurs
    - On modélise uniquement ce qui est nécessaire
    - ➔ C'est le point de vue, la compréhension d'une personne

# Modèle et modélisation (2/3)

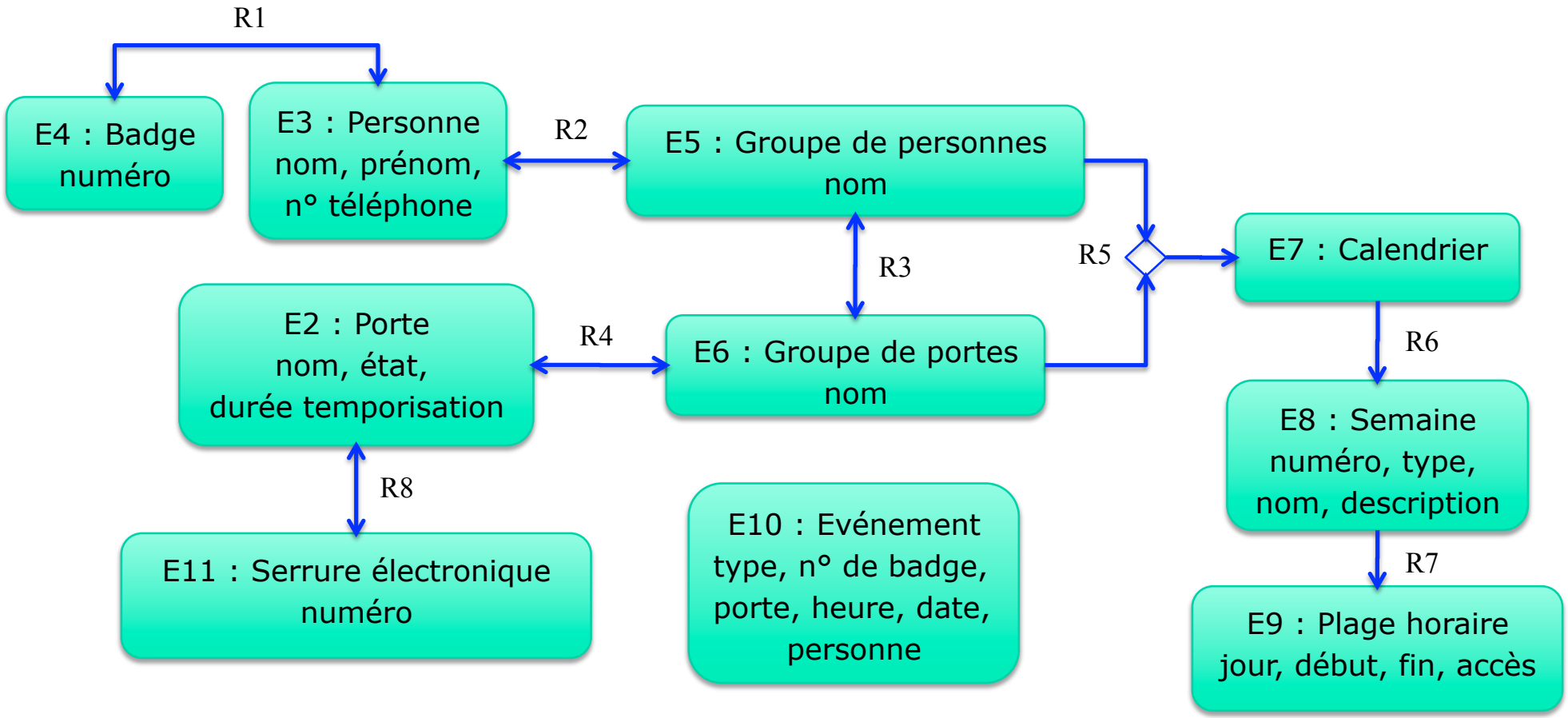
- La modélisation informatique
  - Elle a pour but…
    - de décrire : analyse du domaine métier par exemple
    - de prévoir : conception du fonctionnement des logiciels par exemple
    - et de communiquer : entre membres de l'équipe par exemple
  - La modélisation n'est pas limitée à l'analyse
    - Il faut faire des modèles d'analyse. Par exemples :
      - Modèle (statique) des concepts du domaine et de leurs relations
      - Modèle (dynamique) des interactions des acteurs avec le système
      - Modèle (dynamique) des processus métiers des acteurs
    - Il faut faire des modèles de conception. Par exemples :
      - Modèle (statique) de l'architecture logicielle
      - Modèle (statique) des données
      - Modèle (dynamique) de collaboration des éléments de l'architecture
      - Modèle (dynamique) de programme : les algorithmes !
    - Il faut utiliser/affiner des modèles de conception en programmation et en maintenance
    - Il faut faire des modèles de déploiement
    - Etc.

# Modèle et modélisation (3/3)

- Avantages
  - Un modèle est synthétique
  - Un modèle (semi formel) est moins ambiguë que le langage naturel ou pas du tout (modèle formel)
  - Un modèle est plus universel qu'une langue naturelle
- Inconvénients
  - Il n'y a pas de modèle unique : qu'est-ce qu'un bon modèle ?
  - La complétude d'un modèle n'est pas mesurable
  - La conformité d'un modèle n'est pas toujours appréciable
  - Les modèles semi formels sont « interprétables »

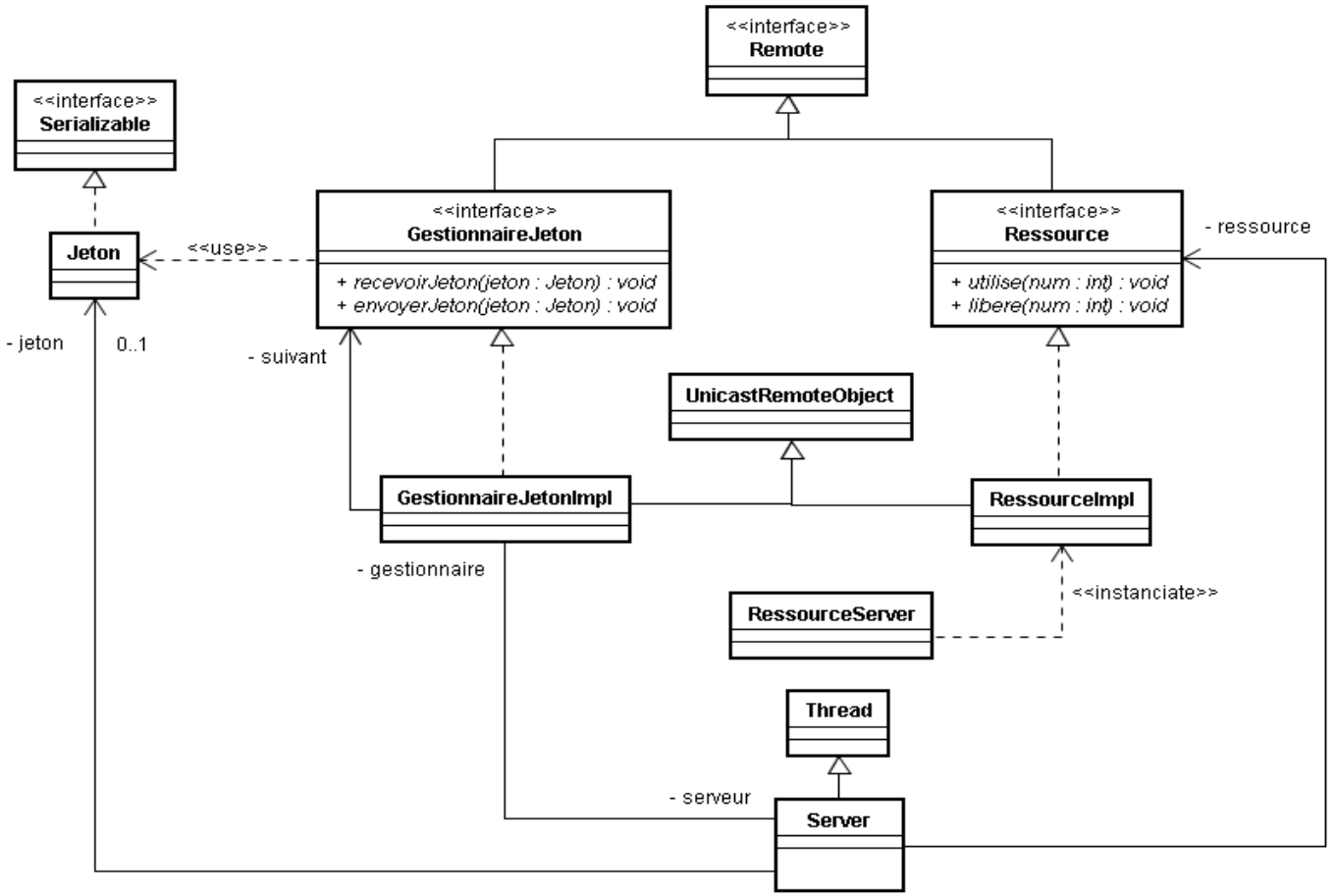
# Exemples de modèles (1/5)

- Exemple de modèle de type « patatoïde »
  - Modèle graphique non normalisé… mais c'est mieux que rien !
  - Par exemple que représente les flèches Ri ?



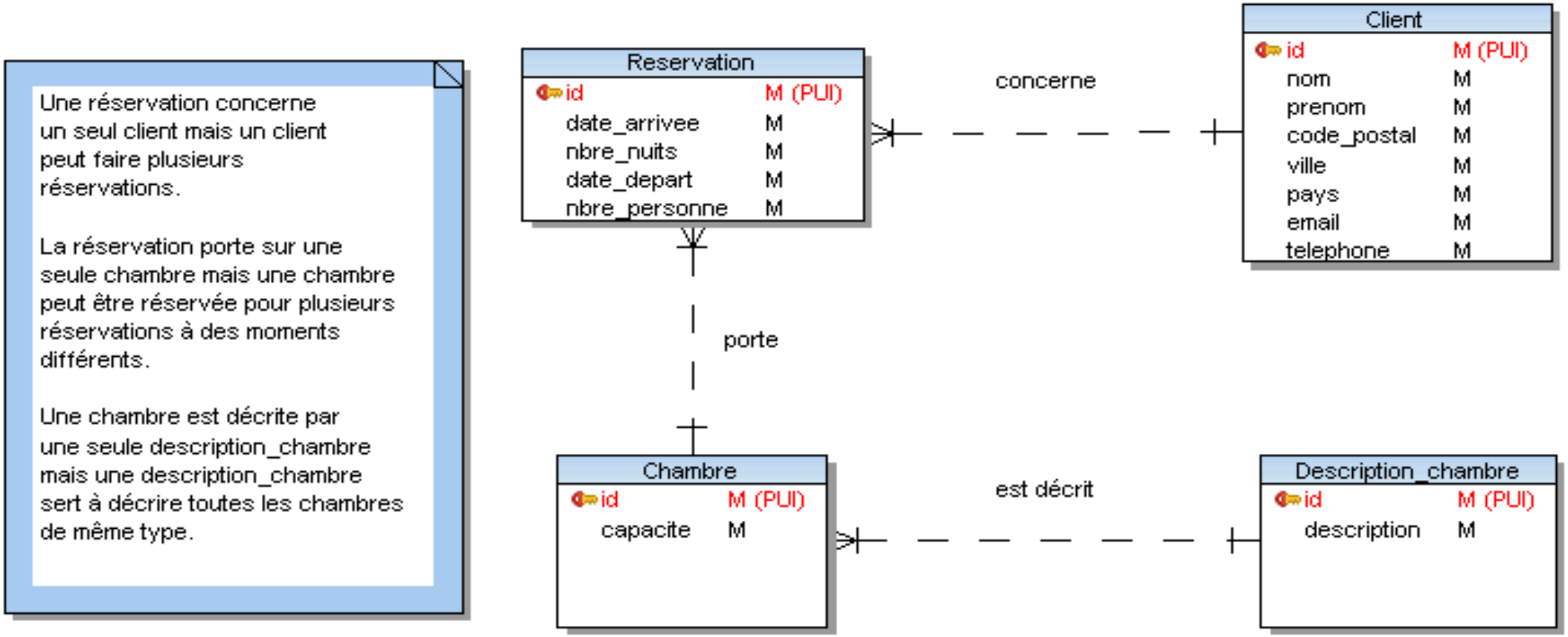
# Exemples de modèles (2/5)

- Exemple de modèle en UML : diagramme de classe
  - Langage standard de l'OMG www.omg.org (≠ normalisé)
  - Langage orienté objet comprenant jusqu'à 11 types de diagrammes (UML 2.0)
  - Conception en Java RMI de l'algorithme d'exclusion mutuelle avec l'anneau à jeton :



# Exemples de modèles (3/5)

- Exemple de modèle de données : diagramme entité-association
  - Permet de définir les données qui seront stockées dans les tables d'une base de données et les relations entre ces données



# Exemples de modèles (4/5)

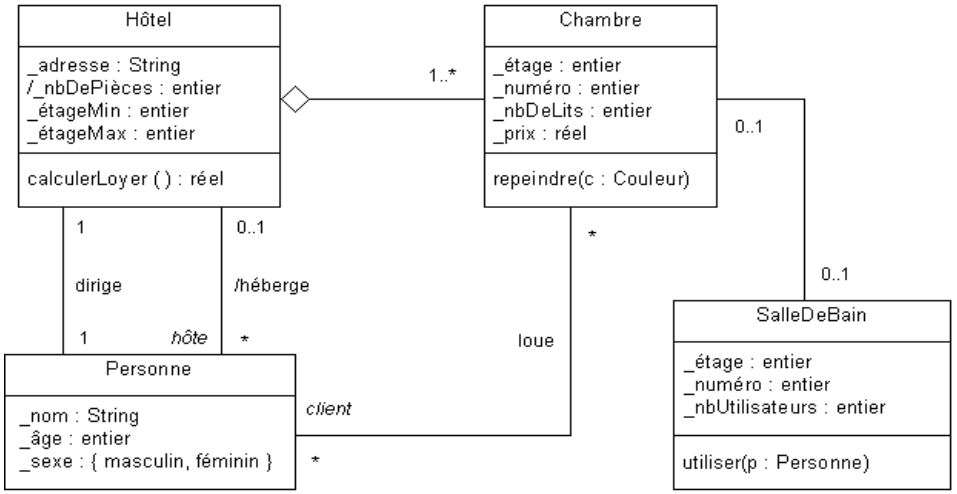
- Exemple de modèle en langage OCL
  - Langage logique orienté objet (standard de l'OMG)
  - Sert à définir des contraintes logique sur les relations, les attributs, les méthodes, etc.

```
context Chambre inv:
   client->size <= _nbDeLits or
   (client->size = _nbDeLits + 1 and client->exists(p : Personne | p._âge < 4))
```

```
context Hôtel inv:
   self.chambre->forAll(c : Chambre | c._étage <= self._étageMax and c._étage >= self._étageMin)
```

```
context Chambre::repeindre(c : Couleur)
   pre: client->isEmpty
   post: _prix = _prix@pre * 1.1
```

```
context Hôtel::calculerLoyer() : réel
 pre:
 post: result =
            self.chambre->select
               (client->notEmpty)._prix->sum
```



# Exemples de modèles (5/5)

- Exemple de modèle avec le langage B
  - Langage de spécification formel : prédicats logiques + théorie des ensembles mathématique
  - Spécification d'un système de réservation :

```
MACHINE
    RESERVATION
CONSTANTS
    nb_max, SIEGES
PROPERTIES
    nb_max ∈ 1..maxint ∧          /* Nombre maximum de sièges */
    SIEGES = 1..nb_max            /* Ensemble des numéros de sièges */
VARIABLES
    occupes, nb_libre
INVARIANT
    occupes ⊆ SIEGES              /* Ensemble des sièges occupés */
     ∧ nb libre ∈ 0..nb max       /* Nombre de sièges libres */
     ∧ nb_libre = nb_max − card(occupes)
INITIALISATION
    occupes, nb_libre := ∅, nb_max
OPERATIONS
    nb ←— place libre =           /* Nombre de places libres */
        BEGIN
            nb := nb_libre
        END ;
    place ←— reserver =           /* Le résultat est la place réservée */
        PRE
            nb libre ≠ 0
        THEN                      /* On prend une certaine valeur pp dans */
            ANY pp WHERE          /* les numéros de sièges libres */
                pp ∈ SIEGES − occupes
            THEN
                place, occupes, nb libre := pp, occupes ∪ {pp}, nb libre − 1
            END
        END ;
    rr ←— liberer(place) =        /* Opération de libération */
        PRE
            place ∈ SIEGES
        THEN                      /* On libère la place indiquée en paramètre */
            IF place ∈ occupes THEN
                rr, occupes, nb_libre := TRUE, occupes − {place}, nb_libre + 1
            ELSE
                rr := FALSE
            END
        END
END
```