



UNIVERSITÉ DE  
SHERBROOKE

# **INF755 Méthodes d'analyse et de conception**

**Hiver 2018**

**Séance-3**

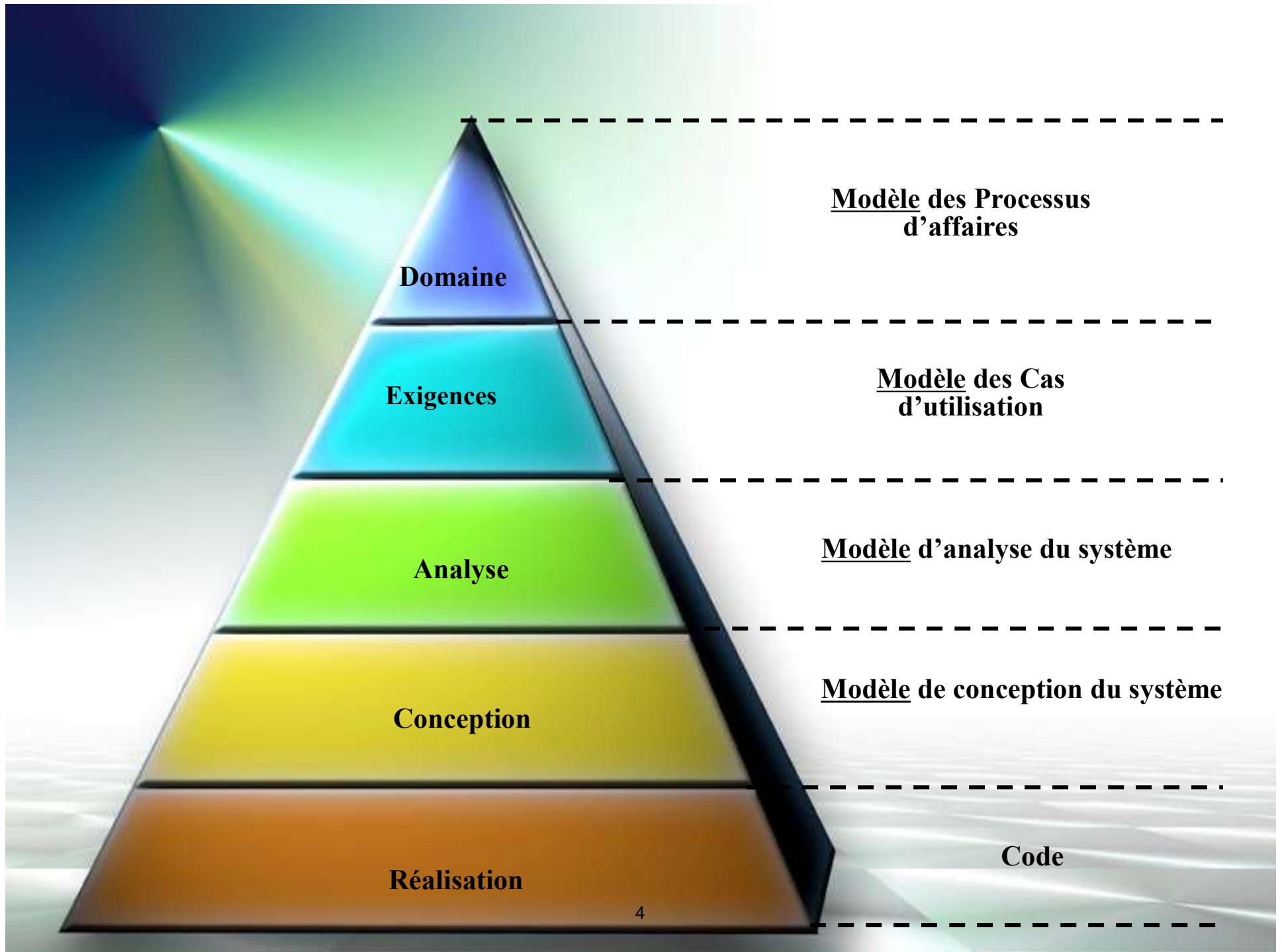
Chargé de cours: Alain Cardinal

# Plan de la séance-3

- Retour séance-2
- Techniques d'explicitation des exigences
- Techniques de prototypage
- Le modèle du domaine
- Exercice en classe (Diagramme d'activités)

## Retour : séance-2

- L'architecture des SI
- Les types de Méthodologie
- L'introduction aux diagrammes UML
  - Le diagramme d'activité
- L'introduction à merise (DFD, MCD)
- Travail d'équipe



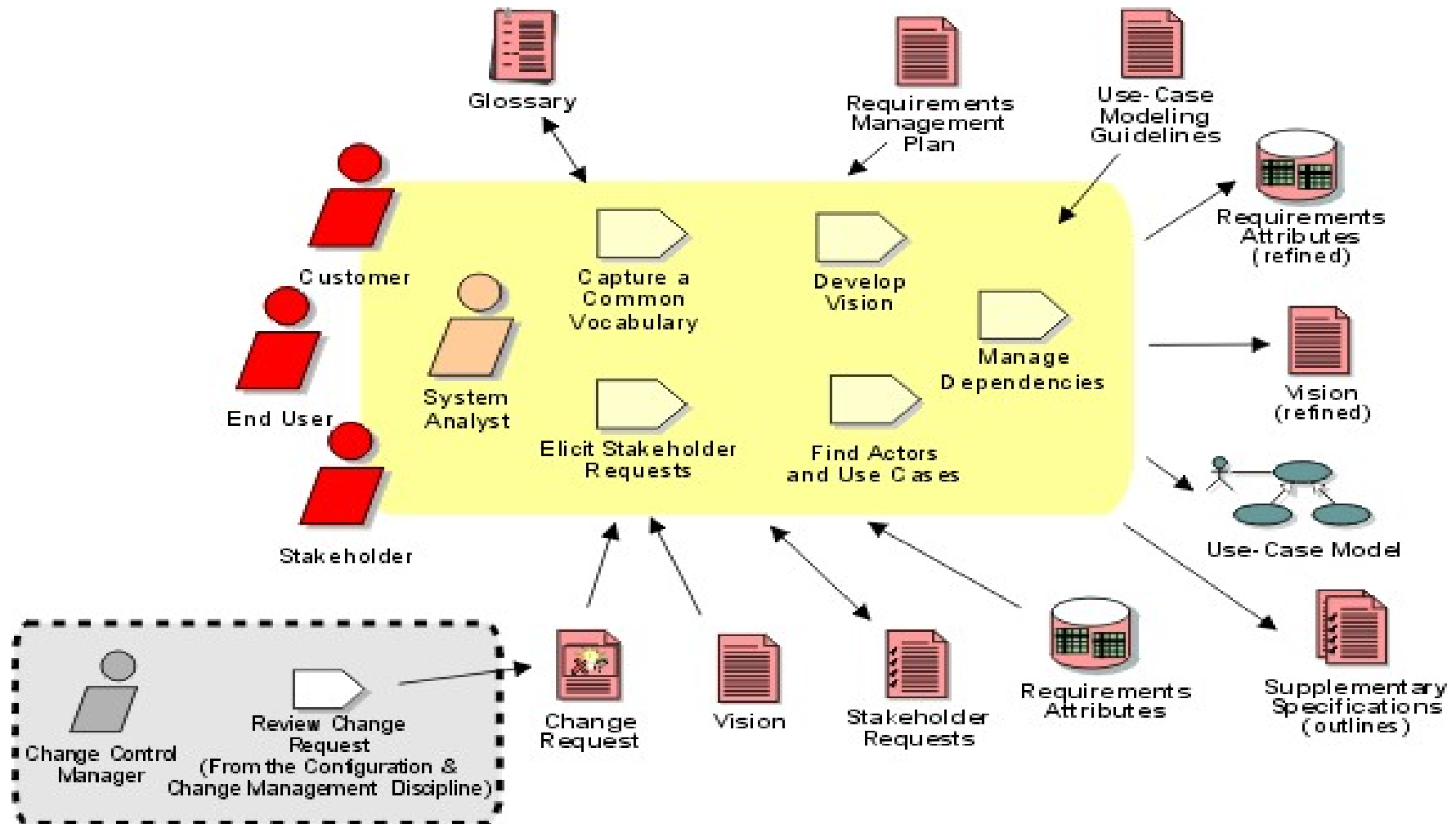


UNIVERSITÉ DE  
SHERBROOKE

# Techniques d'explicitation des exigences



# Comprendre les besoins



# Objectifs

- Rappelez-vous les raisons pour lesquelles les projets échouent...
- L'objectif de ce processus est de collecter et expliciter l'information provenant des intervenants du projet pour **comprendre leurs besoins**.
- Les requêtes peuvent être vues comme une “**Wish list**” qui va être utilisé comme premier intrant pour définir les caractéristiques (features) de haut niveau du système, tel que décrit dans le document de vision.
- Par la suite, ces caractéristiques alimenteront les **exigences logicielles**, telles qu'elles seront décrites dans le modèle des cas d'utilisation, les cas d'utilisation et les spécifications supplémentaires.

# Expliciter les requêtes des intervenants

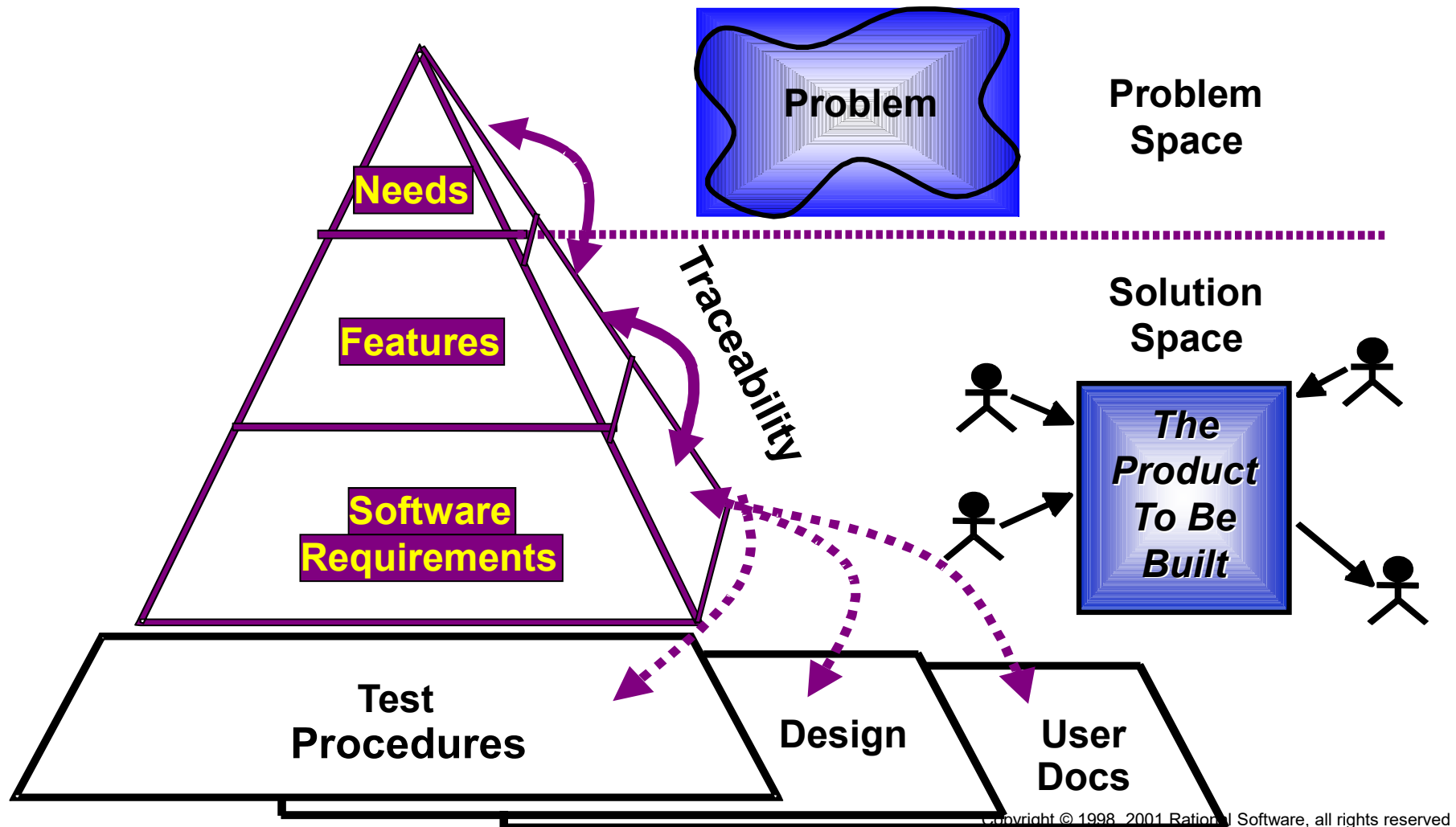
- **Objectifs**
  - Qui sont les **intervenants** du projet?
  - Quels sont les **besoins** auxquels le système doit répondre?
  - **Prioriser** les requêtes des intervenants.



## Expliciter les requêtes des intervenants

- Étapes
  - Identifier les **sources d'information** des exigences
  - Récupérer l'information
  - Réaliser des ateliers d'explicitation des exigences
  - Évaluer les résultats

# bonne vieille pyramide



# Les problèmes de l'explicitation

- Le syndrome du *oui..mais*
- Le syndrome des *ruines non découvertes*
- Le syndrome des *utilisateurs et des développeurs*

## Le syndrome des *ruines non découvertes*

- Combien de ruines reste-t-il à découvrir...
  - Plus vous en trouvez....moins il y en a.
- Même chose pour les exigences...
  - Vous ne saurez jamais si vous les avez tous trouvées
  - Vous espérez en trouver assez.
  - **Périmètre**

## Le syndrome du *oui...mais*

- Lorsque l'utilisateur voit l'implémentation, il y a 2 réactions possibles
  - Wow, c'est cool
  - Oui, mais...hummmm, maintenant que je vois ça, pourrions-nous...et que pensez-vous de....
- Il faut vivre avec...
  - Permet de découvrir de nouvelles exigences
  - Minimiser ce syndrome en explicitant les exigences plus tôt

## **Le syndrome des utilisateurs et des développeurs**

### **L'utilisateur :**

- Ne sait pas ce qu'il veut
- Le sait, mais ne peut l'expliquer
- Pense savoir ce qu'il veut – avant de se le faire dire par le développeur
- L'analyste pense savoir + que l'utilisateur
- Tout le monde pense que tous les autres font de la politique

# Le syndrome des utilisateurs et des développeurs

- Les solutions...
  - **Reconnaître et apprécier** l'utilisateur comme un expert du domaine.
  - **Essayer différentes techniques** d'explicitation des exigences
  - Mettez-vous à la place de l'utilisateur, faites son travail pendant 1 heure ou 2. (jeux de rôles)
  - La politique fait partie de la nature humaine...

## Techniques d'explicitation des exigences

- Interviews et questionnaires
- Atelier d'explicitation d'exigences (Requirements Workshop)
- Session remue-méninges (Brainstorming)
- Scénario-maquette
- Jeux de rôles
- Prototypage



# Choix de la technique d'explicitation

- Type de l'application
- Aptitude de l'équipe de développement
- Aptitude des clients
- L'envergure du problème
- La criticité (nature) du problème
- La terminologie
- Unicité de l'application

## Conseils pour réussir une interview

- Préparation d'un contexte de libre interview (**prendre notes**)
- Avant l'interview, rechercher l'expérience des intéressés et de la compagnie à interviewer
- Noter les réponses aux questions durant l'interview
  - Ou prévoir quelqu'un qui va le faire
- S'assurer que les questions posées sont cohérentes avec le gabarit
- Durant l'interview, il est nécessaire de garder en tête l'objectif, même s'il peut arriver qu'on s'écarte parfois.
- Reformuler les concepts – Facilitateur, réveillez-vous!
- À la fin, revoir les principaux éléments et s'assurer d'une compréhension commune

# Questionnaires

- Limités, car:
  - Difficile de trouver les bonnes questions à l'avance
  - Les questions peuvent influencer les résultats
  - Difficile d'explorer d'autres avenues
  - Difficile de faire un suivi sur des réponses vagues

# Atelier d'explicitation des exigences

- Il aide à construire une équipe efficace, réunie pour un objectif commun : le succès du projet.
- Tous les intéressés ont leur mot à dire, aucun n'est laissé de côté.
- Il se bâtit un accord entre les intéressés et l'équipe de développement sur ce que l'application doit faire.
- Il peut exposer et résoudre les problèmes politiques qui peuvent compromettre le succès du projet.
- **La définition préliminaire** du système au niveau caractéristique est immédiatement disponible suite à l'atelier.

# Remue-méninges

- Elle encourage la participation de toutes les parties présentes
- Elle encourage l'utilisation des idées des autres
- Le « facilitateur » prend note de tout ce qui se dit (rien n'est perdu)
- Elle résulte en un **grand ensemble possible de solutions** au problème posé
- Elle encourage toutes les idées sans contrainte

## Règles pour le remue-méninges

- Pas de critiques ou débats
- Laisser place à **l'imagination**
- Générer autant d'idées que possible
- Combiner et muter les idées

# Étapes du remue-ménages

- Génération d'idées
- Émondage (*Pruning*)
- Regroupement
- Définition des caractéristiques
- Définition des priorités

# Photo de remue-méninges





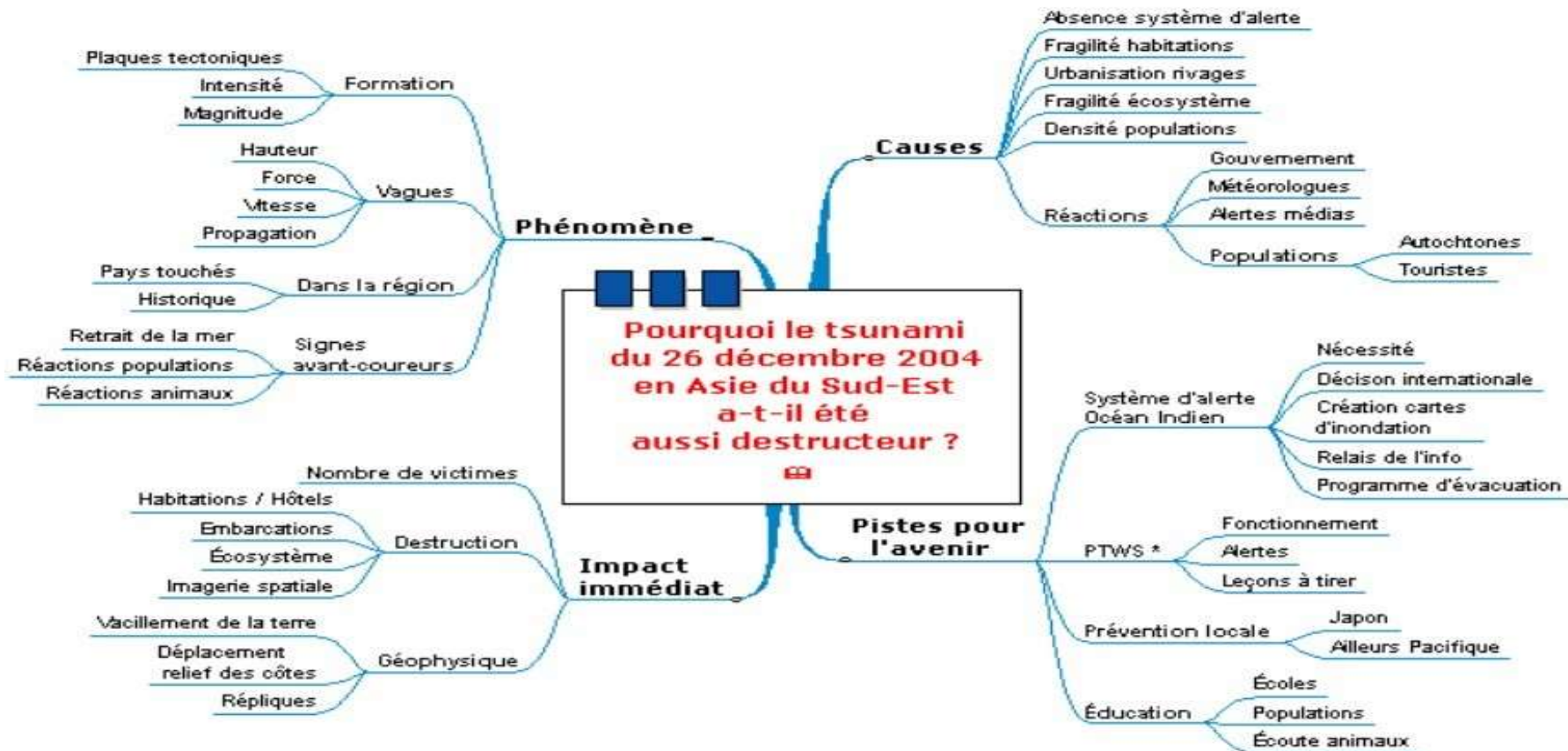


# remue-méninges (carte mentale)

Exemple de **carte mentale** développée en cours de projet



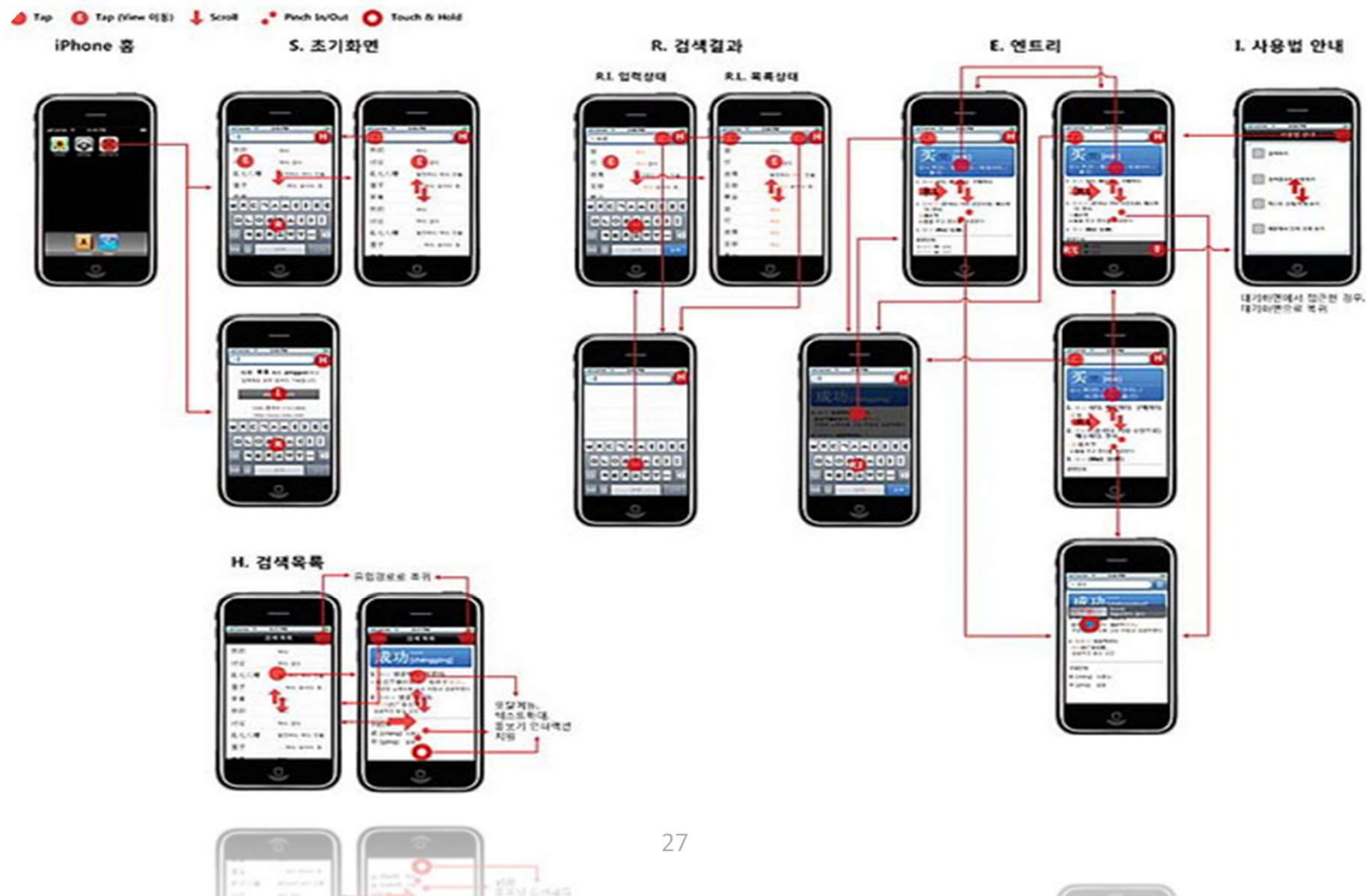
Tsunami du 26 décembre 2004



# Scénario-maquette

- Le « **storyboarding** » sert à observer la réaction des utilisateurs tôt dans le cycle de vie du développement. Il offre les avantages suivants :
  - Très faible coût
  - Convivial, informel et interactif
  - Permet une révision précoce des interfaces utilisateurs du système
  - Facile à créer et à modifier
- **Les scénarios-maquettes** sont aussi un moyen puissant pour surpasser le syndrome de la page blanche


# Scénario-maquette (représentation)





# Types

<b>Passif</b>	<b>Actif</b>	<b>Interactif</b>	Prototypage
Écrans	Présentations		
Règles d'affaires	Animation	Démonstration	
Rapports	Simulation	Présentation interactive	
Coûts et complexité			



## Conseils pour le « scénario-maquette »

- Ne pas trop investir dans le « scénario-maquette »
- Si vous ne changez rien, vous n'avez rien appris
- Ne pas trop bien les faire
- Si possible, faites-les interactifs



## Jeux de rôles

- Les jeux de rôles permettent à l'équipe de développement d'expérimenter le monde des utilisateurs en jouant leurs rôles
- Pour la compréhension des exigences, il faut garder en tête :
  - **Nous devons comprendre que beaucoup d'utilisateurs ne peuvent articuler les besoins qui doivent être définis**



# Jeux de rôles

- Pour la compréhension des exigences, il faut garder en tête :
  - Beaucoup d'utilisateurs n'ont pas la liberté d'admettre qu'ils ne suivent pas une procédure écrite
  - Des utilisateurs individuels ont leurs modèles d'activités de travail profondément enracinés
  - Il est impossible pour n'importe quel développeur d'anticiper toute question qui doit être posée, ou pour tout utilisateur de savoir toute question que le développeur devrait poser.

# Plan de la séance-3

- Retour séance-2
- Techniques d'explicitation des exigences
- **Techniques de prototypage**
- Le modèle du domaine
- Exercice en classe (Diagramme d'activités)



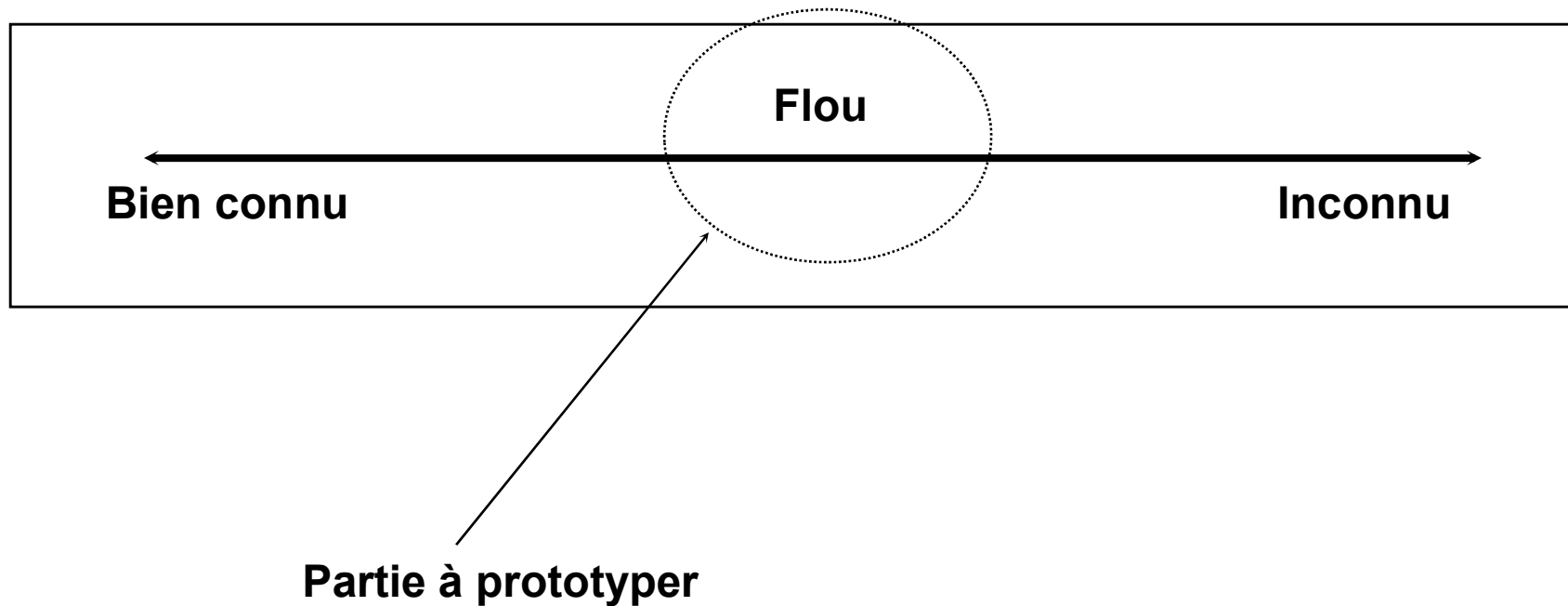


# Prototypage

- Le prototypage sert à identifier les besoins réels des utilisateurs. Ils peuvent interagir de façon concrète avec une partie du système, ce qui permet de :
  - Découvrir d'autres exigences, que les utilisateurs n'ont pas su exprimer
  - Éviter le syndrome «Oui, mais »



## Que faut-il prototyper ?





## Les dangers

- Le prototypage peut parfois glisser vers un prototypage fonctionnel...
- Les clients pensent alors qu'entre votre prototype et l'application finale il n'y a qu'un pas
- Ce n'est pas toujours vrai



## Petit sondage interne

- Quels sont vos techniques d'explicitation des exigences?
  - Interview
  - Questionnaires
  - Atelier d'explicitation d'exigences
  - Session remue-méninges
  - Scénario-maquette
  - Jeux de rôles
  - Prototypage



# EXERCICE-1

- Cas: UPS rivalise avec la concurrence mondiale grâce a la technologie de l'information
  - **Afin de trouver des améliorations durable au cas, en équipes de 4 ou 5 personnes;**
    - **Lecture du cas**
    - **Remue méninge**
      - **Identifier les lacunes actuelles**
      - **Identifier les événements**
        - » **Identifier les déclencheurs**
      - **Identifier les acteurs**
      - **Identifier les entités ou classes**
      - **Présenter les pistes d'améliorations (au moins 2)**
    - **Réaliser le diagramme de cas d'utilisation**

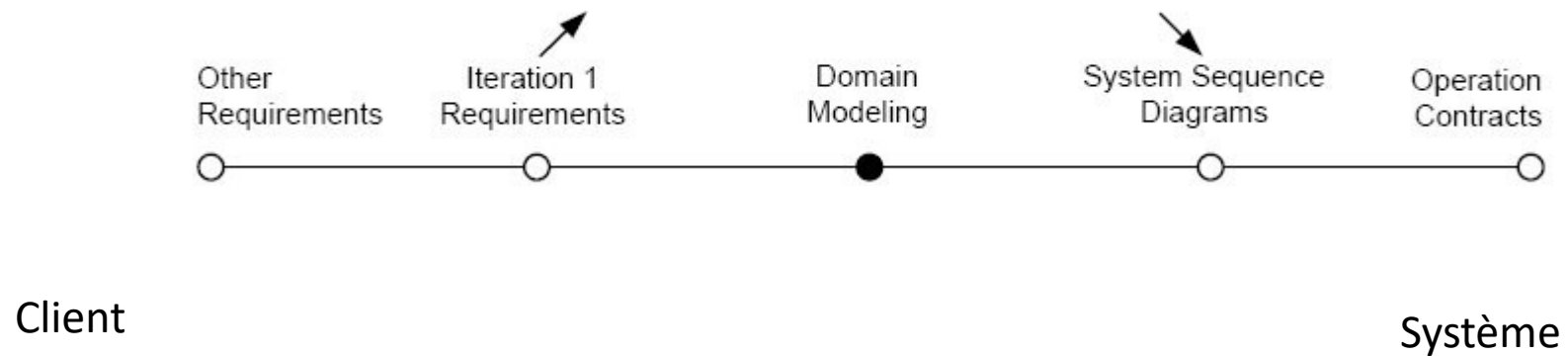


# Plan de la séance-3

- Retour séance-2
- Techniques d'explicitation des exigences
- Techniques de prototypage
- **Le modèle du domaine**
- Exercice en classe (Diagramme d'activités)

# Le modèle du domaine

Ref: Larman



# Le modèle du domaine

- Pourquoi?
  - Visualiser les concepts, les **objets**, les **idées** de l'entreprise ainsi que les **liens** qui les relient
- Motivation?
  - Modéliser les concepts métiers
  - Préparer le terrain à l'équipe logicielle



# Le modèle du domaine

- Réduire l'écart entre le domaine et le « domain layer » des applications

# Le modèle du domaine

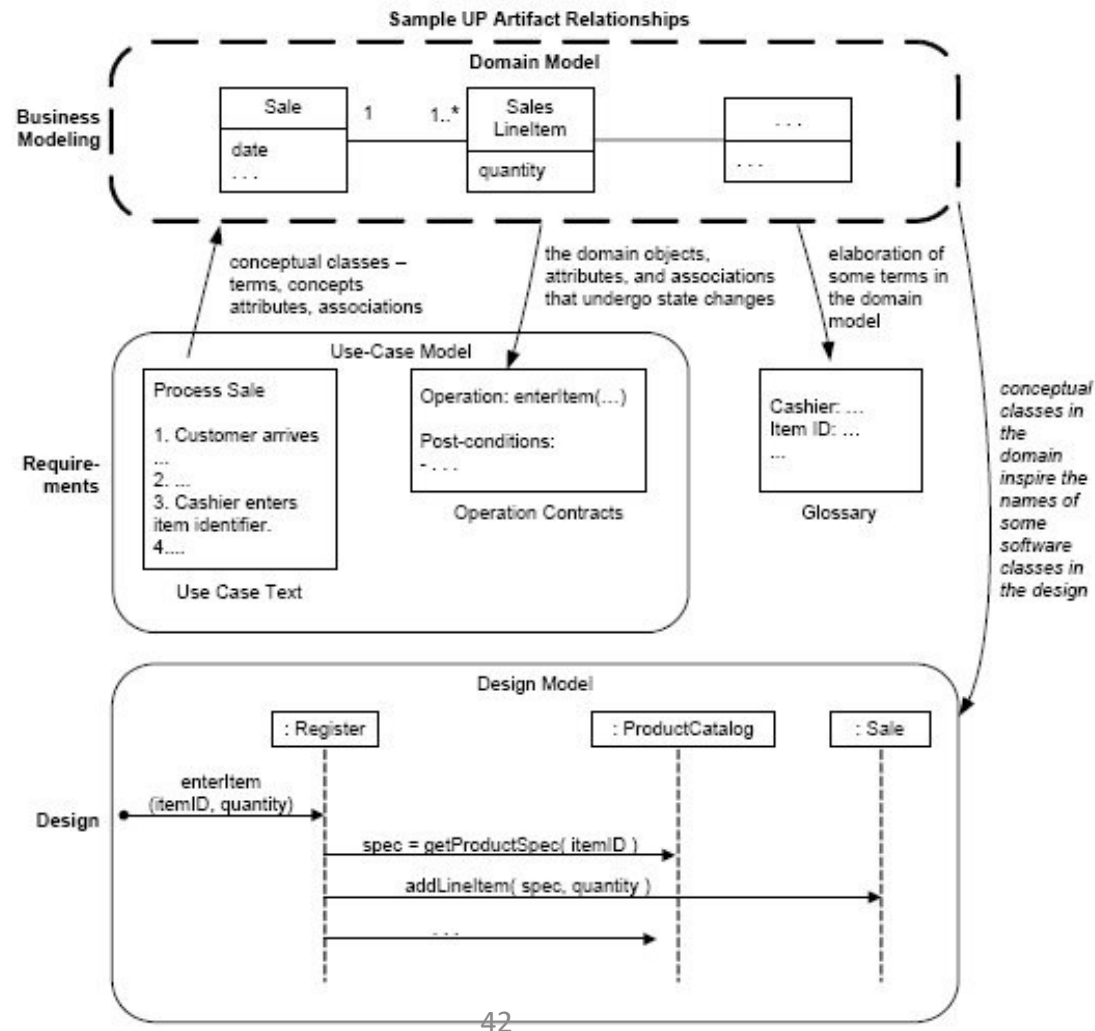


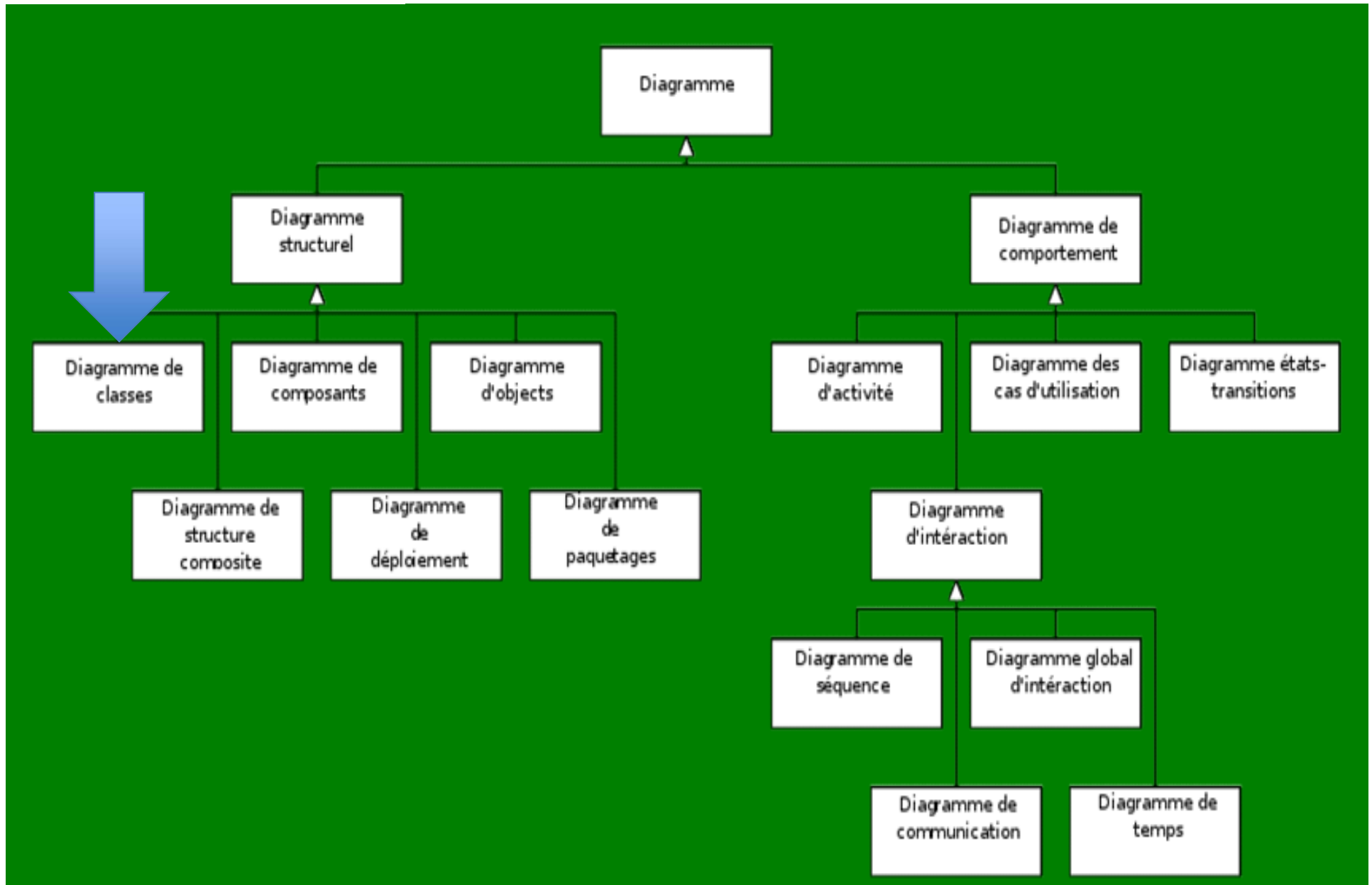
Figure 9.1 Sample UP artifact influence.

# Rappel de quelques notions

- Les classes
- Les attributs
- Les méthodes (attention..)
- Les associations
- Les cardinalités

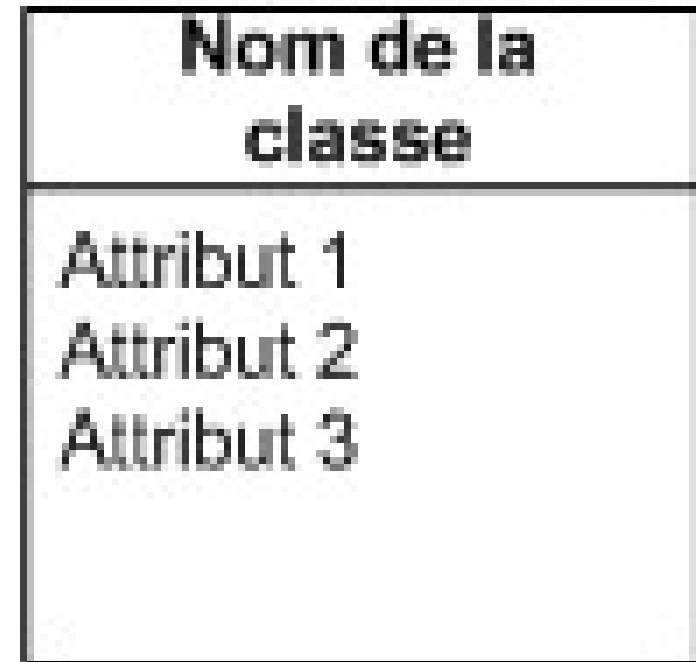


# UML



# Les classes

- Permet de se représenter un concept une idée.
- La section du haut permet de nommer la classe.
- La section du bas permet d'y placer des attributs.





# Les attributs

- Propriétés de la classe
- Selon Larman...
  - Si le concept est un texte ou un chiffre, c'est souvent un attribut
    - L'âge, le NAS, nom, prénom
  - Autrement, c'est une classe
    - Aéroport, destination, provenance...



# Les méthodes

- Dans les classes de conception, on retrouve une troisième section qui permet d'inclure les méthodes (**actions**) que peuvent réaliser les classes.
- À l'extérieur du « *scope* » du modèle du domaine selon Larman mais utilisé par Roques.
- Dans le cadre de ce cours, je veux les méthodes métiers.



## Les associations

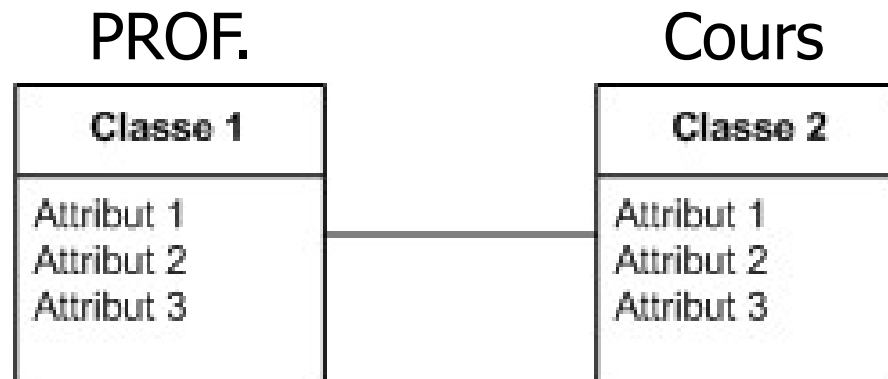
- Utile pour se représenter un **lien** entre deux classes.
- Un besoin de se souvenir, même momentanément, du lien.
- Éviter la multiplication des liens
  - Couplage...





# Association simple

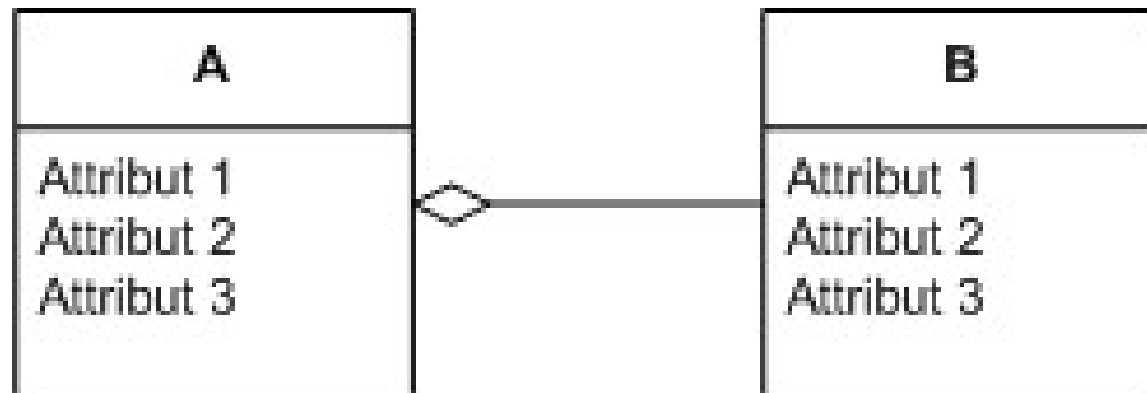
- Permet de représenter un lien entre 2 classes
- Un lien entre un prof et son cours





# Association d'agrégation

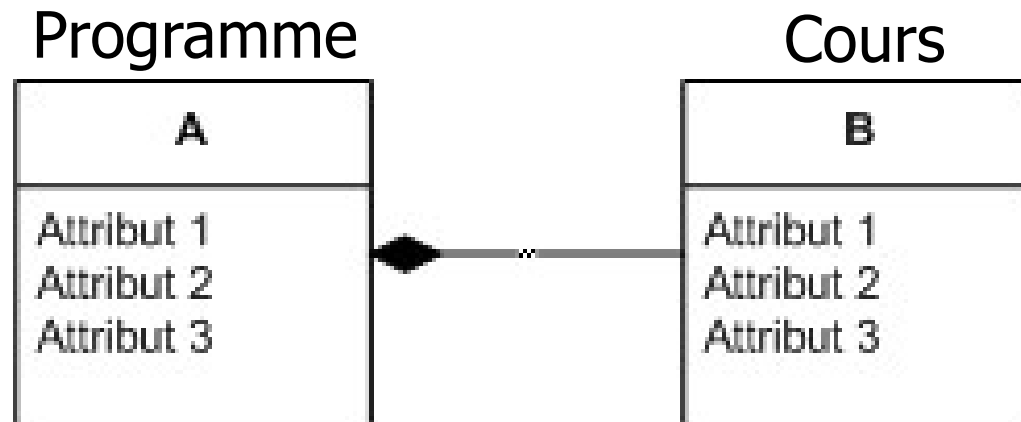
- Indique que la classe A contient des éléments de la classe B.
- La suppression de A n'implique **pas** la suppression de ou des B.





# Association composite

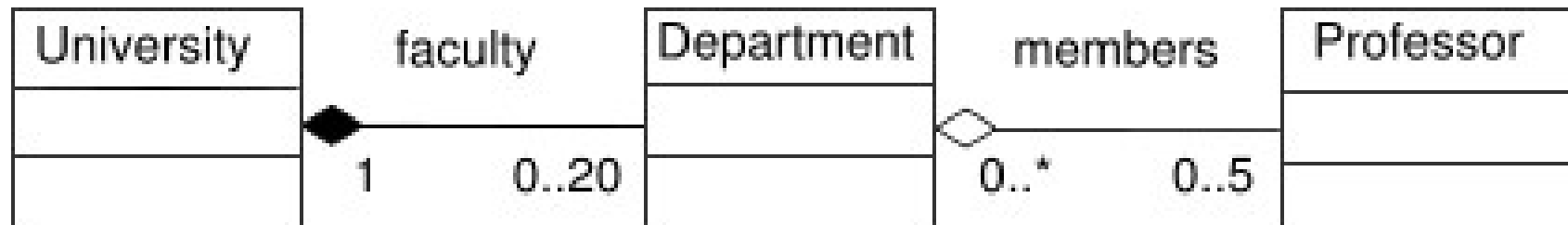
- La composition est une agrégation forte, où il y a un principe d'*ownership*..
- La suppression du contenant A **détruit** inévitablement les éléments contenus B.





# Associations d'agrégation

- Exemple
  - Si l'université ferme, les départements disparaissent, mais les professeurs survivent....fiou!

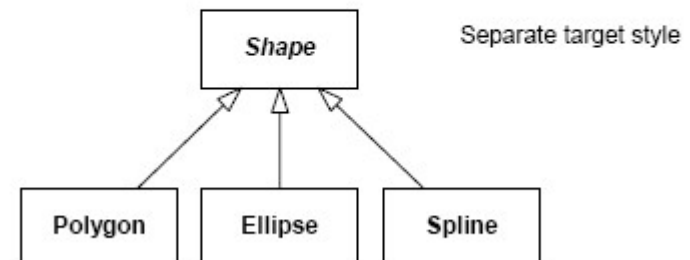
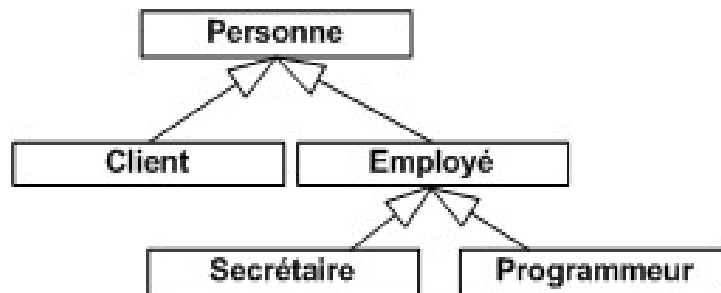




# Généralisation

- Permet de spécialiser une classe à partir d'une classe générale.

## Exemples



- Utiliser avec modération à ce stade



# Les associations

- Lors des premières itérations, on peut se contenter de mettre des associations simples.
- N'intégrer l'agrégation ou la composition que lorsque la logique l'impose.
- Attention à la complexité!



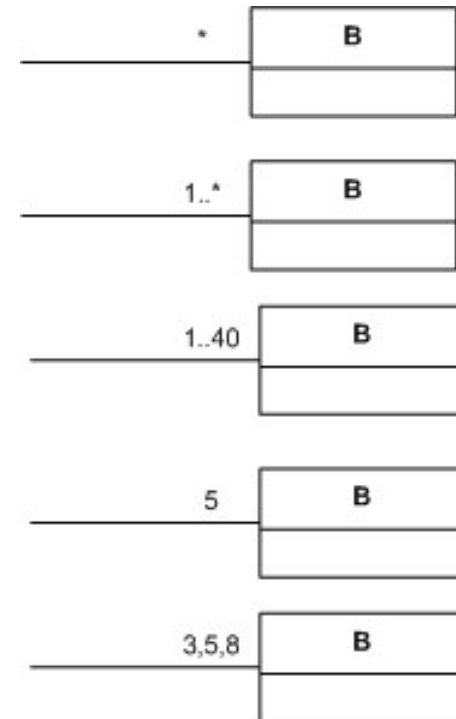
# Les cardinalités

- Les cardinalités permettent d'offrir une information supplémentaire sur la nature du lien
  - Combien d'éléments de  $A$  peuvent être associés à  $B$  et vice et versa.



# Les cardinalités

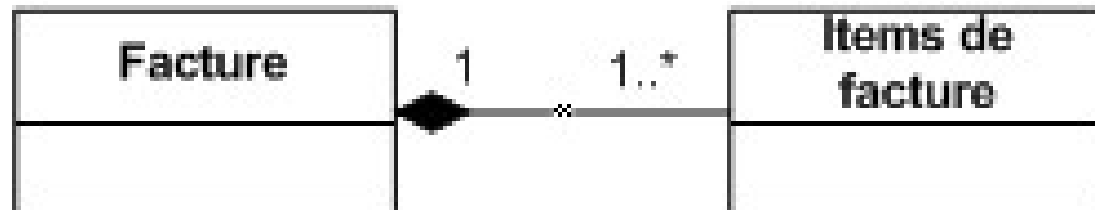
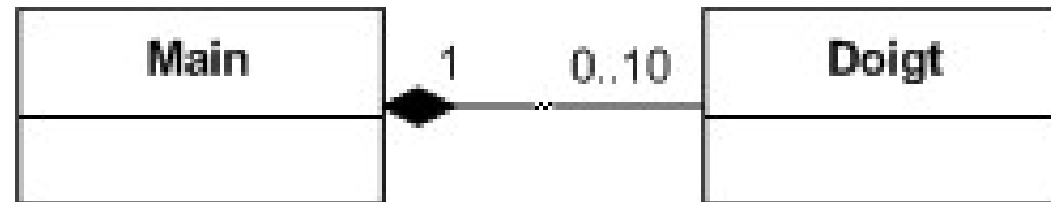
- 0 ou plusieurs
- 1 ou plusieurs
- 1 à 40
- Exactement 5
- 3, 5 ou 8







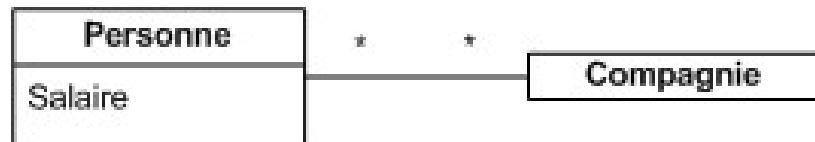
# Exemples



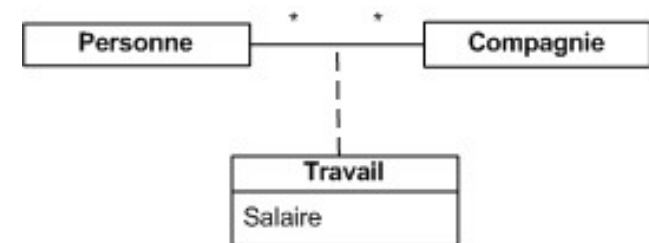


# Classe d'association

- Un peu particulier, lorsque vous voulez lier des attributs ou des méthodes à une association.
- Souvent utilisé avec une cardinalité \* \*



VS?

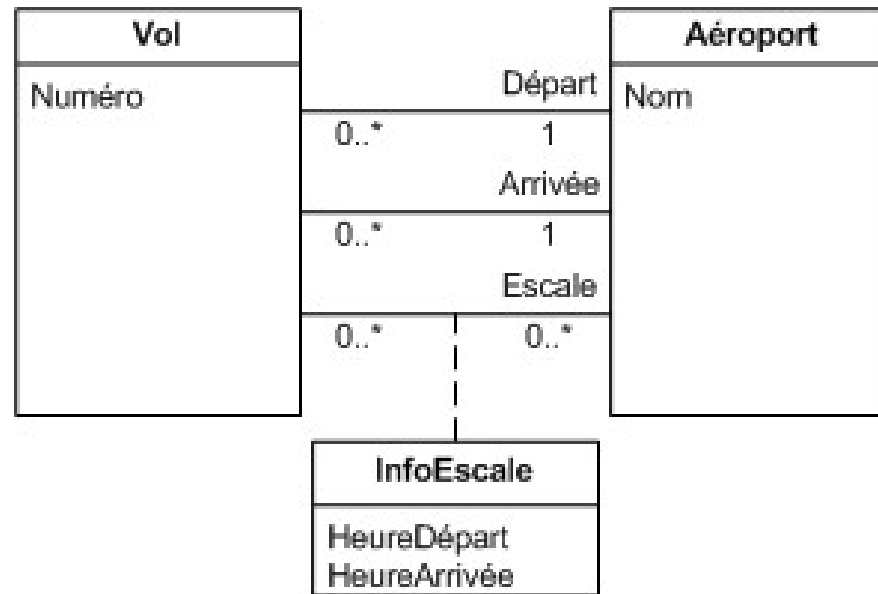


Même salaire pour toutes les cles?



# Liens multiples

- Très puissants pour représenter certains concepts
- Un **vol** part d'un **aéroport**, arrive dans un autre, avec possiblement plusieurs **escales**
- L'aéroport reçoit de 0 à plusieurs **arrivées**, 0 à plusieurs **départs** et 0 à plusieurs escales





# Le modèle du domaine

- **Les classes de description**
  - Existence de la description au-delà de l'existence d'une classe
    - Un **produit** et la **description** du **produit**.
    - Même si je n'ai pas de produit en stock, j'ai sa description.



# Classe de description

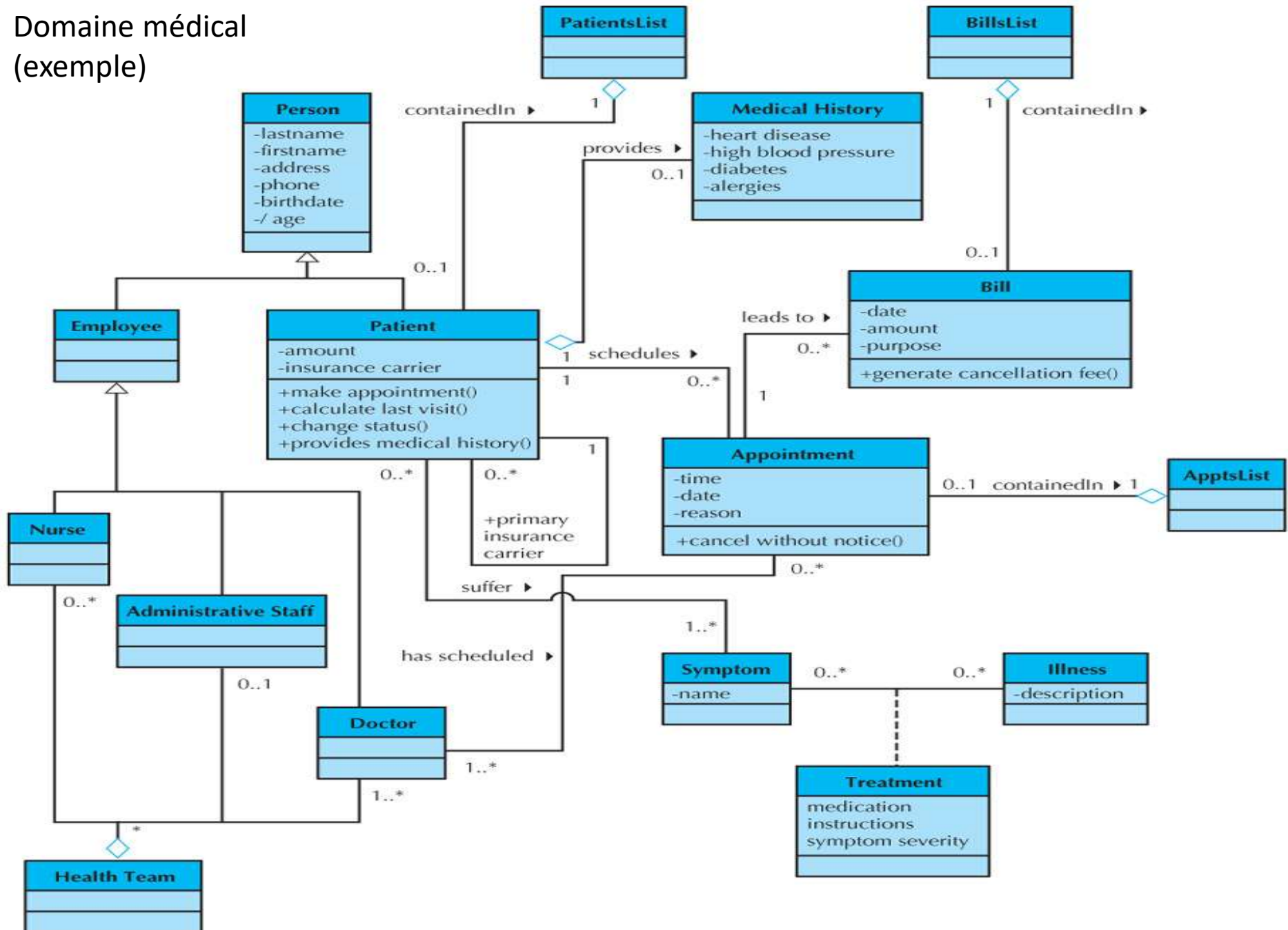
- Permet de conserver une description même si l'objet est absent
  - Je veux avoir la description de mes produits, même si je ne l'ai pas actuellement en stock.



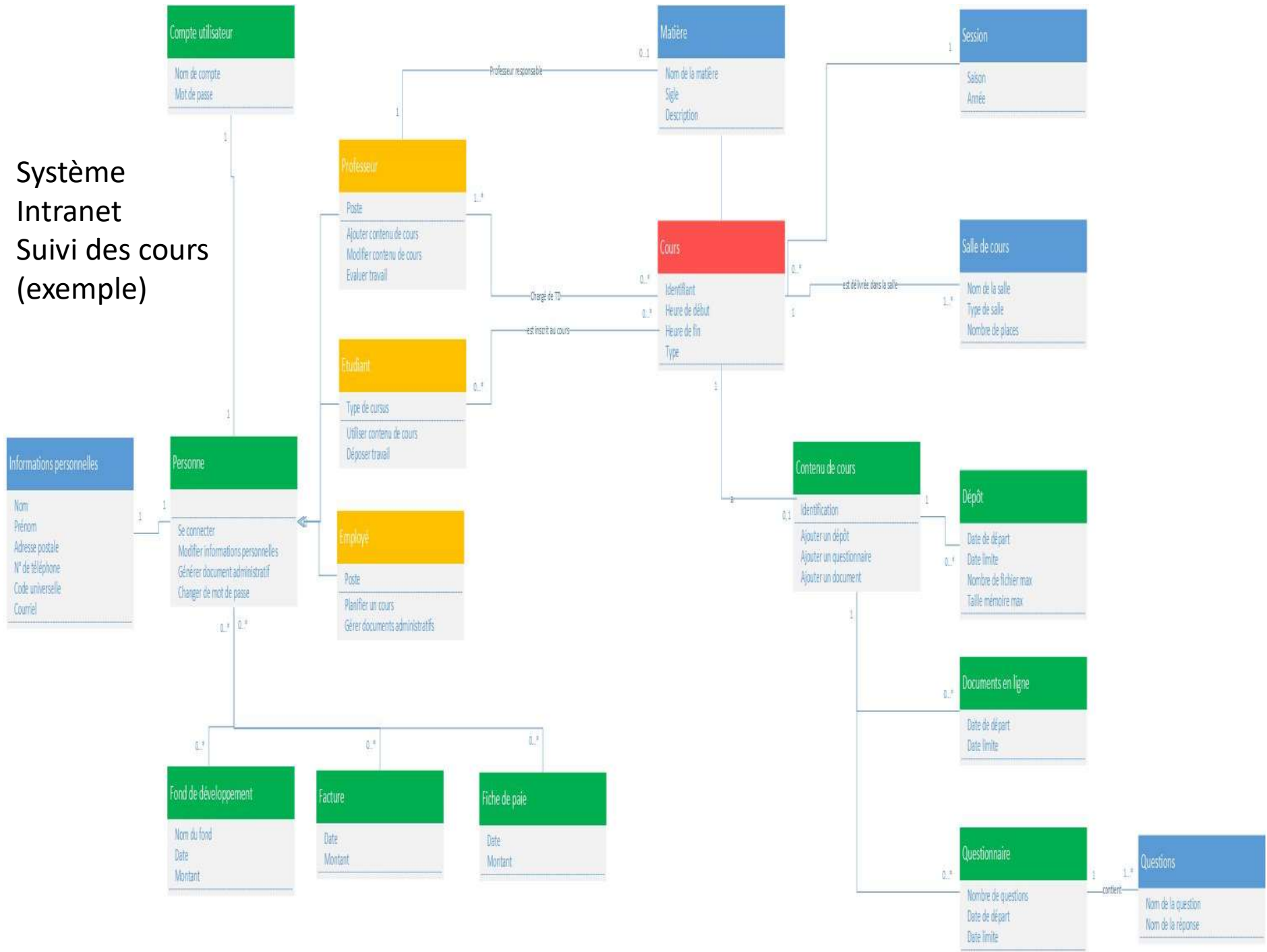
# Étapes pour créer un M. du D.

1. Trouver les classes conceptuelles
2. Dessiner les classes
3. Ajouter les associations
4. Ajouter les attributs
5. Ajouter les méthodes métiers
6. Ajouter les cardinalités

## Domaine médical (exemple)



# Système Intranet Suivi des cours (exemple)







# Prochaine Séance-4

- Les cas d'utilisation  
Diagramme  
Cas d'utilisation
- Le DFD (Diagramme de flux de données)
- Exercice en classe (DFD)
- Lectures:  
Voir Moodle séance-4 références



# Questions?



- Temps pour le travail d'équipe

