Introduction aux Processus Unifiés

Processus de développement

- UML/SysML est un langage (syntaxe et la sémantique précisément définies)
- UML/SysML n'est pas une méthode (pas de préconisation sur la manière d'utiliser le langage dans un projet)
- Le processus de développement (décrivant en particulier l'enchaînement des tâches) est volontairement exclu d'UML/SysML (car considéré comme trop dépendant du domaine d'activité)
- Les auteurs d'UML/SysML recommandent toutefois d'adopter une démarche : guidée par les besoins des utilisateurs, centrée sur l'architecture, ainsi qu'itérative et incrémentale. Aucun processus n'est toutefois à ce jour standardisé par l'OMG

Processus de développement

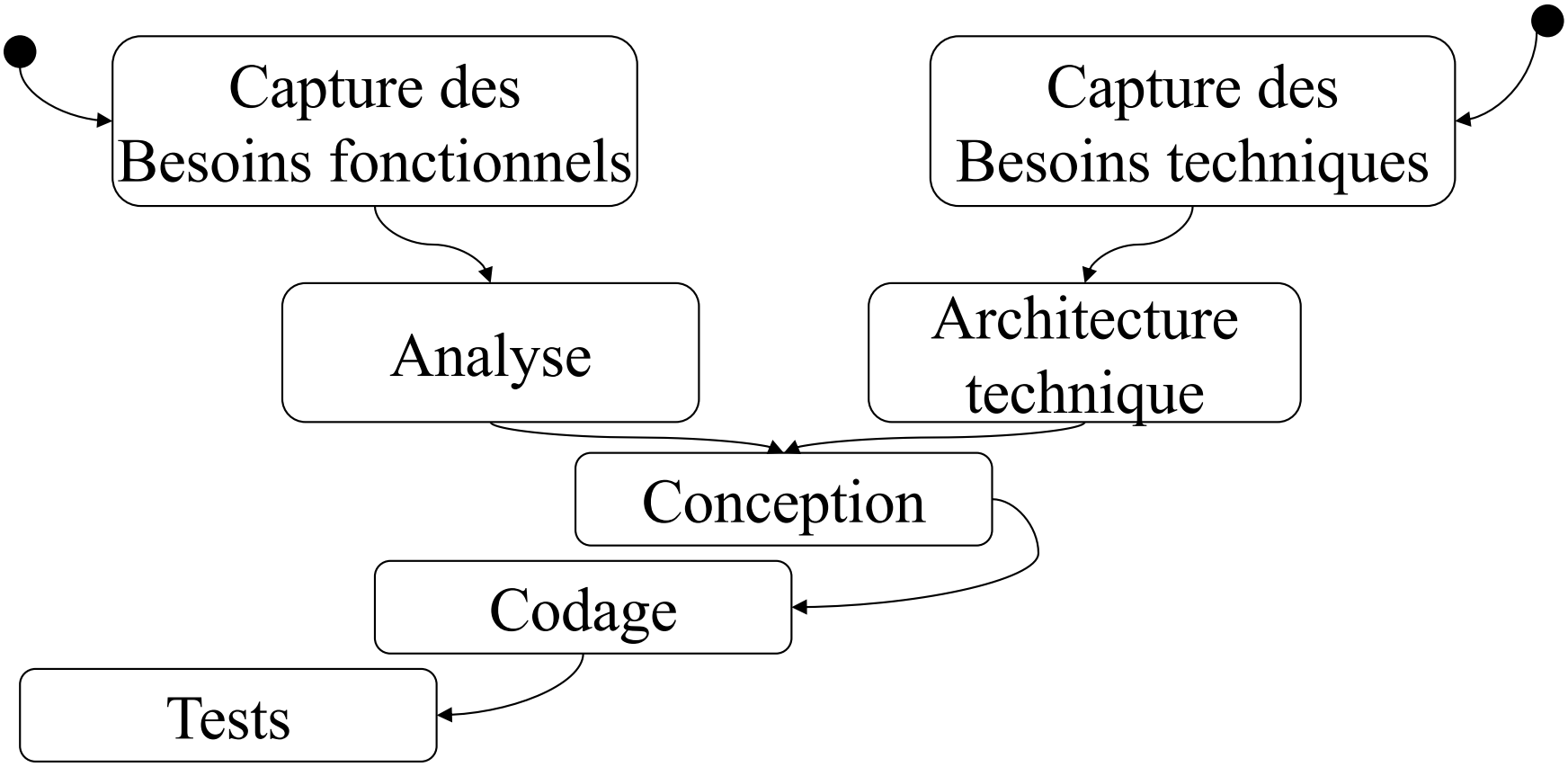
- Plusieurs ouvrages proposent un cadre méthodologique générique basé sur UML/SysML notamment :
- Le Rational Unified Process (RUP) de Rational Software (livre « Le processus unifié » : I. Jacobson, G. Booch, J. Rumbaugh)
- Le 2 Tracks Unified Process (2TUP) de Valtech (livre « UML en action » : P. Roques et F. Vallée)

Dans les deux cas, les processus distinguent des phases

- de capture des besoins
- d'analyse
- de conception

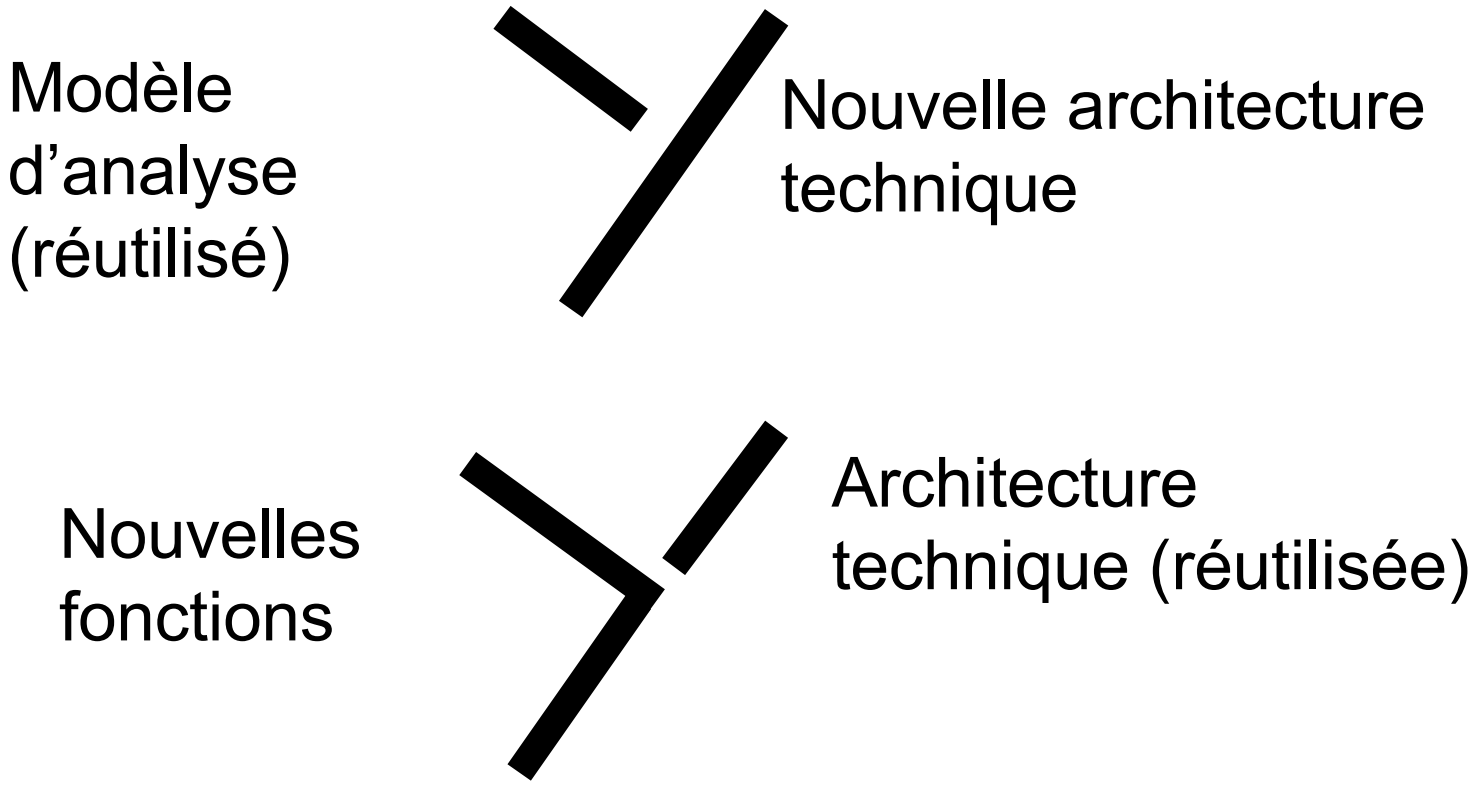
Le cycle en Y du 2TUP

- 2 TUP : Une branche fonctionnelle et une branche technique qui fusionnent à la phase de conception



Le cycle en Y du 2TUP

- Objectif de la séparation fonctionnel / technique : réutilisabilité séparée des savoir-faire métier / technique



Phases du 2TUP

Le 2 TUP définit 4 phases

- Recueil des besoins (pré-étude)
- Capture des besoins
- Analyse objet
- Conception

Elles mêmes décomposées en plusieurs étapes

Phases du 2 TUP

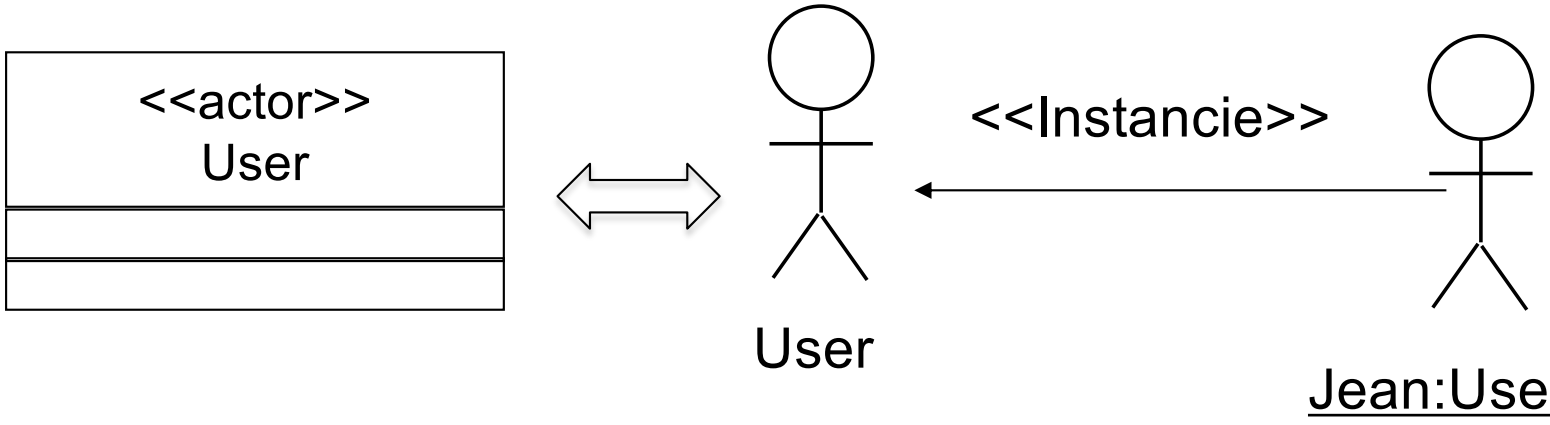
- Recueil des besoins (pré-étude)
  - Identifier les acteurs
  - Identifier les messages
  - Modéliser le contexte
- Capture des besoins
  - Identifier les cas d'utilisation
  - Documenter les cas d'utilisation
  - Organiser les cas d'utilisation
  - Identifier les classes candidates

Phase du 2 TUP

- Analyse objet
  - Découpage en catégories
  - Développement du modèle statique
  - Développement du modèle dynamique
- Conception
  - Conception préliminaire
  - Conception détaillée

Les acteurs

- Acteurs : entités (humaines ou techniques) externes agissant directement sur le système
- Une même personne physique peut jouer plusieurs rôles et être représenté par plusieurs acteurs
- Ne pas confondre acteurs et périphériques d'interface
- Représenté par une classe stéréotypée « Actor »



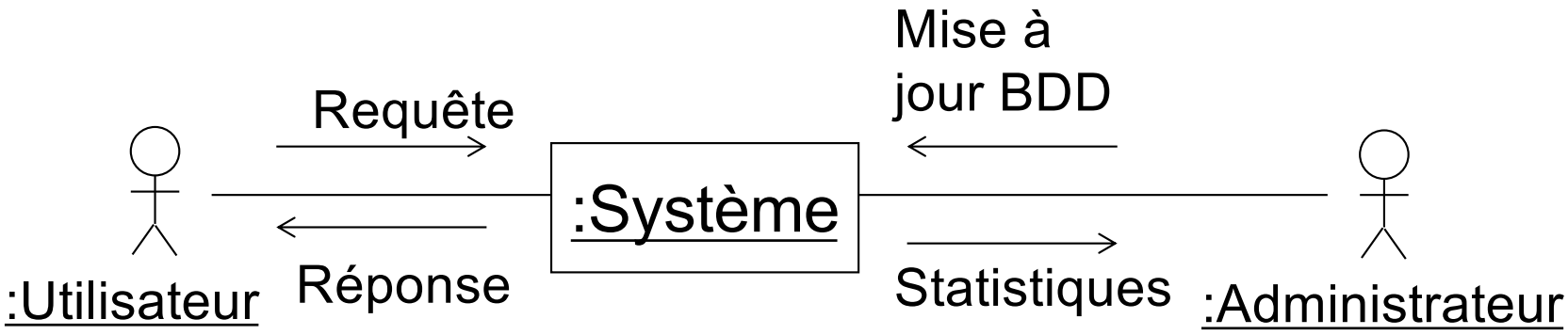
Les messages

- Echanges d'informations de haut niveau acteurs / système
- Pour chaque acteur : rechercher les messages ayant pour but de déclencher une action attendue du système
- Pour le système s'interroger sur les informations attendues par chaque acteur
- Pas d'échanges de messages entre acteurs

Le contexte

- Recueil des besoins (pré-étude)
  - Identifier les acteurs
  - Identifier les messages
  - Modéliser le contexte

- Représenté par un diagramme de contexte dynamique (type diagramme de communication)

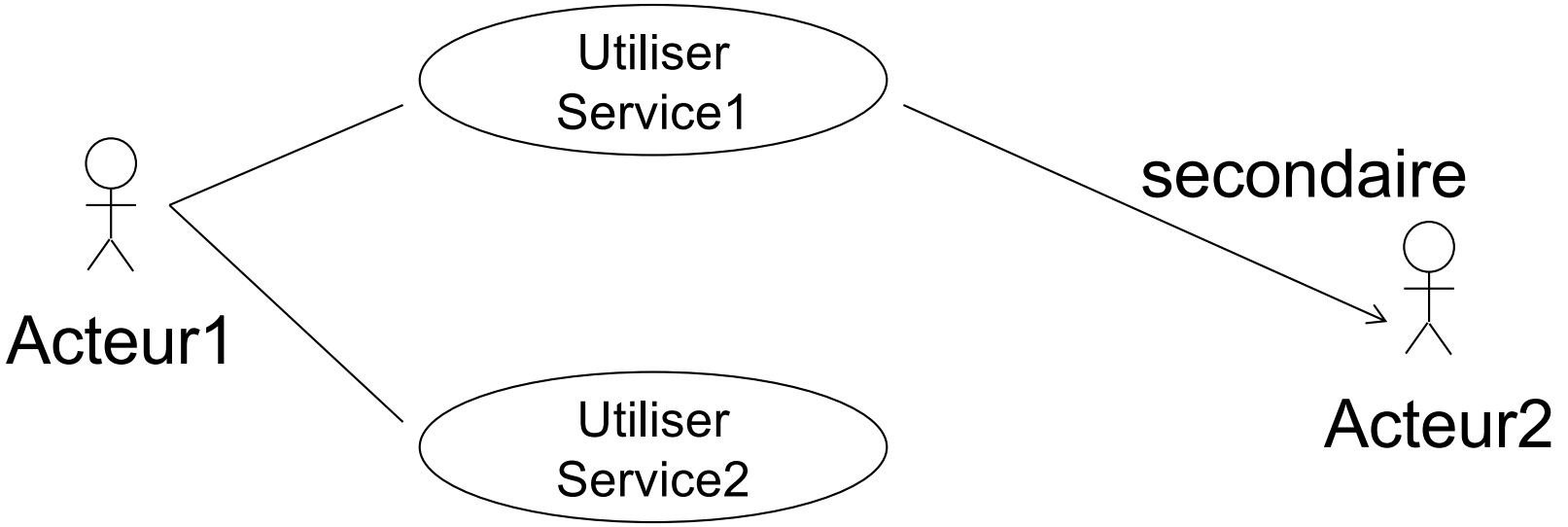


Il peut y avoir plusieurs instances d'un même acteur, mais un seul objet représentant le système

Use cases

- Capture des besoins
  - Identifier les cas d'utilisation
  - Documenter les cas d'utilisation
  - Organiser les cas d'utilisation
  - Identifier les classes candidates

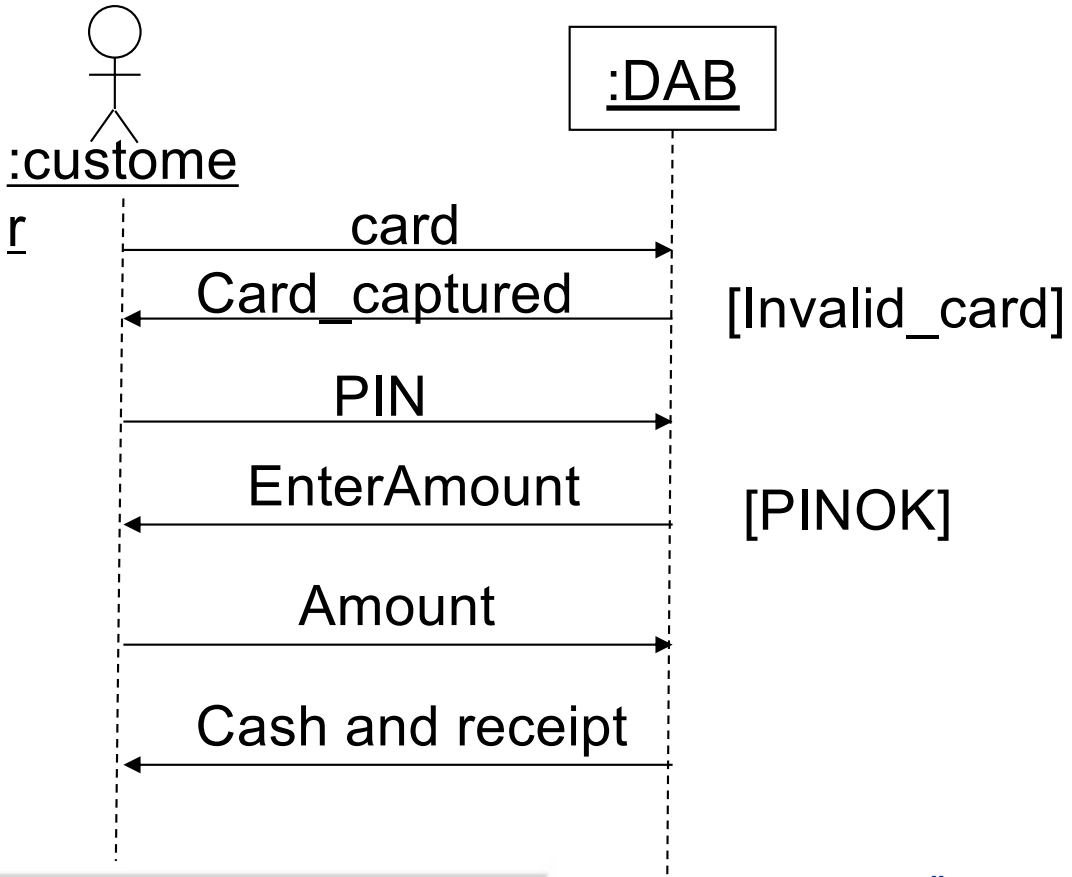
- Modélisent un service rendu par le système à un acteur particulier (acteur principal), peuvent faire intervenir d'autres acteurs acteurs secondaires
- Vues comme des « classes » dont les « instances » sont les scénarios implémentant ce service



Documenter les use cases

- Capture des besoins
  - Identifier les cas d'utilisation
  - Documenter les cas d'utilisation
  - Organiser les cas d'utilisation
  - Identifier les classes candidates

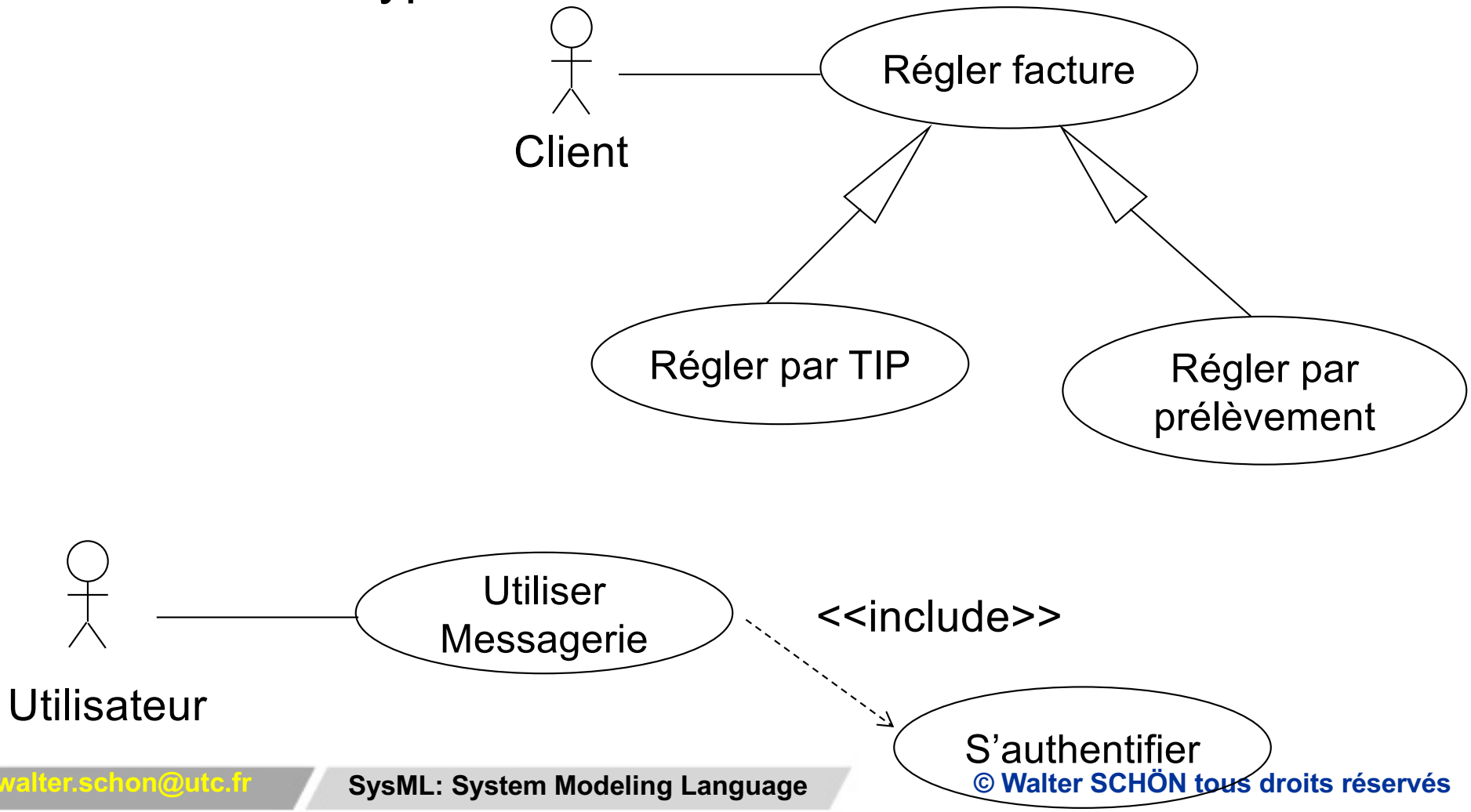
- Présenter les scenarios « instances » (nominaux et d'exception) généralement par diagrammes de séquences



Documenter les use cases

- Capture des besoins
  - Identifier les cas d'utilisation
  - Documenter les cas d'utilisation
  - Organiser les cas d'utilisation
  - Identifier les classes candidates

- Utiliser l'héritage entre acteurs ou use cases ainsi que des relations type inclusion :



Documenter les use cases

- Capture des besoins
  - Identifier les cas d'utilisation
  - Documenter les cas d'utilisation
  - Organiser les cas d'utilisation
  - Identifier les classes candidates

- Pour chaque use case identifier concepts objet par un diagramme des classes participantes

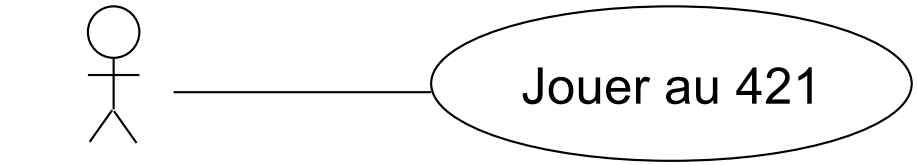
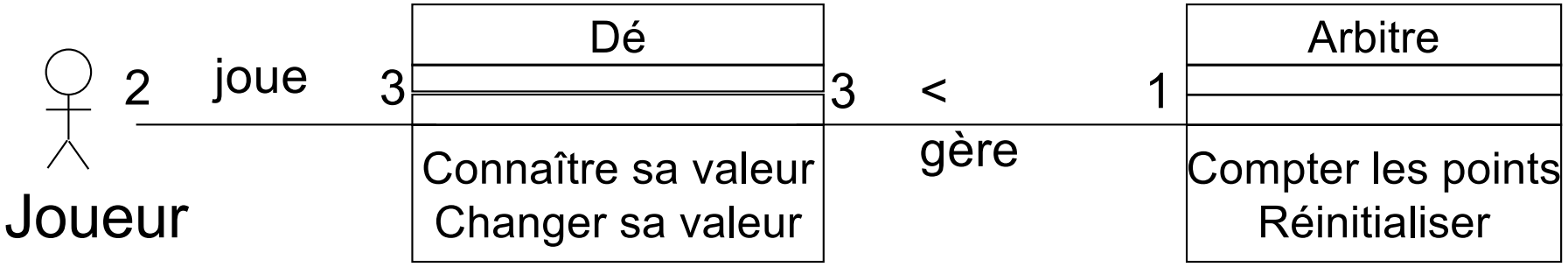


Diagramme des classes participantes



Catégories (Packages)

- Permet de découper un modèle complexe en sous-modèles en utilisant la notion UML/SysML de package.
- Package (catégorie) regroupement de (maximum 10) classes candidates sémantiquement voisines, fortement reliées entre elles plus faiblement reliées aux autres
- Se concentrer sur les dépendances entre catégories (une catégorie doit importer celles dont elle dépend)
- Objectif : catégories peu dépendantes afin d'aboutir à un découpage efficace des développements en parallèle

Modèle statique

- A ce stade (seulement) se développe le modèle statique autour du diagramme de classes :
- Affinage des classes à partir des classes candidates
- Affinage des associations (multiplicité, agrégations, compositions)
- Ajout des attributs (caractérisent l'état) et des méthodes (verbes d'actions)
- Optimiser avec la généralisation
- Ajout de contraintes (OCL…) le plus possible

Modèle dynamique

- Construit en parallèle du modèle statique
- Affinage des scénarios identifiés en capture des besoins
- Utilise diagrammes de séquences ou de communication
- Présentent un ensemble représentatif de scénarios (exhaustivité bien sûr impossible)
- Plusieurs petits scénarios (présentant plusieurs cas), à préférer à un gros scénarios (avec branchements)
- Peut être complété par la dynamique locale à un objet (diagrammes d'activités ou diagrammes d'états)
- Confronter systématiquement modèle statique et modèle dynamique qui doivent rester cohérents.

Conception préliminaire

- Aussi appelée conception architecturale
- Utilise les diagrammes de composants et diagrammes de déploiement pour mettre en évidence l'architecture logicielle voire matérielle
- Phase où se conçoit aussi l'aspect IHM et les classes associées

Conception détaillée

- Phase ultime, c'est là que se conçoivent les détails de l'implémentation des classes (ou blocs)
  - Types d'attributs compatibles avec le langage objet
  - Algorithme des méthodes
  - Utilisation de briques sur étagère (design patterns…)
- Toutes choses auxquelles on a tendance à s'intéresser bien trop tôt !
- Pour en savoir plus : les process objets sont traités de manière plus détaillée dans l'UV LO23 : gestion de projets informatiques.