Master MIDO

2ème année

Spécification et Conception en UML

Maude Manouvrier

- Spécifications initiales
- Analyse
- Conception du système
- Conception des classes

Bibliographie

- ***Modélisation et conception orientées objet avec UML2*** **de Michael Blaha et James Rumbaugh, 2ème édition, Pearson Education France, 2005 – Traduction de l'ouvrage *Applying Object-Oriented Modeling and Design with UML*, Prentice Hall 2005**

- ***The Unified Modeling Language Reference Manual,*** **2nd Edition de James Rumbaugh, Ivar Jacobson et Grady Booch, Addison Wesley Professional, 2004 – Traduction française : *UML 2.0, Guide de Référence*, CampusPress**

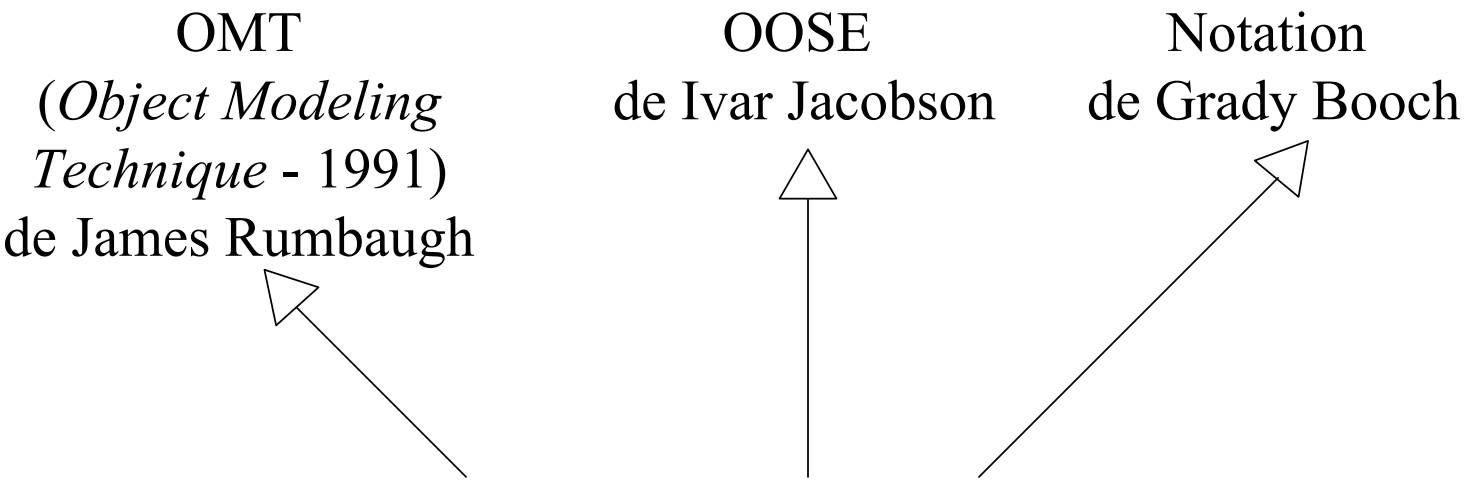
- ***Le guide de l'utilisateur UML*** **de Grady Booch, James Rumbaugh et Ivar Jacobson, Eyrolles, 2000 – Traduction de l'ouvrage *The Unified Modeling Language User Guide;* Addison-Wesley, 1998**

- ***UML 2 par la pratique – Etudes de cas et exercices corrigés*** **de Pascal Roques, 4ème Édition, Eyrolles, 2005**

- **Transparents de cours de Robert Ogor : *http://www-inf.int-evry.fr/COURS/IO21/COURS/CoursENSTBr.pdf***

- **Transparents de cours de Marie-José Blin**

Historique



UML - *Unified Modeling Language*

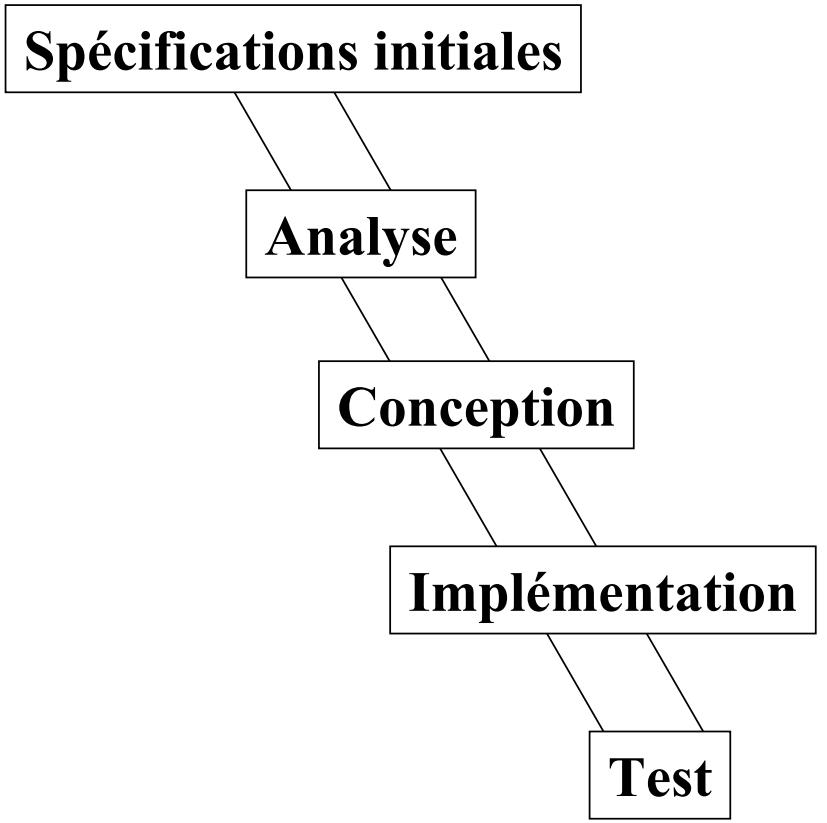
Standard de modélisation objet
adopté en 1997 par l'*Object Management Group* (OMG)

- Révision des spécifications initiales en 2001 – UML 1
- Approbation de la version **UML 2.0** en 2004

Stades de développement (1/4)

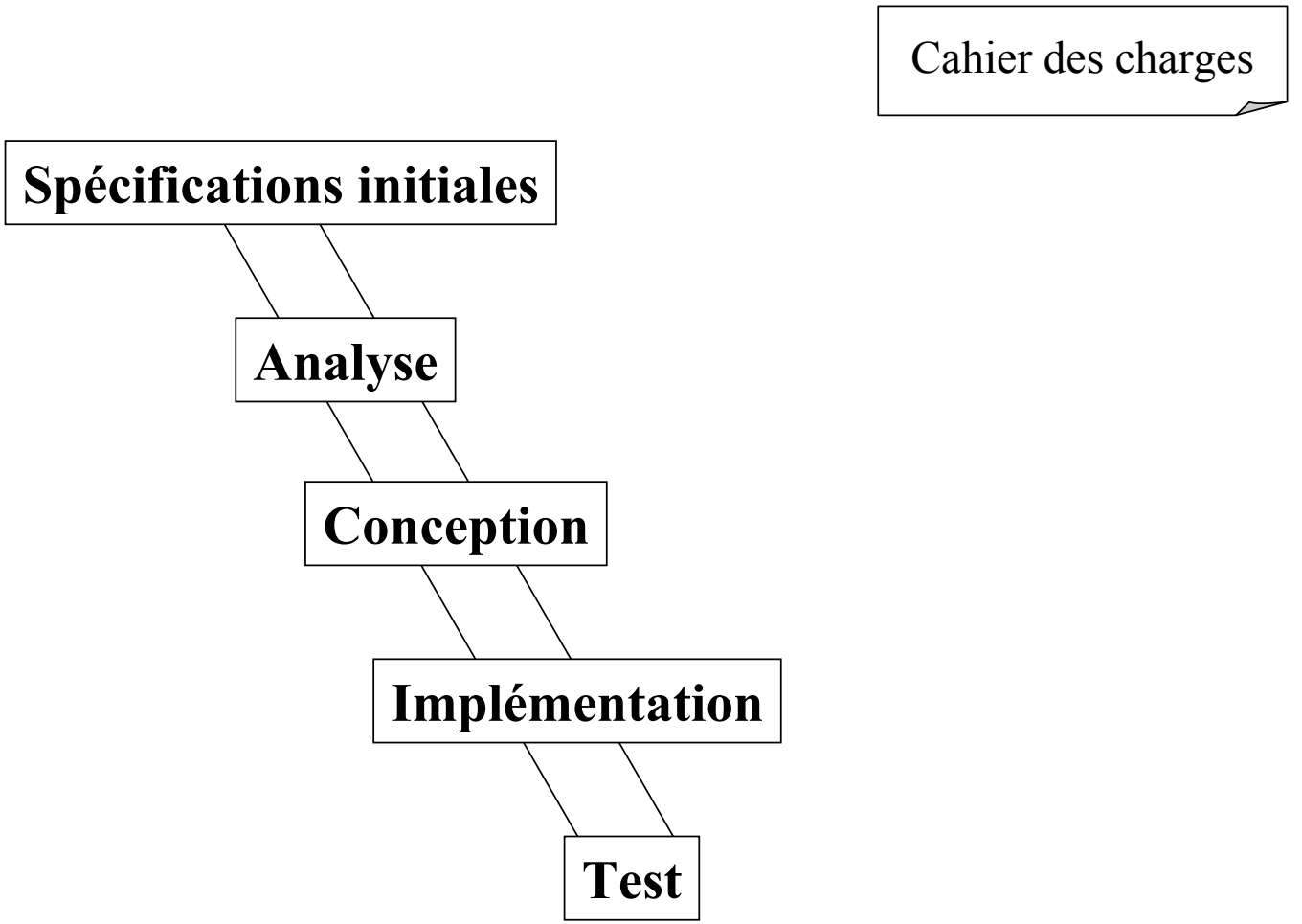
- **Spécifications initiales du système** : définition et formulation des exigences provisoires
- **Analyse** : Compréhension en profondeur des exigences à partir de la construction de modèles
- **Conception du système** : Mise au point de l'architecture du système en instaurant les politiques de conception des classes
- **Conception des classes** :
  - Augmentation et ajustement des modèles du monde réel issus de l'analyse en vue d'une compatibilité avec une implémentation informatique
  - Détermination des algorithmes des opérations
- **Implémentation** : Traduction de la conception en code
- **Test** : Vérification du bon fonctionnement de l'application

Stades de développement (2/4)



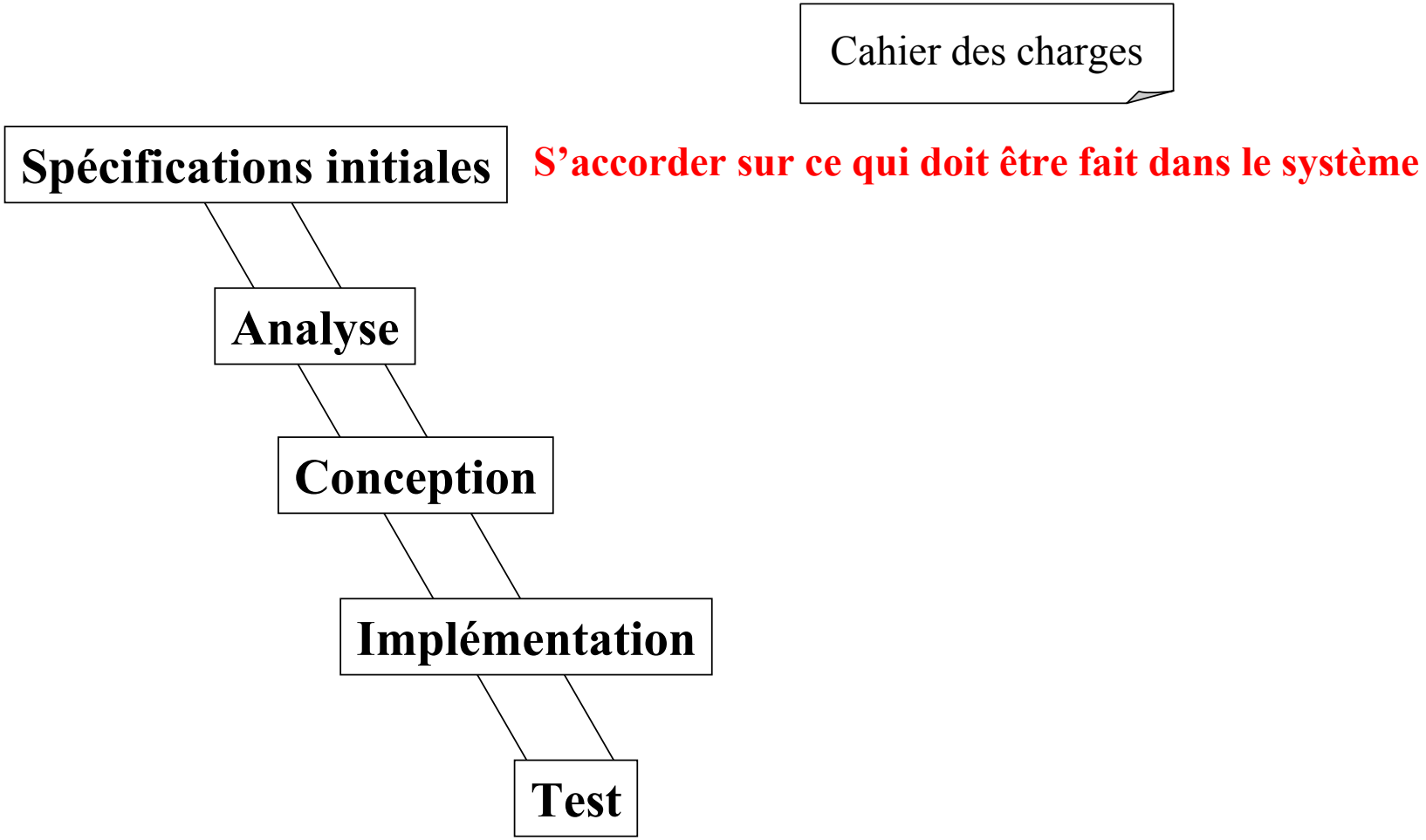
Repris de http://www-inf.int-evry.fr/COURS/IO21/COURS/CoursENSTBr.pdf

Stades de développement (2/4)



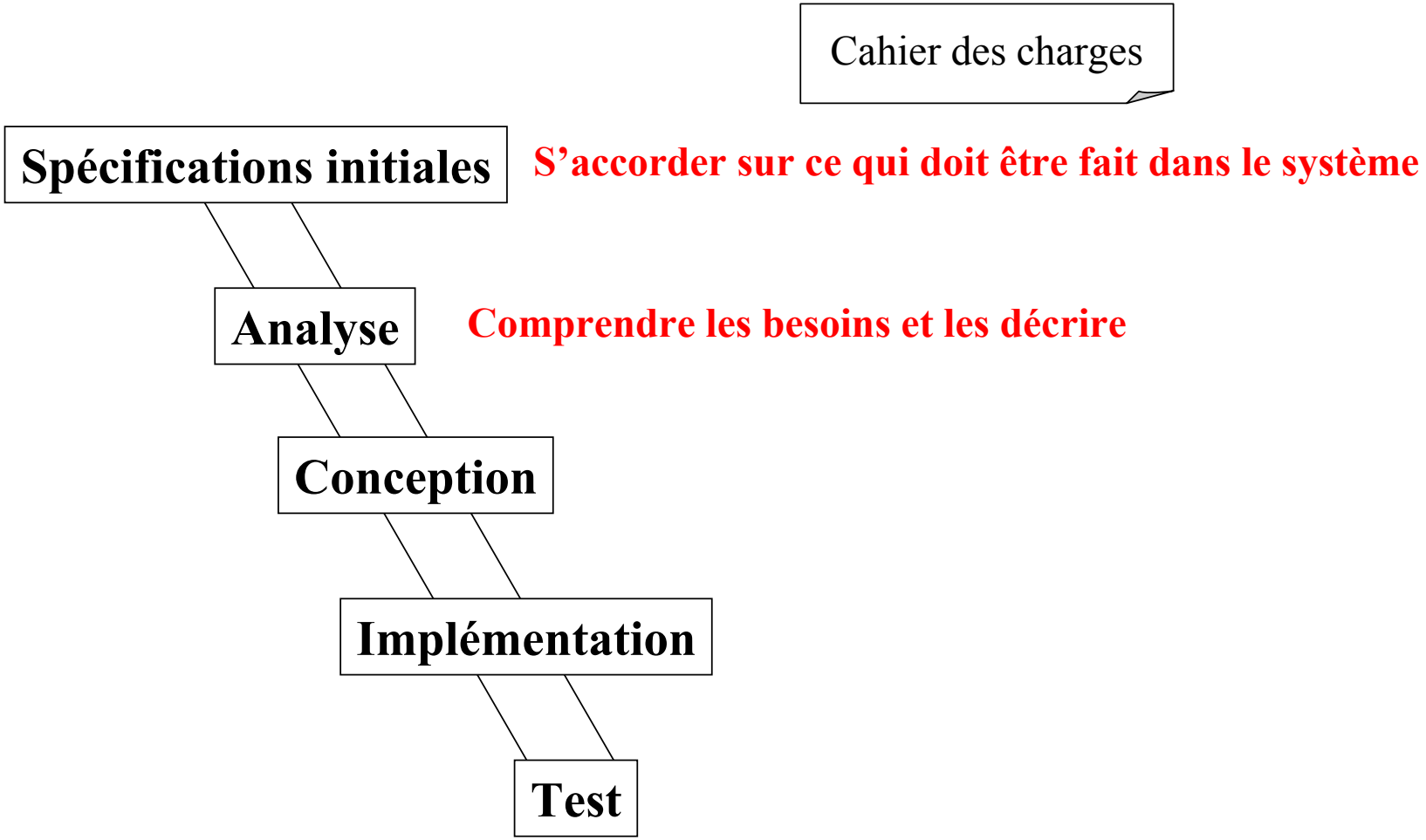
Repris de http://www-inf.int-evry.fr/COURS/IO21/COURS/CoursENSTBr.pdf

Stades de développement (2/4)



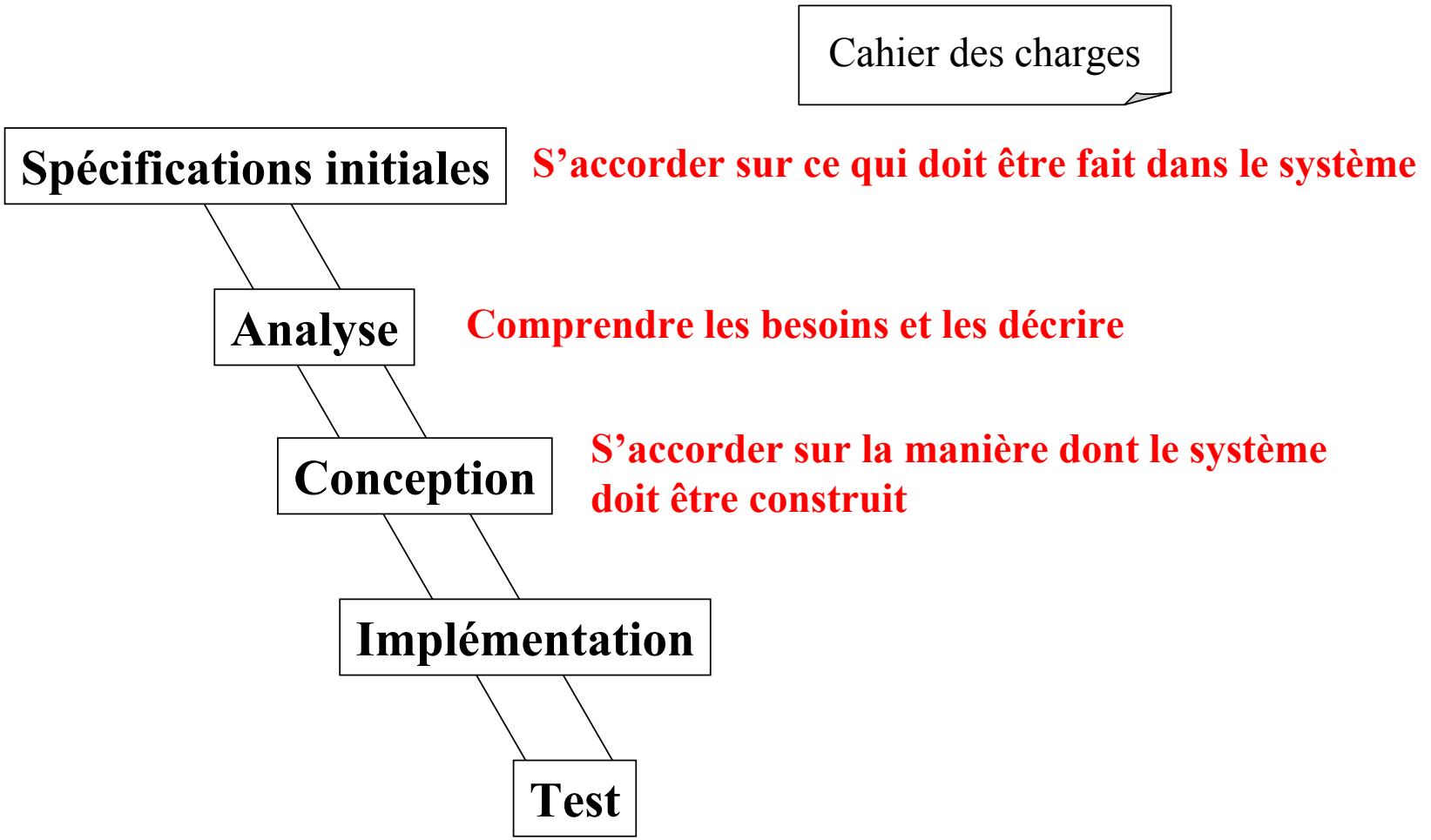
Repris de http://www-inf.int-evry.fr/COURS/IO21/COURS/CoursENSTBr.pdf

Stades de développement (2/4)



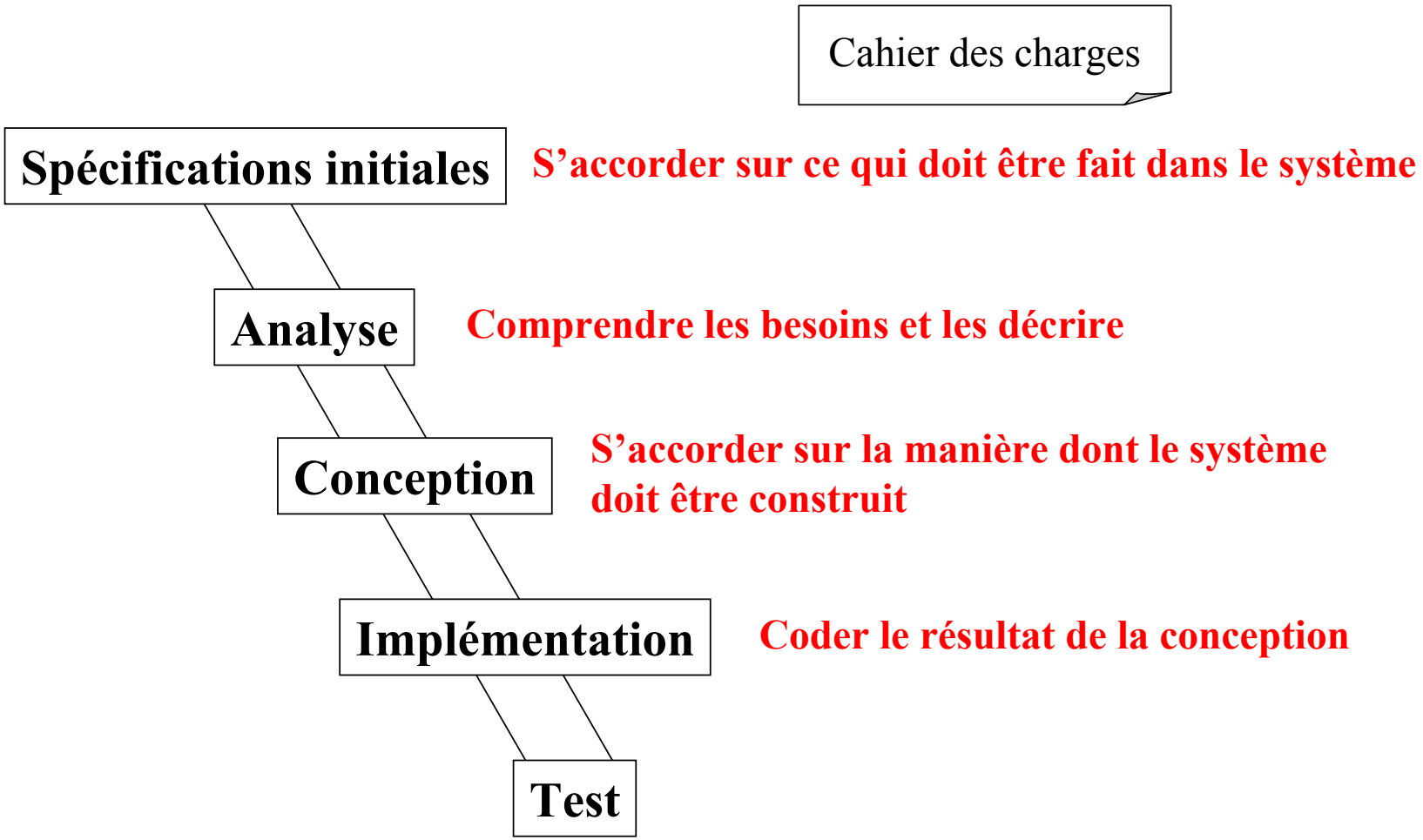
Repris de http://www-inf.int-evry.fr/COURS/IO21/COURS/CoursENSTBr.pdf

Stades de développement (2/4)



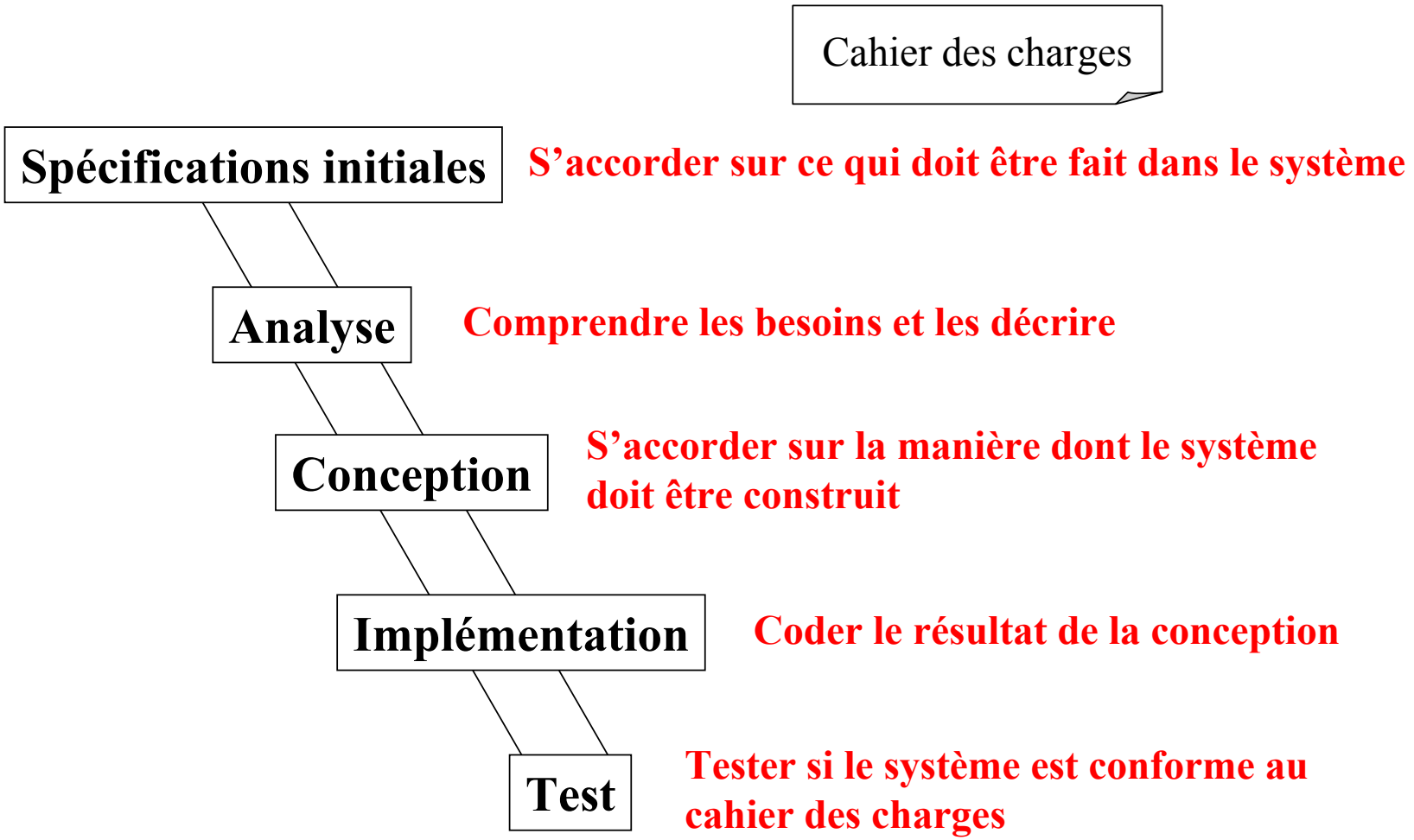
Repris de http://www-inf.int-evry.fr/COURS/IO21/COURS/CoursENSTBr.pdf

Stades de développement (2/4)



Repris de http://www-inf.int-evry.fr/COURS/IO21/COURS/CoursENSTBr.pdf

Stades de développement (2/4)



Repris de http://www-inf.int-evry.fr/COURS/IO21/COURS/CoursENSTBr.pdf

Stades de développement (3/4)

- Élaboration et optimisation des modèles en permanence
- Application des mêmes concepts et de la même notation tout au long du processus mais avec des **changements de points de vue**
  - Premiers stades axés sur les exigences métier
  - Stades ultérieurs axés sur les ressources informatiques
- Grande partie de l'effort pour l'**analyse** et la **conception**

Stades de développement (4/4)

Plusieurs styles de cycle de vie :

- **Développement en cascade :**
  - Séquence linéaire des différents stades
  - Pas de retour en arrière
  - Passage au stade suivant après la fin complète du stade précédent
  - Pour des applications bien comprises avec des exigences bien stabilisées et des résultats d'analyse et de conception prévisibles
  - Pas de livraison d'un système utilisable avant la finalisation complète du système
- **Développement itératif :**
  - Développement en cascade du noyau du système
  - Élargissement du périmètre du système par ajout successif de propriétés et de comportement aux objets existants et de nouveaux types d'objets
  - Plusieurs itérations avant le livrable final – chaque itération comprenant un ensemble complet de stades
  - Pas de construction du système dans son intégralité en une seule fois
  - Valable pour la plupart des applications

Spécifications initiales (1/3)

- Objectif : se forger une idée globale du système en différant les détails
- Questions à se poser [BR05] :
  - A qui l'application est-elle destinée ?
  - Quels problèmes l'application résoudra-t-elle ?
  - Quelles seront les conditions d'utilisation de l'application ?
  - Quand l'application est-elle attendue ?
  - Pourquoi l'application est-elle attendue ?
  - Comment l'application fonctionnera-t-elle ?

Spécifications initiales (2/3)

- **Exigences** : description de la façon dont un système se comporte du **point de vue utilisateur**
- Système = boîte noire dont seul le comportement externe importe
- Rédaction d'un **énoncé des exigences**
  - Exigences : souvent ambiguës, incomplètes voire incohérentes, parfois fausses
  - Énoncé : **Point de départ et moyen de mieux comprendre le problème** mais document non immuable

Spécifications initiales (3/3)

« Préciser ce qui doit être réalisé et non comment l'implémentation doit le réaliser » [BR05] :

Spécifications initiales (3/3)

« Préciser ce qui doit être réalisé et non comment l'implémentation doit le réaliser » [BR05] :

Exigences

- Périmètre du système
- Définition de ce qui est exigé
- Contexte de l'application
- Hypothèses
- Besoins de performances

Spécifications initiales (3/3)

« Préciser ce qui doit être réalisé et non comment l'implémentation doit le réaliser » [BR05] :

Exigences

- Périmètre du système
- Définition de ce qui est exigé
- Contexte de l'application
- Hypothèses
- Besoins de performances

Conception

- Approche générale
- Algorithmes
- Structures de données
- Architecture
- Optimisation
- Planification des ressources

Spécifications initiales (3/3)

« Préciser ce qui doit être réalisé et non comment l'implémentation doit le réaliser » [BR05] :

Exigences

- Périmètre du système
- Définition de ce qui est exigé
- Contexte de l'application
- Hypothèses
- Besoins de performances

Conception

- Approche générale
- Algorithmes
- Structures de données
- Architecture
- Optimisation
- Planification des ressources

Implémentation

- Plates-formes
- Spécifications matérielles
- Bibliothèques logicielles
- Standards d'interface

Spécifications initiales (3/3)

« Préciser ce qui doit être réalisé et non comment l'implémentation doit le réaliser » [BR05] :

Exigences

- Périmètre du système
- Définition de ce qui est exigé
- Contexte de l'application
- Hypothèses
- Besoins de performances

Conception

- Approche générale
- Algorithmes
- Structures de données
- Architecture
- Optimisation
- Planification des ressources

Implémentation

- Plates-formes
- Spécifications matérielles
- Bibliothèques logicielles
- Standards d'interface

« Ne pas prendre trop tôt de décisions de conception ou d'implémentation »

Exemple traité en cours

Système de gestion de demandes de formation

En vue de l'amélioration de son système d'information, souhait d'une entreprise de modéliser le processus de formation des ses employés afin d'automatiser certaines tâches

- Initialisation du processus de formation à la réception d'une demande de formation par le responsable formation de la part d'un employé. Analyse de la demande par le responsable et transmission de l'accord ou du désaccord à l'intéressé.
- En cas d'accord,
  - Recherche par le responsable de formation, dans le catalogue des formations agréées, d'un stage correspondant à la demande.
  - Transmission à l'employé demandeur du contenu de la formation correspondant à la demande et du planning des sessions.
  - Après validation auprès de l'employé, inscription auprès de l'organisme de formation de l'employé par le responsable à la session de formation choisie.
- En cas d'empêchement de l'employé, obligation de l'employé d'informer le responsable au plut tôt pour annuler l'inscription ou la demande.
- A la fin de la formation, remise par le participant au responsable d'une fiche d'appréciation de la formation et d'un document justifiant sa présence au cours de la formation.
- Contrôle, par la responsable, de la facture envoyée par l'organisme de formation avant transmission au service comptable.

Analyse

- **« Étape du développement où on examine le problème réel pour comprendre ses besoins sans planifier l'implémentation » [BR05]**

- Concentration sur la création de modèles

- **Analyse du domaine**

  ⇒ Modèles du domaine

- **Analyse de l'application**

  ⇒ Modèles de l'application

Analyse du domaine (1/21)

- **Objectif** : obtenir un modèle précis, concis, compréhensible et correct du monde réel
- Début d'une compréhension plus claire des exigences
- Mise en évidence des ambiguïtés et des incohérences de l'énoncé du problème
- Abstraction des caractéristiques essentielles de l'énoncé du problème dans un modèle

Analyse du domaine (2/21)

Modèle de classes du domaine

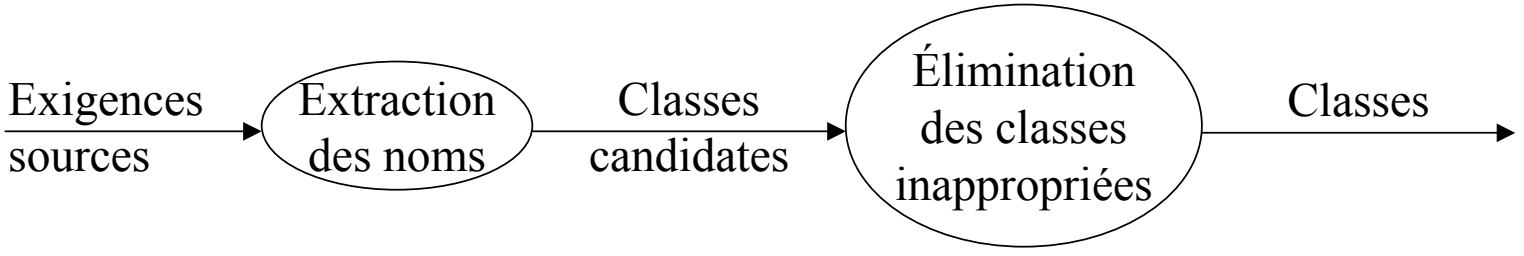
- Description des classes du monde réel et de leurs relations
- Étapes à suivre [BR05] :
  1. **Identifier les classes et conserver les classes pertinentes**
  2. **Préparer un dictionnaire de données**
  3. **Identifier les associations et conserver les associations pertinentes**
  4. **Identifier les attributs des objets et les liens**
  5. **Organiser et simplifier les classes en utilisant l'héritage**
  6. **Vérifier que tous les chemins d'accès existent pour les requêtes probables**
  7. **Itérer et affiner le modèle**
  8. **Réexaminer le niveau d'abstraction**
  9. **Regrouper les classes en *package***

Analyse du domaine (3/21)

1. **Identifier les classes** : trouver les classes pertinentes pour les objets du domaine de l'application

Conseils de [BR05] :

- Au départ, ne pas être trop sélectif et noter tout ce qui vient à l'esprit
- Ne pas se soucier trop de l'héritage ni des classes de haut niveau



Analyse du domaine (4/21)

1. **Identifier les classes :** cas du système de gestion de demandes de formation

Système d'information | Processus de formation | Responsable de formation

Initialisation | Employé | Demande de formation | Analyse | Recherche

Accord | Désaccord | Intéressé | Catalogue de formations agréées | Stage

Transmission | Contenu de formation | Planning | Session

Validation | Responsable | Inscription | Organisme de formation

Obligation | Empêchement | Annulation | Remise

Participant | Appréciation de stage | Justificatif de présences | Facture

Contrôle | Service Comptable

Analyse du domaine (5/21)

1. Conserver les classes pertinentes

Par **élimination** des :

- **Classes redondantes** : classes exprimant le même concept / Conservation du nom le plus évocateur
- **Classes sans intérêt** : classe sans lien avec le contexte ou ne faisant pas partie du périmètre du logiciel
- **Classes vagues** : classes ayant des frontières mal définies ou une portée trop large
- **Attributs** : re-formulation des noms décrivant originellement des objets individuels sous la forme d'attributs
- **Opérations** : appliquées à des objets et non manipulées en tant qu'opérations
- **Rôles** : reflet de la nature intrinsèque d'une classe par son nom et non du rôle de la classe dans une association
- **Éléments d'implémentation** : éléments étrangers au monde réel
- **Classes dérivées** ◁

Analyse du domaine (6/21)

2. **Préparer un dictionnaire de données** : pour éviter les trop nombreuses interprétations
   - Décrire précisément chaque classe par un paragraphe
   - Décrire la portée de chaque classe dans le problème courant, en incluant toutes les hypothèses et les restrictions quant à son utilisation [BR05]
   - Décrire également les attributs, associations, opérations et valeurs énumérées

Analyse du domaine (6/21)

2. **Préparer un dictionnaire de données** : pour éviter les trop nombreuses interprétations
   - Décrire précisément chaque classe par un paragraphe
   - Décrire la portée de chaque classe dans le problème courant, en incluant toutes les hypothèses et les restrictions quant à son utilisation [BR05]
   - Décrire également les attributs, associations, opérations et valeurs énumérées

Demande de formation : Email envoyé par un employé au responsable de formation, précisant un thème, une formation ou une session particulière de formation issues du catalogue. La demande est enregistrée par la système après transmission au responsable de formation.

Analyse du domaine (7/21)

3. Identifier les associations et conserver les associations pertinentes

Par **élimination** des :

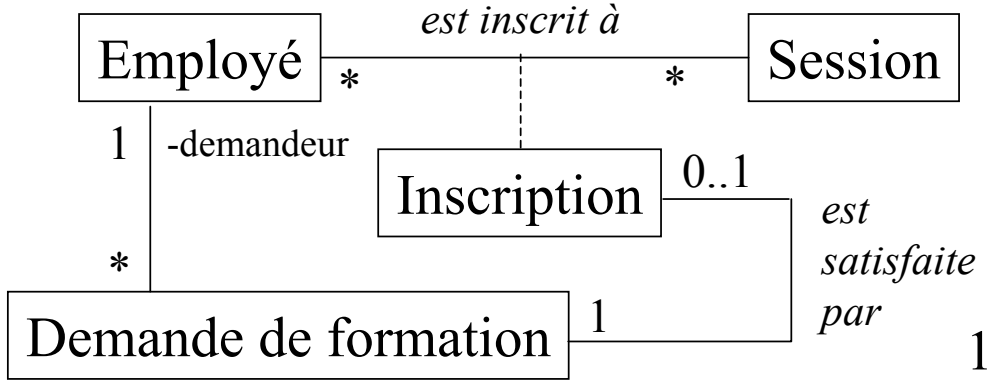
- **Associations non pertinentes ou relevant de l'implémentation**
- **Actions**
- **Associations ternaires** par décomposition associations binaires, associations qualifiées ou classes-associations
- **Associations dérivées** : associations définies en termes d'autres associations (car redondance)

Analyse du domaine (7/21)

3. Identifier les associations et conserver les associations pertinentes

Par **élimination** des :

- **Associations non pertinentes ou relevant de l'implémentation**
- **Actions**
- **Associations ternaires** par décomposition associations binaires, associations qualifiées ou classes-associations
- **Associations dérivées** : associations définies en termes d'autres associations (car redondance)



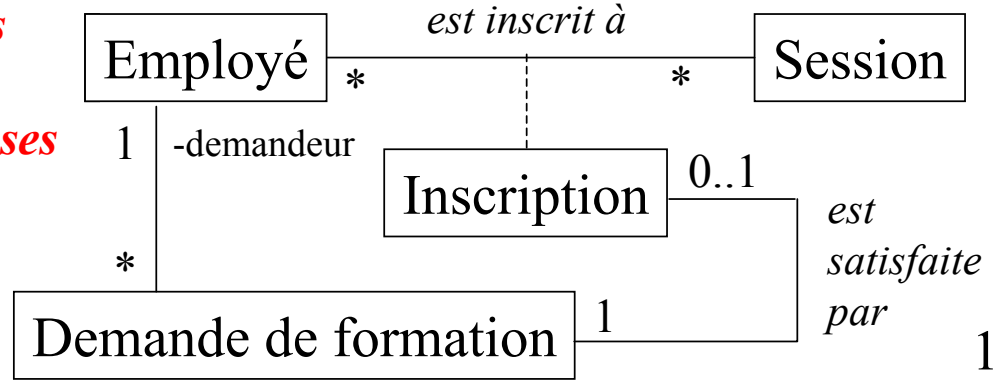
Analyse du domaine (7/21)

3. Identifier les associations et conserver les associations pertinentes

Par **élimination** des :

- **Associations non pertinentes ou relevant de l'implémentation**
- **Actions**
- **Associations ternaires** par décomposition associations binaires, associations qualifiées ou classes-associations
- **Associations dérivées** : associations définies en termes d'autres associations (car redondance)

Toutes les associations formant plusieurs chemins entre des classes n'indiquent pas une redondance



Analyse du domaine (8/21)

3. Identifier les associations et conserver les associations pertinentes (suite)

Précision de la sémantique des associations :

- Éviter les **associations mal nommées :** choix des noms primordial pour la compréhension
- Indiquer les **noms d'extrémités d'associations**
- Identifier les **associations qualifiées**

- Préciser les **multiplicités**
- Ajouter les **associations manquantes**
- Transformer certaines associations en **agrégations**

Analyse du domaine (9/21)

4. Trouver les attributs et conserver les attributs pertinents

Conseils de [BR05] :

- Ne pas pousser la recherche des attributs à l'extrême.
- Ne considérer que les attributs pertinents pour l'application
- Rechercher en premier les attributs les plus importants ; repousser les détails à plus tard
- Omettre les attributs dérivés
- Rechercher les attributs des associations

Élimination des attributs inutiles ou incorrects selon les critères suivants [BR05] :

- Différencier les **objets** de leurs **valeurs**
- Utiliser des **qualificateurs**
- **Ne pas préciser les attributs identificateurs exceptés ceux du domaine de l'application**
- Éliminer les valeurs internes et les détails fins

Analyse du domaine (9/21)

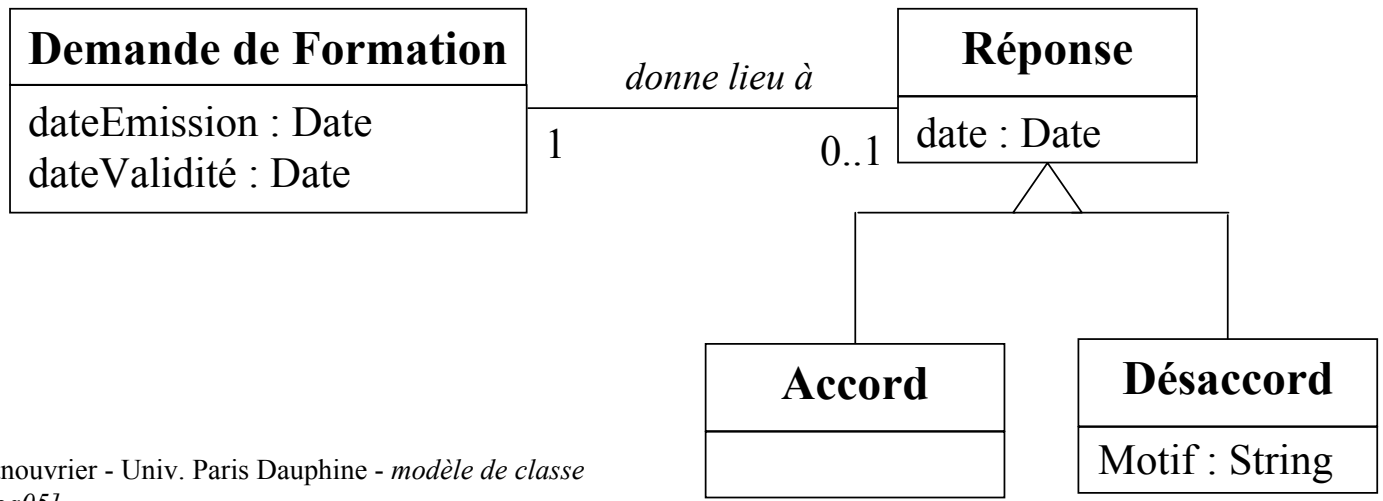
5. Organiser et simplifier les classes en utilisant l'héritage

- Par **généralisation ascendante** : en recherchant les classes ayant des attributs, des associations ou des opérations similaires
- Par **spécialisation descendante** : en évitant les raffinements excessifs
- **Généralisation vs. Énumération** ◁

Analyse du domaine (9/21)

5. Organiser et simplifier les classes en utilisant l'héritage

- Par **généralisation ascendante** : en recherchant les classes ayant des attributs, des associations ou des opérations similaires
- Par **spécialisation descendante** : en évitant les raffinements excessifs
- **Généralisation vs. Énumération**



Analyse du domaine (10/21)

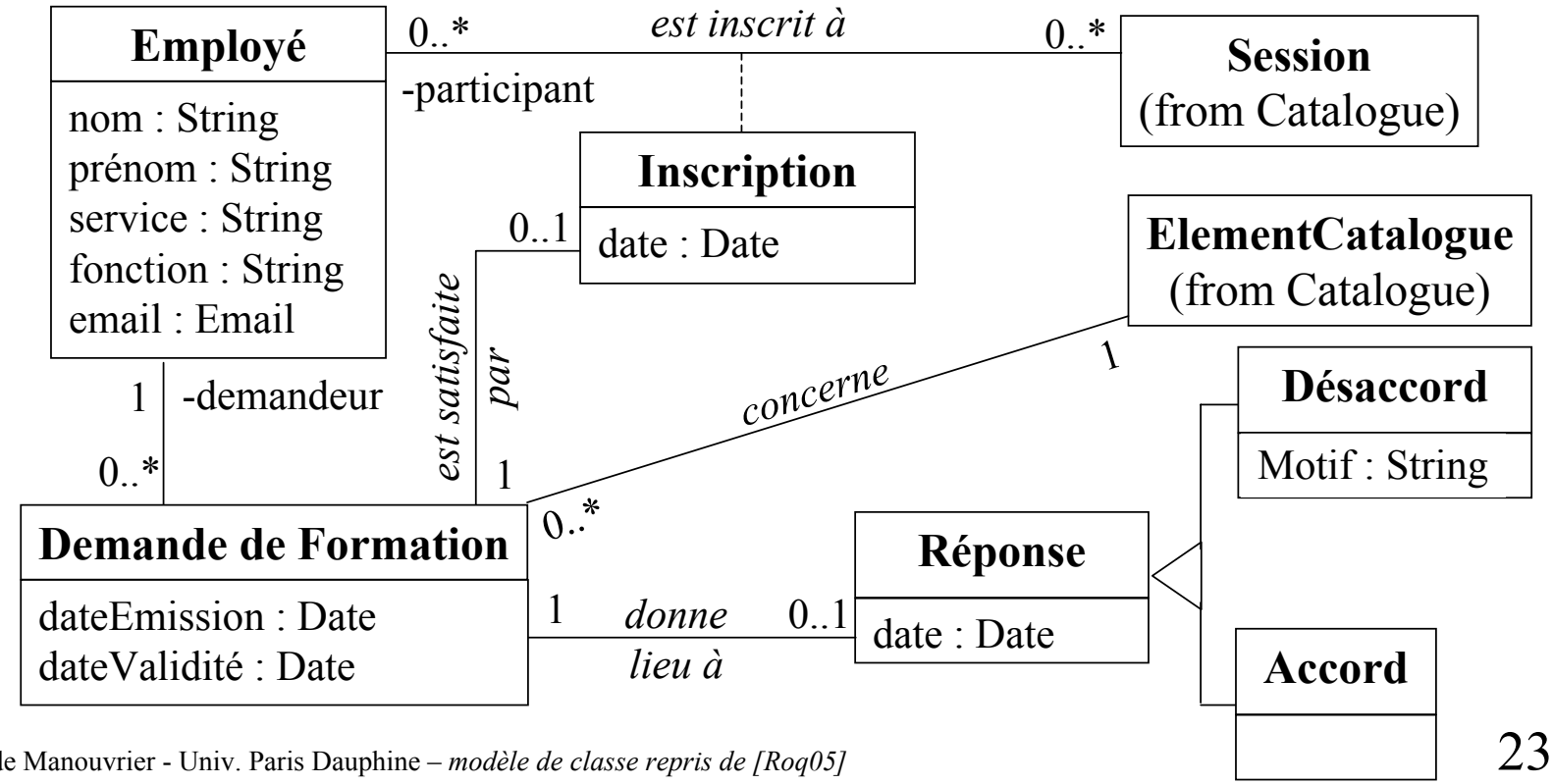
6. Tester les chemins d'accès

- Penser aux questions souhaitées
- Vérifier qu'il n'y a aucune question utile sans réponse

Analyse du domaine (10/21)

6. Tester les chemins d'accès

- Penser aux questions souhaitées
- Vérifier qu'il n'y a aucune question utile sans réponse



Analyse du domaine (11/21)

7. Itérer sur le modèle de classes [BR05]

- Pas d'exactitude du modèle à la première passe
- Nécessité de réaliser des itérations continuelles
- Possibilité d'avoir différentes parties du modèle à différents stades d'achèvement
- Possibilité de revenir à un stade antérieur pour corriger une anomalie détectée
- Raffinements probables après l'achèvement des modèles d'états et d'interactions

Analyse du domaine (12/21)

7. Itérer sur le modèle de classes [BR05] (suite)

- **Recherche des classes manquantes**
  - **Scission de classe** ayant
    - Des attributs et des opérations disparates
    - Plusieurs rôles
  - **Ajout de classe**
    - Création de super-classe en cas d'associations de même but et de même nom
    - Transformation d'une association en classe en cas d'association ayant un rôle façonnant substantiellement la sémantique d'une classe
- **Recherche d'associations manquantes** en cas de chemins d'accès manquants pour répondre aux requêtes
- **Recherche des éléments superflus**
  - Classes avec peu d'attributs, d'opérations ou d'associations
  - Associations redondantes
- **Ajustement des attributs et des associations**
  - Déplacement d'une association en cas de noms d'extrémités d'association avec un sens trop large ou trop étroit
  - Transformation d'une association en association qualifiée en cas de nécessité d'accès à un objet par l'intermédiaire de la valeur d'un attribut

Analyse du domaine (13/21)

8. Élever le niveau d'abstraction

- Après une prise en compte de l'énoncé « au pied de la lettre », nécessité d'élever le niveau d'abstraction pour :
  - Augmenter la souplesse du modèle
  - Réduire le nombre de classes
- Amélioration du modèle par l'**utilisation de patterns d'analyse**

9. Grouper les classes en *packages*

Regroupement des éléments (classes, associations, généralisation et autres packages de niveau inférieur) ayant un thème commun ou fortement couplés afin de plus facilement tracer et visualiser le modèle

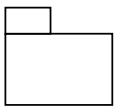
Analyse du domaine (13/21)

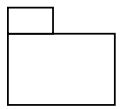
8. Élever le niveau d'abstraction

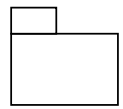
- Après une prise en compte de l'énoncé « au pied de la lettre », nécessité d'élever le niveau d'abstraction pour :
  - Augmenter la souplesse du modèle
  - Réduire le nombre de classes
- Amélioration du modèle par l'**utilisation de patterns d'analyse**

9. Grouper les classes en *packages*

Regroupement des éléments (classes, associations, généralisation et autres packages de niveau inférieur) ayant un thème commun ou fortement couplés afin de plus facilement tracer et visualiser le modèle







	Comptabilité	**Demandes de formation**	**Catalogue de formations**
Entités métier	Service comptable ; Facture	Demande de formation ; Employé ; Inscription ; Réponse ; Accord; Désaccord	Catalogue ; Organisme de formation ; Formation; Session

Analyse du domaine (14/21)

Modèle d'états du domaine

- Description du cycle de vie des objets de certaines classes
- Étapes à suivre [BR05] :

1. **Identifier les classes du domaine ayant des états**
2. **Trouver les états**
3. **Trouver les événements**
4. **Construire les diagrammes d'états**
5. **Évaluer les diagrammes d'états**

Analyse du domaine (15/21)

1. **Identifier les classes ayant des états**

   Recherche des classes ayant un cycle de vie distinct (classes caractérisées par une progression ou représentant un comportement cyclique)

2. **Trouver les états**

   Recherche des états à partir des différences qualitatives dans le comportement, les attributs ou les associations des classes dynamiques

Analyse du domaine (15/21)

1. **Identifier les classes ayant des états**

   Recherche des classes ayant un cycle de vie distinct (classes caractérisées par une progression ou représentant un comportement cyclique)

   Demande de formation

2. **Trouver les états**

   Recherche des états à partir des différences qualitatives dans le comportement, les attributs ou les associations des classes dynamiques

Analyse du domaine (15/21)

1. **Identifier les classes ayant des états**

   Recherche des classes ayant un cycle de vie distinct (classes caractérisées par une progression ou représentant un comportement cyclique)

   Demande de formation

2. **Trouver les états**

   Recherche des états à partir des différences qualitatives dans le comportement, les attributs ou les associations des classes dynamiques

Création **EnAttenteDeRéponse** **EnAttenteInscription**

Satisfaite **Réalisée**

Analyse du domaine (16/21)

3. Trouver les événements

- Recherche des événements déclenchant les transitions entre les états
- Capturer les informations véhiculées par un événement sous la forme d'une liste de paramètres

Analyse du domaine (16/21)

3. Trouver les événements

- Recherche des événements déclenchant les transitions entre les états
- Capturer les informations véhiculées par un événement sous la forme d'une liste de paramètres

valider / envoyer email au responsable

refuser / envoyer refus à l'employé

accepter / envoyer accord à l'employé

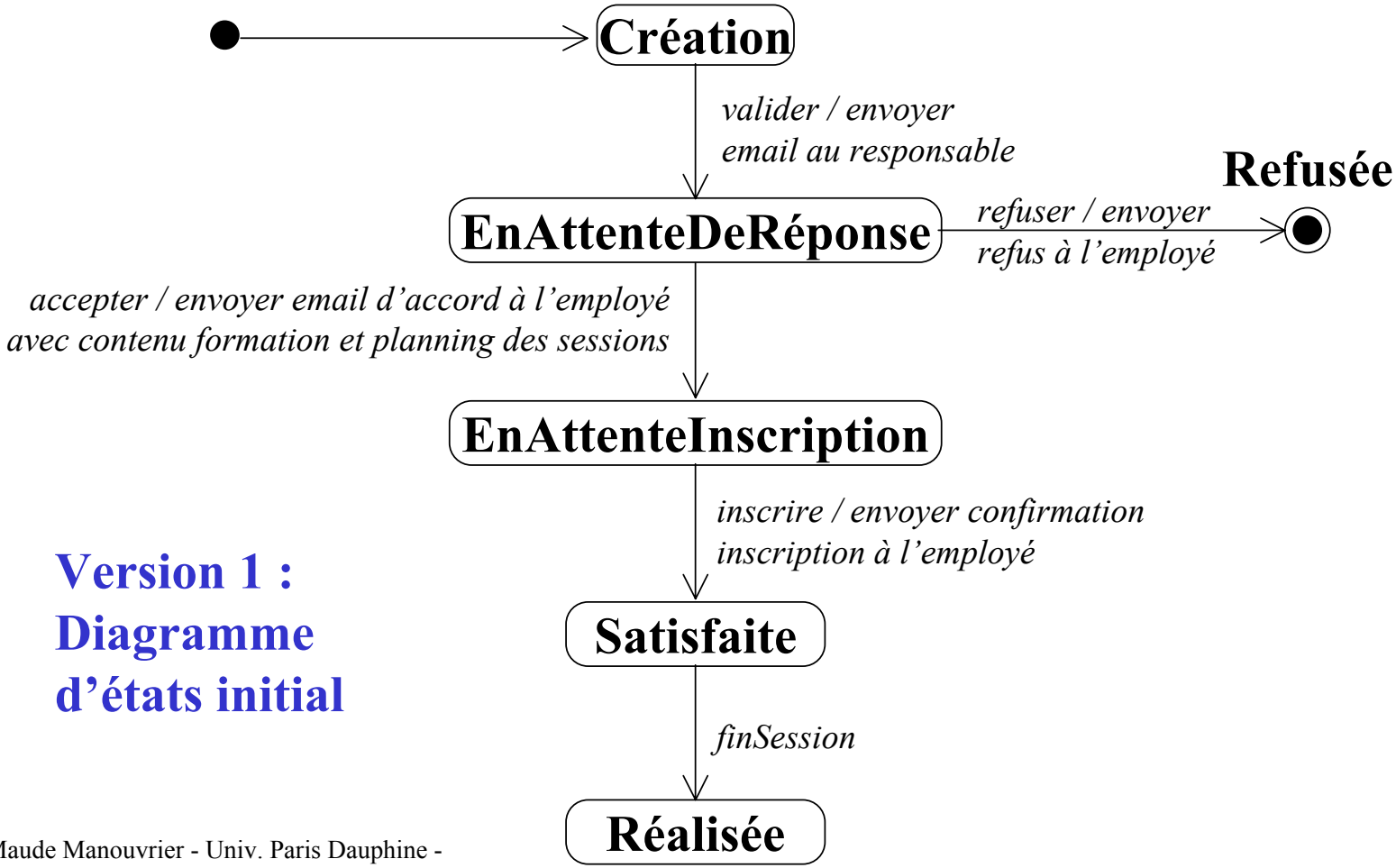
inscrire / envoyer confirmation inscription à l'employé

Analyse du domaine (17/21)

4. **Construire des diagrammes d'états** (exemple)

Analyse du domaine (17/21)

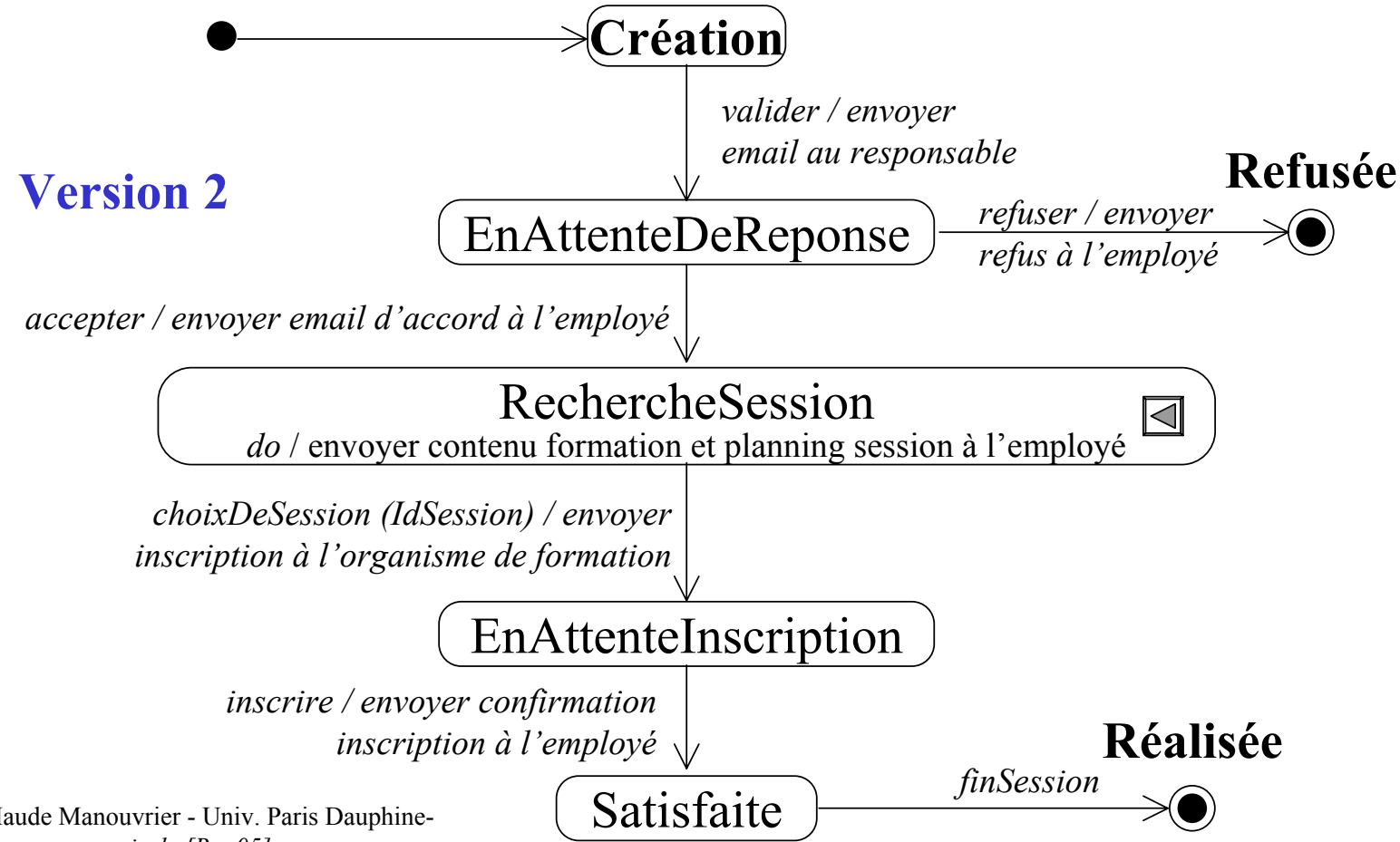
4. Construire des diagrammes d'états (exemple)



Version 1 : Diagramme d'états initial

Analyse du domaine (18/21)

4. Construire des diagrammes d'états (exemple)



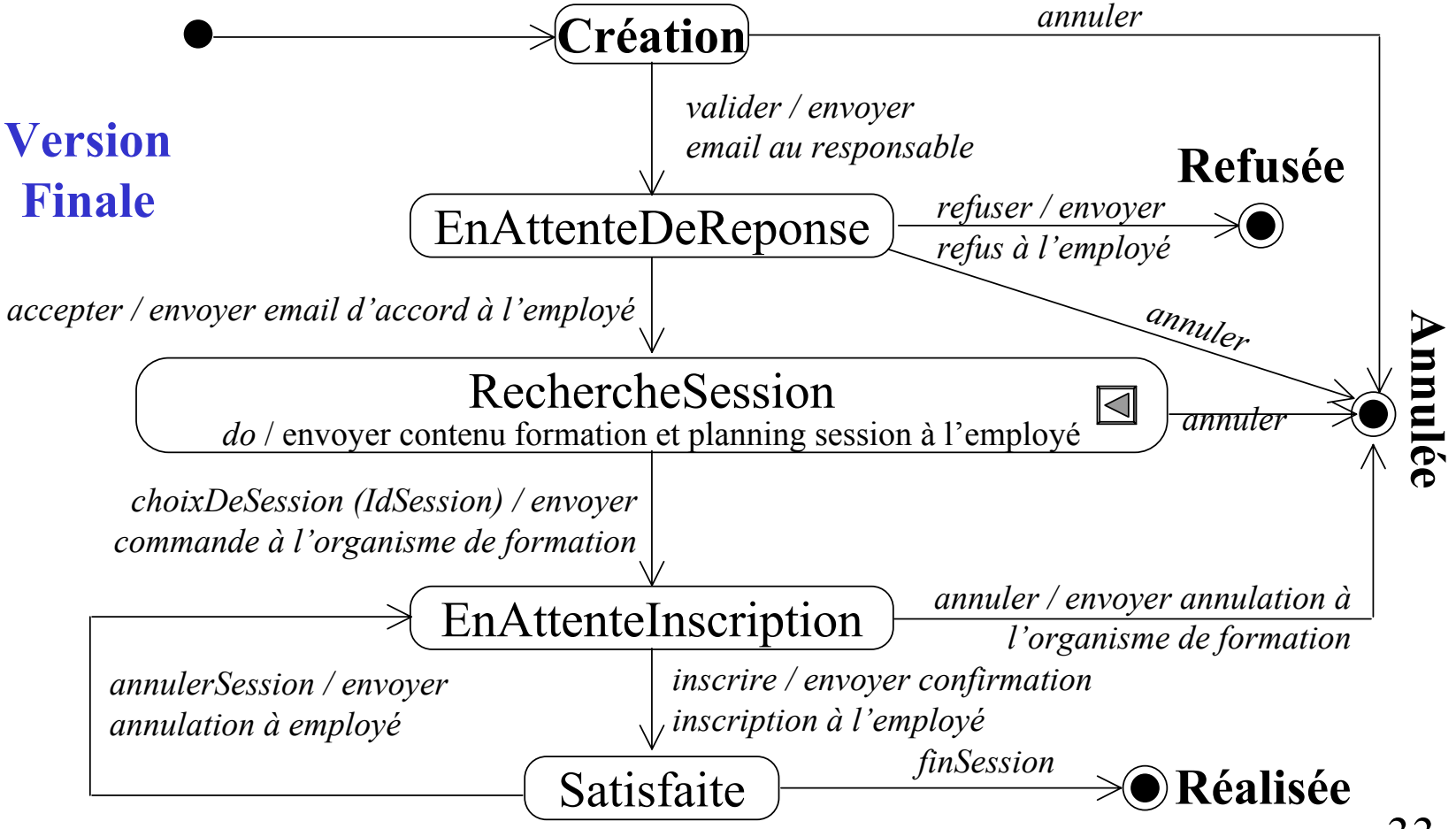
Analyse du domaine (19/21)

5. Évaluer les diagrammes d'états [BR05]

- Examiner chaque modèle d'états et vérifier la connexion de tous les états
- Prêter une attention particulière aux chemins d'accès (y-a-t-il un chemin conduisant de l'état initial à un état final ?)
- Vérifier que la présence des variations attendues
- En cas de classe cyclique, vérifier la présence de la boucle principale
- Ajouter les états manquants (identifiés par les chemins manquants)

Analyse du domaine (20/21)

5. Évaluer les diagrammes d'états (exemple)



Analyse du domaine (21/21)

- Modèle d'interactions : rarement intéressant pour l'analyse du domaine ◁
- Itération de l'analyse [BR05]
  - **Nécessité de réaliser plusieurs itérations** pour obtenir un modèle complet
  - Affinage de l'analyse au fur et à mesure de l'amélioration de la compréhension
  - Vérification du résultat avec le rédacteur de l'énoncé du problème et les experts du domaine
- Analyse vs. conception
  - Utilisation du modèle d'analyse comme base pour l'architecture, la conception et l'implémentation du système
  - Difficulté pratique pour éviter une « contamination » du modèle d'analyse par l'implémentation

Analyse de l'application (1/28)

- **Objectif** : ajouter les « artefacts majeurs » au modèle du domaine
- Concentration de l'attention sur les détails de l'application et prise en compte des interactions ◁
- Étapes à suivre **pour le modèle d'interactions** [BR05] :
  1. **Déterminer la frontière du système**
  2. **Trouver les acteurs**
  3. **Trouver les cas d'utilisation**
  4. **Trouver les événements initiaux et finaux**
  5. **Rédiger les scénarios standards**
  6. **Ajouter des scénarios de variations et d'exceptions**
  7. **Trouver les événements externes**
  8. **Préparer des diagrammes d'activités pour les cas d'utilisation complexes**
  9. **Organiser les acteurs et les cas d'utilisation**
  10. **Vérifier avec le modèle de classes du domaine**

Analyse de l'application (2/28)

1. Déterminer la frontière du système

- Nécessité de bien connaître le périmètre précis de l'application pour bien spécifier ses fonctionnalités
- Durant l'analyse : **détermination de la finalité du système et de son aspect pour les utilisateurs**

Analyse de l'application (2/28)

1. Déterminer la frontière du système

- Nécessité de bien connaître le périmètre précis de l'application pour bien spécifier ses fonctionnalités
- Durant l'analyse : **détermination de la finalité du système et de son aspect pour les utilisateurs**

> *Le système à développer doit gérer les demandes de formation des employés d'une entreprise. Les formations agréées sont disponibles dans un ou plusieurs catalogues. La gestion et la mise à jour des catalogues ne fait pas partie des fonctionnalités du système. De même, le suivi de la présence et du bon déroulement des sessions ne font pas partie du système. En revanche, le système doit permettre d'automatiser les actions suivantes : Rédiger une demande de formation; Évaluer une demande de formation; Chercher un stage; Inscrire un employé à un stage; Commander et payer un stage.*

Analyse de l'application (3/28)

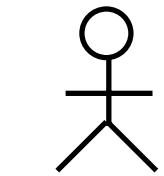
2. Trouver les acteurs

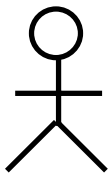
- Identification des objets externes agissant avec le système
- Recherche d'archétypes de comportement (et non pas d'individus)
- Correspondance possible entre un acteur et plusieurs objets extérieurs ou un objet extérieur et plusieurs acteur (un acteur par facette)

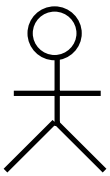
Analyse de l'application (3/28)

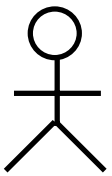
2. Trouver les acteurs

- Identification des objets externes agissant avec le système
- Recherche d'archétypes de comportement (et non pas d'individus)
- Correspondance possible entre un acteur et plusieurs objets extérieurs ou un objet extérieur et plusieurs acteur (un acteur par facette)


Employé


Responsable de formation


Organisme de formation


Service comptable

Analyse de l'application (4/28)

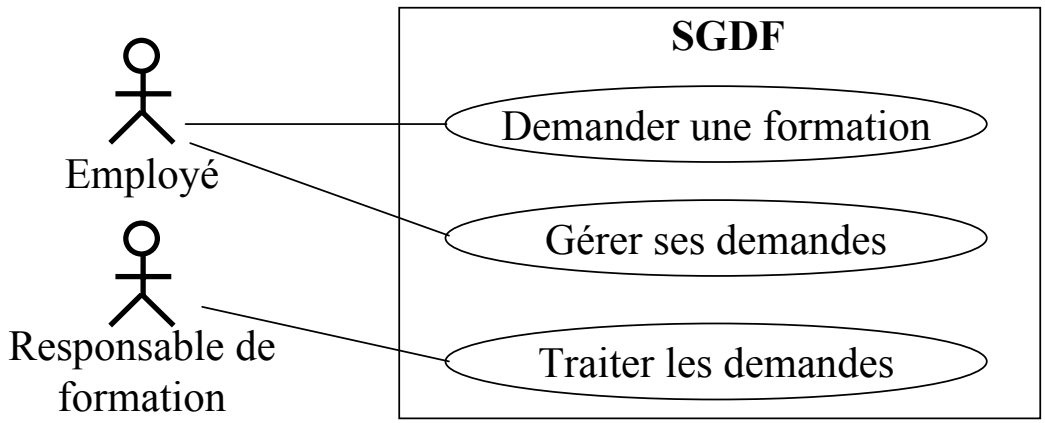
3. Trouver les cas d'utilisation ◁

- Dresser, pour chaque acteur, la liste de ses façons fondamentalement différentes d'utiliser le système
- Représenter un cas d'utilisation pour chaque type de service fourni par le système
- Conserver un niveau de détails similaire pour tous les cas d'utilisation
- Tracer **un diagramme de cas d'utilisation préliminaire**

Analyse de l'application (4/28)

3. Trouver les cas d'utilisation

- Dresser, pour chaque acteur, la liste de ses façons fondamentalement différentes d'utiliser le système
- Représenter un cas d'utilisation pour chaque type de service fourni par le système
- Conserver un niveau de détails similaire pour tous les cas d'utilisation
- Tracer **un diagramme de cas d'utilisation préliminaire**



Analyse de l'application (5/28)

4. Trouver les événements initiaux et finaux

- Partition des fonctionnalités du système en unités discrètes (les cas d'utilisation) indiquant les acteurs impliqués mais **pas de représentation claire du comportement**
- **Compréhension du comportement à partir de la compréhension des séquences d'exécution de chaque cas d'utilisation**
- Recherche des événements initiaux et finaux de chaque cas d'utilisation

Analyse de l'application (5/28)

4. Trouver les événements initiaux et finaux

- Partition des fonctionnalités du système en unités discrètes (les cas d'utilisation) indiquant les acteurs impliqués mais **pas de représentation claire du comportement**
- **Compréhension du comportement à partir de la compréhension des séquences d'exécution de chaque cas d'utilisation**
- Recherche des événements initiaux et finaux de chaque cas d'utilisation

Demander une formation :

Événement initial : Saisie d'une demande par un employé

Événement final : Enregistrement de la demande

Gérer ses demandes :

Événement initial : Ouverture d'une demande existante par l'employé

Événement final : (1) Annulation de la demande ou (2) Enregistrement des modifications

Analyse de l'application (6/28)

5. Préparer des scénarios standards

- Préparation d'un ou plusieurs dialogues types pour avoir une idée du comportement attendu du système
- Illustration par des scénarios des interactions importantes, des formats d'affichage externe et des échanges d'informations
- Ne pas rédiger le cas général directement, mais penser en terme d'exemples d'interactions
- Commencer par rédiger les scénarios des « cas normaux »

Analyse de l'application (6/28)

5. Préparer des scénarios standards

- Préparation d'un ou plusieurs dialogues types pour avoir une idée du comportement attendu du système
- Illustration par des scénarios des interactions importantes, des formats d'affichage externe et des échanges d'informations
- Ne pas rédiger le cas général directement, mais penser en terme d'exemples d'interactions
- Commencer par rédiger les scénarios des « cas normaux »

Exemple de scénario pour le cas d'utilisation « Demander une formation »

Albert Gamotte (employé) se connecte au système. Il va dans le menu « Créer une demande ». La date du jour est automatiquement affectée à la demande. Albert Gamotte choisit parmi les thèmes disponibles le thème « Formation UML 2 ». Il enregistre sa demande et se déconnecte.

Analyse de l'application (7/28)

6. Ajouter des scénarios de variations et d'exceptions

- Considérer les cas particuliers tels que les entrées omises, les valeurs minimales et maximales, les valeurs dupliquées
- Prendre en compte les cas d'erreur (ex. valeurs invalides, absence de réponse)
- Envisager les interactions pouvant se superposer aux interactions de base

- Mot de passe de l'employé erroné
- Serveur indisponible à la connexion
- Connexion à la base de données coupée
- Panne du serveur au moment de l'enregistrement de la demande

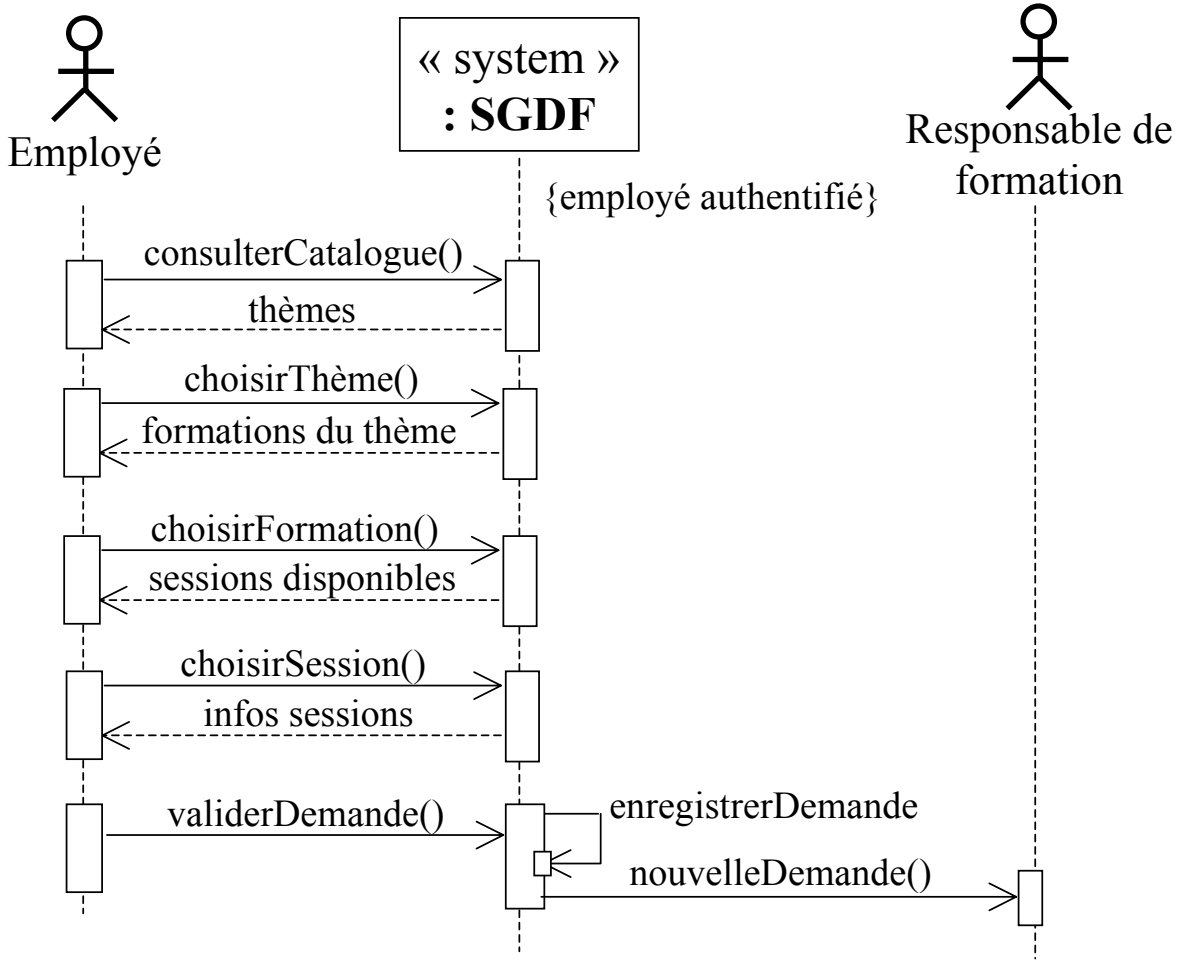
Analyse de l'application (8/28)

7. Trouver les événements externes

- Examiner les scénarios pour identifier les événements externes :
  - Entrées
  - Décisions
  - Interruption
  - Interactions provenant ou dirigées vers des utilisateurs ou des équipements externes
- Regrouper les événements ayant le même effet sous un même nom (même en cas de valeurs de paramètres différentes)
- Préparer un diagramme de séquences pour chaque scénario ◁

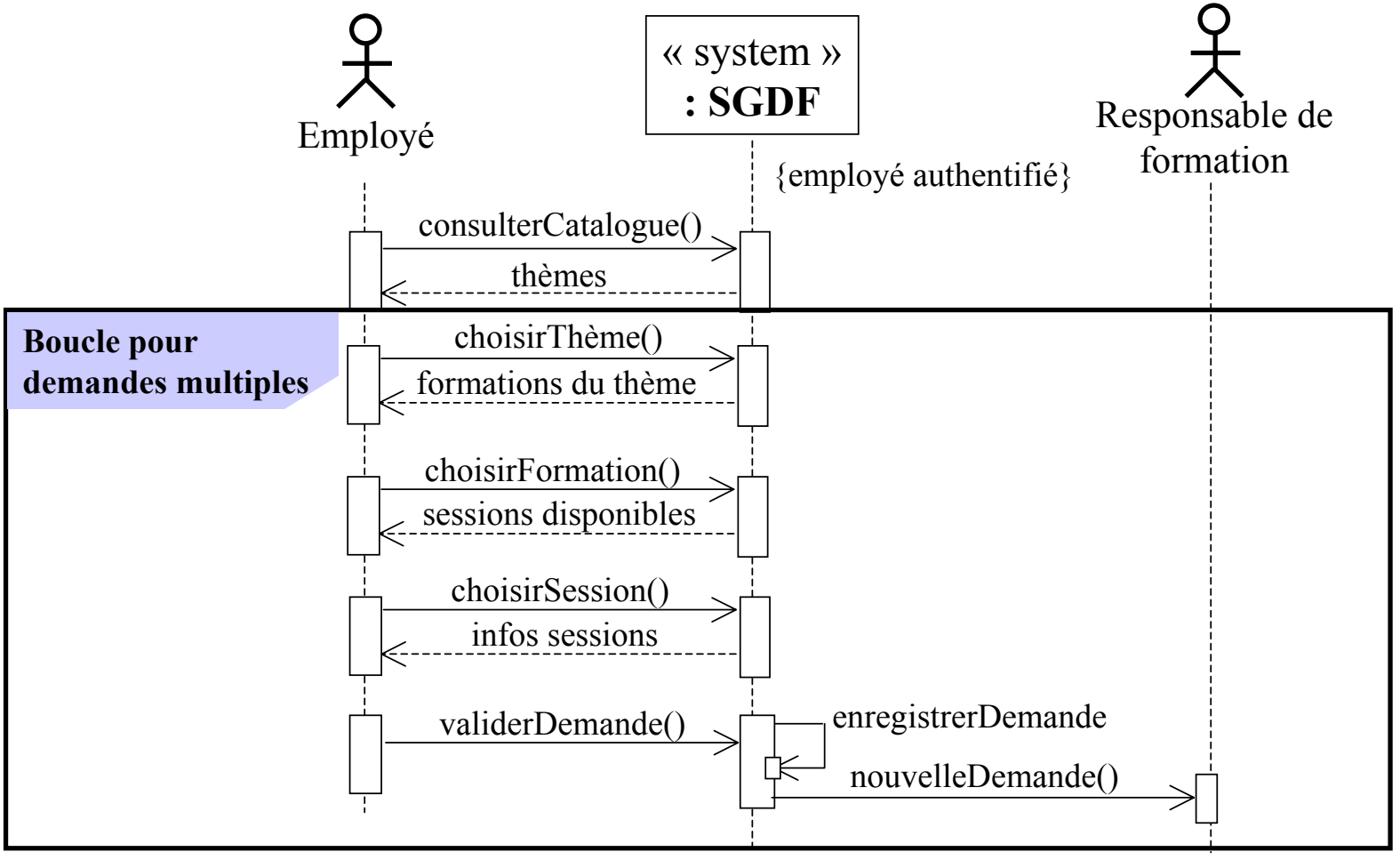
Analyse de l'application (9/28)

7. Trouver les événements externes (exemple)



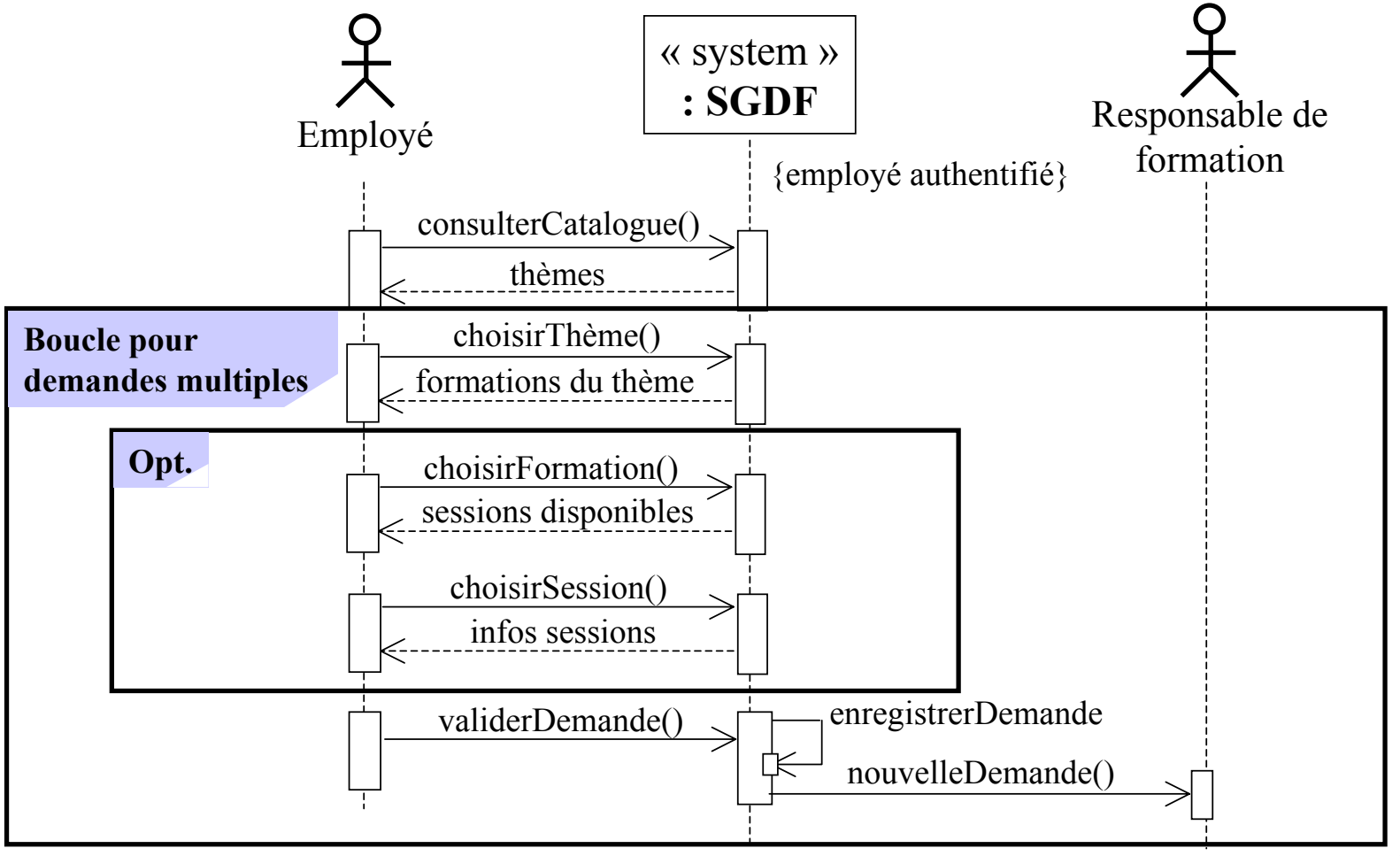
Analyse de l'application (9/28)

7. Trouver les événements externes (exemple)



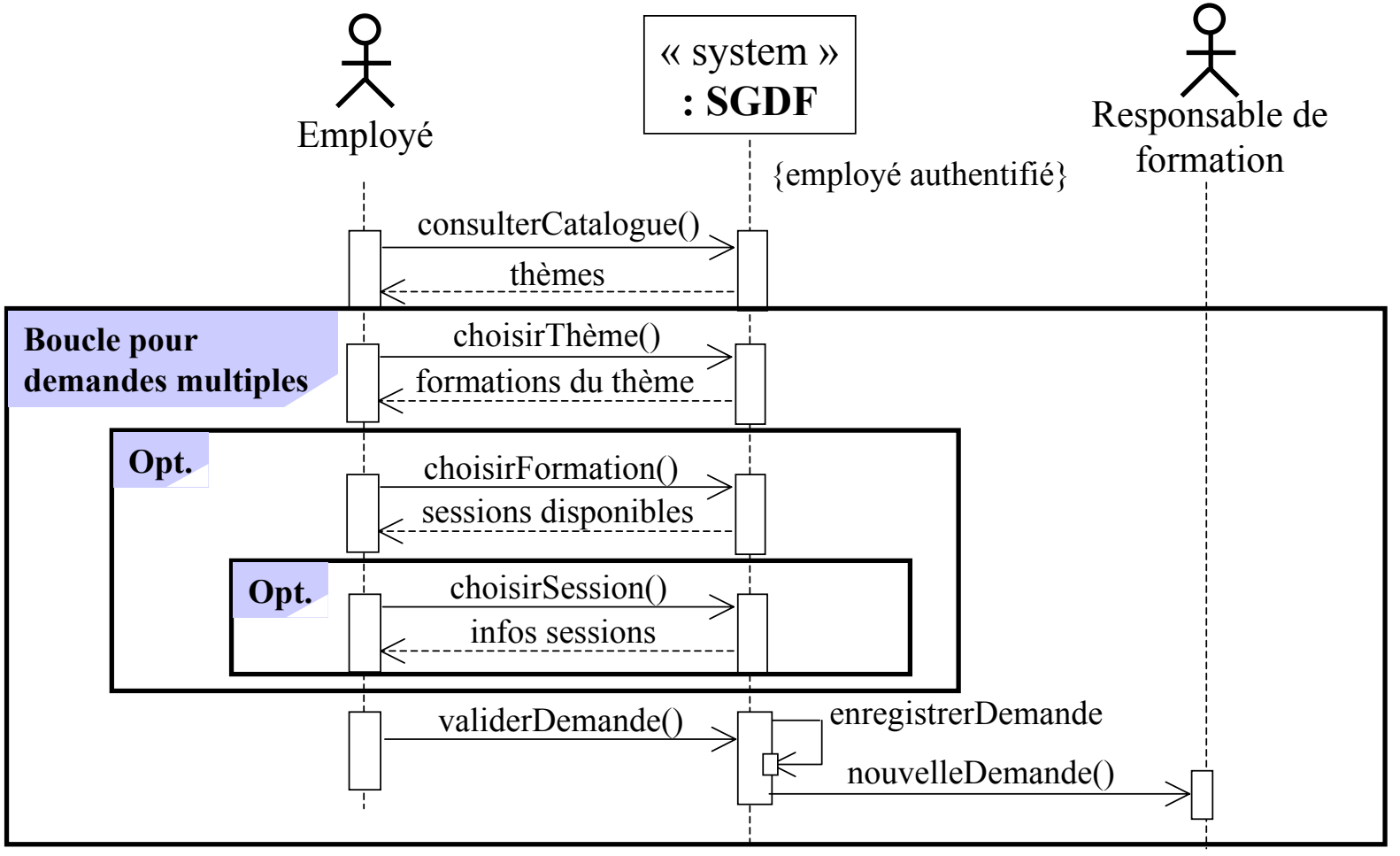
Analyse de l'application (9/28)

7. Trouver les événements externes (exemple)



Analyse de l'application (9/28)

7. Trouver les événements externes (exemple)



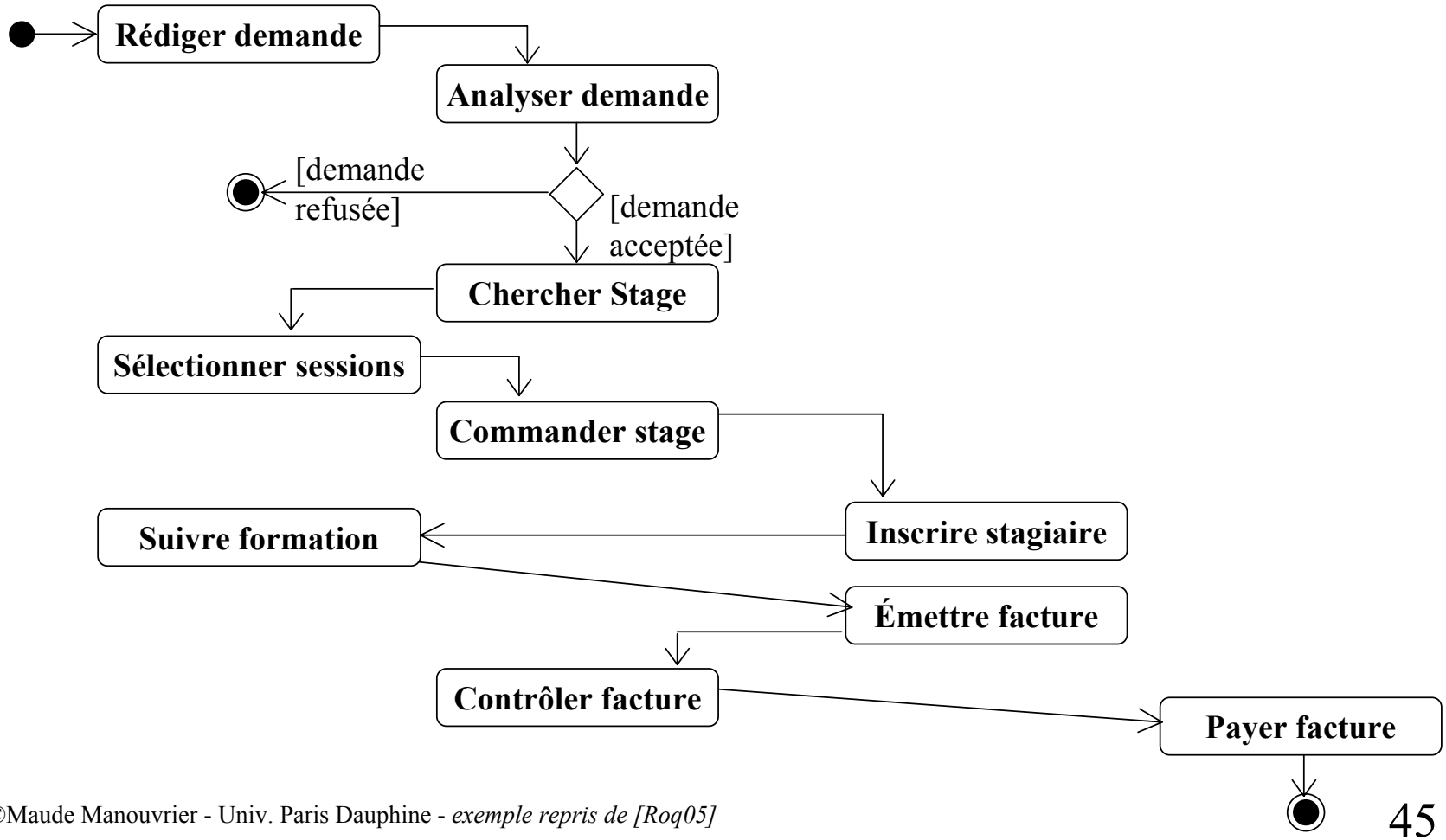
Analyse de l'application (10/28)

8. **Préparer des diagrammes d'activité pour les cas d'utilisation complexe** [BR05] ◁

   - Utiliser des diagrammes d'activité **pour documenter la logique métier pendant l'analyse**
   - Ne pas utiliser les diagrammes d'activité « comme une excuse pour commencer l'implémentation »

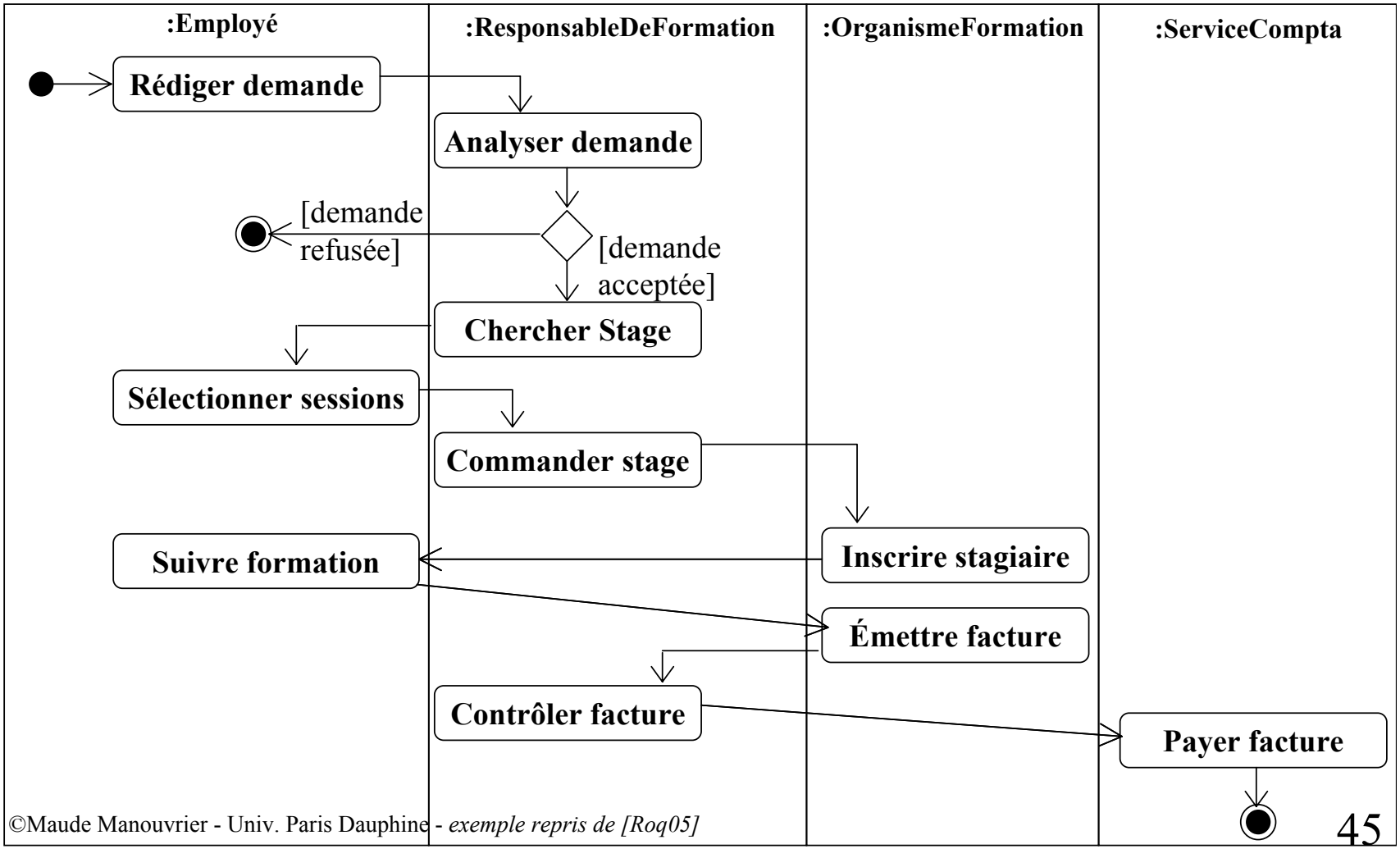
Analyse de l'application (11/28)

8. Préparer des diagrammes d'activité pour les cas d'utilisation complexe (exemple) ◁



Analyse de l'application (11/28)

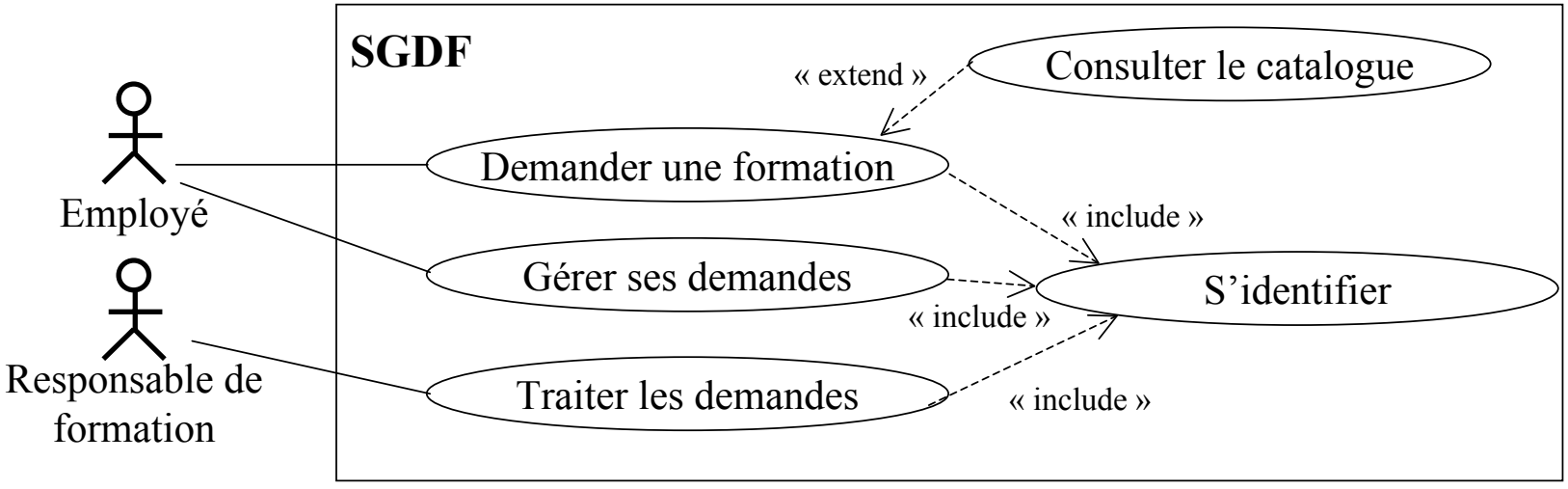
8. Préparer des diagrammes d'activité pour les cas d'utilisation complexe (exemple)



Analyse de l'application (12/28)

9. Organiser les acteurs et les cas d'utilisation

- Pour les systèmes de grande taille ou les systèmes complexes, organiser les cas d'utilisation à l'aide des relations *include* et *extend*
- Possibilité d'organiser les acteurs par des généralisations



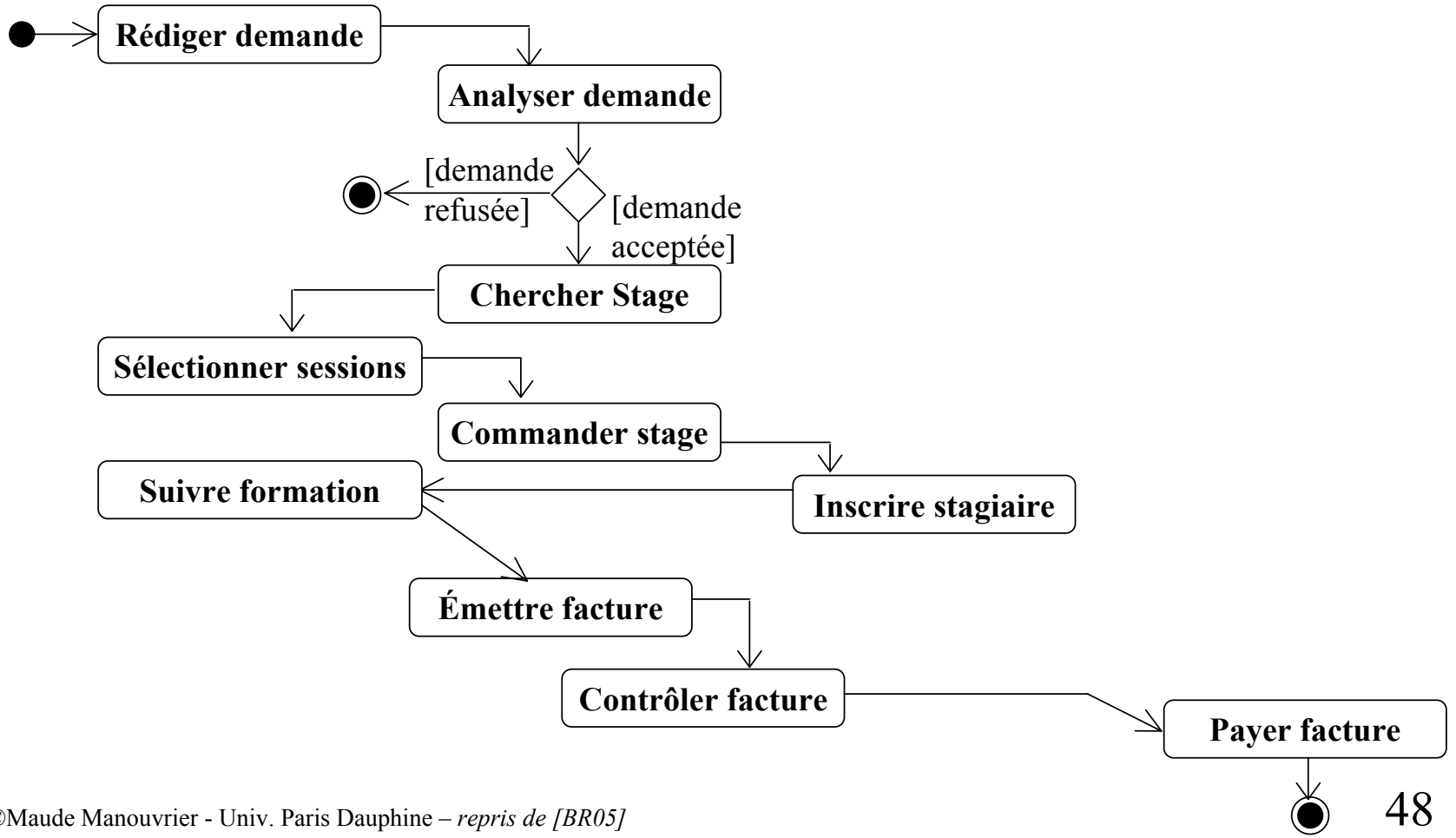
Analyse de l'application (13/28)

10. Vérifier avec le modèle de classes du domaine

- Vérifier la cohérence entre le modèle du domaine et celui de l'application
- Examiner les scénarios pour s'assurer que le modèle du domaine possède toutes les données nécessaires
- Vérifier que le modèle du domaine couvre tous les paramètres d'événements

Analyse de l'application (14/28)

10. Vérifier avec le modèle de classes du domaine (exemple)



Analyse de l'application (14/28)

10. Vérifier avec le modèle de classes du domaine (exemple)

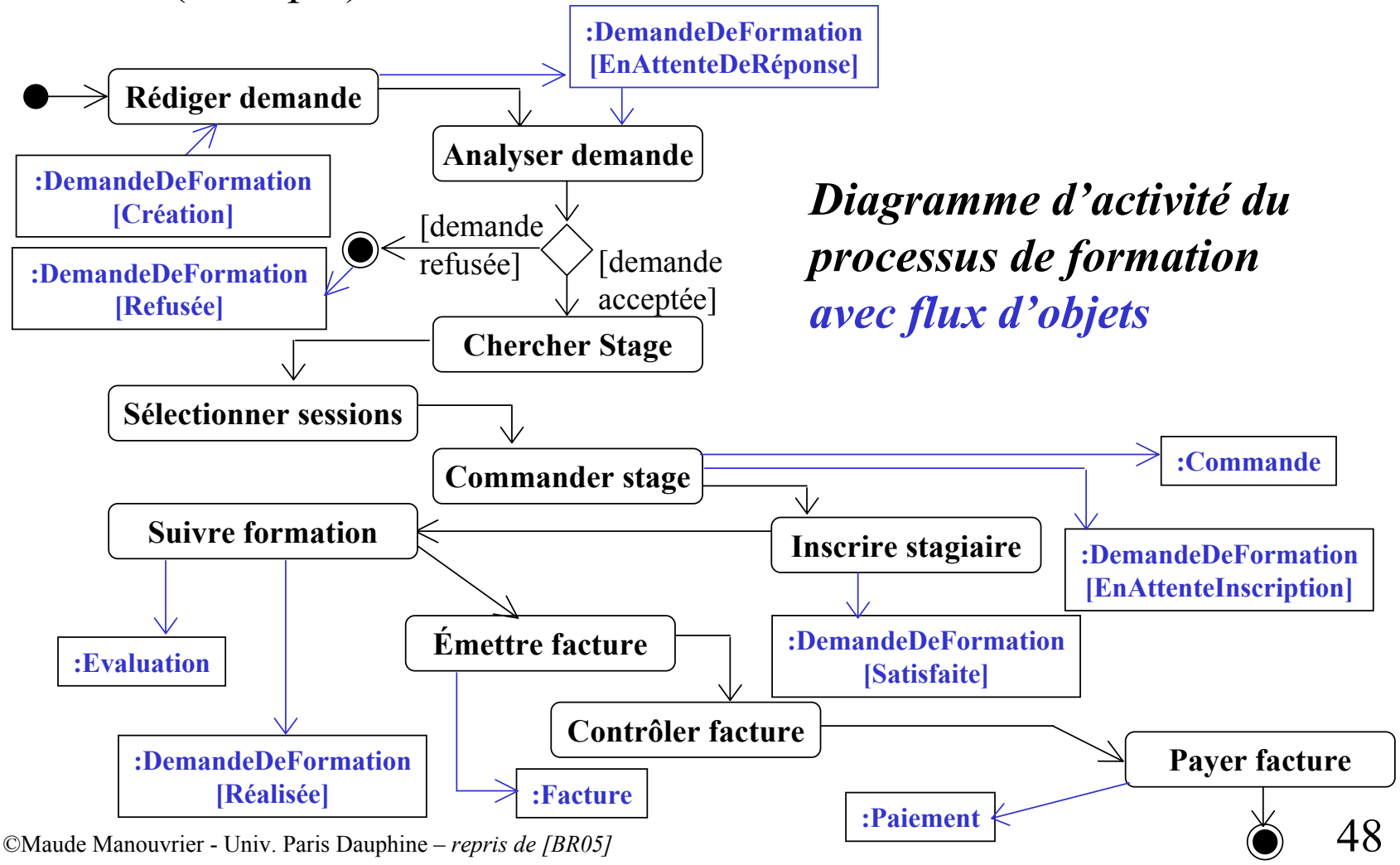


Diagramme d'activité du processus de formation avec flux d'objets

Analyse de l'application (15/28)

Construire un **modèle de classes d'application** [BR05] :

1. **Spécifier les interfaces utilisateur**
2. **Définir les classes frontières**
3. **Déterminer les contrôleurs**
4. **Vérifier avec le modèle d'interactions**

Analyse de l'application (16/28)

1. Spécifier les interfaces utilisateurs (IHM)

- IHM = « objet ou groupe d'objets fournissant aux utilisateurs d'un système un moyen cohérent d'accès aux objets du domaine, aux commandes et aux options de l'application » [BR05]
- Pendant l'analyse, mettre l'accent sur le flux d'informations et le contrôle et non sur le format de présentation
- Pendant l'analyse, considérer l'interface de manière grossière et ne pas se préoccuper des différentes manières d'entrer les données
- Esquisser néanmoins un croquis de l'interface pour aider à visualiser le fonctionnement de l'application et s'assurer de ne rien avoir oublié d'important

Analyse de l'application (17/28)

2. Définir les classes frontières

- Classe frontière (*boundary*) : « classe modélisant les interactions entre le système et ses acteurs » [Roq05]
- Gestion, via une classe frontière, du format d'une ou plusieurs sources externes et conversion des informations pour les transmettre au système et inversement [BR05]
- Utilité de définir des classes frontières pour isoler la partie interne du système du monde extérieur [BR05]

Analyse de l'application (17/28)

2. Définir les classes frontières

- Classe frontière (*boundary*) : « classe modélisant les interactions entre le système et ses acteurs » [Roq05]
- Gestion, via une classe frontière, du format d'une ou plusieurs sources externes et conversion des informations pour les transmettre au système et inversement [BR05]
- Utilité de définir des classes frontières pour isoler la partie interne du système du monde extérieur [BR05]

« boundary » **ÉcranGeneralEmployé**

« boundary » **ÉcranDemandeFormation**

Analyse de l'application (17/28)

2. Définir les classes frontières

- Classe frontière (*boundary*) : « classe modélisant les interactions entre le système et ses acteurs » [Roq05]
- Gestion, via une classe frontière, du format d'une ou plusieurs sources externes et conversion des informations pour les transmettre au système et inversement [BR05]
- Utilité de définir des classes frontières pour isoler la partie interne du système du monde extérieur [BR05]

« boundary » **ÉcranGeneralEmployé**

« boundary » **ÉcranDemandeFormation**

Apparition uniquement des souches des classes frontières dans le modèle d'application, l'élaboration de ces classes n'étant pas encore claire à ce stade du processus de développement [BR05]

Analyse de l'application (18/28)

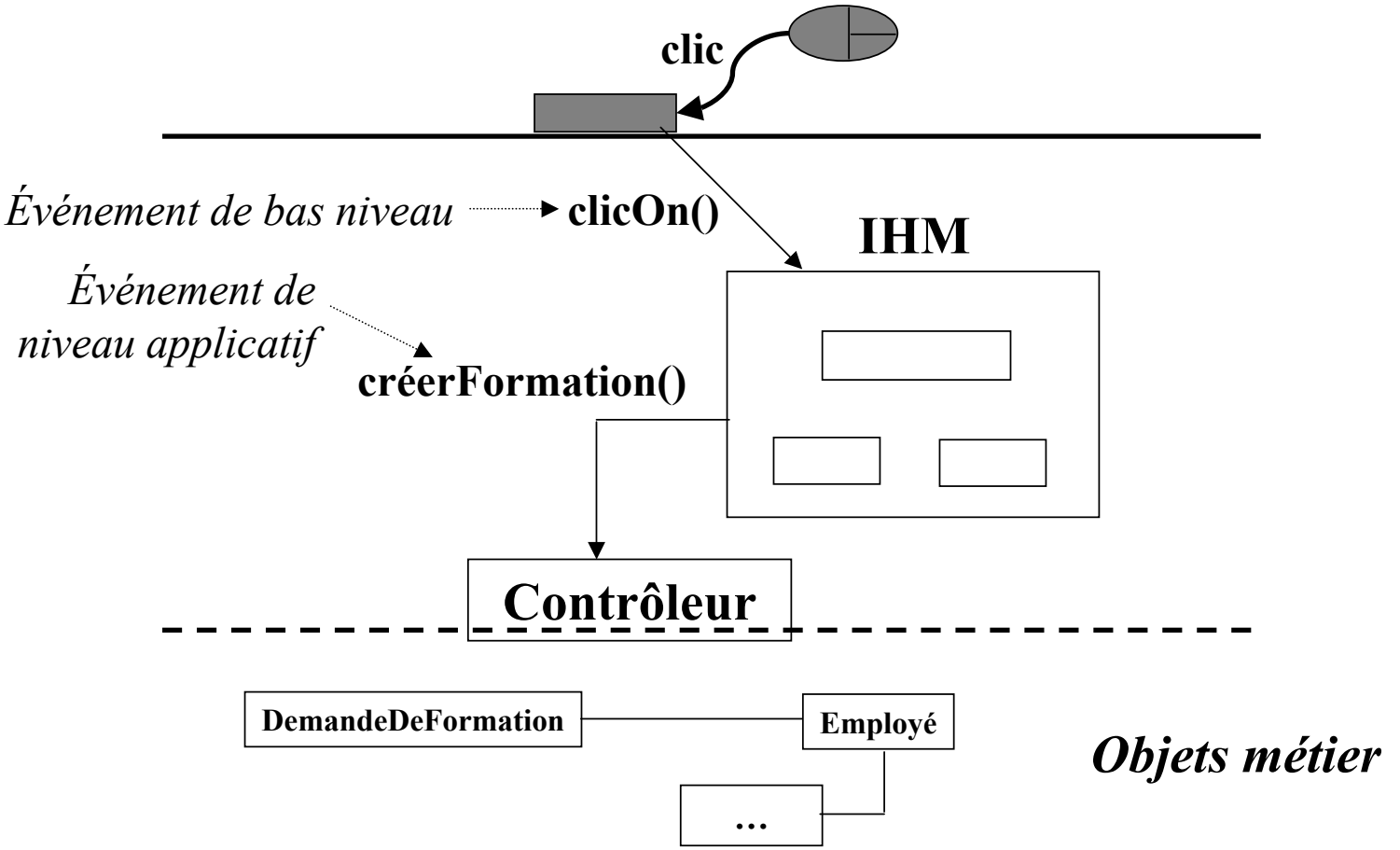
3. Déterminer les contrôleurs

- Contrôleur : « objet actif gérant le contrôle au sein d'une application » [BR05]
  - Réception de signaux par le contrôleur provenant du monde extérieur ou des objets internes au système
  - Réaction à ces signaux
  - Invocation des opérations sur les objets du système
  - Envoi de signaux au monde extérieur
- Un ou plusieurs contrôleurs par application pour séquencer le contrôle [BR05]
- Liaison en général entre un contrôleur et un cas d'utilisation particulier [Roq05]

« control »
ContrôleurDemandeFormation

Analyse de l'application (19/28)

3. Déterminer les contrôleurs (suite)



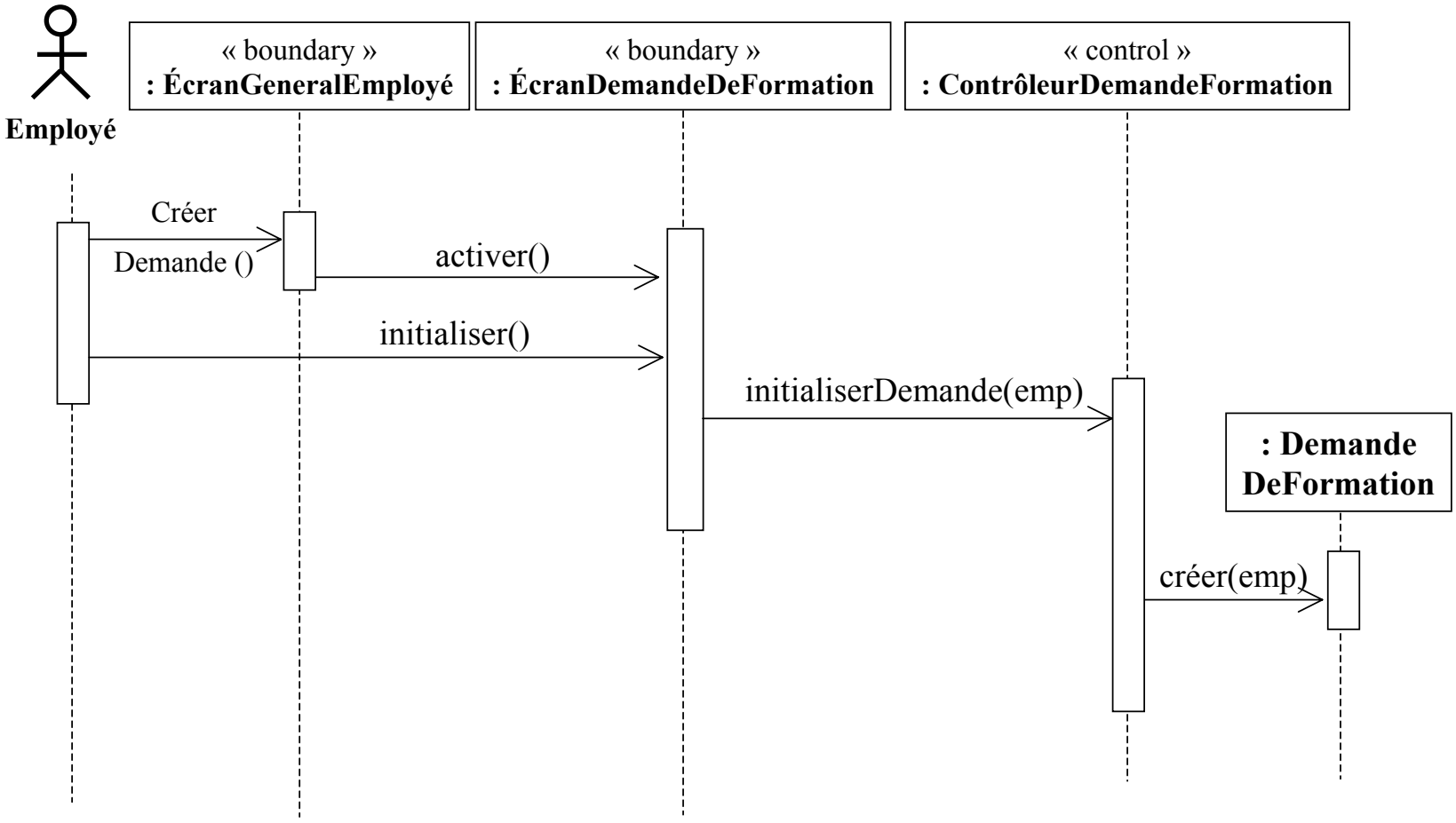
Analyse de l'application (20/28)

4. Vérifier avec le modèle d'interactions

- Construire le modèle de classes de l'application en revoyant les cas d'utilisation et en réfléchissant à leur manière de fonctionner
- A la fin de l'analyse, possibilité de de simuler un cas d'utilisation avec les classes de deux modèles (domaine et application)

Analyse de l'application (21/28)

4. Vérifier avec le modèle d'interactions (exemple)



Analyse de l'application (22/28)

Modèle **d'états de l'application** [BR05] :

1. **Déterminer les classes d'application ayant des états**
2. **Identifier les événements**
3. **Construire des diagrammes d'états**
4. **Comparer aux autres diagrammes d'états**
5. **Comparer au modèle de classes**
6. **Comparer au modèle d'interactions**

Analyse de l'application (23/28)

1. **Déterminer les classes d'application ayant des états**

   Bons candidats pour les diagrammes d'états : les **classes de l'interface** et les **classes contrôleurs**

2. **Trouver les événements**

   Étudier les scénarios pour en extraire les événements

Analyse de l'application (23/28)

1. **Déterminer les classes d'application ayant des états**

   Bons candidats pour les diagrammes d'états : les **classes de l'interface** et les **classes contrôleurs**

2. **Trouver les événements**

   Étudier les scénarios pour en extraire les événements

Contraste entre l'élaboration du modèle d'états du domaine et celui de l'application [BR05] :

- **Pour le modèle du domaine : recherche des états puis des événements**
  Concentration du modèle du domaine sur les données
  Regroupements de données significatifs qui forment des états sujets à des événements
- **Pour le modèle de l'application : sens inverse**
  Concentration du modèle d'application sur les comportements
  Élaboration de cas d'utilisation relevant des événements

Analyse de l'application (24/28)

3. **Construire des diagrammes d'états** [BR05]

- Construire un diagramme d'états pour chaque classe d'application ayant un comportement temporel
- Envisager un diagramme de séquence pour chaque classe ayant un comportement temporel
  - Diagramme d'états initial = séquences d'événements et d'états
  - Correspondance entre chaque scénario ou diagramme de séquences et un chemin dans le diagramme d'états
- **Diagramme d'états complet ⇒ Prise en compte de tous les scénarios et traitement de tous les événements pouvant affecter un état**

Analyse de l'application (25/28)

3. Construire des diagrammes d'états (exemple)

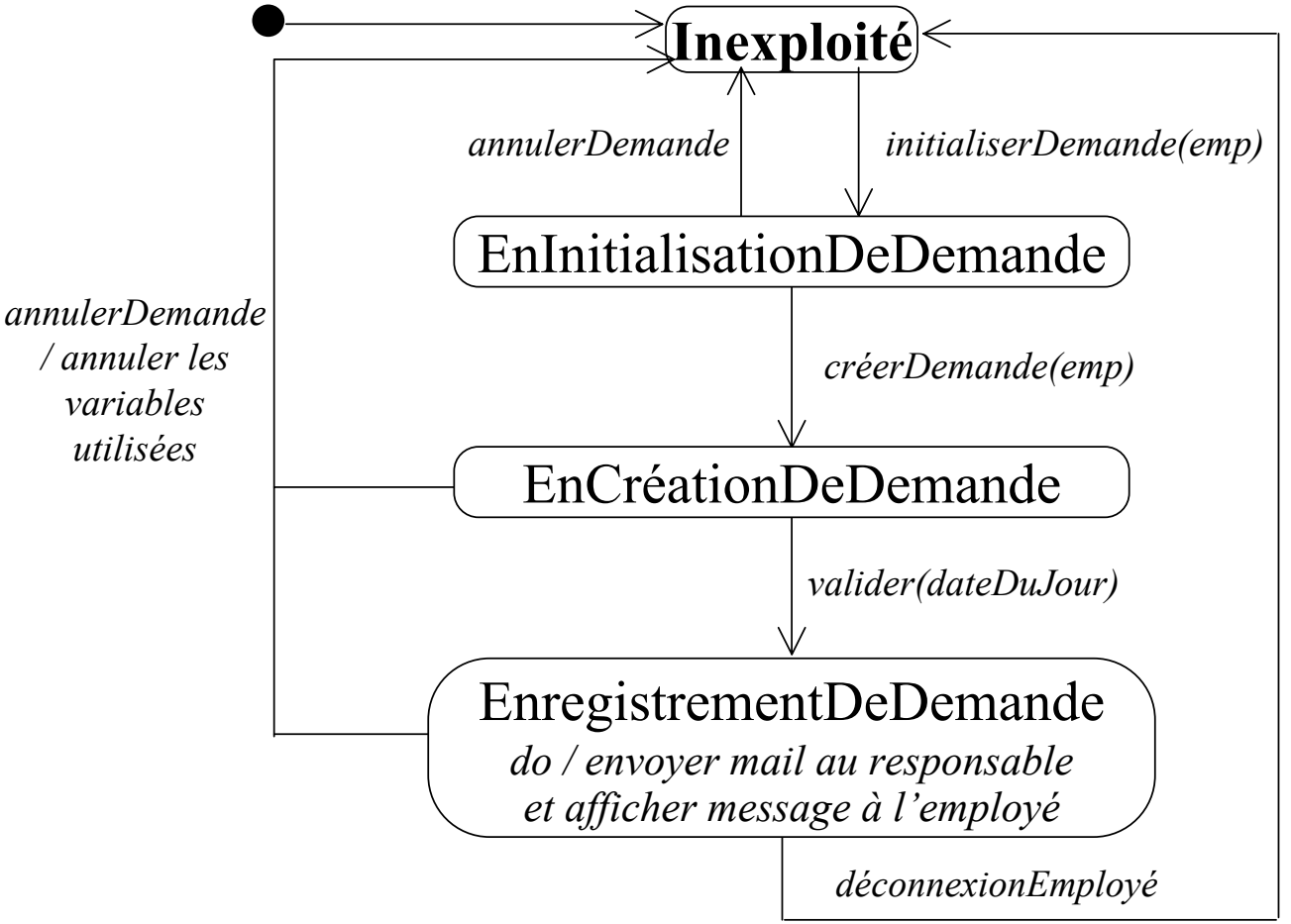


Diagramme d'états initial pour le contrôleur de demandes de formation (à compléter)

Analyse de l'application (26/28)

4. Vérifier avec les autres diagrammes d'états

Vérifier la complétude et la cohérence de chaque diagramme d'états et vérifier la cohérence entre les événements de différents diagrammes

5. Vérifier avec le modèle de classes

Vérifier la cohérence avec le modèle de classes du domaine et celui de l'application

6. Vérifier avec le modèle d'interactions

- Comparer le modèle d'états avec les scénarios du modèle d'interactions
- Simuler chaque chaque séquence manuellement et vérifier le reflet correct du comportement sur le diagramme d'états
- En cas de découverte d'erreur, modifier le diagramme d'états ou les scénarios
- Identifier les chemins légitimes dans le modèle d'états ($\Rightarrow$scénarios supplémentaires) et vérifier qu'ils ont un sens

Analyse de l'application (27/28)

Ajouter des opérations [BR05] :

1. **Opérations issues du modèle de classes**

   Inutile de représenter les lectures ou écritures des attributs ou de liens d'associations = opérations implicites

2. **Opérations issues des cas d'utilisation**

   Correspondance entre les opérations et les activités

3. **Opérations « aide-mémoire »**
   - Issues du comportement des classes dans le monde réel
   - Indépendantes d'une application particulière mais avec un intérêt intrinsèque
   - Élargissement de la définition d'une classe au-delà des besoins stricts du problème immédiat

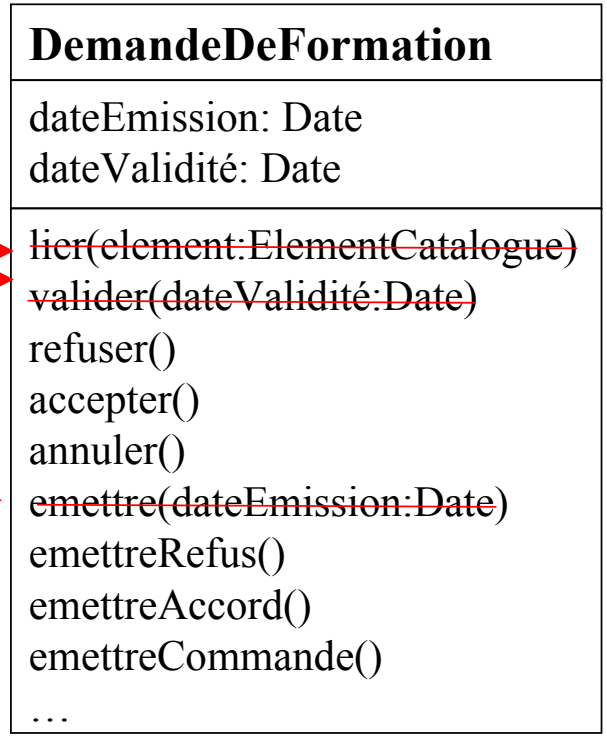
Analyse de l'application (28/28)

DemandeDeFormation
dateEmission: Date dateValidité: Date
lier(element:ElementCatalogue) valider(dateValidité:Date) refuser() accepter() annuler() emettre(dateEmission:Date) emettreRefus() emettreAccord() emettreCommande() …

4. **Simplifier les opérations** [BR05]
   - Rechercher les opérations similaires et le variations de forme d'une même opération
   - Élargir la définition d'une opération pour englober ses variations et les cas particuliers
   - Utiliser l'héritage pour réduire le nombre d'opérations

Analyse de l'application (28/28)

Opérations de mise à jour d'attributs ou de liens ne devant pas apparaître lors de l'analyse



4. **Simplifier les opérations** [BR05]
   - Rechercher les opérations similaires et le variations de forme d'une même opération
   - Élargir la définition d'une opération pour englober ses variations et les cas particuliers
   - Utiliser l'héritage pour réduire le nombre d'opérations

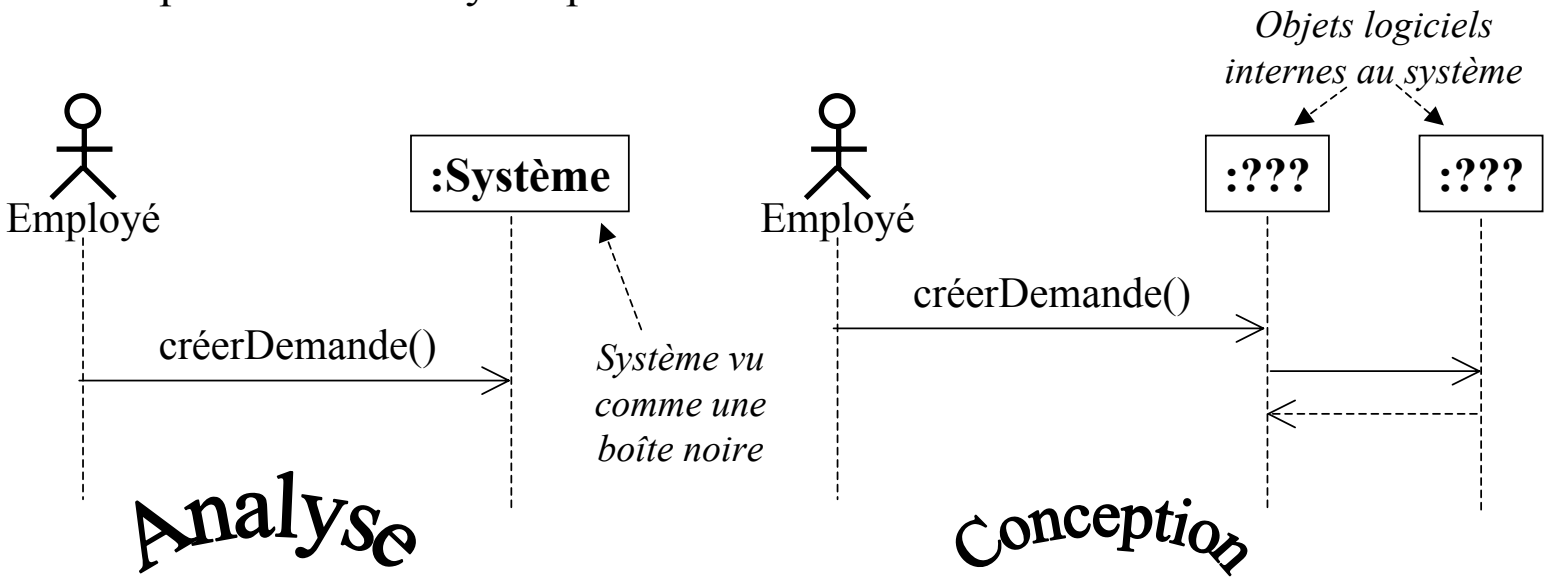
Analyse : résumé (1/2)

- Objectif de l'analyse : « comprendre un problème afin de pouvoir définir une conception correcte du système »
- Capture des propriétés essentielles du problème sans introduire d'artefacts d'implémentation
- 2 phases :
  - Analyse du domaine :
    - Capture de la connaissance générale d'une application
    - Traduction en modèle de classes, parfois d'états, rarement d'interactions
  - Analyse de l'application :
    - Concentration sur les artefacts principaux, importants et visibles par les utilisateurs et approuvés par les utilisateurs
    - Domination du modèle d'interactions, mais importance également des modèles de classes et d'états

Analyse : résumé (2/2)

Importance de l'abstraction tout au long de l'analyse [BR05] :

- Réfléchir de manière large en construisant les modèles
- Ne pas lier l'application à des pratiques métier arbitraires et pouvant changer dans le temps
- Essayer de trouver une souplesse anticipant les futures évolutions et permettant de s'y adapter



Conception du système (1/11)

- **Analyse : le quoi**, indépendamment du comment
- **Conception** (*design*) : **le comment**, avec des prises de décisions, dans un premier temps de haut niveau, puis à des niveaux de plus en plus détaillés
- **Conception du système** :
  - **Première étape de la conception**
  - Mise au point d'une stratégie de haut niveau : l'**architecture du système**

Conception du système (2/11)

Étapes a suivre [BR05] :

1. Estimer les performances du système
2. Mettre au point un plan de réutilisation
3. Organiser le système en sous-systèmes
4. Identifier les questions de concurrence inhérentes au problème
5. Allouer les sous-systèmes aux équipements matériels
6. Gérer le stockage des données
7. Gérer les ressources globales
8. Choisir une stratégie de contrôle du logiciel
9. Traiter les cas limites
10. Arbitrer les priorités
11. Sélectionner un style architectural

Conception du système (3/11)

1. **Estimer les performances du système**
   - Objectif : déterminer la faisabilité du système
   - Émettre des hypothèses simplificatrices
   - Se contenter d'approximations, d'estimations et de suppositions si nécessaires
2. **Mettre au point un plan de réutilisation**
   - Deux aspects de réutilisation :
     - Utilisation d'éléments existants
     - Création de nouveaux éléments
   - Nécessité d'avoir de l'expérience pour concevoir des éléments réutilisables
   - Exemples d'éléments réutilisables : bibliothèque, *framework* et *pattern*

Conception du système (4/11)

- **Bibliothèque** [BR05] : « collection de classes utiles dans de nombreux contexte »
- ***Framework*** [BR05] : « squelette de programme devant être étoffé pour construire une application complète »
- **Pattern** [Roq05] : « Solution de modélisation récurrente et documentée, applicable dans un contexte donné »

Conception du système (5/11)

3. Décomposer un système en sous-systèmes

- **Sous-système** :
  - Définition cohérente de traiter une partie du problème
  - Ensemble de classes, d'associations, d'opérations, d'événements, de contraintes, reliés et ayant des interfaces bien définies (et restreinte) avec d'autres sous-systèmes
  - Défini en termes des services qu'il fournit
- **Service** : ensemble de fonctionnalités apparentées servant le même but
- **Interface** :
  - Spécification de la forme des interactions et des flux d'informations traversant la frontière du sous-système
  - Sans lien avec l'implémentation interne du sous-système
- Conception d'un sous-système indépendante et sans affectation sur les autres sous-systèmes

Conception du système (6/11)

3. Décomposer un système en sous-systèmes (suite)

- **Décomposition en couches** :

  Construction d'une couche en terme de celles qui se trouvent en-dessous et fournissant une base d' implémentation pour celles se situant au-dessus

- **Décomposition en partitions** :

  Division d'un système en sous-systèmes indépendants et faiblement couplés, chacun fournissant un type de service

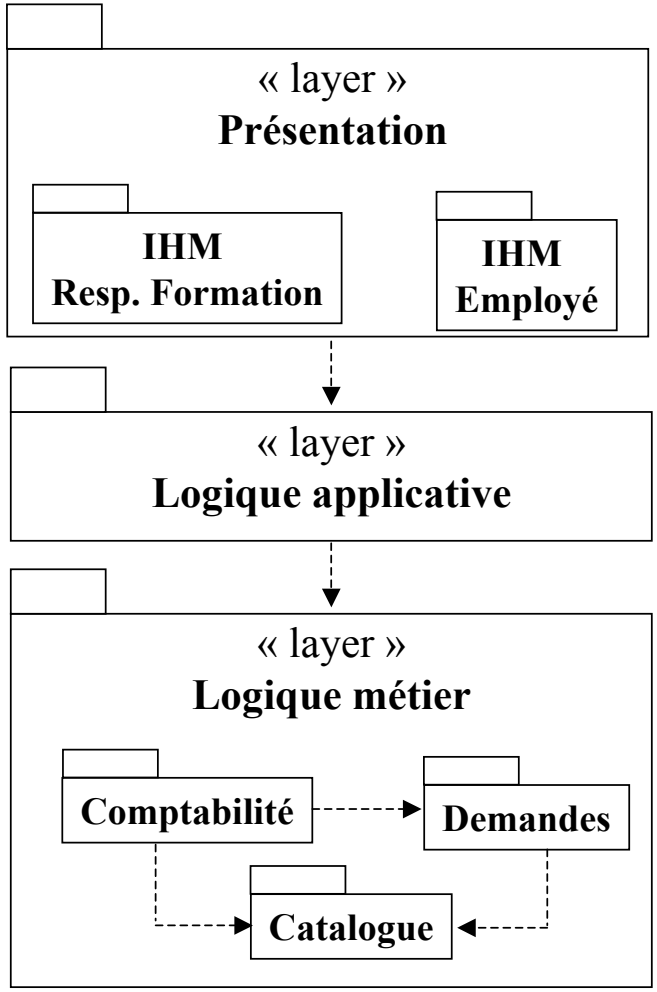
- **Combinaison de couches et de partitions** :

  Nécessité pour la plupart des grands systèmes

Conception du système (7/11)

3. Décomposer un système en sous-systèmes (exemple)

Exemple d'architecture en couches pour le système de gestion de demandes de formation



Conception du système (8/11)

4. Identifier la concurrence

- **Concurrence** = « deux ou plusieurs activités ou événements dont l'exécution peut se chevaucher dans le temps »
- Identifier la concurrence intrinsèque à l'aide du modèle d'états
- Décomposition en sous-systèmes de préférence indépendants
- ***Thread* de contôle** : « chemin d'exécution unique dans un ensemble de diagrammes d'états où un seul objet est actif à la fois»
- Définir les tâches concurrentes en groupant des objets non concurrents dans un même ***thread* de contôle** (ou tâche)

Conception du système (9/11)

5. Allouer des sous-systèmes

a) Estimer les exigences en ressources matérielles

b) Arbitrer entre matériel et logiciel

c) Allouer des tâches aux processeurs

d) Choisir l'organisation et la forme de connexion des différentes unités physiques (topologie des connexions, unités redondantes, communications)

6. Gérer le stockage des données

Structure de données en mémoire, fichiers ou bases de données

5. Gérer les ressources globales

Unités physiques (ex. processeurs), espaces (ex. espace disques), noms logiques (ex. noms de fichiers ou de classes), accès aux données partagées (ex. bases de données)

Conception du système (10/11)

8. Choisir une stratégie de contrôle du logiciel

- **Contrôle externe** : flux d'événements visibles à l'extérieur entre les objets du système
- **Contrôle interne** : flux d'événements au sein d'un processus

9. Gérer les cas limites

- Initialisation
- Terminaison
- Échecs

10. Arbitrer les priorités

- Arbitrage entre temps et espace, matériel et logiciel, simplicité et généralité, efficacité et maintenabilité
- Compromis dépendant des buts de l'application

Conception du système (11/11)

11. Sélectionner un style architectural

- **Transformation par lots** (*batch*) : « Transformation de données exécutée en une seule fois sur un ensemble complet de données d'entrées »
- **Transformation continue** : « Transformation opérée continuellement sur des données d'entrées variant »
- **Interface interactive** : « Système dominé par des interactions externes »
- **Simulation dynamique** : « Système simulant l'évolution des objets du monde réel »
- **Système temps réel** : « Système dominé par des contraintes de temps strictes »
- **Gestionnaire de transactions** : « Système avec pour rôle le stockage et la mise à jour de données, et impliquant souvent des accès concurrents provenant de différents emplacement physiques »

Conception des classes (1/13)

- **Analyse** : détermination de ce qui devra être réalisé par l'implémentation
- **Conception du système** : détermination du « plan d'attaque »
- **Conception des classes** :
  - Finalisation de la définition des classes et des associations
  - Choix des algorithmes des opérations

Conception des classes (2/13)

Conceptions des classes

- Ajout de détails et prise de décisions fines
- Choix des différentes façons d'implémenter les classes d'analyse
- Nécessité d'avoir plusieurs itérations à des niveaux successifs d'abstraction
- Critères de facilité d'implémentation, de maintenabilité et d'extensibilité

Conception des classes (3/13)

Étapes à suivre [BR05] :

1. Combler le fossé entre exigences de haut niveau et services de bas niveau
2. Réaliser les cas d'utilisation par des opérations
3. Formuler un algorithme pour chaque opération
4. Décomposer récursivement jusqu'à obtenir des opérations de conception desservant des opérations de bas niveau
5. Remanier le modèle pour obtenir une conception plus nette
6. Optimiser les chemins d'accès aux données
7. Réifier les comportements devant être manipulés
8. Ajuster la structure des classes pour augmenter l'héritage
9. Organiser les classes et associations

Conception des classes (4/13)

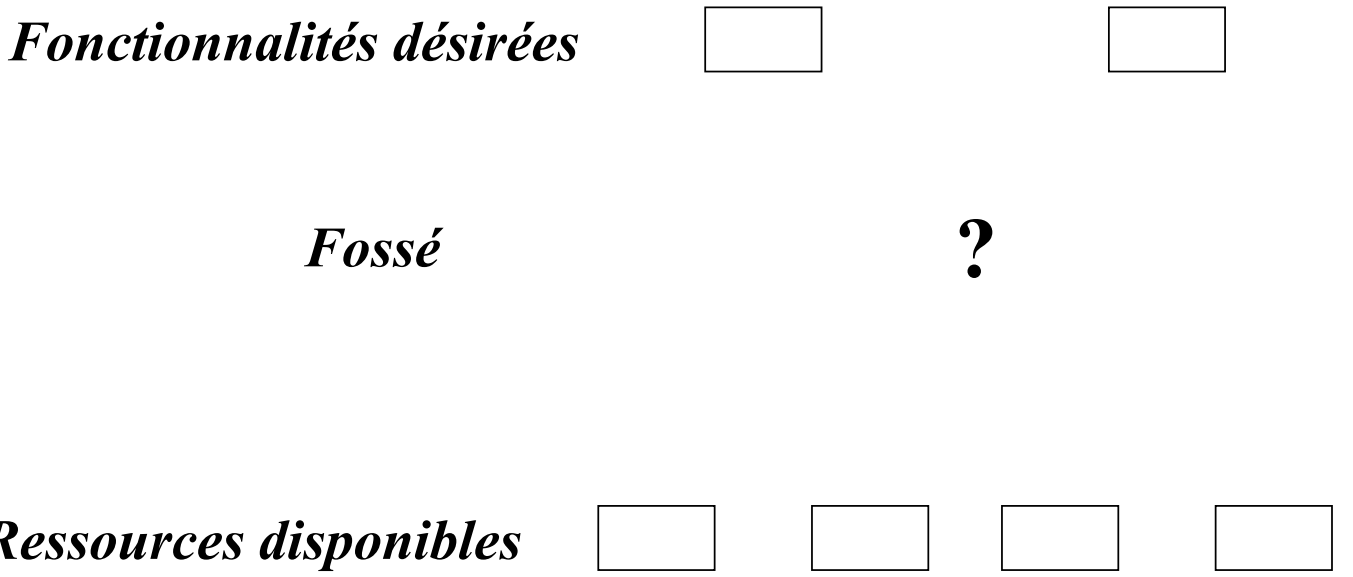
1. **Combler le fossé entre exigences de haut niveau et services de bas niveau**

Fonctionnalités désirées

Ressources disponibles

Conception des classes (4/13)

1. **Combler le fossé entre exigences de haut niveau et services de bas niveau**



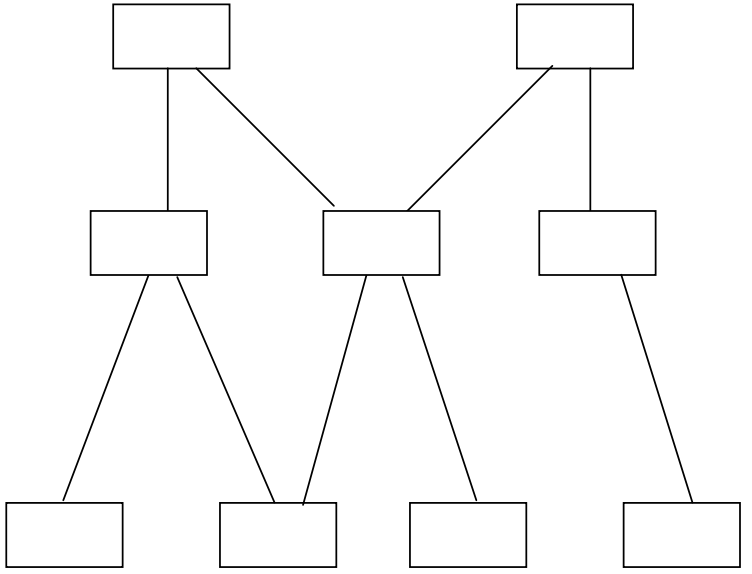
Conception des classes (4/13)

1. **Combler le fossé entre exigences de haut niveau et services de bas niveau**

Fonctionnalités désirées

Éléments intermédiaires

Ressources disponibles



Conception des classes (5/13)

2. Réaliser les cas d'utilisation

- **Cas d'utilisation** = définition d'un comportement mais pas de sa réalisation
- **Conception** $\Rightarrow$ choix entre les différentes options et préparation de l'implémentation
- Étapes :
  a) Dresser la liste des **responsabilités** d'un cas d'utilisation
  b) Regrouper les responsabilités en groupes cohérents $\Rightarrow$ une seule opération de bas niveau pour réaliser les responsabilités d'un même groupe
  c) Affecter les nouvelles opérations de bas niveau aux classes

Conception des classes (6/13)

3. Concevoir les algorithmes

a) Choisir des algorithmes qui minimisent le coût d'implémentation des opérations

- « 20% des opérations comprennent 80% du temps » ⇒ concentrer la recherche d'algorithmes à ces opérations
- Prise en compte de la complexité des calculs, de la facilité d'implémentation et de de compréhension, et de la souplesse

b) Sélectionner les structures de données appropriées aux algorithmes

Utilisation de **classes conteneurs** (*bags*, ensembles, tableaux, listes, files, piles, dictionnaires, arbres etc.)

c) Déterminer de nouvelles classes et opérations internes en cas de nécessité

Conception des classes (7/13)

4. Opérer par décomposition récursive

- Organiser les opérations en couches : invocation des opérations de bas niveau par les opérations de haut niveau
- Opérer de manière descendante : opérations de haut niveau puis celle de bas niveau
- Deux manières d'opérer :
  - Par fonctionnalités
    Attention à ne pas trop dépendre des fonctionnalités de haut niveau
    ⇒ « Système hypersensible aux changements »
  - Par mécanismes
    Construction du système en s'appuyant sur des mécanismes de base (mémorisation des informations, séquençage du flux de contrôle, coordination des objets, transformation des données …)
    ⇒ « Système ne faisant réellement rien d'utile »
  - **Sélectionner une combinaison appropriée entre conception par fonctionnalités et conception par mécanismes**

Conception des classes (8/13)

5. Remanier le modèle pour obtenir une conception plus nette

- ***Refactoring*** :
  - « processus consistant à modifier la structure interne d'un programme pour améliorer sa conception sans en changer les fonctionnalités externes »
  - « Partie essentielle de tout processus de génie logiciel »
  - $\Rightarrow$ Maintien de la viabilité de la conception pour la suite du développement
- Conception initiale $\Rightarrow$ incohérences, redondances, et points faibles en terme d'efficacité
- Nécessité de revoir la conception et de retravailler les classes et les opérations $\Rightarrow$ au final satisfaction de tous leurs usages et cohérence du point de vue conceptuel

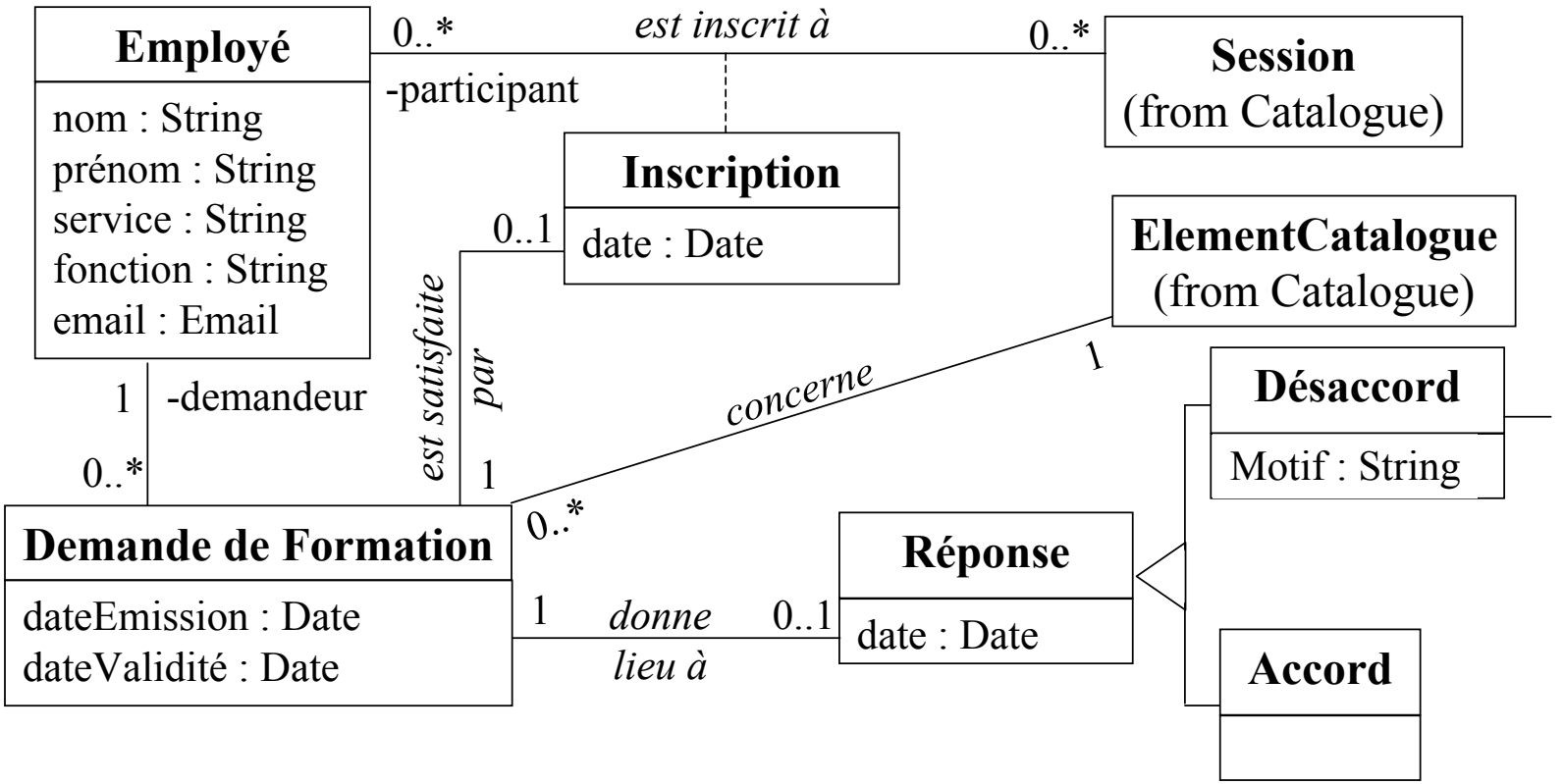
Conception des classes (9/13)

6. Optimiser la conception

- Trouver le juste équilibre entre efficacité et clarté
- Ne pas prendre en compte l'efficacité prématurément
- Bien comprendre la logique de fonctionnement du système avant de penser à l'optimiser
- Étapes à suivre [BR05] :
  - Fournir des chemins d'accès efficaces
    Par ajout d'associations redondantes ▶
  - Réorganiser les calculs pour améliorer l'efficacité
  - Sauvegarder des calculs intermédiaires pour éviter de refaire des calculs

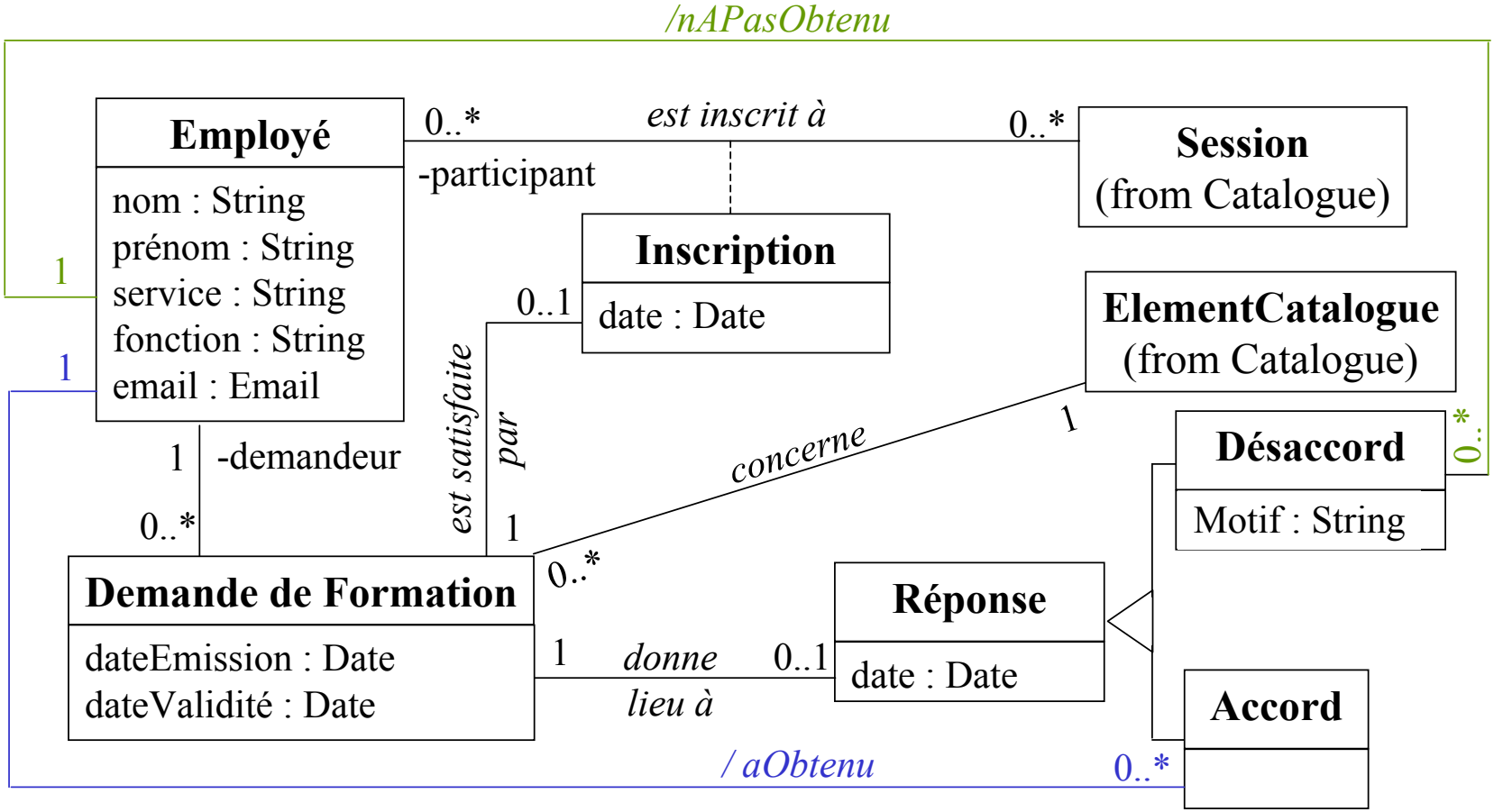
Conception des classes (10/13)

6. **Optimiser la conception** (exemple)



Conception des classes (10/13)

6. Optimiser la conception (exemple)



Conception des classes (11/13)

7. Réifier les comportements

- **Réification** = « promouvoir au rang d'objet quelque chose n'étant pas un objet »
- **Réification d'un comportement** $\Rightarrow$ encodage du comportement dans un objet et décodage lors de son exécution
- Existence de *patterns* comportementaux réifiant les comportements

Conception des classes (12/13)

8. Ajuster l'héritage

Étapes à suivre [BR05] :

- Réorganiser les classes et les opérations pour accroître l'héritage
- Extraire les comportements communs à plusieurs groupes de classes

  ⇒ Création de super-classes abstraites
- Employer la délégation pour partager les comportements en cas d'impossibilité d'utiliser l'héritage du point de vue sémantique

  Délégation = « Mécanisme d'implémentation où un objet, disposant d'une opération, y répond en la transmettant à un autre objet pour qu'il l'exécute »

Conception des classes (13/13)

9. Organiser la conception des classes

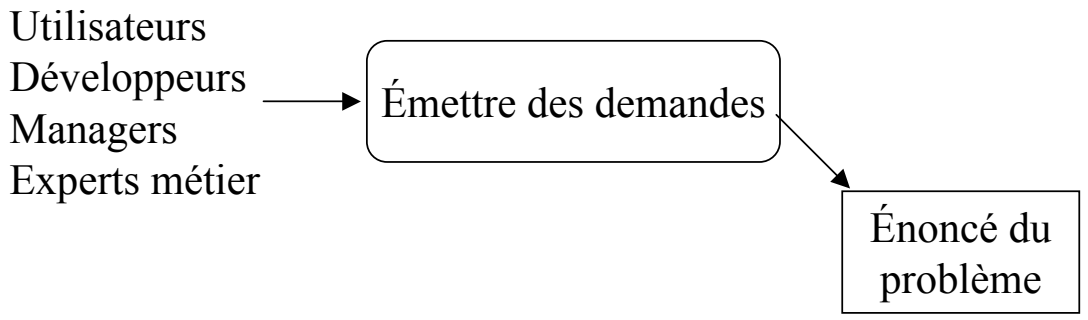
Étapes à suivre [BR05] :

- Masquer l'information interne de la vue externe
  - Séparer soigneusement les spécifications externes de l'implémentation interne
  - Limiter la portée de la navigation dans le modèle de classes
  - Ne pas accéder aux attributs étrangers
  - Définir des interfaces à haut niveau d'abstraction
  - Cacher les objets externes
  - Éviter les appels en cascade de méthodes
- Maintenir la cohérence des entités
  - Séparer la stratégie de l'implémentation
  - **Méthode de stratégie** = instruction d'E/S, structures conditionnelles, et accès aux données
  - **Méthode d'implémentation** = encodage de l'algorithme sans prise de décision, ni d'émission d'hypothèses, ni de traitement par défaut
- Raffiner les packages
  - Définir des packages pour les éditer et les compiler plus facilement et pour la commodité de l'équipe de développement
  - Définir des packages avec des interfaces minimales et bien définies
  - Définir des packages cohérents et les organiser autour d'un thème commun

Résumé du processus (1/3)

Résumé du processus (1/3)

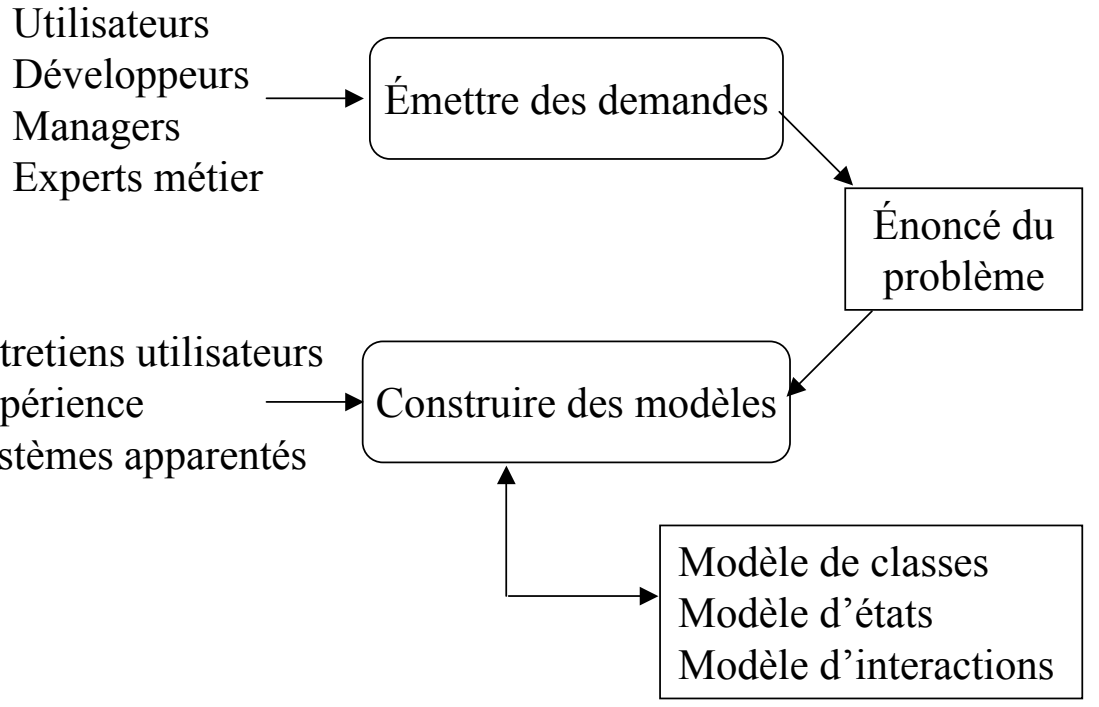
Spécifications initiales du système



Résumé du processus (1/3)

Spécifications initiales du système

Analyse :
Analyse du domaine
Analyse de l'application

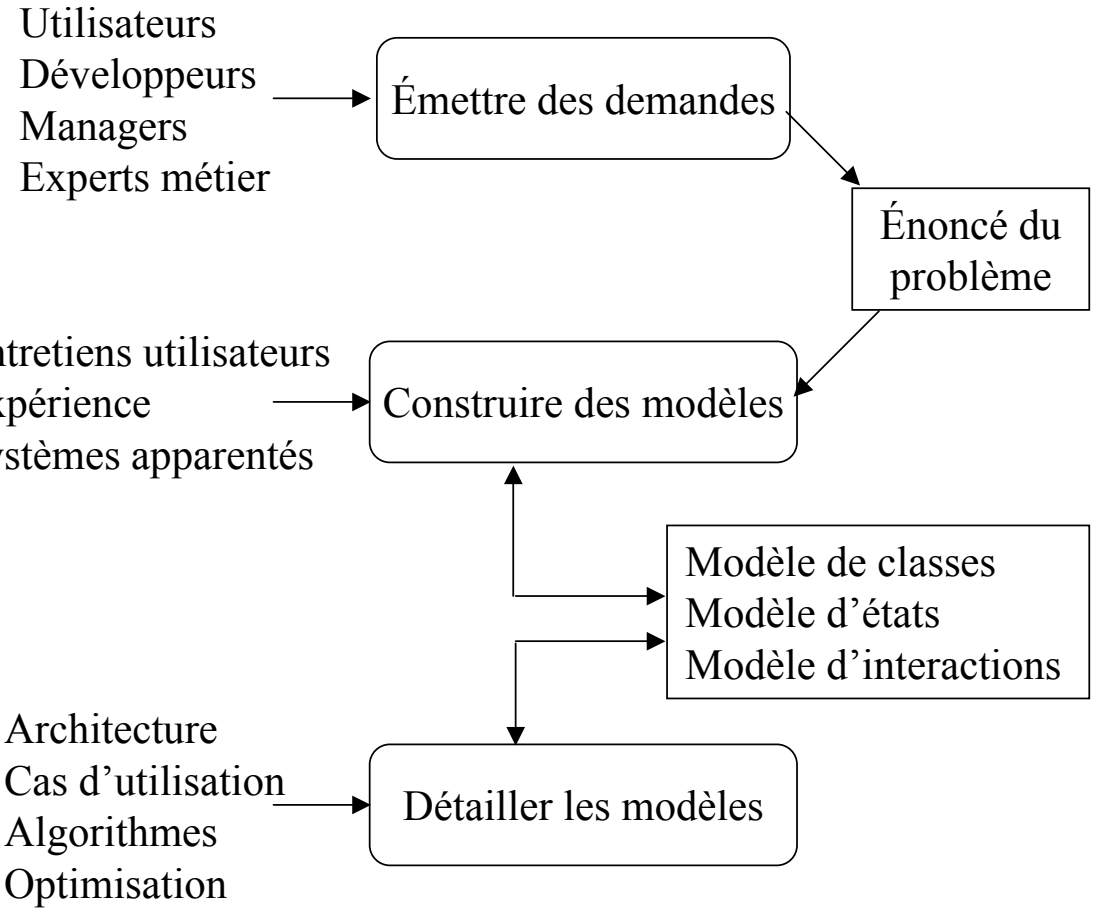


Résumé du processus (1/3)

Spécifications initiales du système

Analyse :
Analyse du domaine
Analyse de l'application

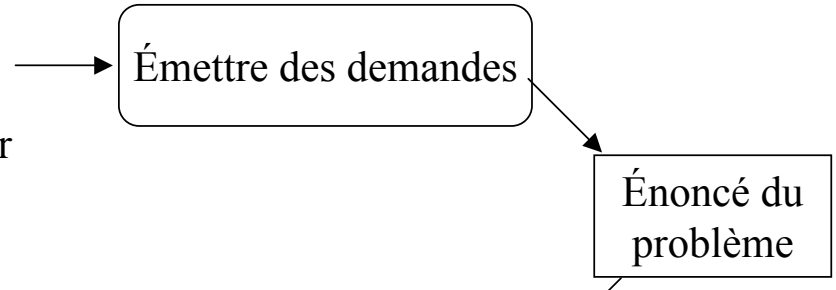
Conception :
Conception du système
Conception des classes



Résumé du processus (1/3)

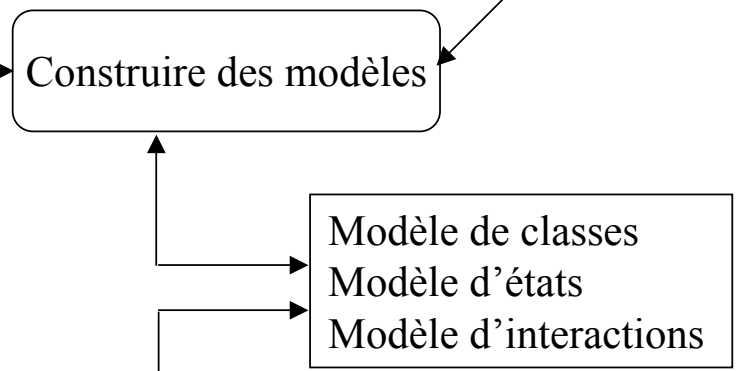
Spécifications initiales du système

Utilisateurs
Développeurs
Managers
Experts métier



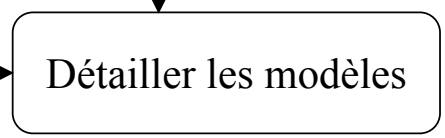
Analyse :
Analyse du domaine
Analyse de l'application

Entretiens utilisateurs
Expérience
Systèmes apparentés



Conception :
Conception du système
Conception des classes

Architecture
Cas d'utilisation
Algorithmes
Optimisation



Il s'agit d'un processus itératif

Résumé du processus (2/3)

- **Spécifications initiales** : genèse de l'application
  - Entrée : idée brute d'une nouvelle application
  - Sortie : énoncé du problème = point de départ d'une analyse approfondie
- **Analyse** : construction de modèles pour bien comprendre les exigences en termes de ce qui doit être réalisé et non de comment cela doit être réalisé
  - Entrée : énoncé du problème
  - Sortie : modèles de classes, d'états et d'interactions
  - 2 stades :
    - **Analyse du domaine** : pour capturer les connaissances générales d'une application
    - **Analyse de l'application** : pour capturer les aspects informatiques de l'application visibles pour les utilisateurs

Résumé du processus (3/3)

- **Conception** : détermination de comment l'application doit être réalisée
  - Entrée : modèles d'analyse
  - Sortie : modèles de classes, d'états et d'interactions détaillés
  - 2 stades :
    - **Conception du système** : mise au point de l'architecture du système
    - **Conception des classes** : Augmentation et ajustement des modèles issus de l'analyse pour qu'ils soient prêts à être implémentés