COO

Licence 3 Informatique parcours MIAGE Semestre 6

Philippe Renevier-Gonin
Simon Urli
Chrisitian Brel

http://deptinfo.unice.fr/~renevier/COOA

Université Nice SOPHIA ANTIPOLIS

D'après un cours de Ph. Collet

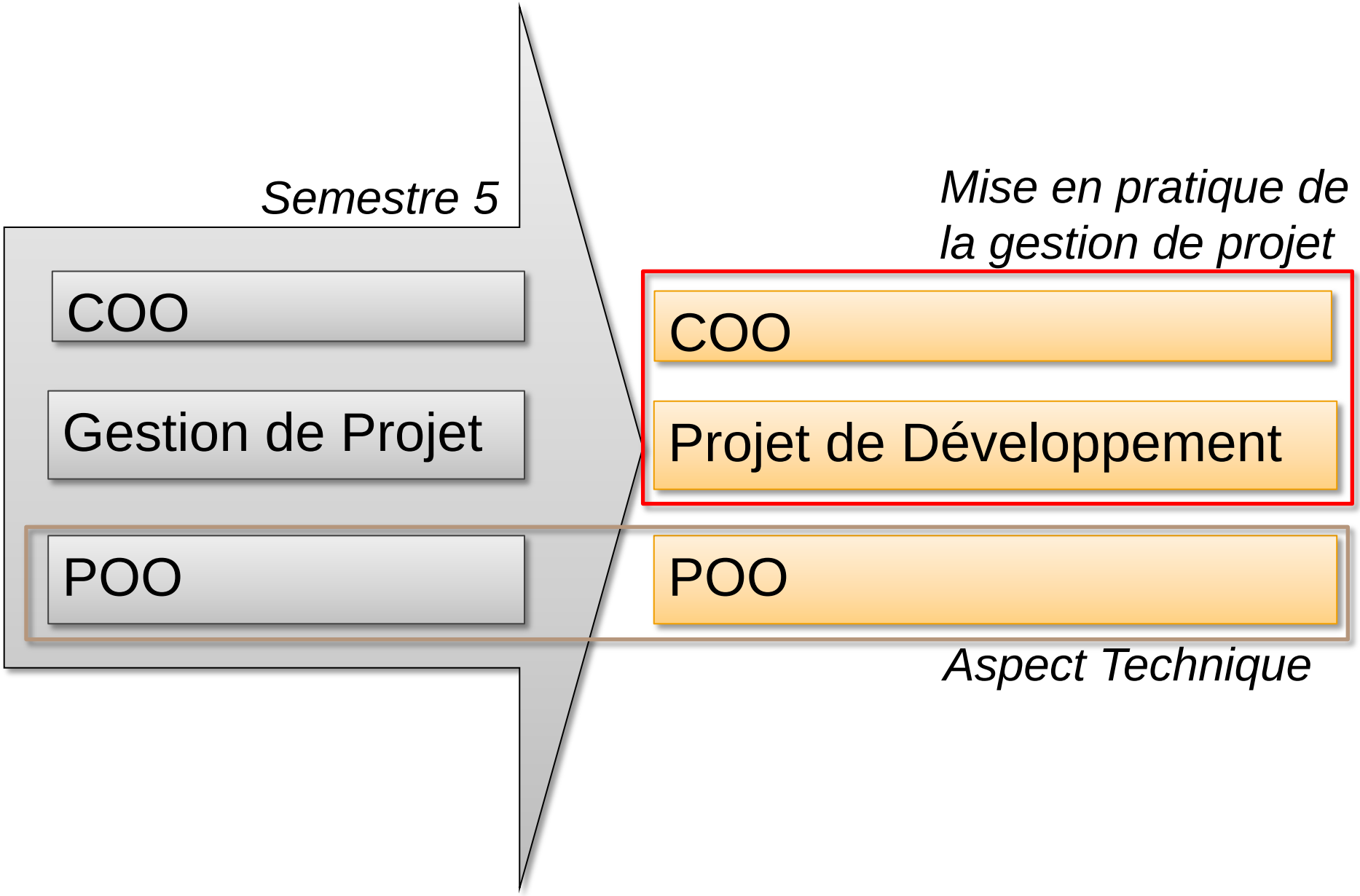
Pourquoi ?

- Connaître (l'essentiel d') UML ne suffit pas pour réaliser de bonnes conceptions
- UML n'est qu'un langage, une notation…
- En plus, il faut :
  - Savoir penser et coder en termes d'objets (cf. POO)
  - Savoir organiser l'analyse et la conception
  - À grande échelle
  - A l'aide de guides méthodologiques adaptés au(x) problème(s)
  - Sans vision dogmatique
  - Mais en réutilisant des bons principes communs

Objectif du cours

- Vous sensibiliser à l'ingénierie des systèmes d'information.
- Vous donner une vision complète de l'activité de conception au sens large (analyse, conception, spécification) dans le cycle de développement logiciel.
- Liaison IHM (maquette) – Système Information
- Vous faire mettre en pratique les différentes activités qui constituent l'étape de conception d'un processus de développement.
- Vous faire travailler en équipe, mais aussi à plusieurs équipes communicantes !

Positionnement dans le programme du GL



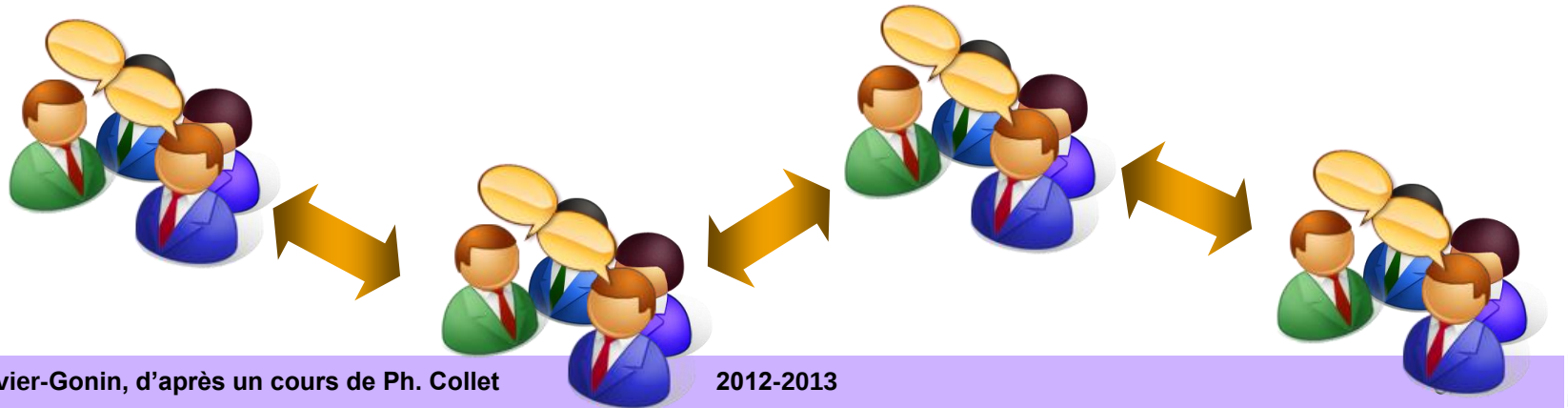
Programme

- Introduction et motivations
- Rappels sur UML et les activités d'analyse et de conception
- Processus de conception : définition
- Rappel
  - Processus Unifié UML : RUP
  - Processus agiles
- Conclusion

- 6h de cours
- 1er TD : ce matin et demain après-midi

Organisation

- Crédit : coeff 2
- Enseignement
  - 6h de cours
  - 30h de TD
- Evaluation
  - Projet tout le long de l'enseignement par équipe de 4
  - 2 rendus :
    - à mi parcours (40 %)
    - Soutenance (20 %) et final (40%)



Rappels

(ou pas)

Génie logiciel

- Définition (*software engineering*)
  - ensemble de méthodes, techniques et outils pour la production et la maintenance de composants logiciels de qualité
- Principes
  - rigueur et formalisation, séparation des préoccupations, modularité, abstraction, prévision du changement, approche générique, approche incrémentale
- Besoins
  - Langages — *pour décrire*
  - Outils — *pour manipuler*
  - **Méthodes** — *pour décider*
  - Théories — *pour démontrer*
  - Professionnels — *pour réaliser*
  - Logistique — *pour supporter*

Qualités pour l'utilisateur (phases d'exploitation)

- **Fiabilité = Validité + Robustesse**
  - Validité ≡ correction, exactitude : assurer exactement les fonctions attendues, définies dans le cahier des charges et la spécification, en supposant son environnement fiable
  - Robustesse: faire tout ce qu'il est utile et possible de faire en cas de défaillance: pannes matérielles, erreurs humaines ou logicielles, malveillances…
- **Convivialité**
  - Réaliser tout ce qui est utile à l'utilisateur, de manière simple, ergonomique
- **Performance**
  - Utiliser de manière optimale les ressources matérielles : temps d'utilisation des processeurs, place en mémoire, précision…
- **Extensibilité, Compatibilité, Intégrité (sécurité)…**

Qualités pour le développeur (phases de devt)

- **Documentation**
  - Tout ce qu'il faut, rien que ce qu'il faut, là où il faut, quand il faut, correcte et adaptée au lecteur
- **Modularité = Fonctionnalité + Interchangeabilité + Réutilisabilité**
  - Fonctionnalité : Localiser un phénomène unique, facile à comprendre et à spécifier
  - Interchangeabilité : Pouvoir substituer une variante d'implémentation sans conséquence fonctionnelle (et souvent non-fonctionnelle) sur les autres parties
  - Réutilisabilité : Aptitude à être réutilisé, en tout ou en partie, tel que ou par adaptation, dans un autre contexte : autre application, machine, système…
- **Vérifiabilité**
  - Aptitude d'un logiciel à être testé
- **Portabilité**
  - Aptitude d'un logiciel à être transféré dans des environnements logiciels et matériels différents

Génie logiciel : le défi

❒ Contradictions apparentes

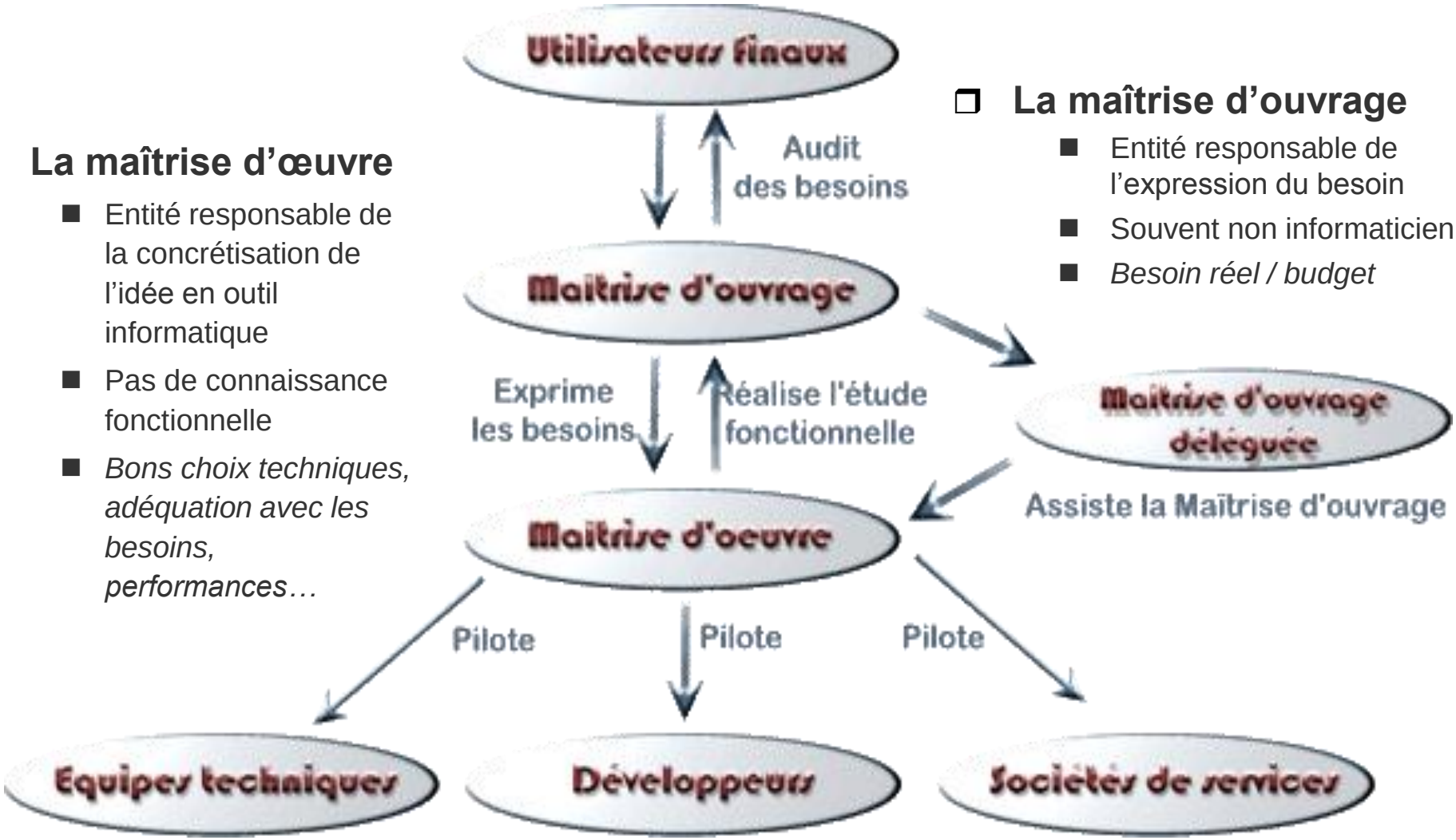
- Qualités vs coût du logiciel
- Qualités pour l'utilisateur vs qualités pour le développeur
- Contrôler vs produire

❒ Conséquences

- ☞ Chercher sans cesse le meilleur compromis
- ☞ Amortir les coûts
  - ◆ Premier exemplaire de composant coûteux à produire ou à acheter, puis amortissement…
- ☞ Gérer un projet informatique de manière spécifique (avec des **méthodes** spécifiques)

Organisation d'un projet informatique

- **La maîtrise d'œuvre**
  - Entité responsable de la concrétisation de l'idée en outil informatique
  - Pas de connaissance fonctionnelle
  - *Bons choix techniques, adéquation avec les besoins, performances…*



- **La maîtrise d'ouvrage**
  - Entité responsable de l'expression du besoin
  - Souvent non informaticien
  - *Besoin réel / budget*

Comment gérer un projet informatique ?

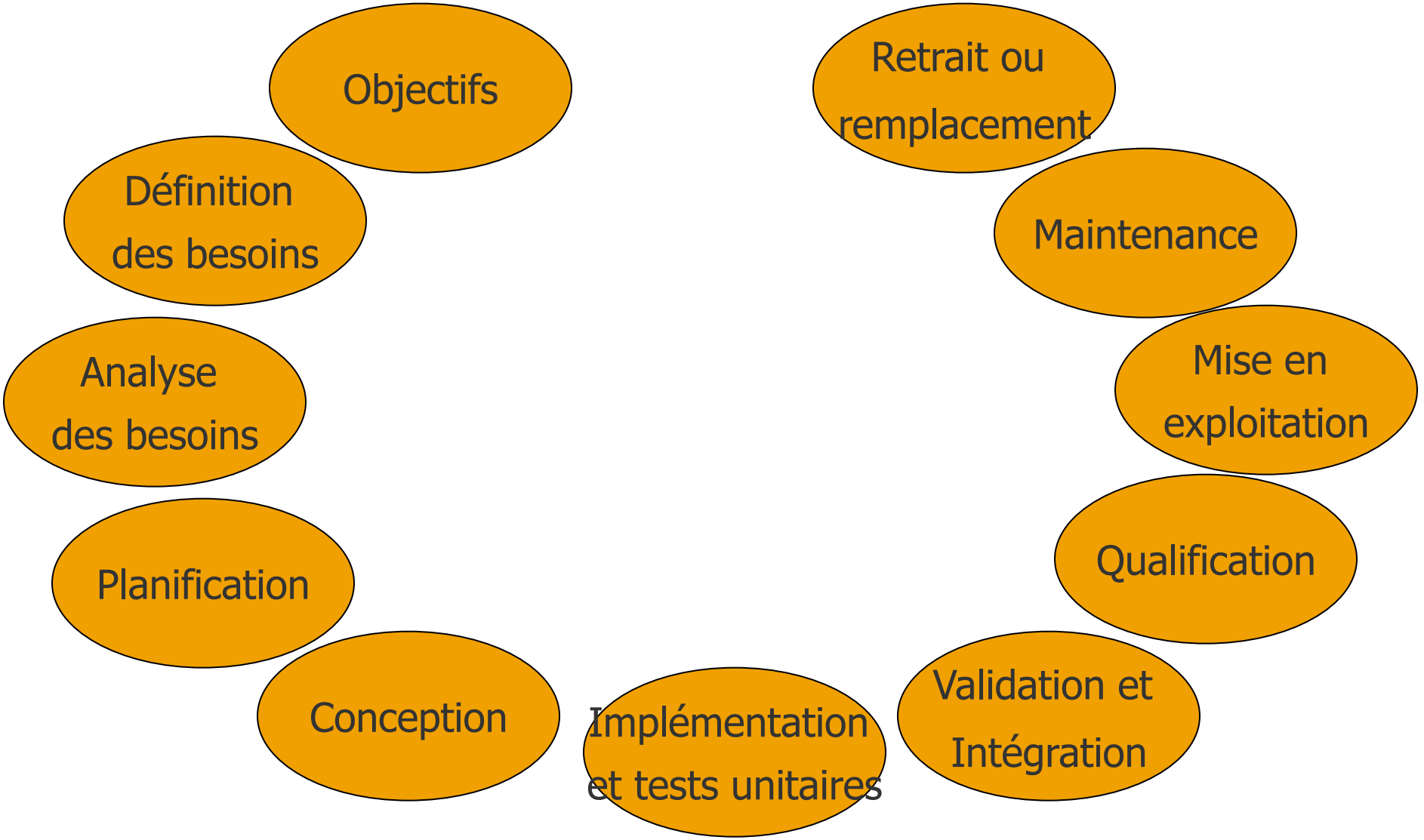
❒ Définir et utiliser des méthodes

- spécifiant des processus de développement
- organisant les activités du projet
- définissant les artefacts du projet
- se basant sur des modèles
  - pas des modèles objets, des modèles de **cycle de vie**

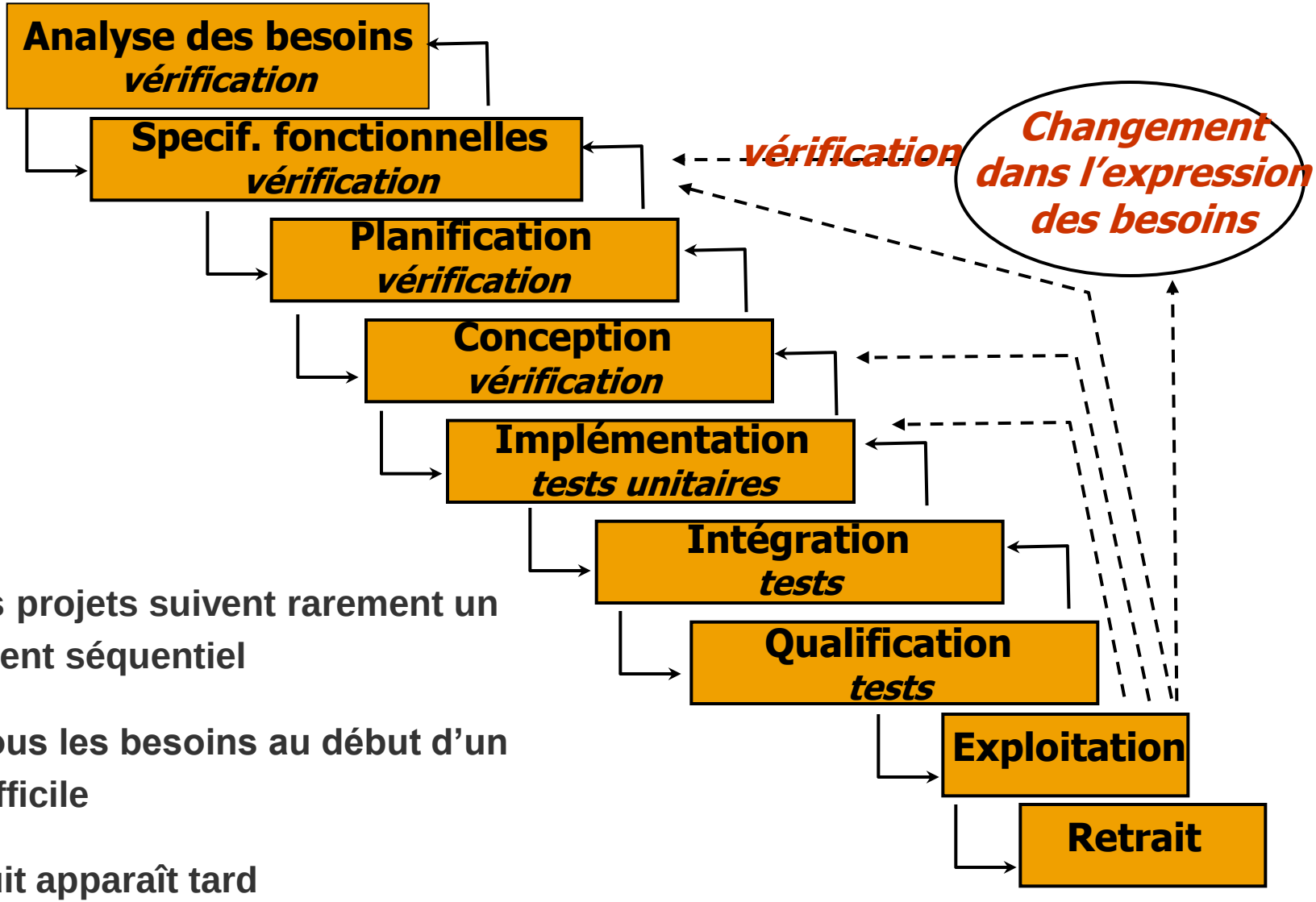
❒ Modèles de cycle de vie

- organiser les différentes phases du cycle de vie pour l'obtention d'un logiciel fiable, adaptable et efficace, etc.
- guider le développeur dans ses activités techniques
- fournir des moyens pour gérer le développement et la maintenance

cycle de vie: organiser des phases

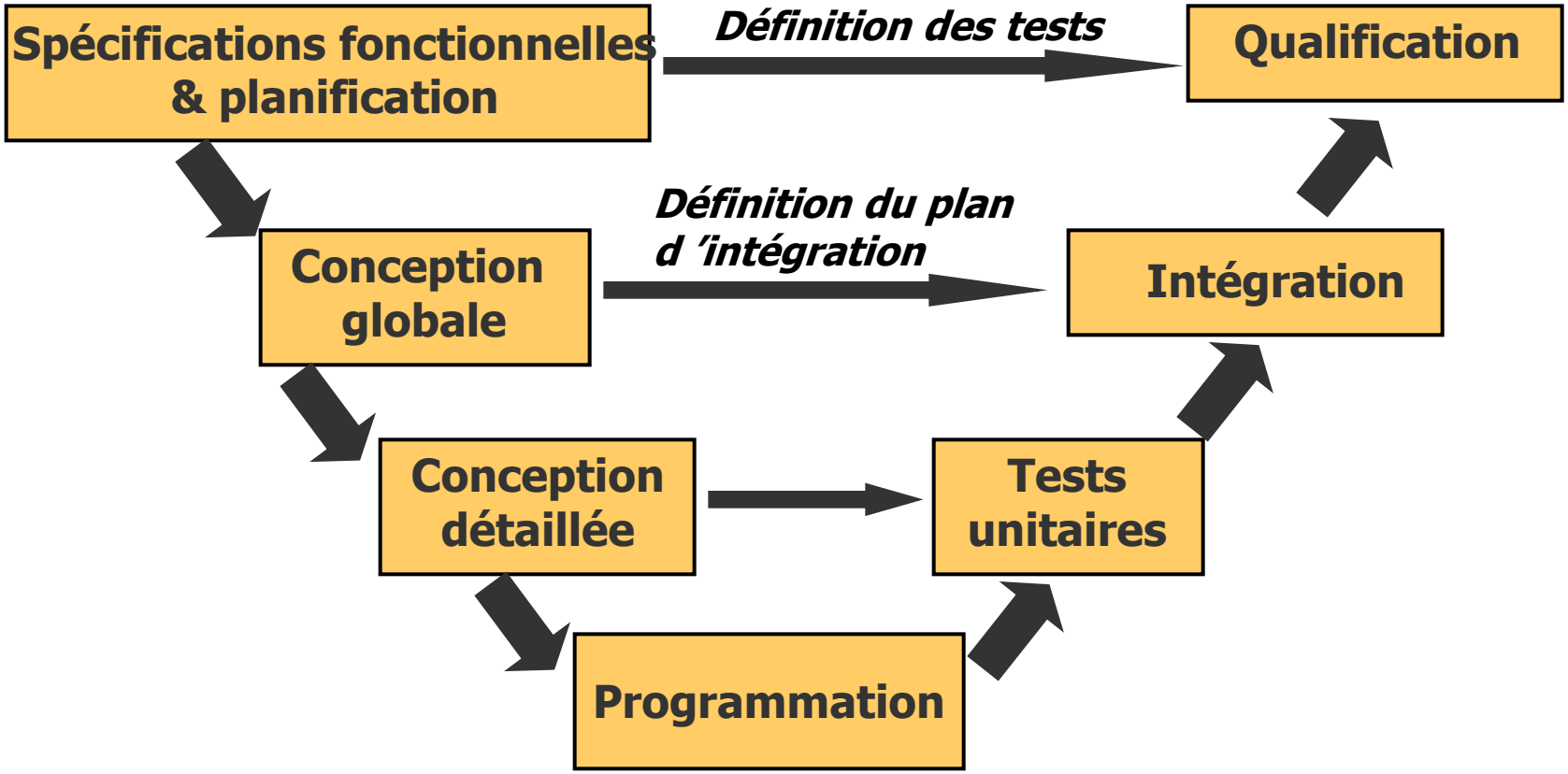


Modèle en cascade (1970)



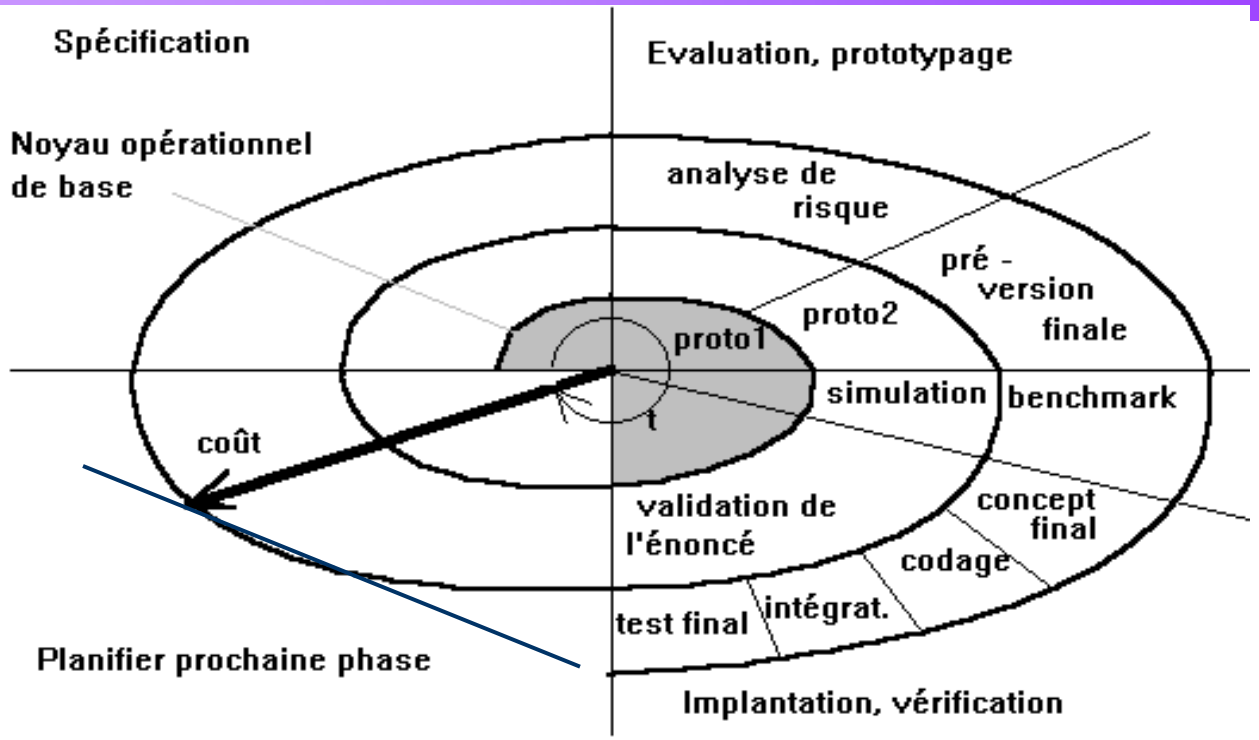
- Les vrais projets suivent rarement un développement séquentiel
- Établir tous les besoins au début d'un projet est difficile
- Le produit apparaît tard
- Echecs majeurs sur de grands systèmes

Modèle en V



- Meilleure anticipation
- Bonne structuration des tests
- Cadre de développement rigide
- Le produit apparaît toujours tard

Modèle en spirale (Boehm, 1988)



- **Incréments successifs => itérations**
  - Approche souvent à base de prototypes
  - Nécessite de bien spécifier les incréments
  - Figement progressif de l'application
- **Mais gestion de projet pas évidente**
- **Pourtant, les méthodes objets dérivent de ce modèle !**

Méthodes et processus de conception

Université Nice SOPHIA ANTIPOLIS

Définitions

- **Modèle (comme UML)**
  - Notation pour représenter, formalisme
  - Pour partager l'information, uniformiser, mécaniser (transformation vers le code)
- **Méthode**
  - Guide plus ou moins formalisé
  - Démarche reproductible permettant d'assurer la qualité
- **Processus**
  - A peu près la même définition qu'une méthode…
  - « ensemble de directives et d'étapes destinées à produire des logiciels de manière contrôlée et reproductible, avec des coûts prévisibles, présentant un ensemble de bonnes pratiques autorisées par l'état de l'art »

Définitions (suite)

- **Une méthode définit**
  - des concepts de modélisation
    - obtenir des modèles à partir d'éléments de modélisation, sous un angle particulier, représenter les modèles de façon graphique
  - une chronologie des activités (quand construit-on les modèles)
  - un ensemble de règles et de conseils
- **Une méthode peut donc reposer sur un processus…**
  - **= notation + artefact (ce qui est manipulé) + démarche**
- **On distingue souvent :**
  - **Les processus de développement technique**
  - **Les processus de gestion du développement lui-même**

Evolution des méthodes

- **Premières méthodes** (années 60-70)
  - Guide par découpage en sous-problèmes, analyse fonctionnelle
  - Méthodes d'analyse structurée
- **Méthodes orientées bases de données** (années 80 et suivantes)
  - Concevoir la base de données est central au système d'information
  - Méthodes globales qui séparent données et traitements
- **Méthodes pour la conception avec réutilisation** (années 95 et suivantes)
  - Frameworks, Design Patterns, bibliothèques de classes (orientées objets)
  - Méthodes incrémentales unifiées par une notation commune (UML)

Quelles activités sont couvertes par les méthodes ?

❒ Cinq grandes activités émergent de la pratique de développement et de gestion de projet

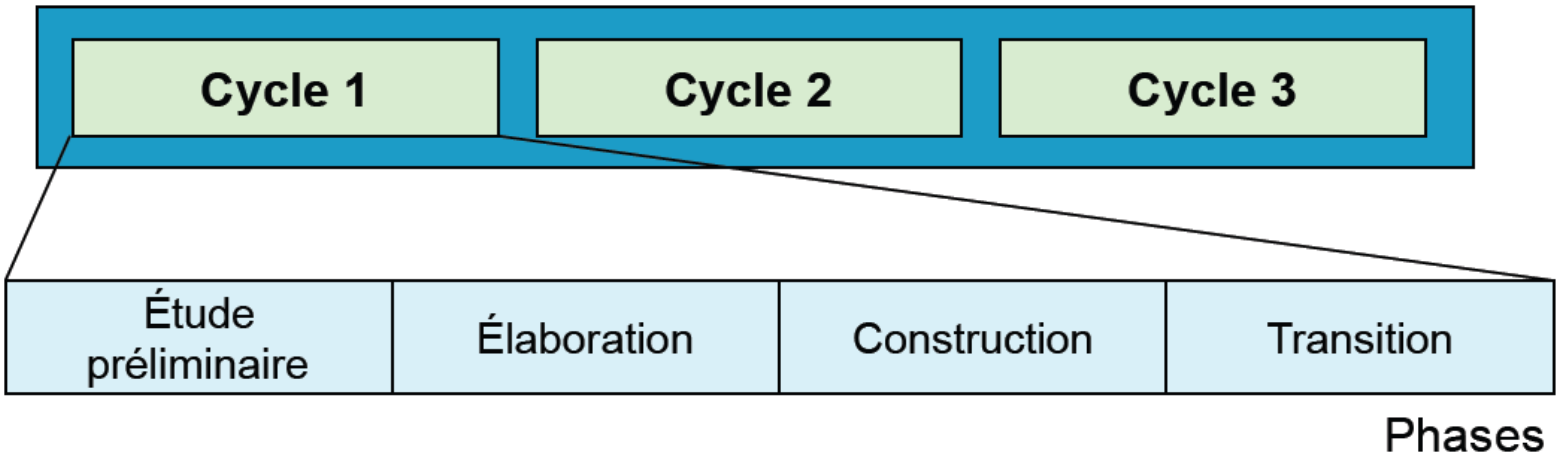
dans ce cours…

- Spécification des besoins
- Analyse (recherche du bon système)
- Conception (liée à l'implémentation, la construction)
- Implémentation (activité la plus coûteuse)
- Tests (activité la plus négligée)

Unified Software Development Process : Principes

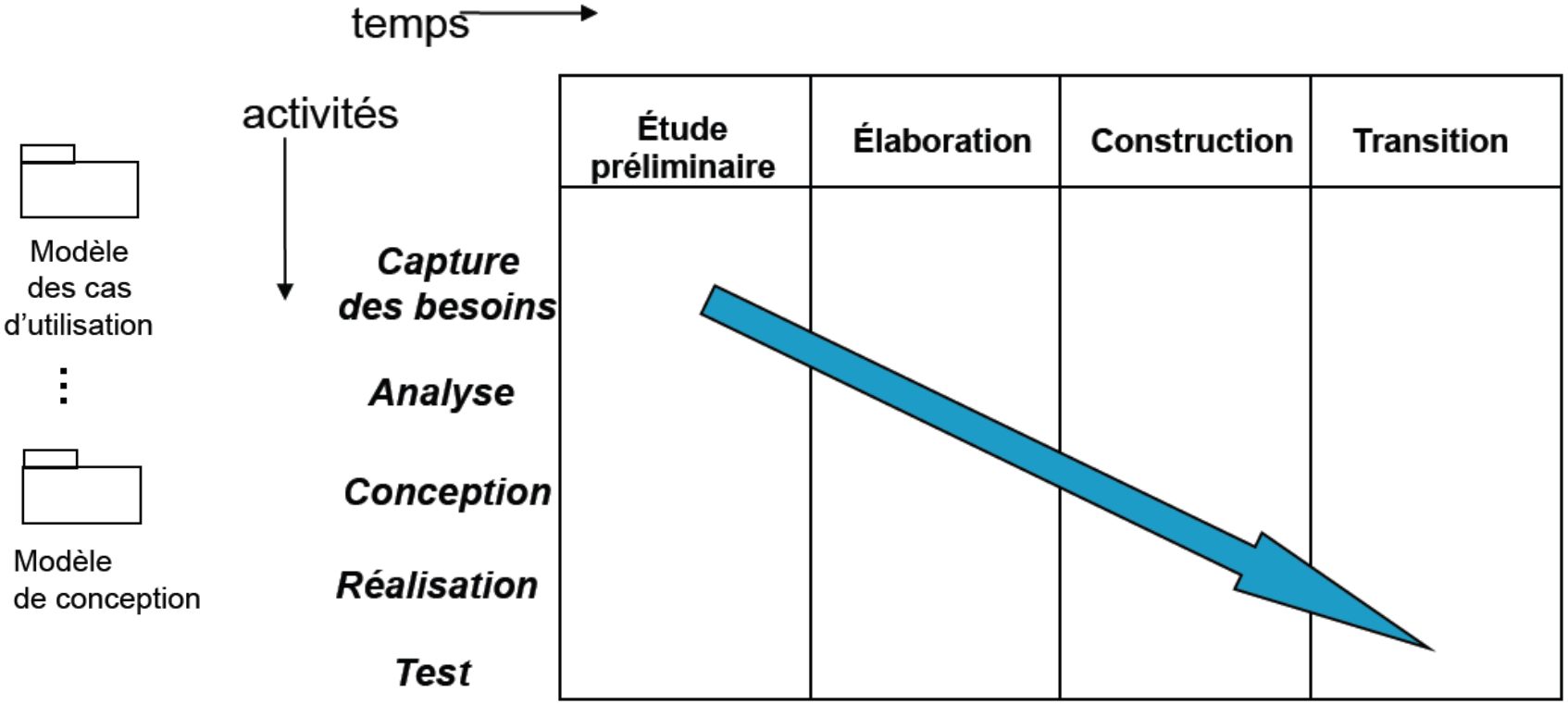
- Considérer un produit logiciel quelconque par rapport à ses versions
  - un cycle produit une version

- Gérer chaque cycle de développement comme un projet ayant quatre phases
  - chaque phase se termine par un point de contrôle (ou jalon) permettant de prendre des décisions



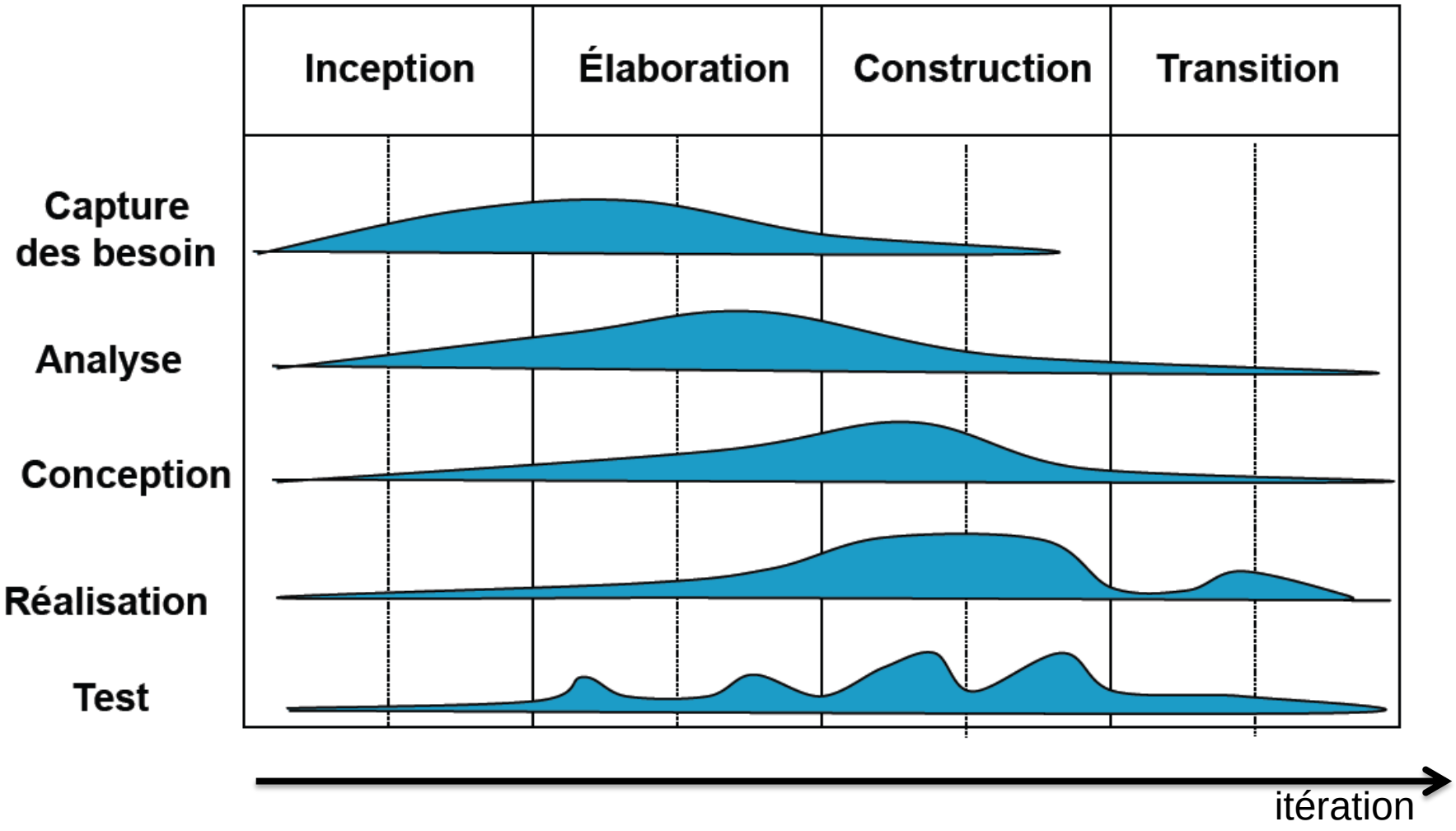
Phases et activités

- Le cycle met en jeu des activités
  - vue du développement sous l'angle technique
  - Les activités sont réalisées au cours des phases, avec des importances variables, lors d'**ITERATION**
  - Les activités consistent notamment à créer des modèles



Efforts par itération dans une phase

- Chaque phase spécifie les activités à effectuer



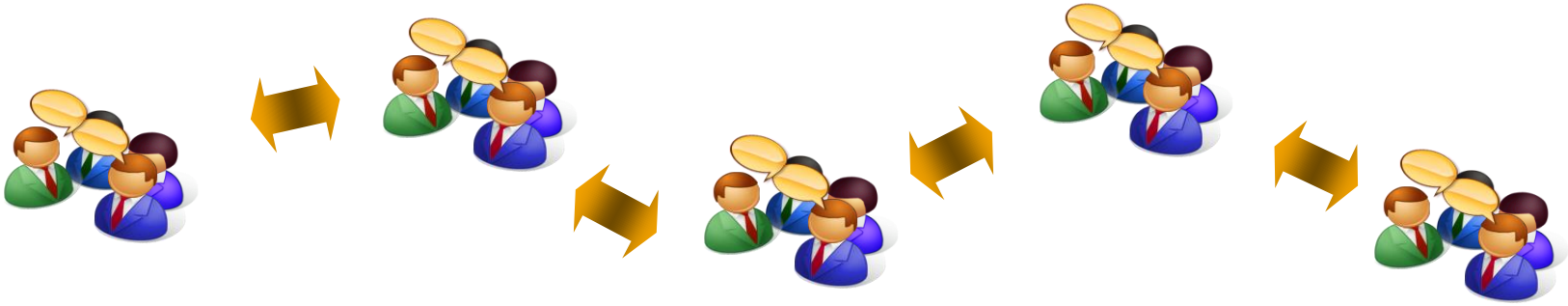
Déroulement des TD

Organisation, travail attendu, erreurs classiques

Université Nice SOPHIA ANTIPOLIS

TD, équipes, sujets…

- 6 équipes de 5
- Les équipes sont formées par consensus par les étudiants d'un même groupe de TD
- Les sujets sont tirés au sort



- Chaque sujet a au moins un lien avec trois autres sujets, il nécessitera donc une collaboration entre les deux équipes concernées, au moins au niveau des cas d'utilisation
- Chaque séance de TD est l'occasion d'avancer dans une partie de la modélisation

Déroulement et évaluation

- **5 séances de TD (3h) pour démarrer la conception : 1ère itération**
- **5 séances de TD (3h) pour terminer la conception : 2ième itération**
- **Portable(s) : travail d'abord sur PAPIER !!**
- **Travail itératif**
  - **RENDU A MI PARCOURS**
  - **DOSSIER ET SOUTENANCE FINALE**
- **Le chargé de TD guide les équipes et joue le rôle de client**
- **25 avril :** Dossier rendu

03 Mai : 1 soutenance par équipe (20 minutes + 10 minutes)

Contenu du dossier de conception

- Ensemble des diagrammes de cas d'utilisation, cohérents vis-à-vis des autres systèmes d'informations.
- Diagramme détaillé de classes (signature typée des attributs et des méthodes).
- Un diagramme de séquences par cas d'utilisation
- Une maquette de(s) IHM(s) en lien avec les diagrammes
- Au moins un diagramme d'activités (il est bien sûr pertinent de l'utiliser pour détailler un cas d'utilisation particulièrement complexe...)
- Au moins un diagramme d'états (il est bien sûr pertinent de l'utiliser pour détailler une classe dont les états sont remarquables...)
- Des spécifications OCL (préconditions, postconditions, invariants) sur les classes qui vous semblent les plus pertinentes.

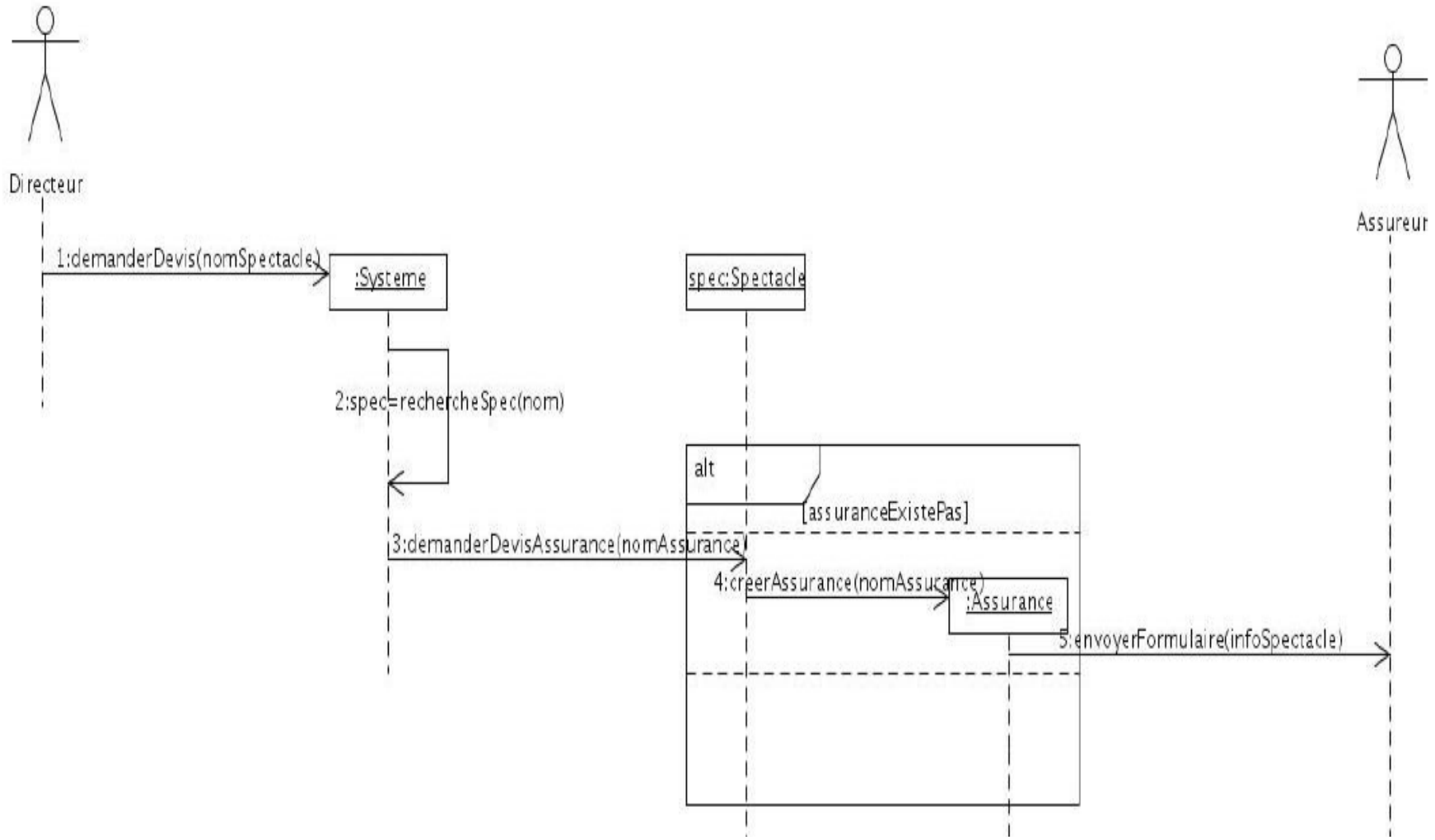
Maquette de l'IHM : concrétiser la conception

- Problème d'abstraction au niveau de la conception
- Faire la maquette
  - « Concrétiser » l'utilisation du système
- Y associer les use-cases
  - si le use case fait intervenir un acteur humain et qu'il n'y a pas d'Ihm, alors ce n'est pas un use-case
- Y associer diagramme de séquence
  - Déterminer comment les infos sont présentées, comment les paramètres sont entrés
- (voire Y associer diagramme de classe pour la création ou la manipulation d'objet)
- Vérifier la Cohérence

Erreurs classiques

- Faux cas d'utilisation (des interactions entre humains)
- Trop modéliser n'est pas modéliser juste. Bien délimiter le sujet, comprendre ce que l'on me demande et bien le faire.
- Les diagrammes de classe
  - les acteurs apparaissent très souvent sous forme de classe !
  - relation, classe, attributs, arités, opérations... construites au fur et à mesure.
- Les diagrammes de séquence
  - Les paramètres qui tombent du ciel,
  - des séquences illogiques qui ne mènent à rien....
- OCL : certains ont fait l'impasse, les autres ont réussis à condition de ne pas tomber dans le piège des collections !
- Manque de cohérence entre les diagrammes...
  - Ne pas hésiter à revenir en arrière

Exemple d'incohérence : ce qui a été rendu (1/3)



Description: Le directeur envoi une demande de devis, pour un spectacle et une assurance donné, via l'interface du Système, celui ci entame alors une recherche du spectacle concerné pour ensuite envoyer une requête de demande de devis. Une instance d'assurance est alors crée pour envoyer le formulaire demandé à l'assurance concernée.

Exemple d'incohérence : problème au début (2/3)

Nom du devis : Mariage PRG

Demander le devis

- demanderDevis( ) ?
- D'où vient le nom de l'assurance (création) ?
- Il en faut +

Nom du devis : Mariage PRG

Il n'y a pas d'assurance associée à l'événement.

Assureur : Azur Assurance ▼

Choisir Assureur

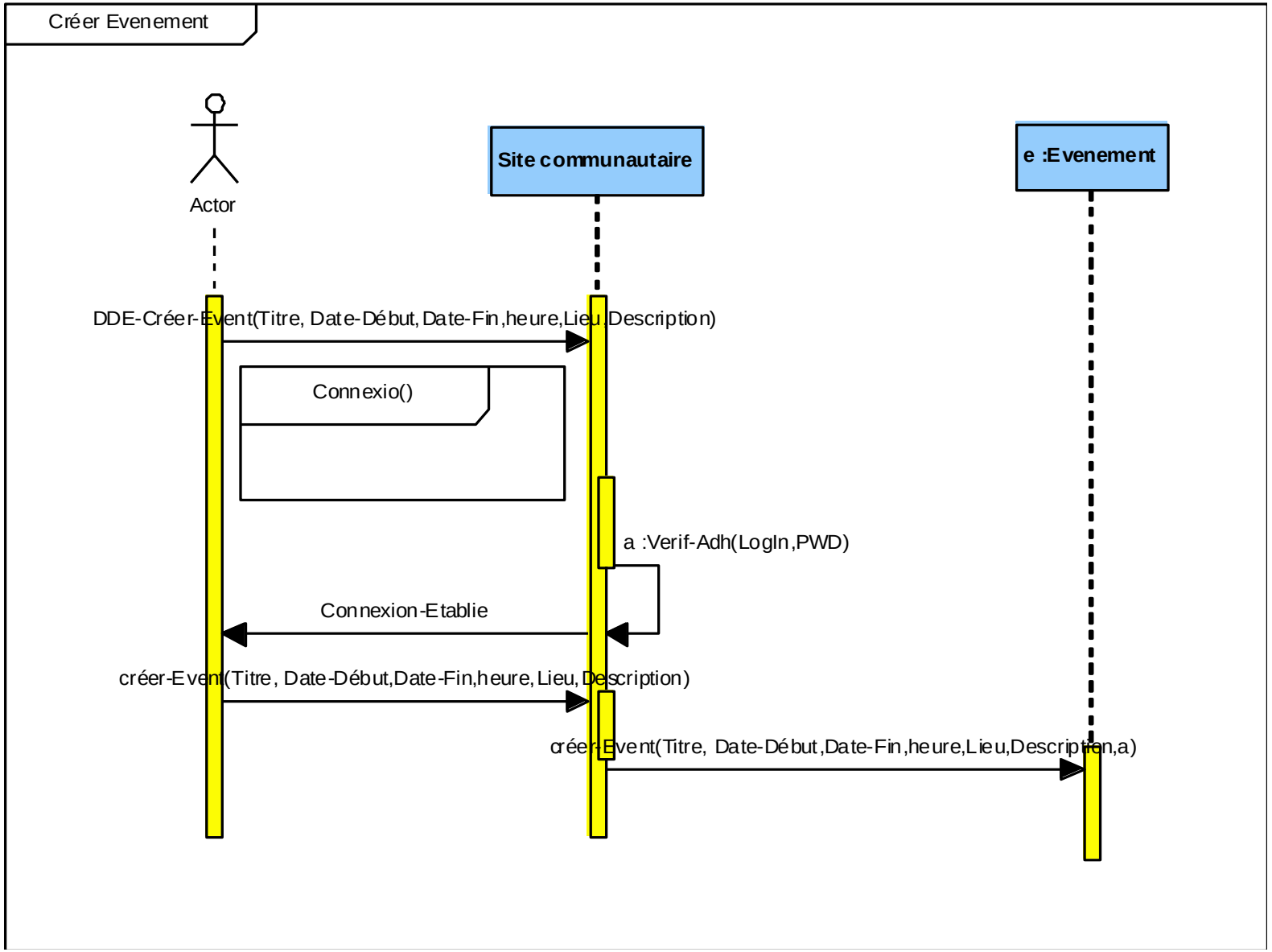
Système

```
+parcourirDemandeMairie()
+getListeContratAssurance()
+getListePersonnel()
+getListeEquipement()
+getListePlacesDispo()
+verifierPlacesDispo()
+demandeAutorisation()
+creerRepresentation()
+creerBillet()
+recuClient()
+envoyerErreur()
+rechercherSpec()
+ajouterSinistre()
+demandeAssurance()
+chercherDevis()
+validationDevis()
+MiseAJourPlacesDispo()
+calculDépenseRepresentation(): float
+calculBénéficeRepresentation(): float
+calculBénéficesAnnuel(): float
```

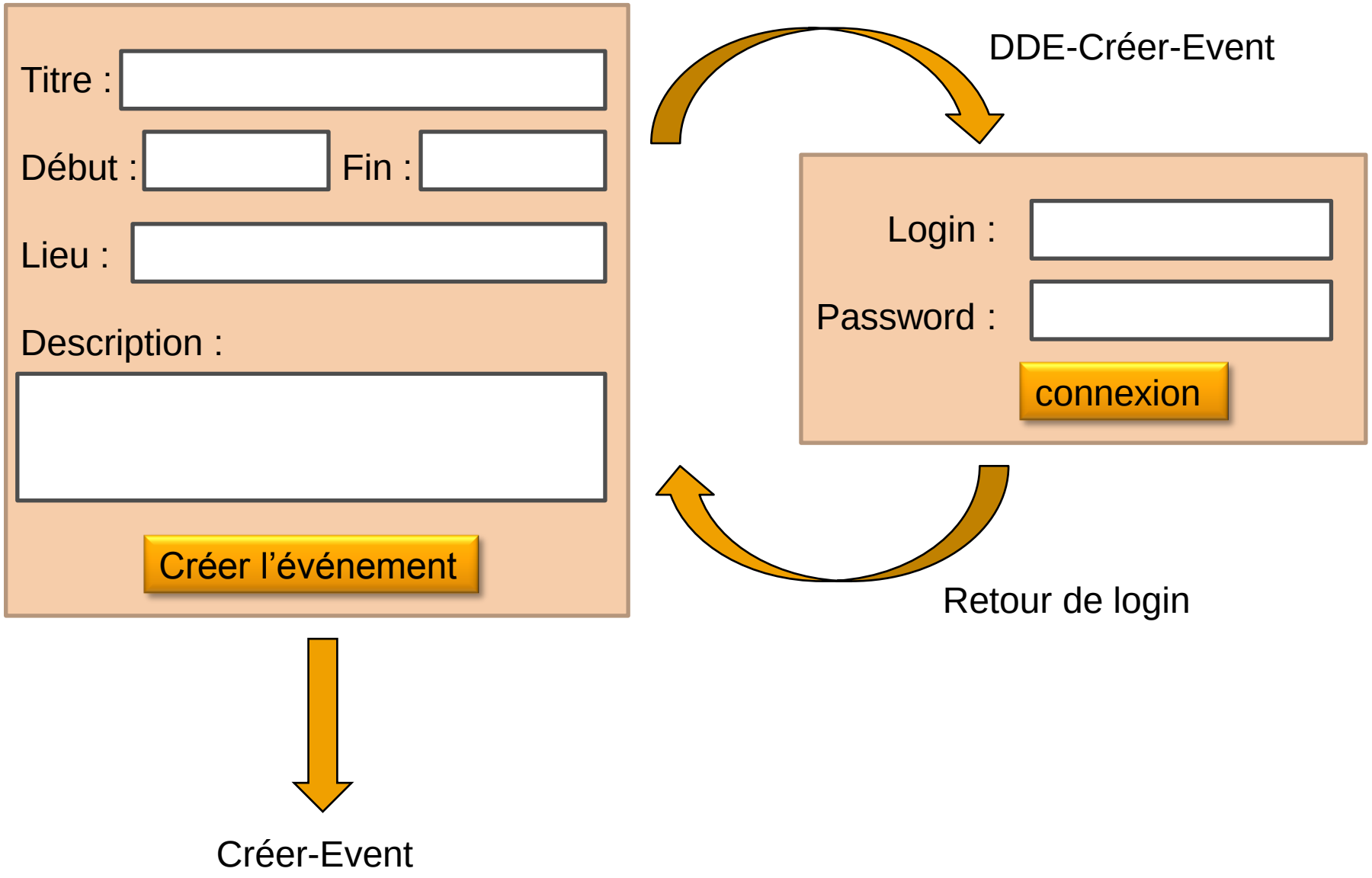
Exemple d'incohérence : problème à la fin (3/3)

- Message vers un Acteur Humain
  - L'acteur est présent devant le système au moment où la séquence l'atteint...
  - Possible… mais pensez vous que les entreprises partenaires vont mettre un employé à attendre vos requêtes 24h/24 ?
- Solution
  - Parler au système : « stockage » des requêtes
  - Use Case / séquence pour traiter les requêtes par l'acteur humain ciblé…
  - Se mettre d'accord si l'acteur n'est pas dans l'entreprise… (peut être hors sujet pour un des groupes)

Autre exemple d'incohérence (1/2)

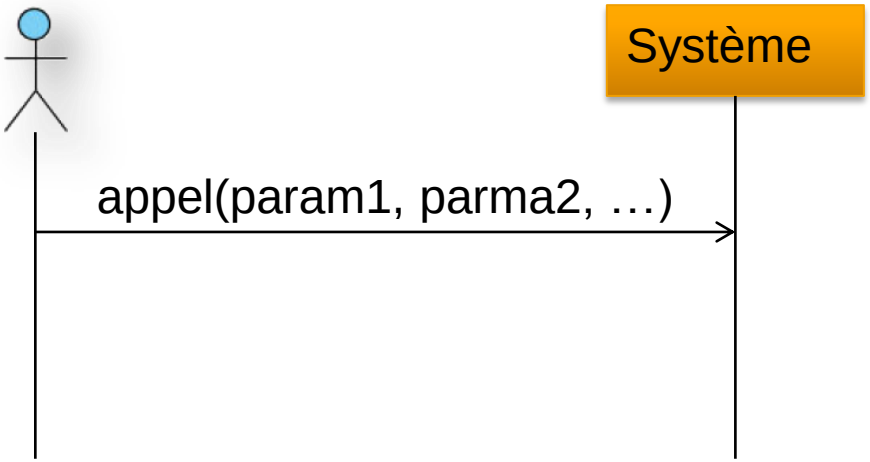


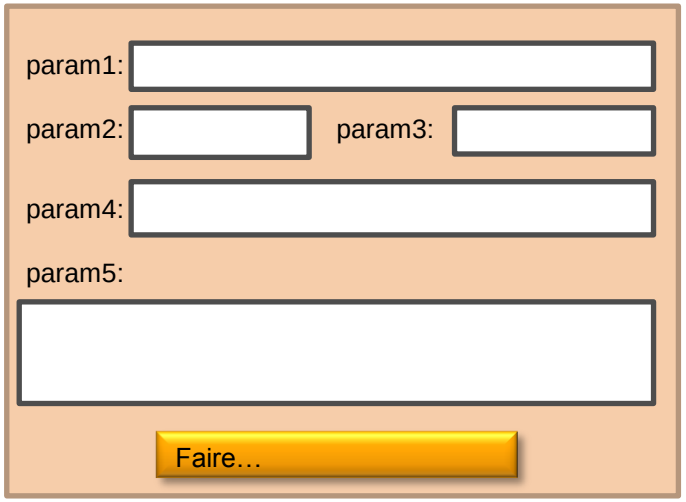
Autre exemple d'incohérence (2/2)



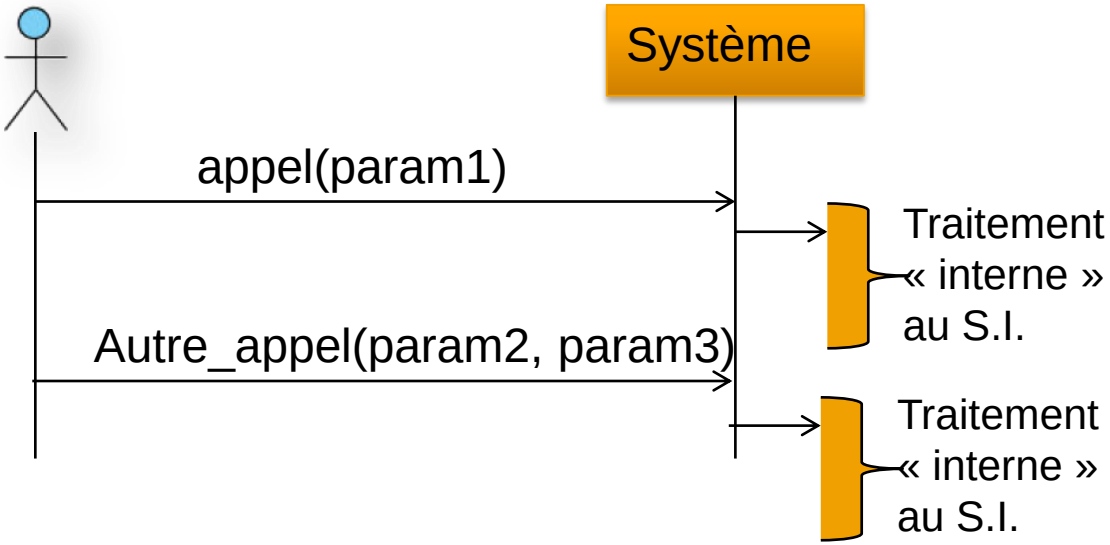
Variation dans le dialogue avec les utilisateurs

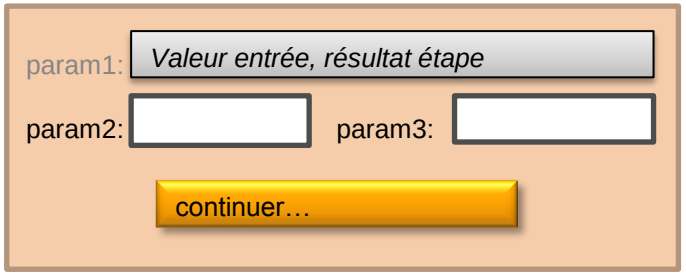
- tout **afficher** ou tout **demander** en une seule fois





- **afficher** ou **demander** au fur et à mesure





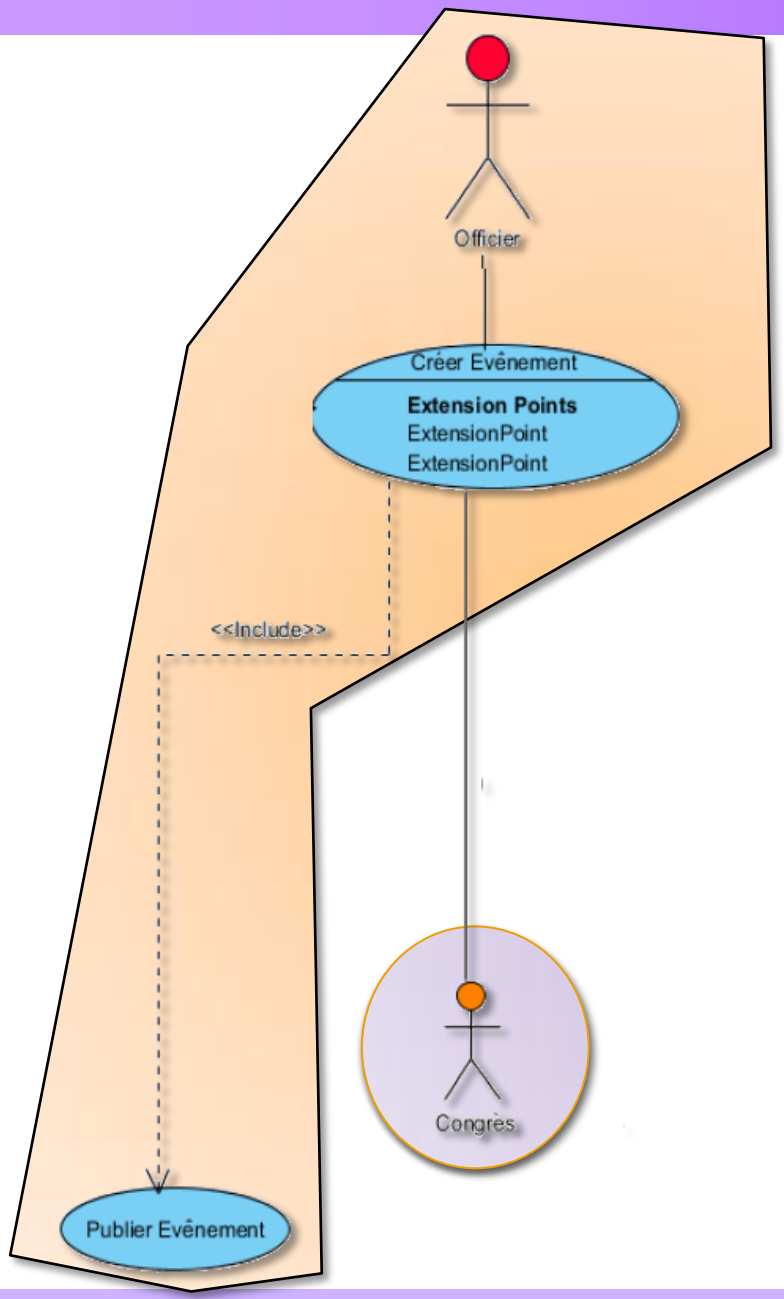
Déroulement des Tds

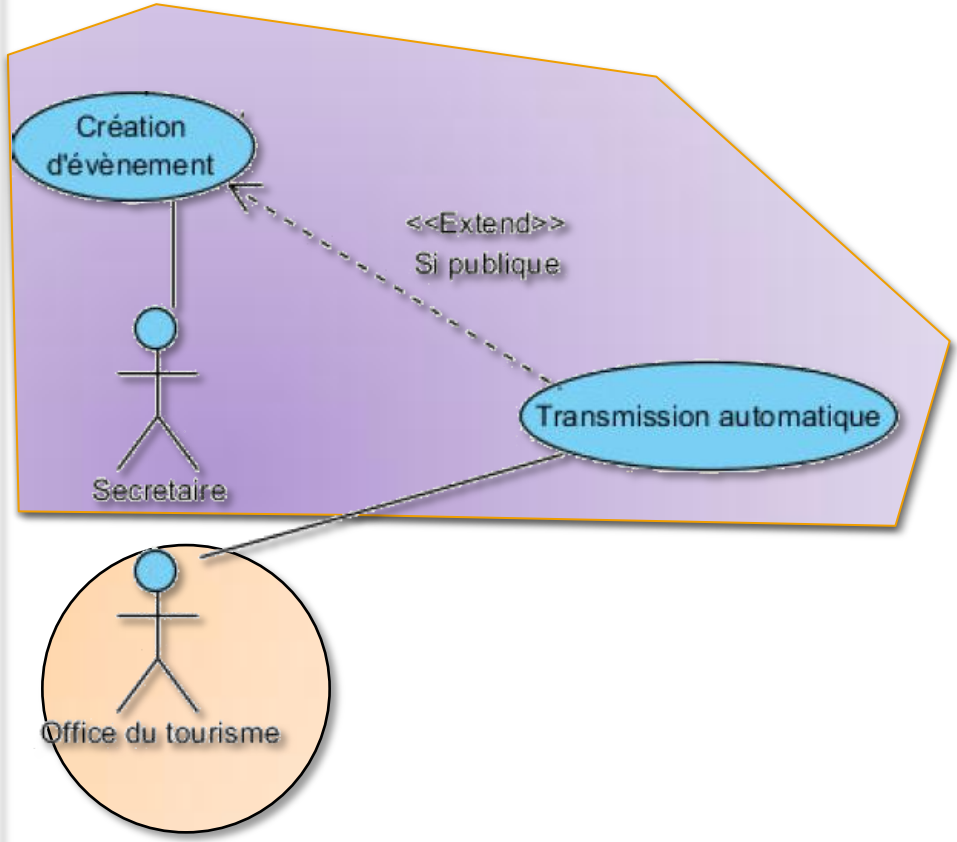
- Contrôle Continu et note(s) différenciée(s)
- Critères n°1 :
  - La méthode
  - La cohérence
  - La crédibilité
- Préparation de la séance
- Scénario

Collaboration entre S.I.

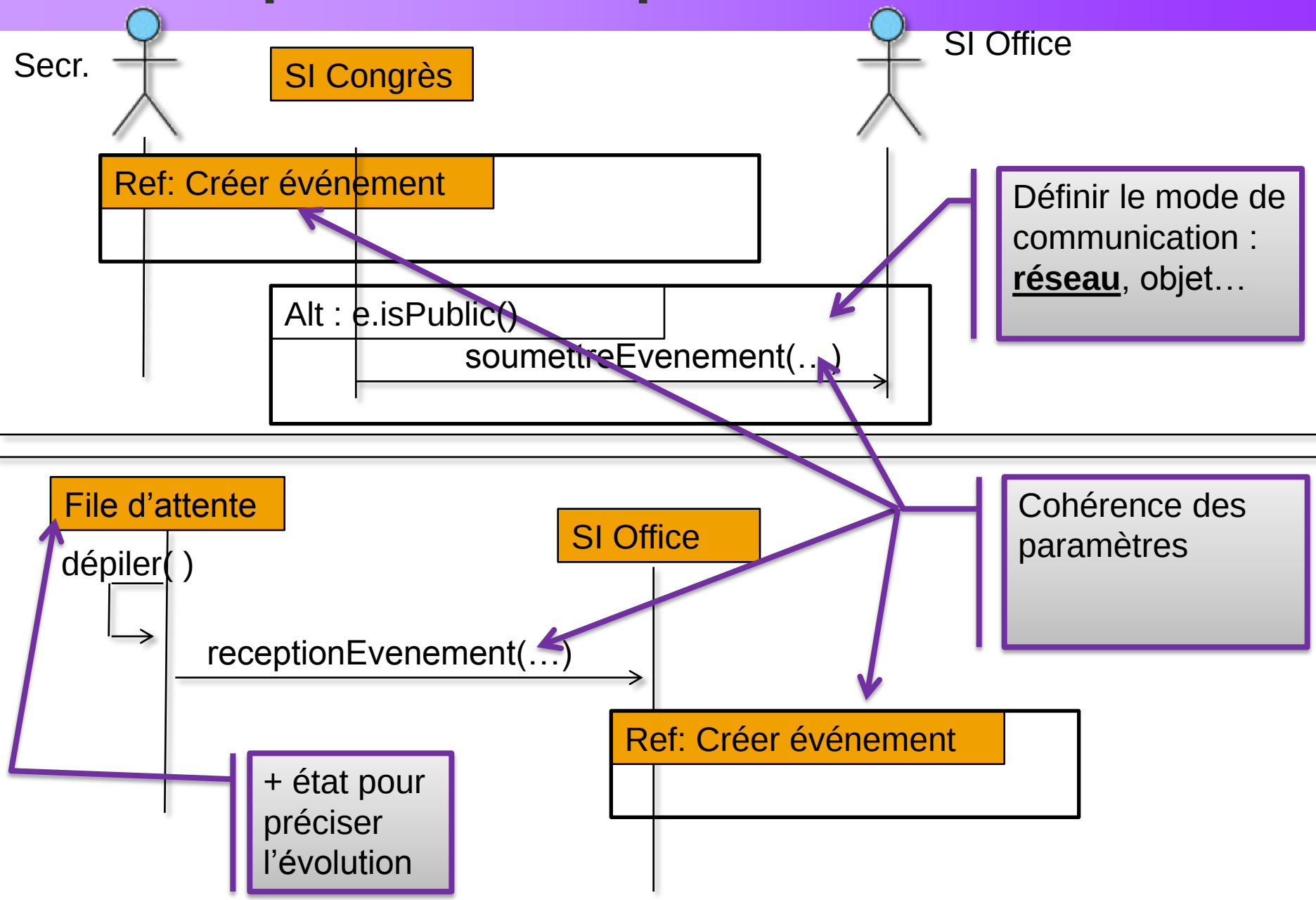
- Use Cases & scénario
- Séquences complémentaires
- Classes « compatibles » mais pas nécessairement les mêmes
- Cohérences dans la transmission
  - Informations échangées
  - média

Correspondance de Use Case





Correspondance de Séquences



Canevas de rapport intermédiaire (1/3)

1. Introduction

- Résumé du sujet
- Résumé des points en interaction avec les autres équipes
- Problématiques à soulever

2. Point de vue général de l'architecture

- **Un glossaire**
- **Une représentation générale (diagramme d'activité)**

Pour la suite : TOUT DIAGRAMME DOIT ETRE EXPLIQUE

Canevas de rapport intermédiaire (2/3)

3. Point de Vue Statique

- Cas d'utilisation
  - Acteurs
  - Diagrammes de Cas d'utilisation
  - Scénarios (sous forme textuelle, un par *use case*)
  - Un morceau de Maquette IHM pour chaque use case
- Diagrammes de Classes lisible en format A4 (une découpe « logique » avec répétition de certaine classe).

4. Point de vue Dynamique

- Diagrammes de Séquence
- Chaque diagramme de séquence devra référencer un use-case.
- Morceau(x) d'IHM associé(s)
- Diagrammes de Machine d'Etat

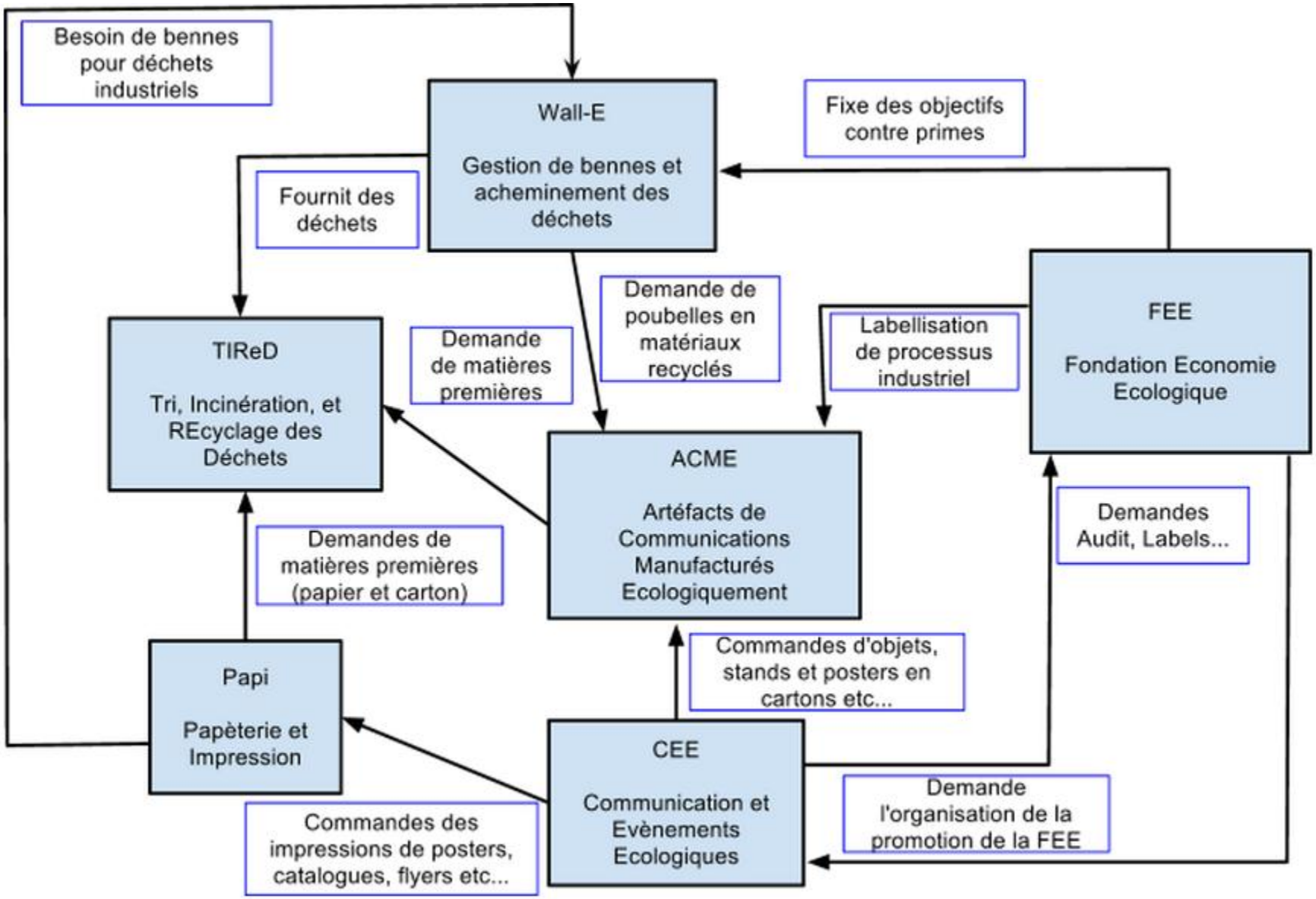
Canevas de rapport intermédiaire (3/3)

5. Interactions avec les autres S.I.

6. Maquette de l'interface
   - une maquette globale rattachée aux diagrammes présentés dans le document (synthèse des points 3-4-5)
   - Lien avec les scénarios

7. Conclusion
   - Analyse de votre solution : points forts et points faibles

Les sujets


Besoin de bennes pour déchets industriels
Wall-E
Gestion de bennes et acheminement des déchets
Fixe des objectifs contre primes
Fournit des déchets
TIReD
Tri, Incinération, et REcyclage des Déchets
Demande de matières premières
Demande de poubelles en matériaux recyclés
Labellisation de processus industriel
FEE
Fondation Economie Ecologique
ACME
Artéfacts de Communications Manufacturés Ecologiquement
Demandes Audit, Labels...
Demandes de matières premières (papier et carton)
Commandes d'objets, stands et posters en cartons etc...
Papi
Papèterie et Impression
CEE
Communication et Evènements Ecologiques
Demande l'organisation de la promotion de la FEE
Commandes des impressions de posters, catalogues, flyers etc...