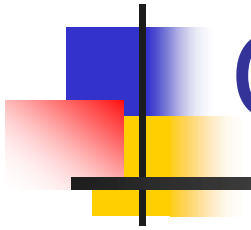


Cours d'architecture logicielle
Conception d'architecture

Lydie du Bousquet

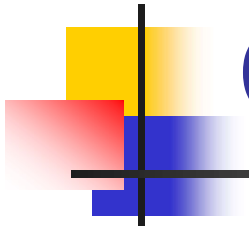
Lydie.du-bousquet@imag.fr

Philippe Lalanda

Rappel

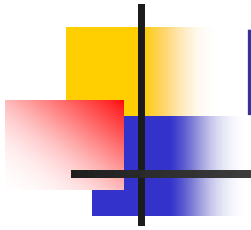
Architecture = composants + connecteurs

Représentation = ensemble de vues



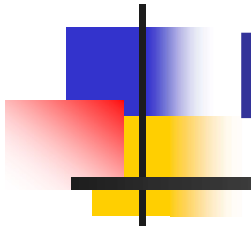
Conception/réflexion

- Exprimer l'architecture permet de discuter autour des solutions à mettre en œuvre
- La description dépend de l'interlocuteur
  - Fonctionnelle
  - Techniques
- Dans ce cours,
  - **Représentation**
  - Travail sur les premières phases de **conception**
    S'intéresse surtout au côté fonctionnel
  - Evaluation

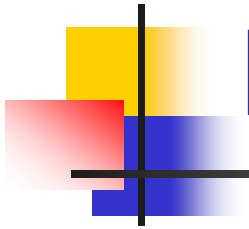


Exercice : CyberRadar

- Diagramme de contexte
- Liste fonctionnalités
- Identifier les composants
- Proposer une/des vues logiques
- …

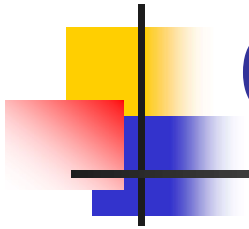


Pourquoi est-ce difficile ?



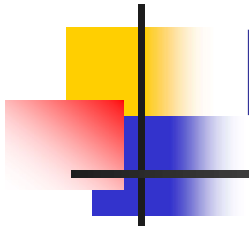
Difficulté de la conception

- Lien difficile entre la phase de gestion des exigences et la phase d'architecture
  - **Phase exigences** :
    - identifier les stakeholders,
    - comprendre les buts de chacun,
    - identifier et résoudre les conflits, …
  - **Phase architecturale** :
    - définir des composants, des connecteurs, des propriétés, …
  - problème aujourd'hui non résolu



Objectif de ce cours

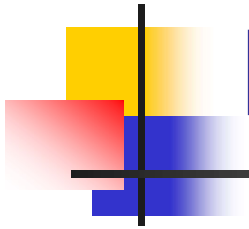
- Présenter
  - la notion de conception architecturale
  - les grands principes à suivre lors de la conception
  - les notions de cohésion et de couplage



La conception reste un défi

- Problèmes
  - Trouver les bons composants avec le bon niveau de granularité
  - Trouver les bons COTS
  - Trouver la bonne organisation et les bonnes interactions
- Et surtout, s'assurer que
  - les exigences fonctionnelles (et non fonct.) sont satisfaites
  - L'environnement de calcul est pris en compte
  - Les évolutions sont possibles, …
- Et savoir pourquoi !

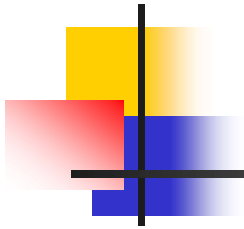
Notion de traçabilité



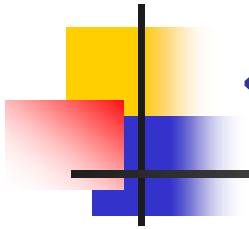
Plan

- Introduction
- **Une activité incrémentale**
- Méthode de conception
- Cohésion et couplage
- Conclusion

Conception : activité incrémentale



- Face à un problème complexe, plusieurs étapes sont nécessaires pour
  - Comprendre, appréhender
  - Exprimer
  - Détailler
  - Valider



2. Conception : activité incrémentale

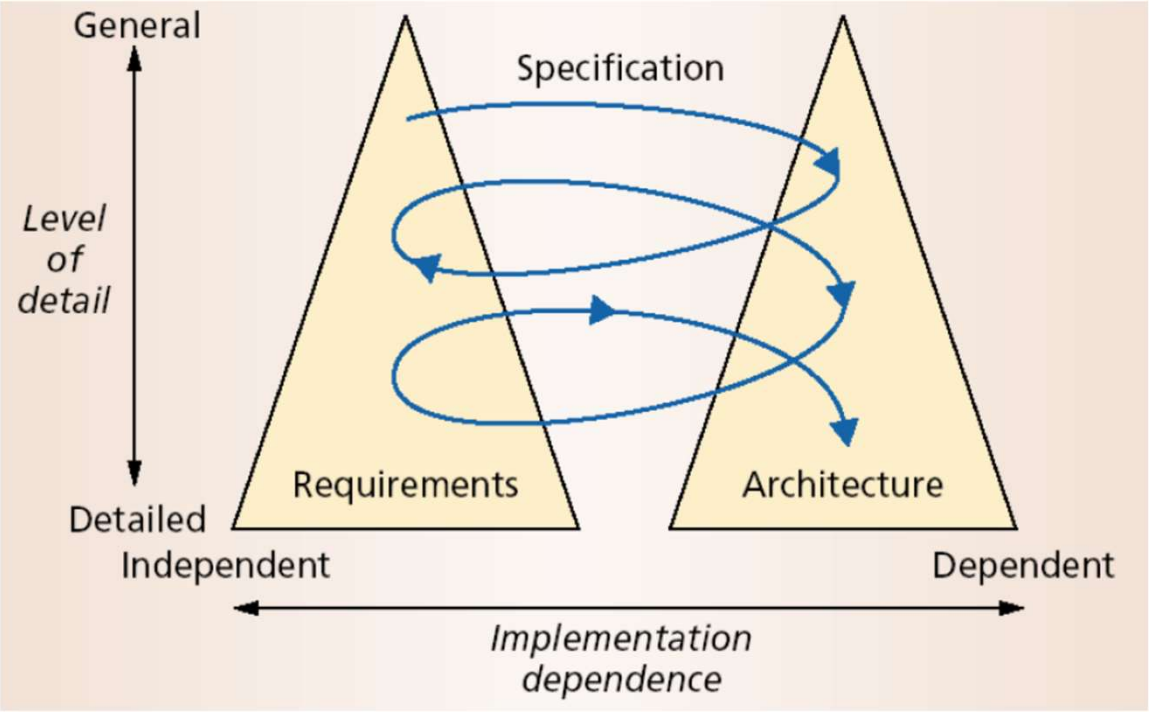
« Chevauchement » des phases

- A partir des exigences
- Un premier modèle est construit
  - Permet de retranscrire la compréhension du problème
- Permet de repérer de nouvelles exigences
- Un second modèle va être construit
  - Prise en compte des nouvelles exigences
  - Découvertes de nouveaux points
- Et ainsi de suite…

2. Conception : activité incrémentale

« Chevauchement » des phases

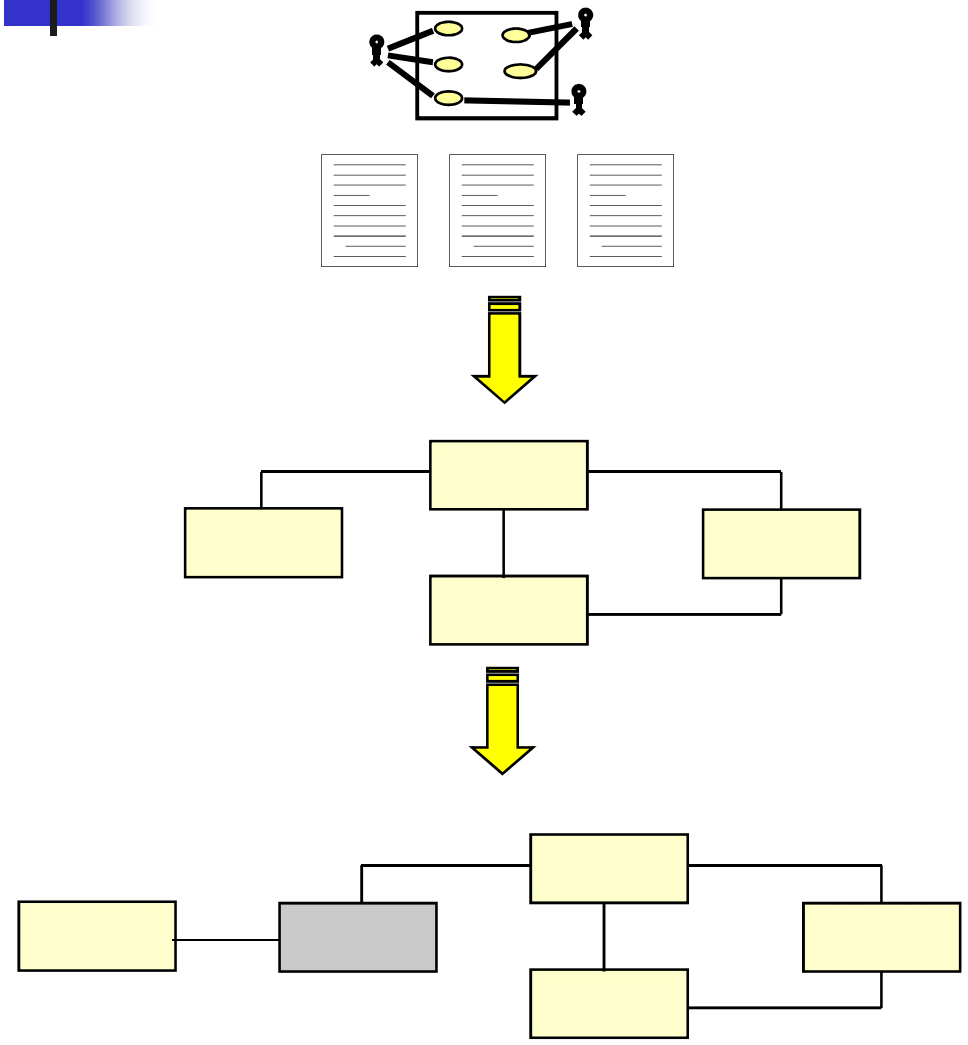
Nuseibeh : Twin Peak model



Importance d'une gestion explicite des exigences

2. Conception : activité incrémentale

Une succession de modèles



Modèle du domaine

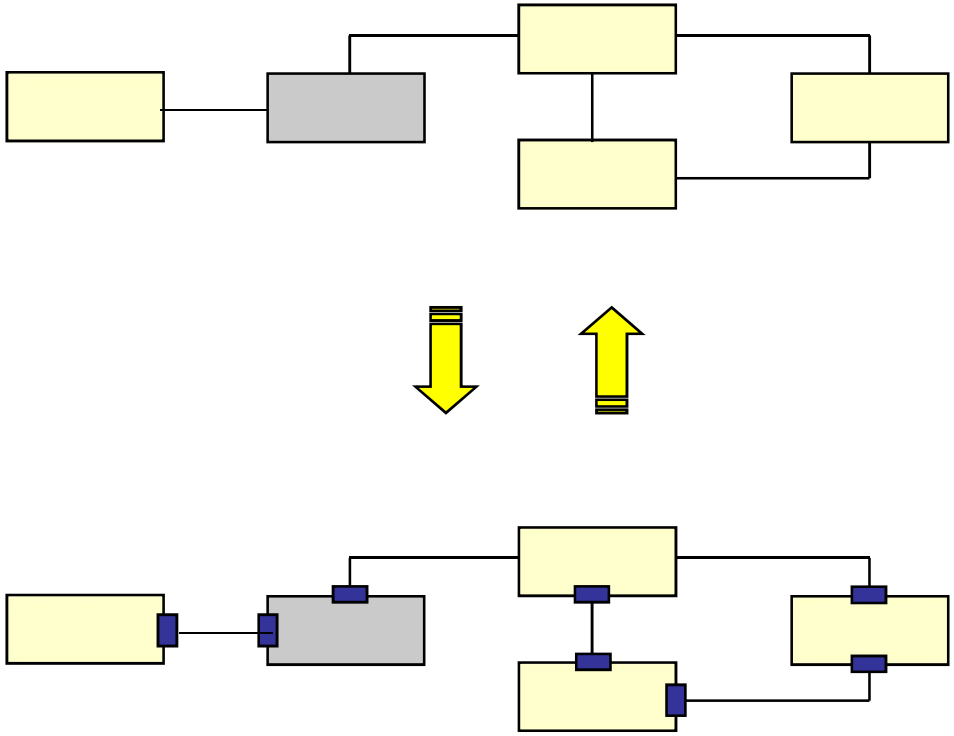
Architecture logicielle

Architecture (technique)

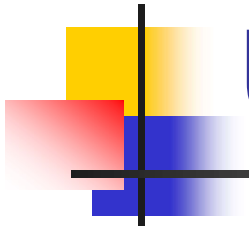
2. Conception : activité incrémentale

Spécification des interfaces

Définition incrémentales des interfaces

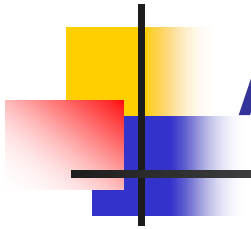


Une succession de modèles



- Les liens entre les différents modèles se perdent en cours de développement
  - Le développeur fait des changements
  - Les modifications ne sont pas répercutées (trop coûteux)
  - Le développeur a le dernier mot !

  - Grand soucis du MDA ...

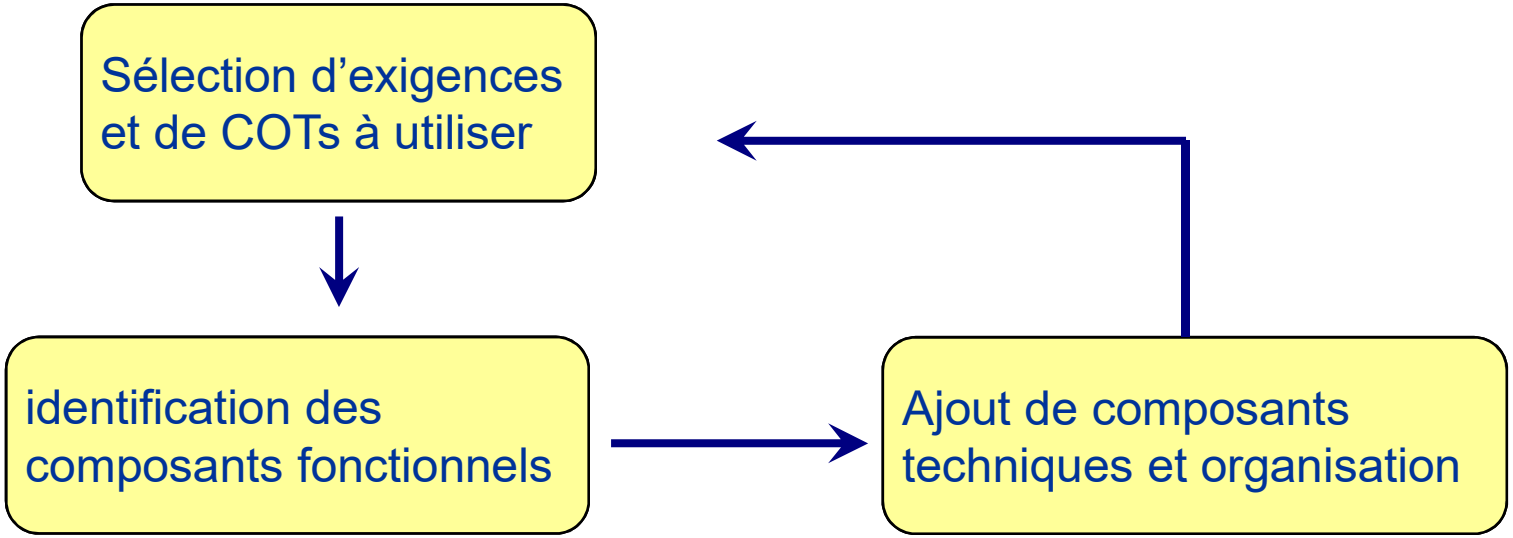


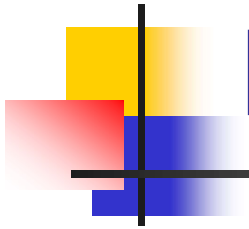
Approche pour la conception

- Identification des composants
  - à partir du document de spécification
    - exigences
    - contraintes d'utilisation de COTS
- Organisation des composants
  - Utilisation de patrons/styles d'architecture et
  - tactiques de conception
- Évaluation :
  - mesure de la cohésion et du couplage
  - Revue d'architecture

3. Méthode de conception

Approche pour la conception

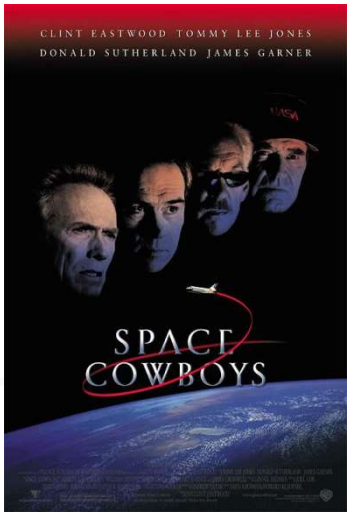




Exemple

- Simulateur pour l'entraînement à la conduite
  - l'élève prend place dans une cabine grandeur réelle et la pilote. Les effets de ses actions sont calculés et rendus en temps réel dans la cabine par différents vérins.
  - La cabine est dotée d'un écran de visualisation du terrain
  - Ces systèmes existent pour les chars, les camions, etc.
- Nous allons spécifier, en partie, les différentes vues architecturales du simulateur

Illustration : Simulateur de Vol « Space Cowboys »



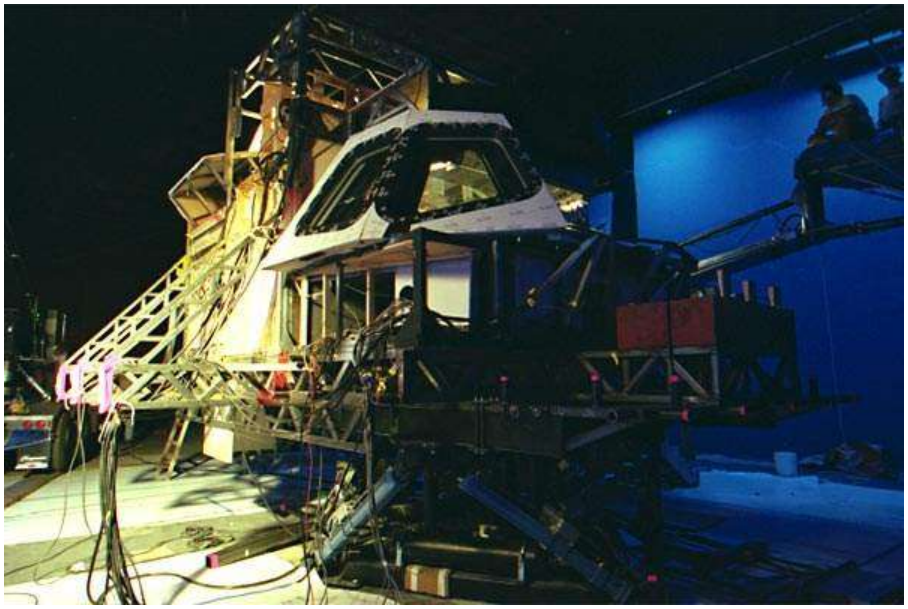
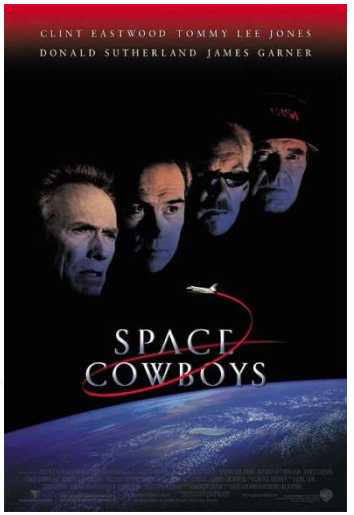
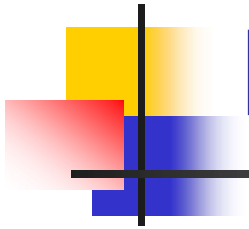


Illustration : Simulateur de Vol « Space Cowboys »





Rappel : spécifications

- Plan du document de spécification
  - Contexte (positionnement, documents utilisés, …)
  - Diagramme de contexte
  - Périmètre fonctionnel
  - Description des fonctionnalités
    - éventuellement sous forme de scénarios (en partie)
  - Qualités logicielles
  - IHM
  - Contraintes d'implantation

Vue de contexte

Élève

Présentation IHM

Actions de conduite

Instructeur

configurations

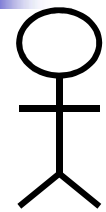
états

Système de conduite

positions

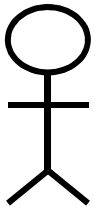
Cabine

Acteurs



Élève

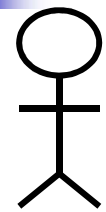
Un **élève** est une personne inscrite à la formation de conduite. Il doit s'introduire dans une cabine de simulation, choisir un exercice et effectuer un exercice. Cela revient à utiliser les pédales et le volant pour se déplacer dans un environnement simulé.



Instructeur

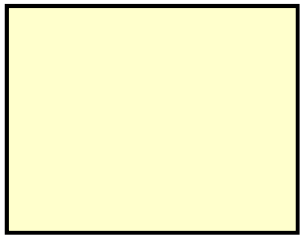
Un **instructeur** est une personne experte en conduite qui est chargé de surveiller l'évolution d'un élève au cours d'un exercice et de l'évaluer lorsqu'un exercice est terminé. Il peut également en temps réel modifier l'environnement simulé.

Acteurs



Cabine

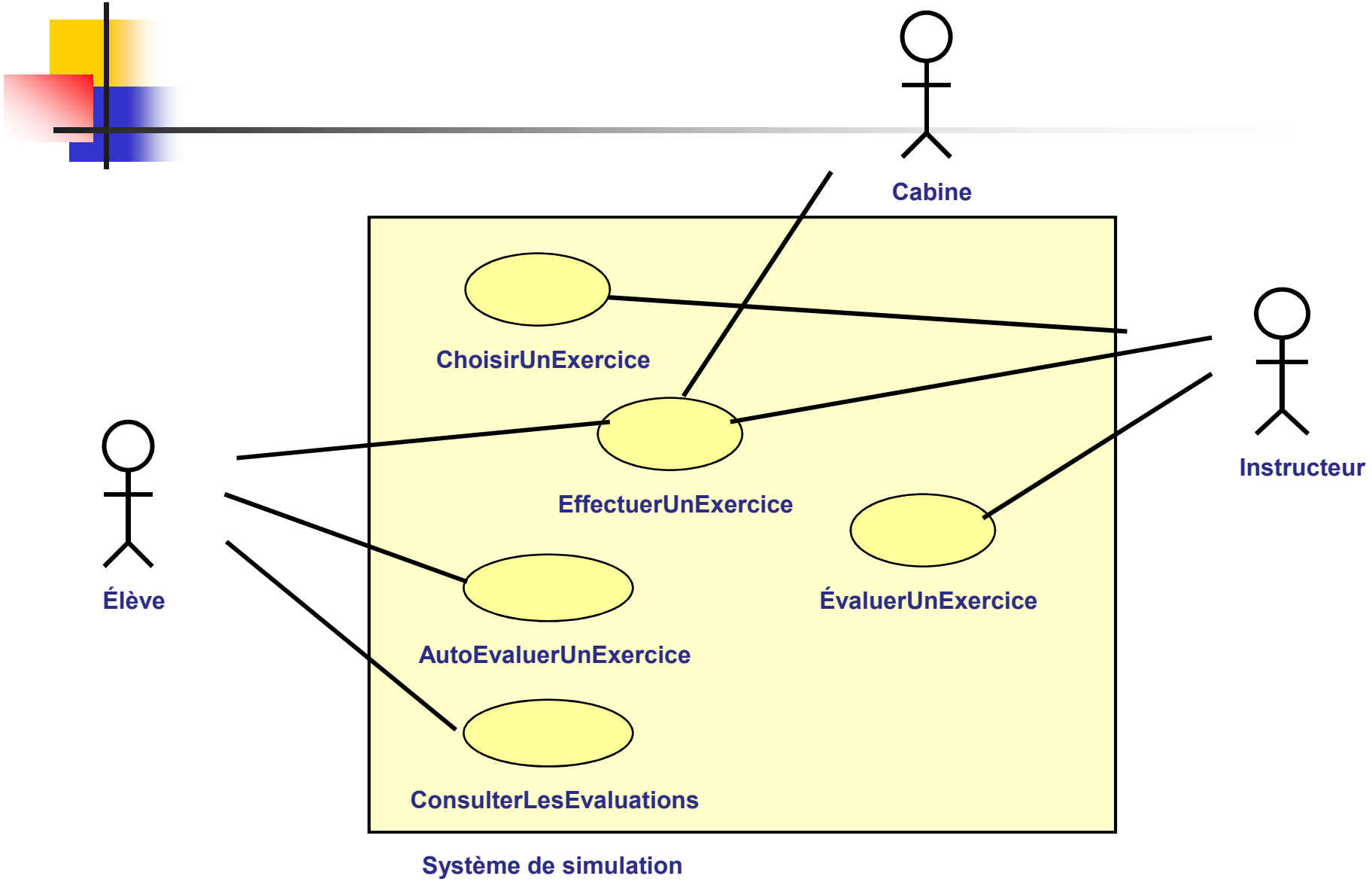
Une **cabine** est un équipement reproduisant en taille réelle l'habitacle d'un véhicule. Elle comprend tous les éléments d'interaction fournis par un véhicule (pédales, frein à main, volant, rétro, …). Elle est montée sur vérins de façon à reproduire les secousses ressenties en réel.



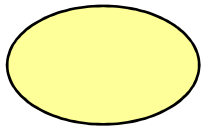
Simulateur

Le **simulateur** est un logiciel proposant des exercices de conduite et des capacités d'évaluation. Il pilote une cabine de simulation et prend en compte en temps réel les actions de l'élève.

Use cases – un sous-ensemble



Use cases – suite

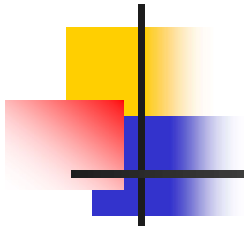


Effectuer UnExercice

Pré-condition: l'élève a choisi un exercice …
Début: l'élève appuie sur le bouton START
Fin …
Post-condition…
Déroulement nominal
• l'élève appuie sur le bouton START
• le *système* demande le code de l'élève pour l 'identifier
• le *système* lance la simulation
• l'élève interagit avec le système
• le *système* affiche en temps réel le terrain et reproduit les mouvements de la cabine. Le cas échéant, il envoie l'iHM vers le poste instructeur si celui-ci est connecté
• La simulation s'arrête à la fin de l'exercice
Variantes …
Contraintes non-fonctionnelles

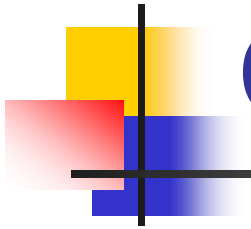
Exigences – un sous ensemble

Libellé	Numéro	Catégorie
Restitution sensation de mouvements, secousses	E 3 SYS	MVT
Restitution des sensations de vibrations du char	E 4 SYS	MVT
Vision partie visible du canon	E 6 SYS	VIS
Symbologie pilote pour épiscope central	E 7 SYS	VIS
Rétroviseurs dans les épiscopes latéraux	E 8 VIS	VIS
Visuel : restitue conditions climatiques (EAU)	E 9 VIS	VIS
Visuel : restitue type et conditions d'observation	E 10 VIS	VIS
Visuel : Feux de signalisation	E 12 VIS	VIS
Restitue l'environnement sonore du poste de pilotage	E 14 SYS	SON
Simulation bruits de roulement	E 15 SIM	SON
Simulation bruits de châssis (moteur, trans, venti)	E 16 SIM	SON
Simulation bruits tourelle et de tirs 120 mm	E 17 SIM	SON
Emet les alertes vocales	E 19 PHO	SON
PCA : Suivre et contrôler le travail des élèves	E 21 PCA	INS
PCA : Créer et modifier des exercices	E 22 PCA	INS
PCA : Répétition état pupitres et commandes pilote	E 23 PCA	INS
Piloter à l'aide d'un mini-manche	E 27 SYS	INS
Générer et annuler des pannes	E 28 SYS	INS
Geler et dégeler la simulation	E 29 SYS	INS
Mémoriser et rejouer une séquence	E 30 SYS	INS



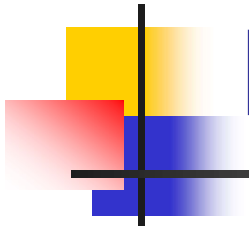
Exigences non fonctionnelles

- Performance
  - Le système devra réagir en moins d'un dixième de seconde
- Disponibilité
  - Une heure d'indisponibilité maximale par semaine – la nuit
- Utilisabilité
  - Une personne non experte devra être capable d'utiliser le logiciel au bout d'une ½ heure l'apprentissage



Contraintes

- Méthode
  - Le système devra être développé en utilisant UML
  - Chaque phase se soldera par un document (livrable)
- Plate-forme d'exécution
  - Le système devra être développé au dessus de Linux
  - TCP/IP devra être utilisé pour la communication



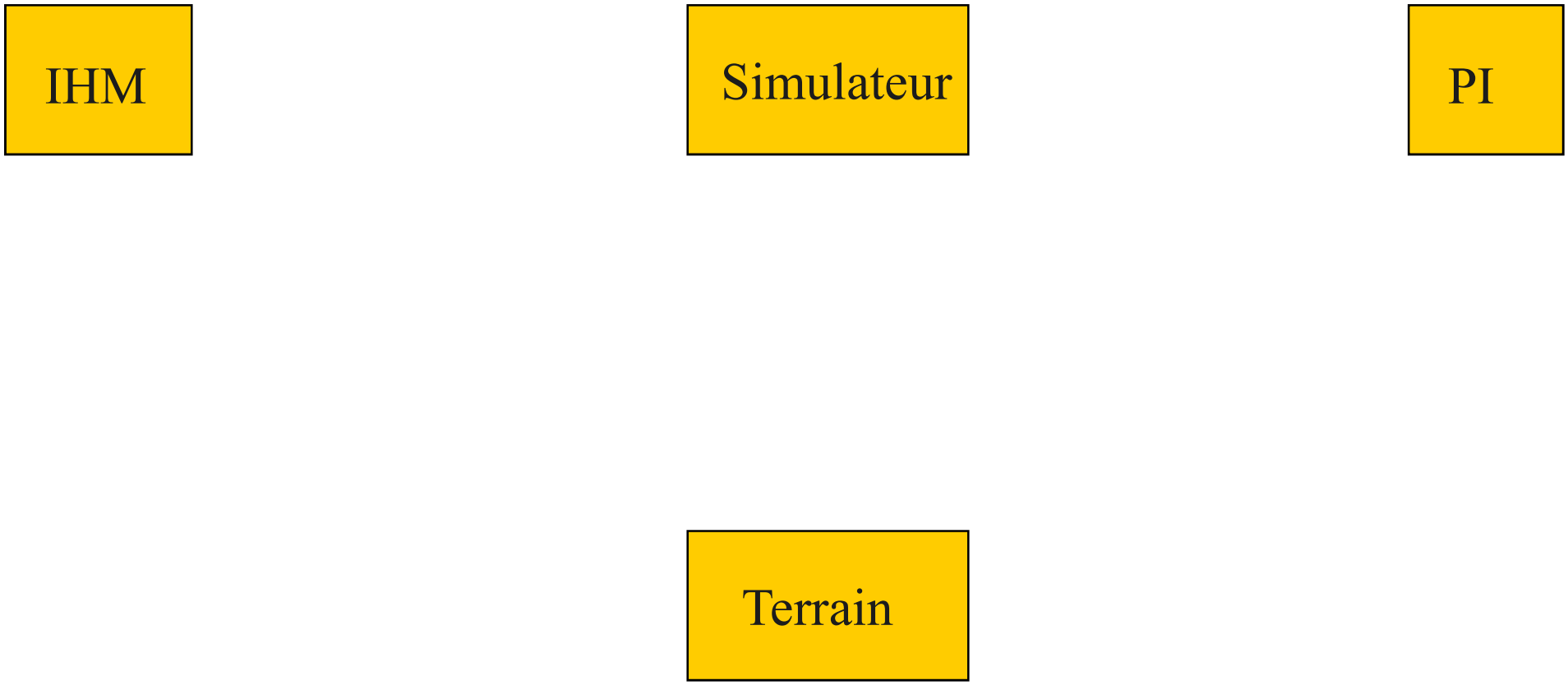
Plan

- Exemple : spécifications
- Exemple : architecture

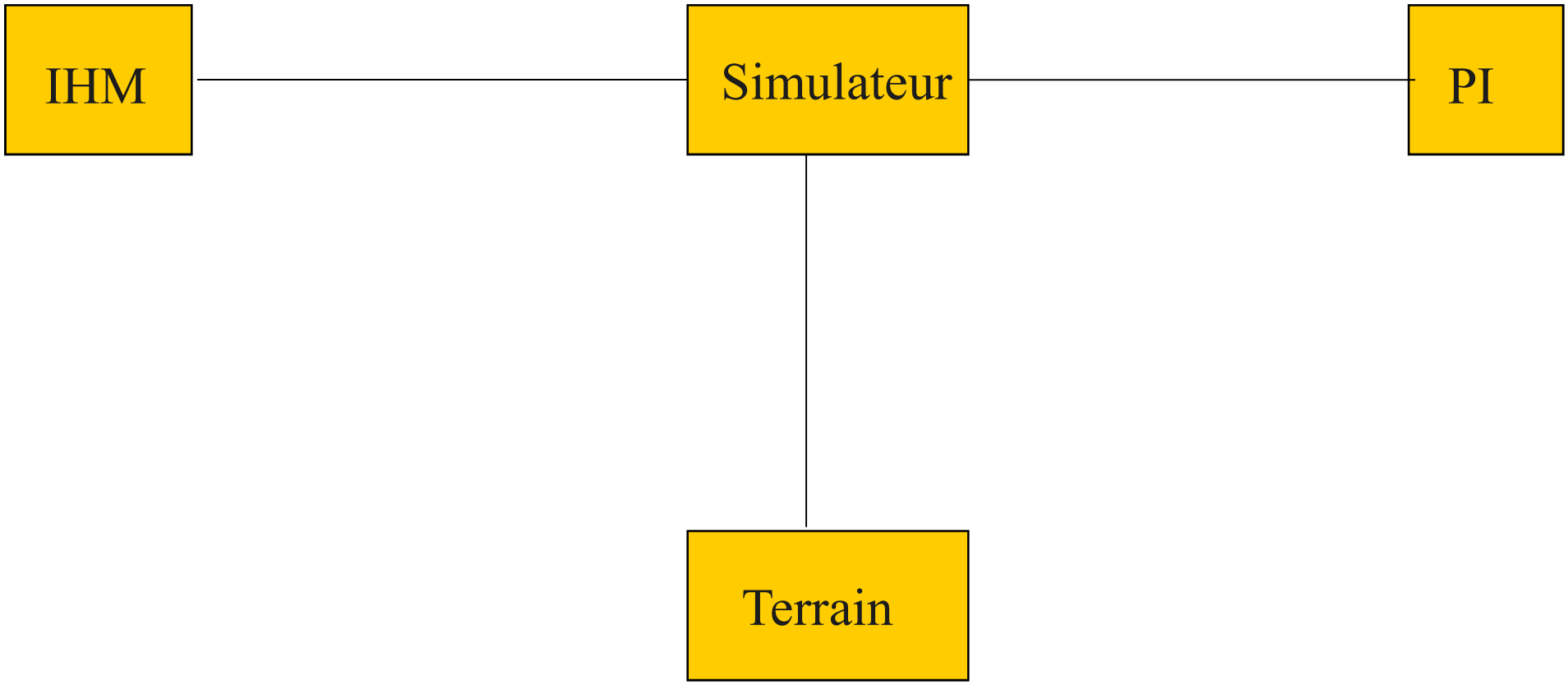
Construction des vues
Identification des composants importants

- L'IHM cabine
- Le poste instructeur (PI)
- La base terrain
- Le simulateur

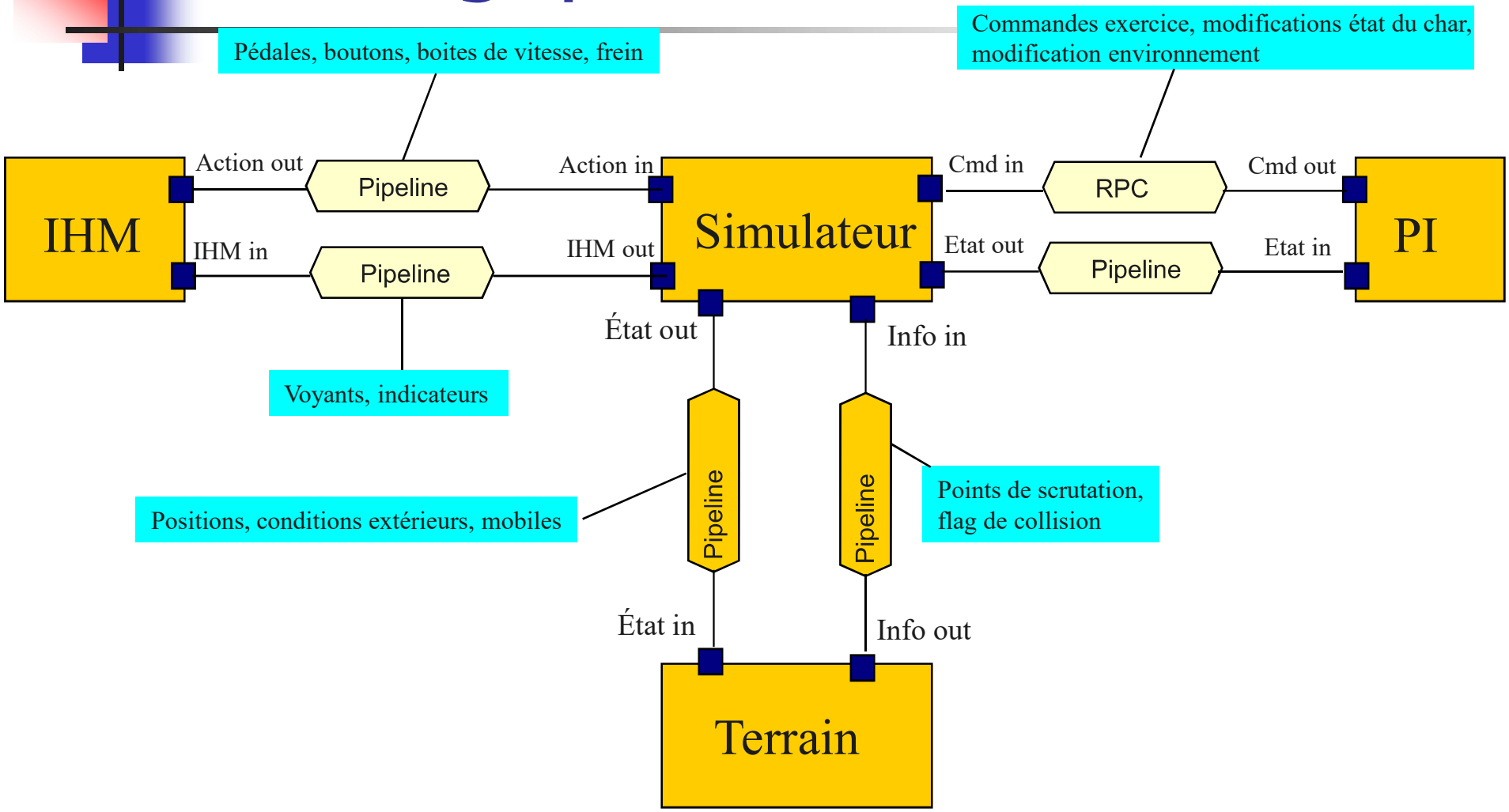
Vue logique



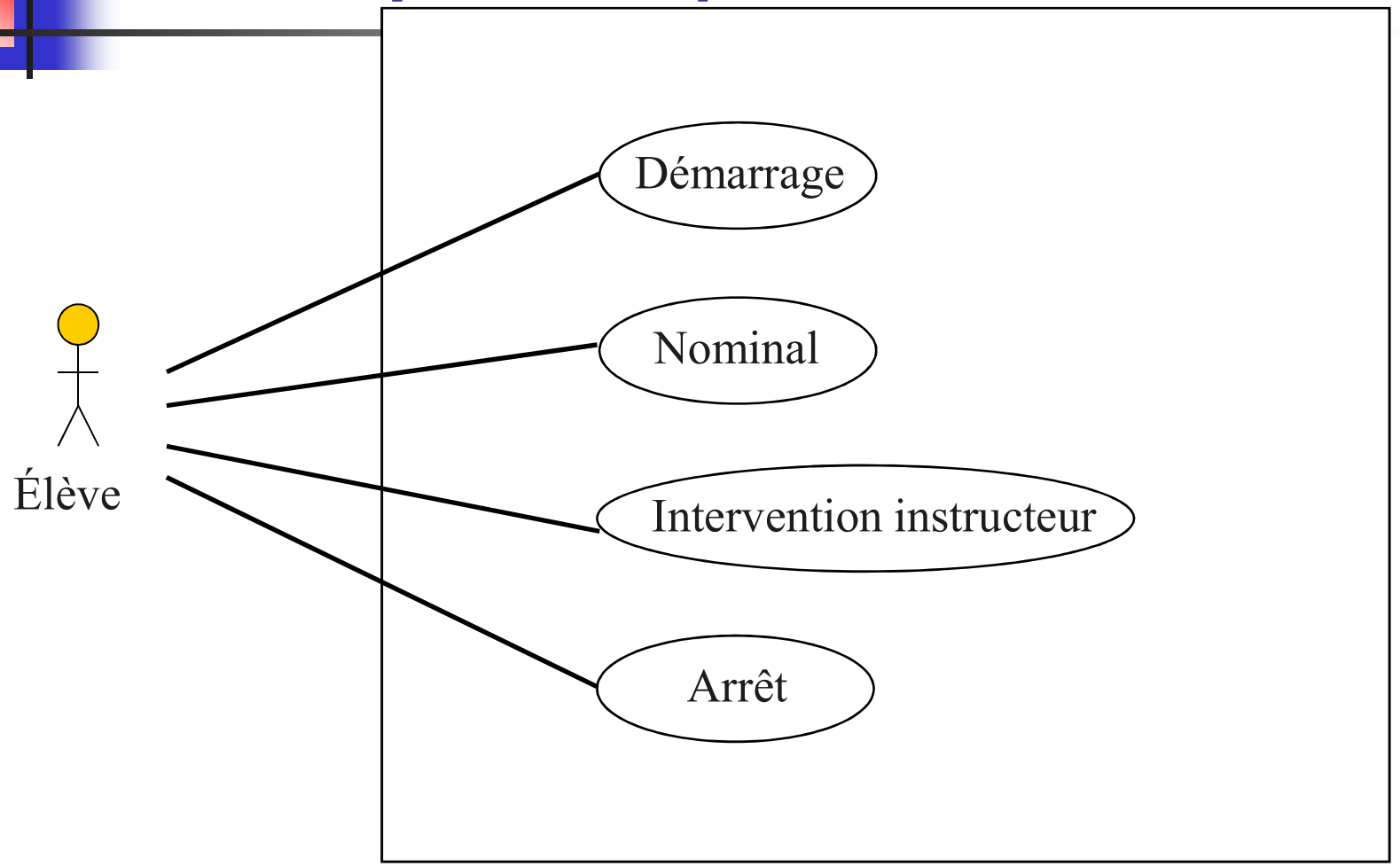
Vue logique



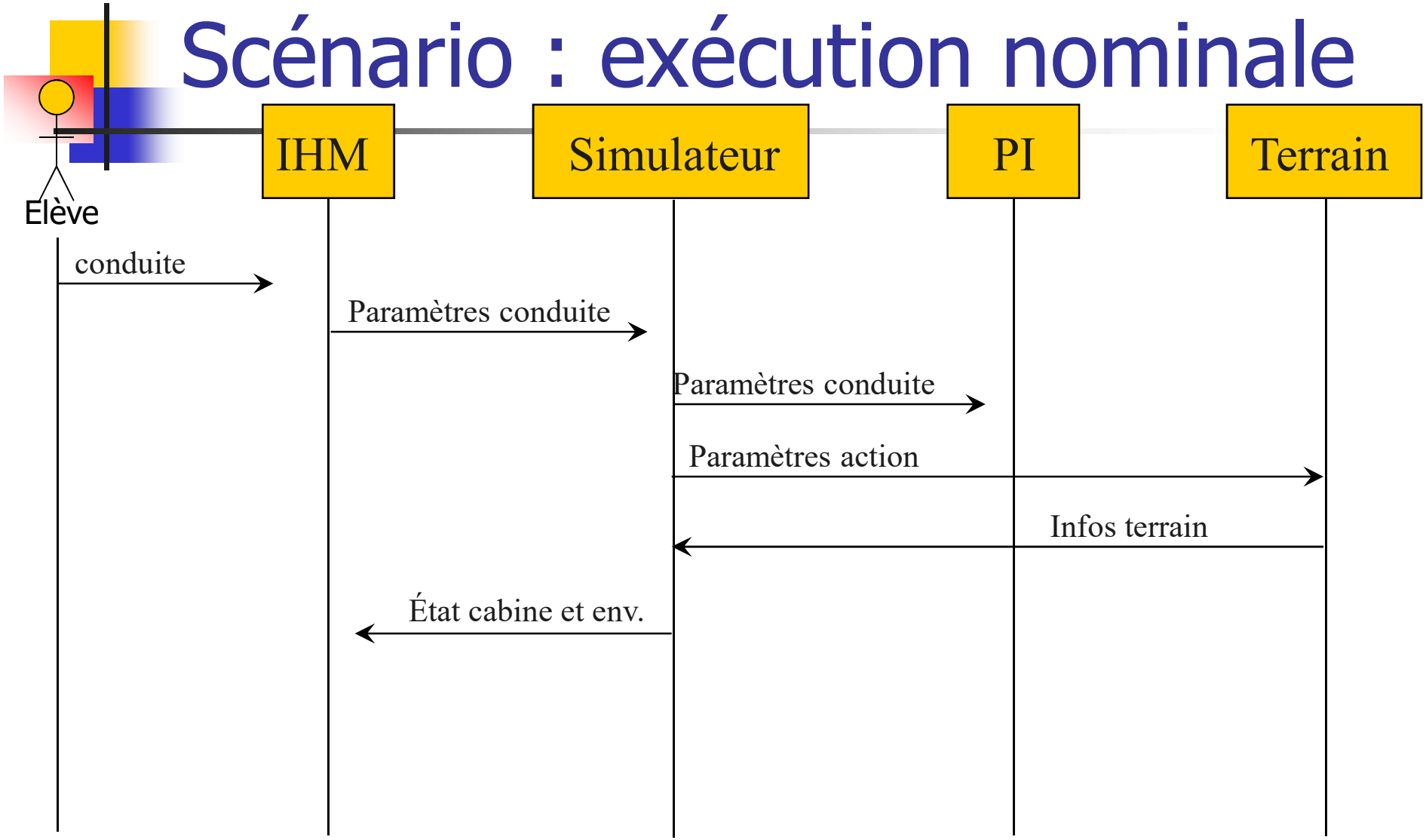
Vue logique


Pédales, boutons, boites de vitesse, frein
Commandes exercice, modifications état du char, modification environnement
IHM
Action out
Pipeline
Action in
Simulateur
Cmd in
RPC
Cmd out
PI
IHM in
Pipeline
IHM out
Etat out
Pipeline
Etat in
État out
Info in
Voyants, indicateurs
Positions, conditions extérieurs, mobiles
Pipeline
Pipeline
Points de scrutation, flag de collision
État in
Info out
Terrain

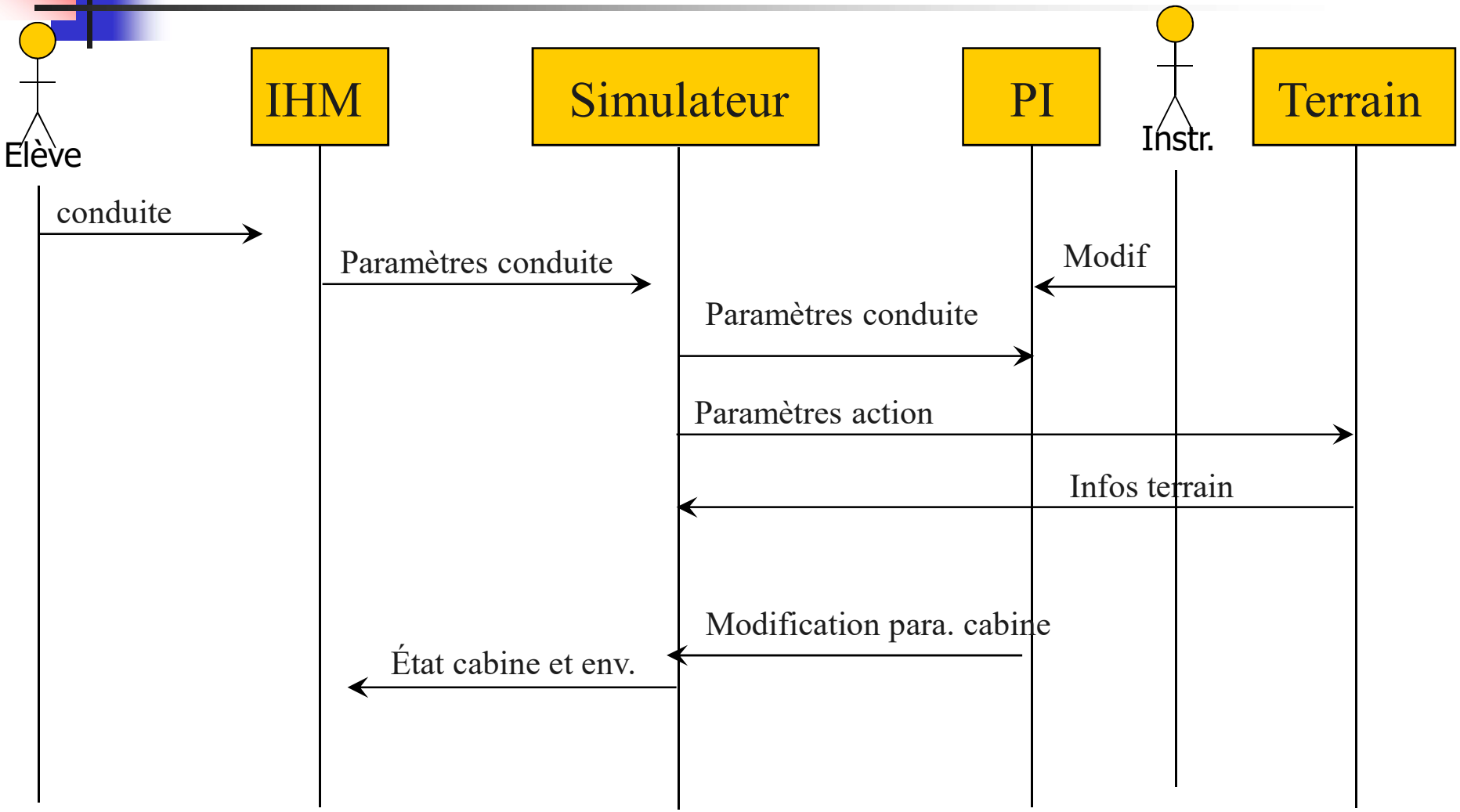
Vue dynamique : scénarios



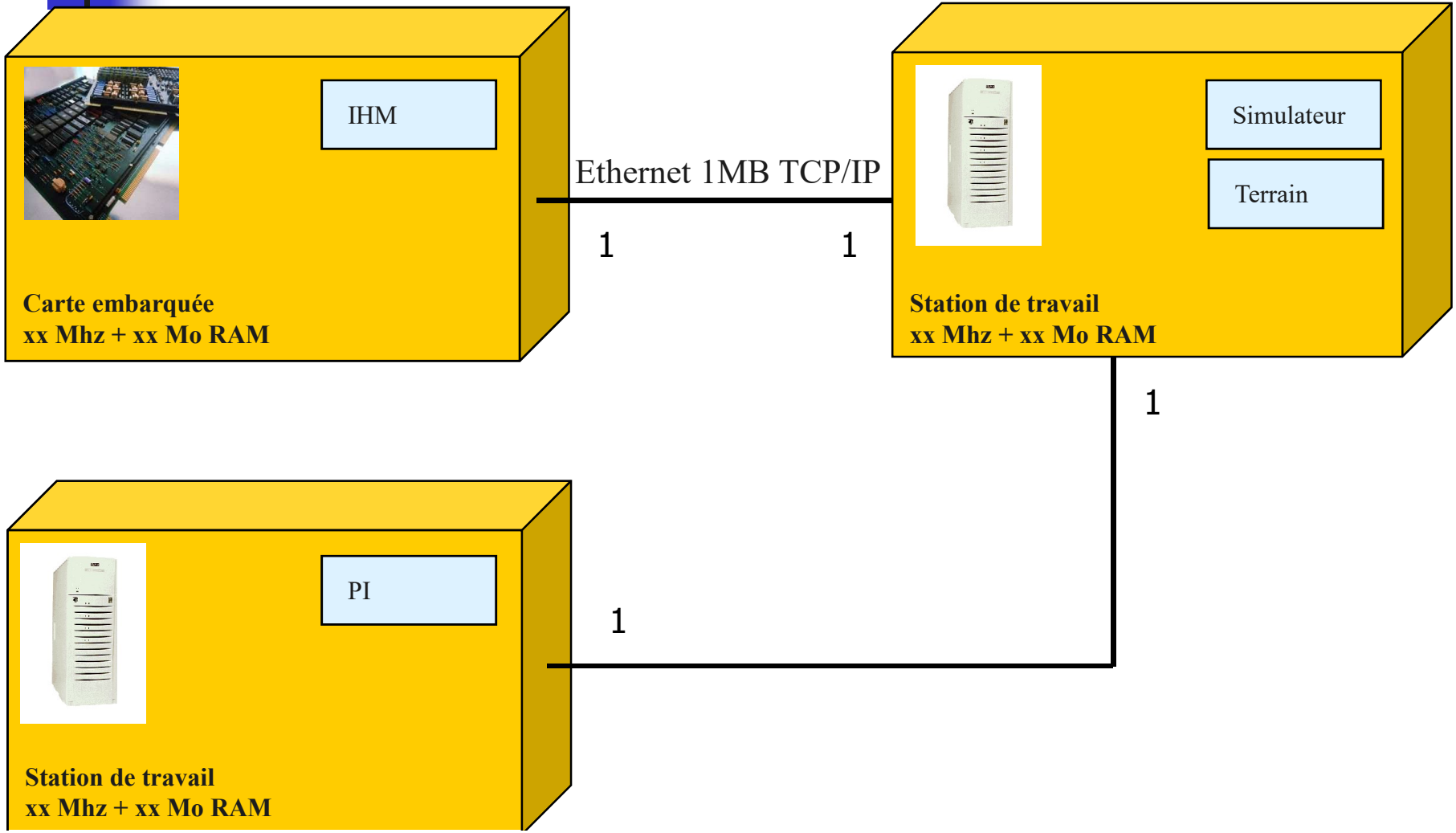
Scénario : exécution nominale



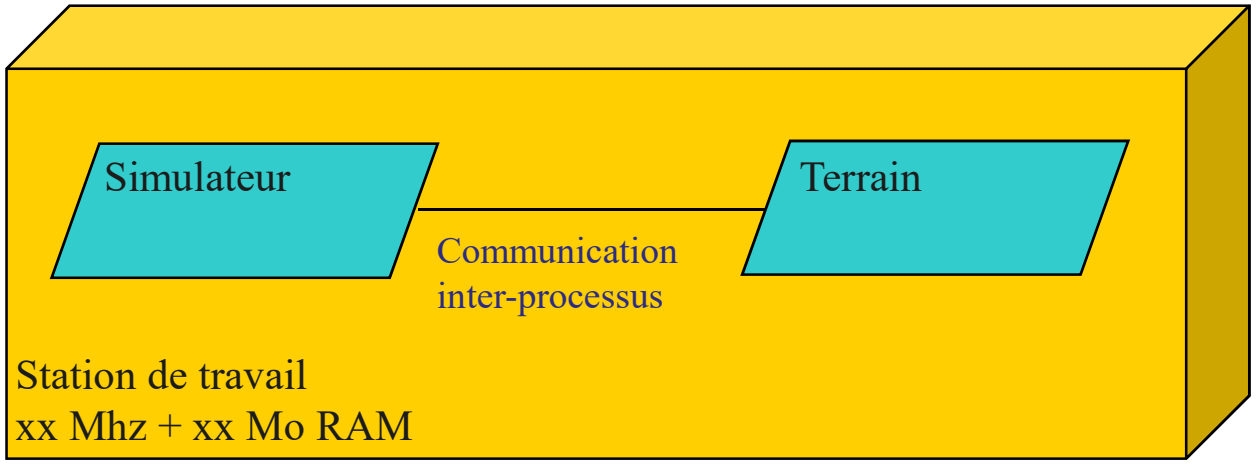
Scénario : intervention instructeur

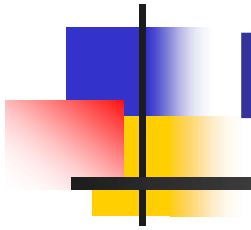


Vue physique



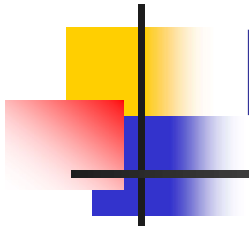
Vue processus - partielle





Et après ?

Quelle architecture pour
le composant Simulateur ?

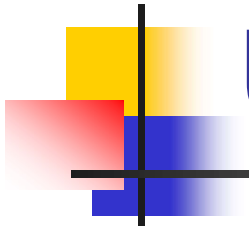


Plan

- Contexte de conception
- Une activité incrémentale
- **Méthode de conception**
- Cohésion et couplage
- Conclusion

3. Méthode de conception

Utilisation des exigences



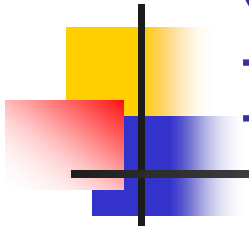
- Méthode CBSP (University of Southern California)
  - Les exigences contiennent des informations architecturales (de façon implicite ou explicite)
- Principe de la méthode
  - Identifier les exigences qui ont un impact sur l'architecture et les reformuler
  - Exprimer l'impact sur des composants, des connecteurs, des propriétés
  - Identifier les conflits et les résoudre

3. Méthode de conception - CBSP

Exemple : analyse des exigences

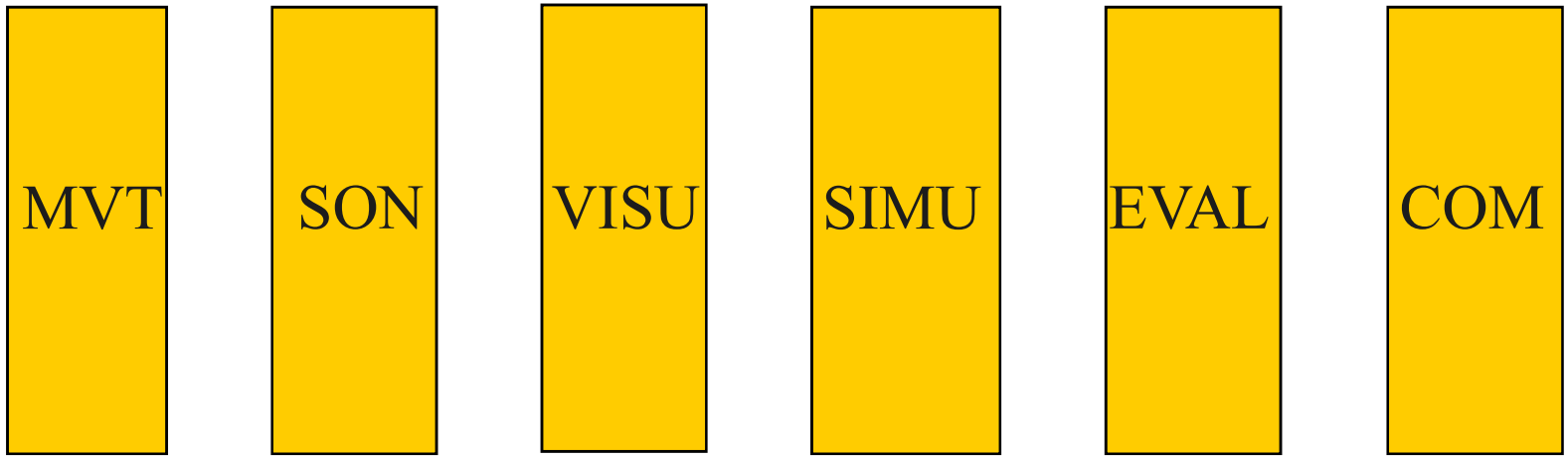
Libellé	Numéro	Catégorie
Restitution sensation de mouvements, secousses	E 3 SYS	MVT
Restitution des sensations de vibrations du char	E 4 SYS	MVT
Vision partie visible du canon	E 6 SYS	VIS
Symbologie pilote pour épiscope central	E 7 SYS	VIS
Rétroviseurs dans les épiscopes latéraux	E 8 VIS	VIS
Visuel : restitue conditions climatiques (EAU)	E 9 VIS	VIS
Visuel : restitue type et conditions d'observation	E 10 VIS	VIS
Visuel : Feux de signalisation	E 12 VIS	VIS
Restitue l'environnement sonore du poste de pilotage	E 14 SYS	SON
Simulation bruits de roulement	E 15 SIM	SON
Simulation bruits de châssis (moteur, trans, venti)	E 16 SIM	SON
Simulation bruits tourelle et de tirs 120 mm	E 17 SIM	SON
Emet les alertes vocales	E 19 PHO	SON
PCA : Suivre et contrôler le travail des élèves	E 21 PCA	INS
PCA : Créer et modifier des exercices	E 22 PCA	INS
PCA : Répétition état pupitres et commandes pilote	E 23 PCA	INS

Construction des vues
Identification des composants importants

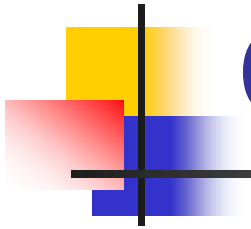


- L'IHM cabine
- Le poste instructeur (PI)
- La base terrain
- Le simulateur
  - Gestion du mouvement de la cabine (MVT)
  - Gestion du son (SON)
  - Gestion de l'écran (VISU)
  - Calcul des simulations (SIMU)
  - Évaluation du pilote (EVAL)

Exemple : composants identifiés



Cohérence et cohésion des composants

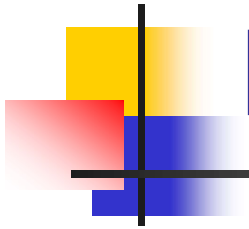


Organisation des composants

- La topologie est liée à des choix techniques
  - Atteinte de propriétés non fonctionnelles
  - Performance, disponibilité, sécurité, …
- Utilisation de patterns
  - Patterns d'architecture (styles)
  - Tactiques de conception

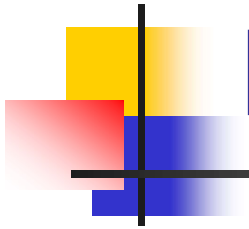
3. Méthode de conception

Les patterns / style



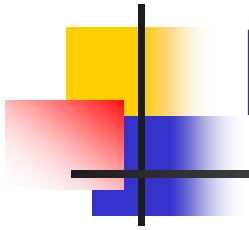
- Un pattern de conception (patron) est la description d'un problème et d'une solution
  - <problème type, solution type> validé par l'expérience

- La solution doit être validée par l'expérience. Elle décrit des composants en interaction
  - types de composants
  - types de relations
  - gestion du contrôle et des données



Les patterns - suite

- Pattern = réutilisation de (solution de) conception
  - il est plus facile de réutiliser une solution de conception de haut niveau qu'une implantation
  - c'est une façon de décrire des bonnes pratiques, des bonnes conceptions et de transmettre une connaissance
- Les patterns majeurs sont connus des programmeurs qui utilisent un vocabulaire précis pour échanger
- Exemple : MVC



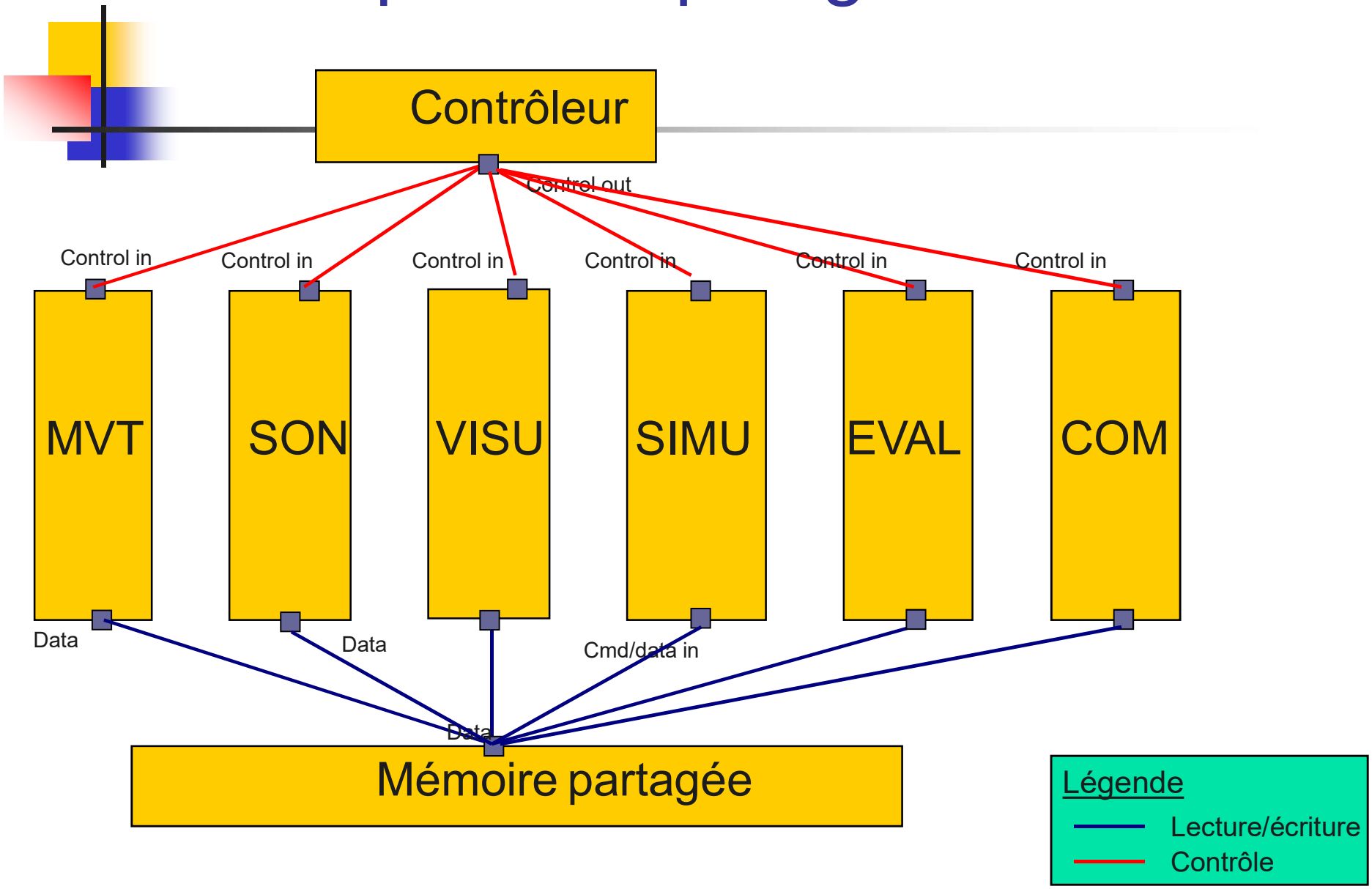
Les tactiques de conception

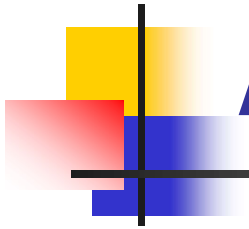
- Une tactique est une stratégie locale qui a pour but de structurer un sous ensemble de composants
  - elle sous entend une organisation logique et une organisation dynamique

- Une tactique ne structure pas l'ensemble du système

  contrairement aux styles

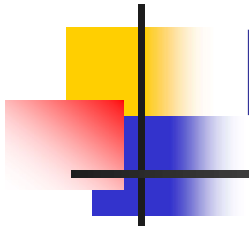
Exemple de topologie





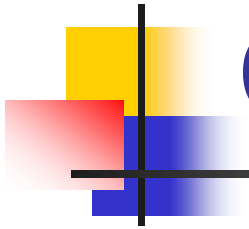
Synthèse : Approches pour la conception

- Recherche des composants fonctionnels
  - analyse fine des exigences
  - prise en compte des composants à réutiliser
- Utilisation de patterns
  - organisation des composants (communication, …)
  - ajout de composants plus « techniques »
- Validation de l'architecture
  - Mesure de la cohésion et du couplage
  - Revue d'architecture



Plan

- Introduction
- Une activité incrémentale
- Méthode de conception
- **Cohésion et couplage**
- Conclusion

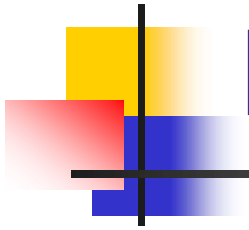


Cohésion et couplage

- Deux critères importants pour l'évaluation d'une architecture :
  - La cohésion de chaque module
  - Le couplage entre les modules

- Évaluer la cohésion d'un module c'est répondre à la question
  Pourquoi les éléments d'un module ont-ils été mis ensemble ?

- Évaluer le couplage d'un module c'est répondre à la question
  quels liens existent entre ces deux modules ?



La cohésion

- **« largeur » fonctionnelle** d'un module
- Cohérence de ce module
  - c'est une mesure **qualitative**

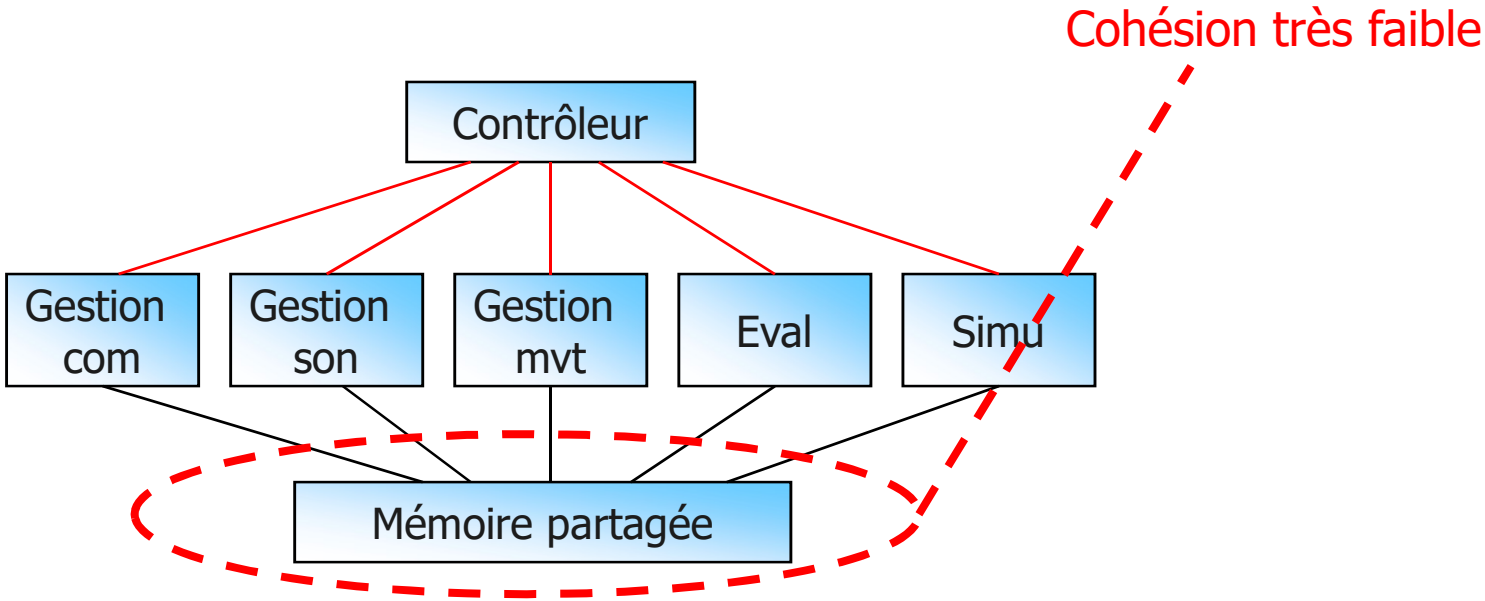
- Une grande cohésion favorise l'évolution
  - facilite la recherche du module à modifier
  - concentre l'ensemble des modifications dans un minimum de modules

Niveaux de cohésion

- Au niveau architectural, on peut distinguer quatre types de cohésion
  - accidentelle
  - syntaxique
  - temporelle
  - fonctionnelle

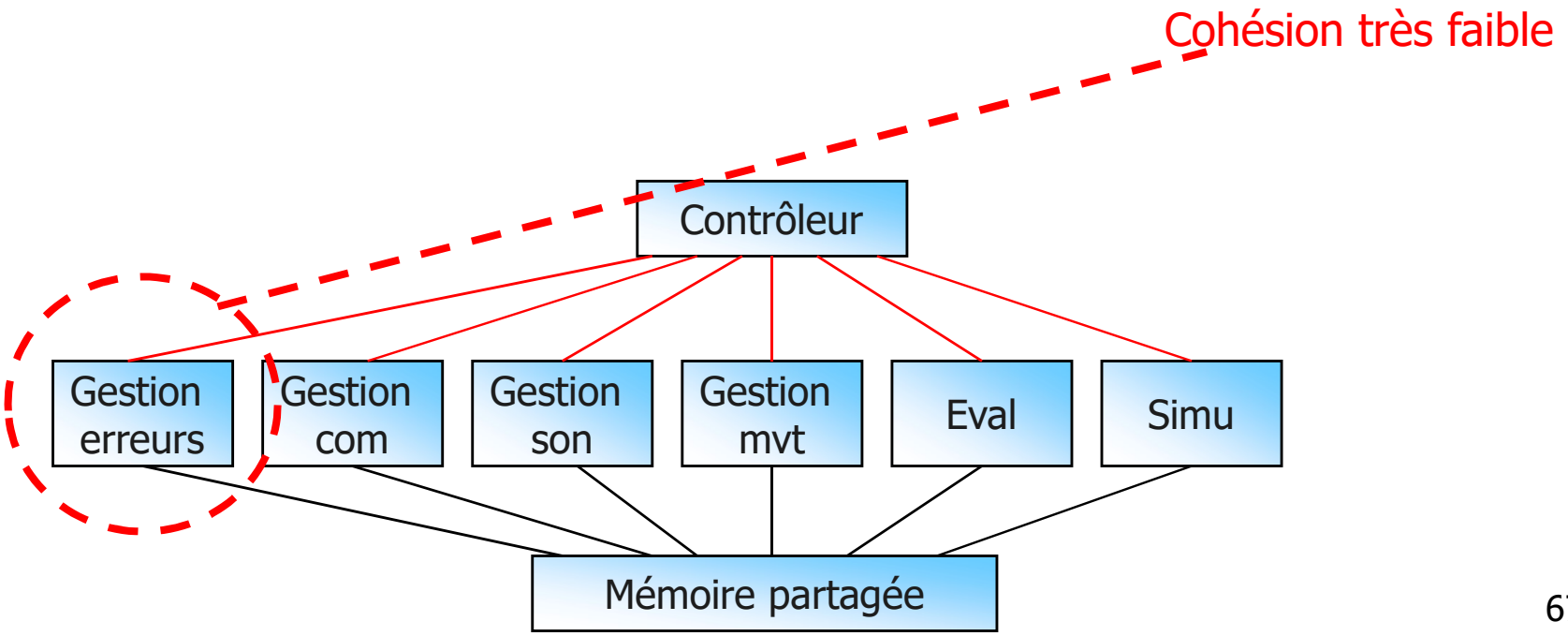
Cohésion accidentelle

- Les éléments d'un composant sont regroupés sans logique apparente, voire « au hasard »
  - modules de type « fourre-tout » (misc)
  - entrepôts de données utilisés par de nombreux composants



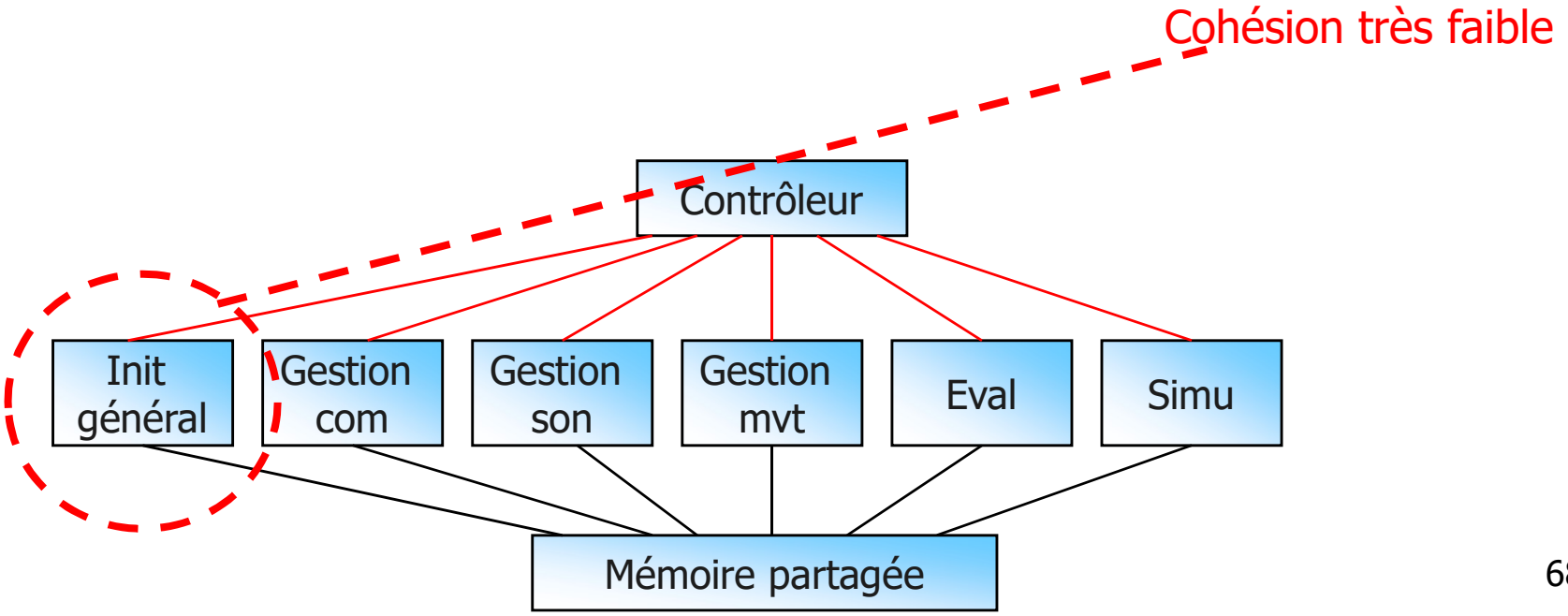
Cohésion syntaxique

- Les éléments d'un composant sont groupés sur une base « syntaxique »
  - Par nom de classes ou de procédures
  - Par type de classes (gestion des erreurs, gestion des communication)



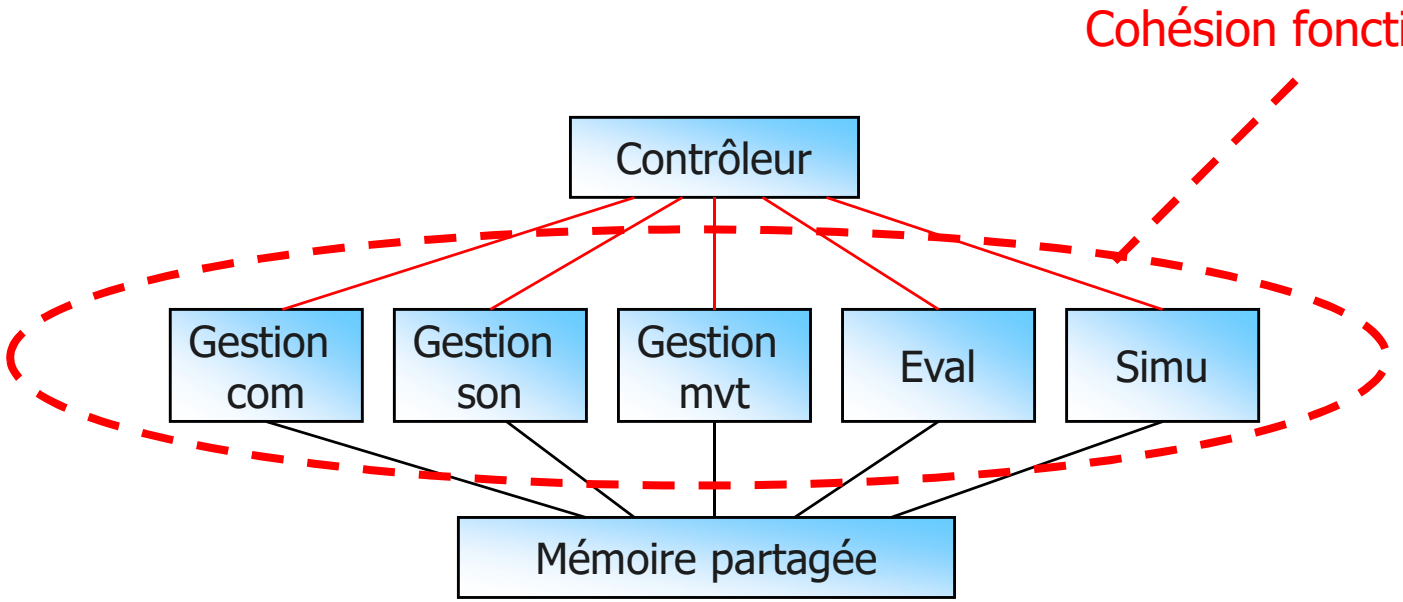
Cohésion temporelle

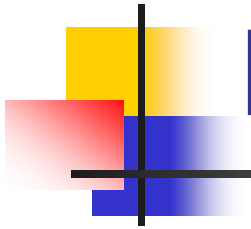
- Les éléments d'un composant sont groupés sur une base temporelle
  - éléments qui sont activés au même moment
  - éléments qui sont utilisés au même moment



Cohésion fonctionnelle

- Un composant regroupe tous les éléments nécessaires à l'exécution d'une fonction



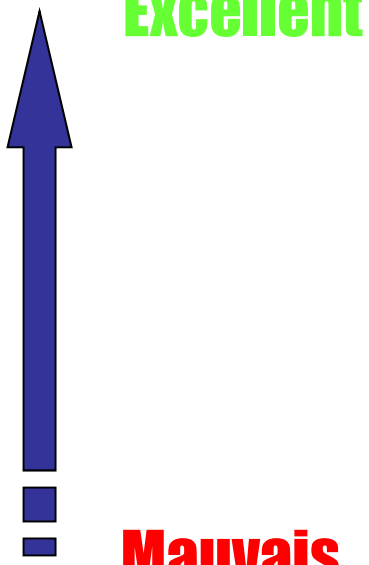


Le couplage

- Le couplage définit le niveau de dépendance entre deux ou plusieurs composants
  - c'est une mesure **qualitative**
- Un faible couplage favorise l'évolution
  - Concentre les modifications sur un composant
  - Pas ou peu d'effet de bord

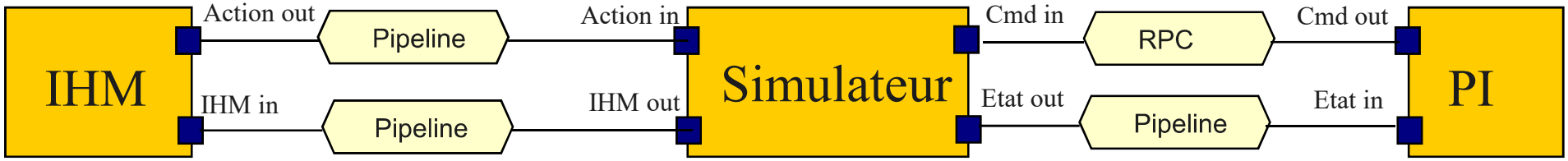
Niveaux de couplage

- On peut distinguer les niveaux de couplage suivants
  - absence de couplage direct
  - couplage de données
  - couplage par référence
  - couplage de contrôle
  - couplage externe
  - couplage commun
  - couplage de contenu



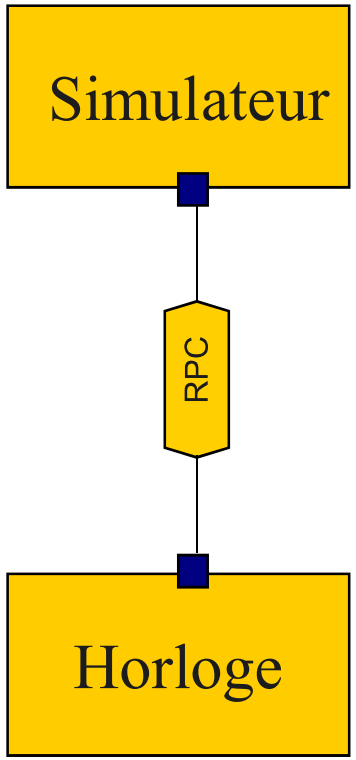
Absence de couplage direct

- Composants n'interagissant pas et ne partageant pas d'information
- Exemple
  - Deux modules très éloignés dans la structure architecturale qui ne partagent ni types ni données : IHM et PI
- Évolution
  - Aucun impact



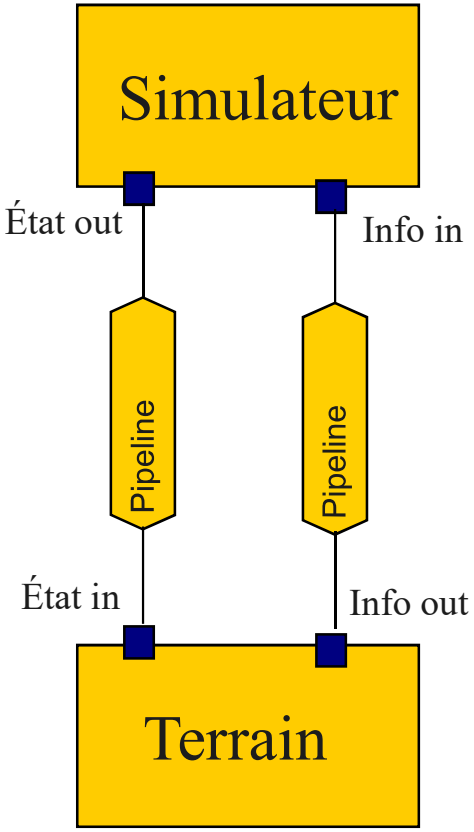
Couplage de données

- Deux composants n'échangent que des données simples via leurs interfaces (passage par valeur – pas de partage de données)
- Exemple
  - Simulateur et Horloge
- Les composants sont sensibles aux évolutions
  - des interfaces



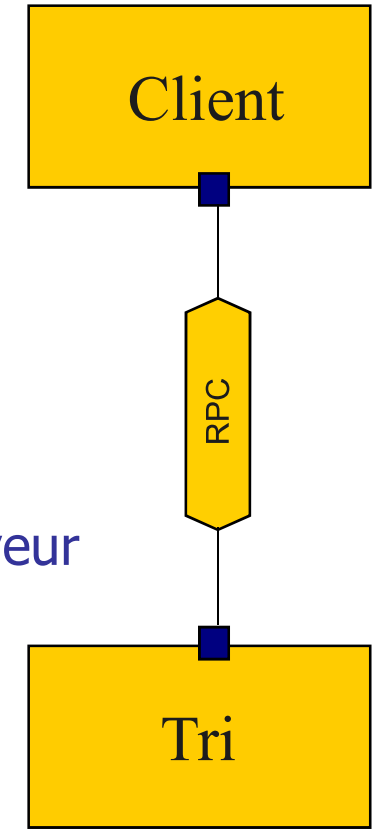
Couplage par référence

- Deux composants échangent des données structurées via leurs interfaces (éventuellement passage par adresse)
- Exemple
  - Simulateur et Terrain
- Les composants sont sensibles aux évolutions
  - des interfaces et
  - des structures de données



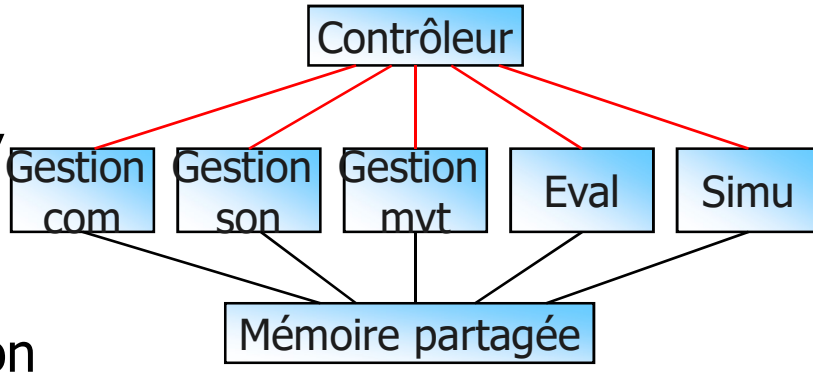
Couplage de contrôle

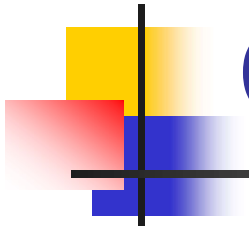
- L'interface d'un composant permet d'influencer son comportement
- Exemple :
  - Un composant de tri à qui on passe en paramètre la méthode de tri
- Le composant client est sensible aux évolutions
  - des interfaces,
  - des structures de données et
  - des fonctions internes du composant serveur



Couplage externe

- Deux composants communiquent par l'intermédiaire d'un tiers
- Exemple :
  - Les composants de simulation
- Évolution
  - Dépendance sur les données, voire sur le comportement. Le canal de communication n'étant pas identifié, il risque d'être oublié lors de l'évolution



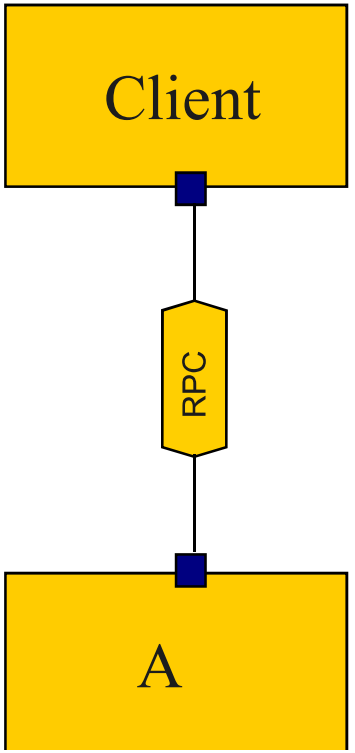


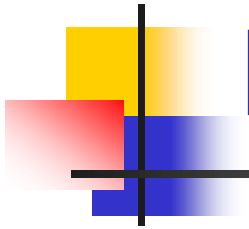
Couplage commun

- Deux composants partagent des variables globales
- Exemples :
  - Communication à base de variables partagées
  - Instruction COMMON de Fortran
- Évolution
  - L'utilisation de variables globales réclame une grande discipline d'utilisation
  - Plus une variable est utilisée, plus il est difficile de faire évoluer sa structure

Couplage de contenu

- Un composant connaît et exploite le contenu d'un autre (accès à des variables privées, à des constantes, à la structure logique,…). Ces informations n'étant pas publiées sur les interfaces.
- Exemples
  - Un composant « sait » que le résultat d'un composant de tri est une liste triée
- Évolution
  - Une fois que le contenu d'un module est connu et exploité, il ne peut plus évoluer.





Plan

- Introduction
- Une activité incrémentale
- Méthode de conception
- Cohésion et couplage
- **Conclusion**