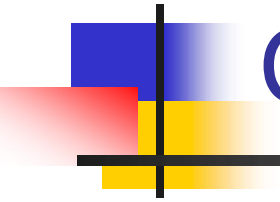
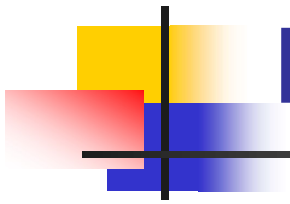


Génie Logiciel

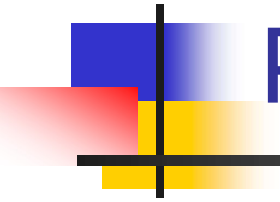
Lydie du Bousquet

Lydie.du-Bousquet@univ-grenoble-alpes.fr



Plan du cours

- Architecture (5 semaines)
- Test (4 semaines)



Partie architecture logicielle

Construite avec
Philippe Lalanda

pourquoi se préoccuper de l'architecture ?

L'histoire du Vasa

- Navire de guerre suédois
  - Construit entre 1626 et 1628
- Exigences
  - Très gros navire (2 ponts)
  - Canons sur les 2 ponts
  - Grand et pas large
- Résultat : le navire a coulé
  - à 1 mille marin de son point de départ,
  - lors de son 1er voyage
- Raisons de cet échec ?
  - Inexpérience
  - Non gestion des exigences

Images issues de Wikipédia

Que nous apprend cette histoire ?

- Les exigences étaient incompatibles
  - Le second bateau sur ce modèle mesurait 1,5 mètres de plus en largeur
  - Il n'a pas coulé
- L'architecte n'a pas bien fait son travail
  - les exigences fixées par le roi
  - Inexpérience

Que nous apprend cette histoire ?

- Le client impose toujours ses exigences
- Certaines sont incompatibles
- La phase d'architecture vise à identifier les conséquences des exigences
- En cas d'incompatibilités, l'architecte doit faire des **compromis**
- En cas d'échec, c'est la **responsabilité de l'architecte**, pas du roi !

« La phase d'architecture vise à identifier les conséquences des exigences »

Comment va-t-on s'y prendre ?

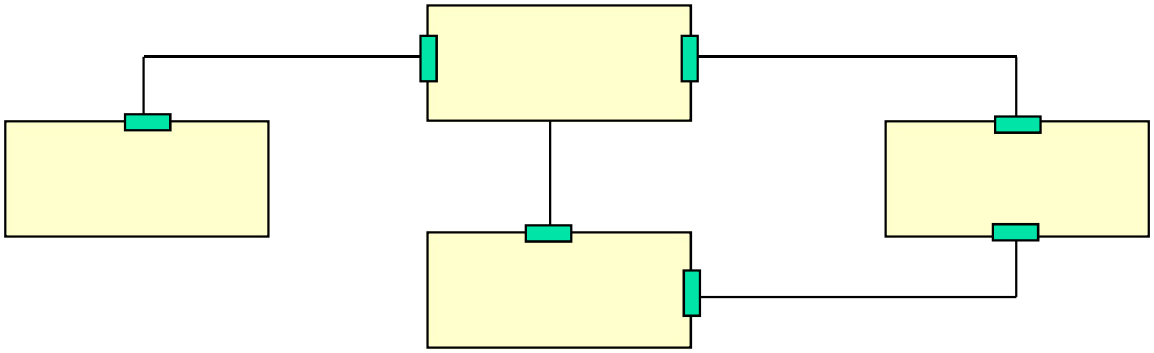
… identifier les conséquences des exigences

Distinguer une étape dans le développement

- **Représenter** une architecture
- **Évaluer** une solution
  - Ses qualités intrinsèques
  - Par rapport aux exigences
- **Capitaliser** l'expérience

Architecture logicielle : point de départ

- Une **représentation abstraite** d'un système



Une représentation

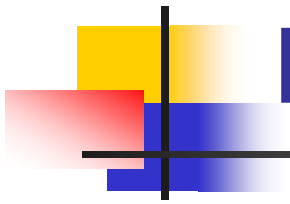
- C’est un support
  - de réflexion
  - de communication
- Que l’on peut utiliser pour
  - concevoir
  - modifier

Le contenu du cours

- Une méthode de **représentation** d'architecture

- Des éléments pour **concevoir** une architecture
  - méthode CBSP (University of Southern California)
  - styles architecturaux et de tactiques
  - démarche incrémentale

- Des éléments pour **évaluer** une architecture



Les savoir-faire qui seront évaluées

- Capacité à **produire** une solution architecturale raisonnable qui s'appuie sur des styles et des tactiques

- Capacité à **représenter**
  cette solution de façon rigoureuse
  avec la méthode de représentation vue en cours

- Capacité à **justifier** correctement vos choix

Les savoir-faire qui seront évaluées

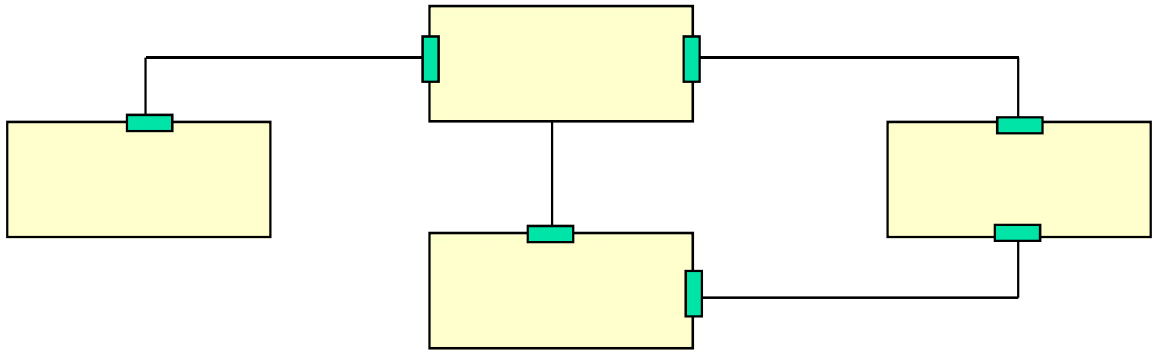
- Capacité à **produire** une solution architecturale raisonnable qui s'appuie sur des styles et des tactiques

- Capacité à **représenter**
  cette solution de façon rigoureuse
  avec la méthode de représentation vue en cours

- Capacité à **justifier** correctement vos choix

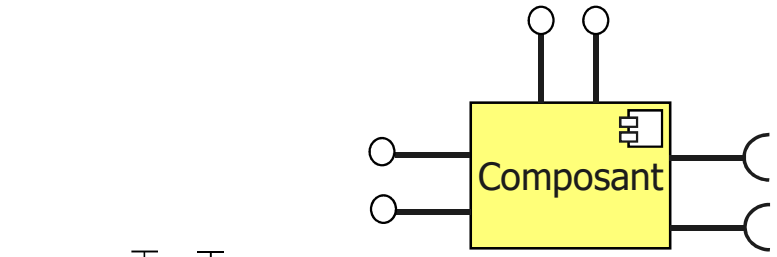
Définition de l'architecture

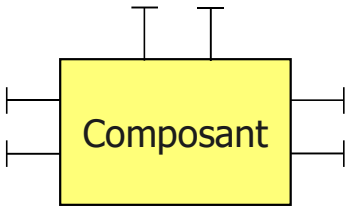
- Une architecture logicielle est une **représentation abstraite** d'un système exprimée essentiellement à l'aide de composants logiciels (et des connecteurs)



Composants logiciels

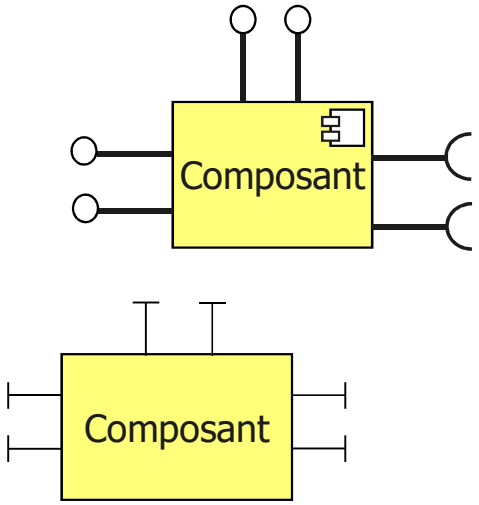
- Les composants sont des **spécifications d'unités fonctionnelles**
  - clairement définies
  - sémantiquement cohérentes et compréhensibles
- Développés ou acquis
- Ne pas confondre
  - spécification
  - réalisation





Description des composants

- Propriétés fonctionnelles
  - Services requis
  - Services fournis
  - Cycle de vie
- Contraintes
  - Type de communication
  - Ordonnancement
- Propriétés non fonctionnelles
  - Performance, robustesse, …

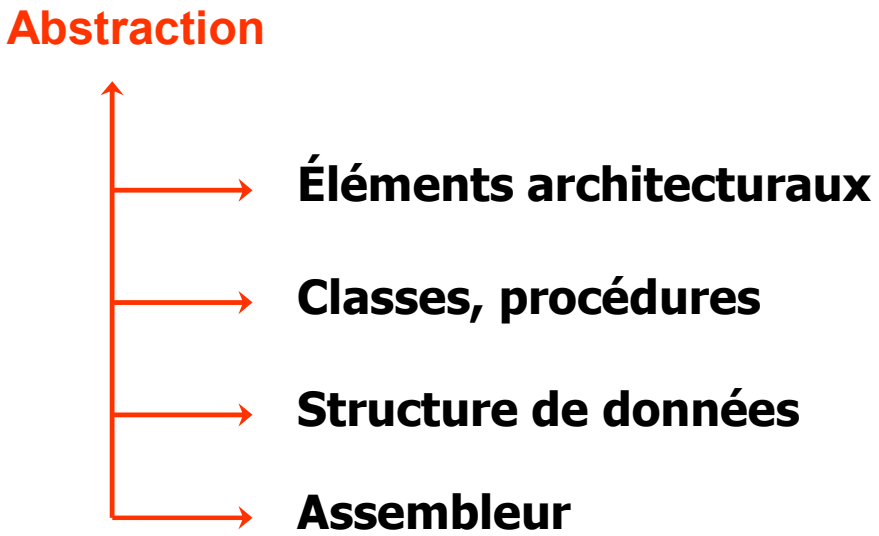


Connecteurs

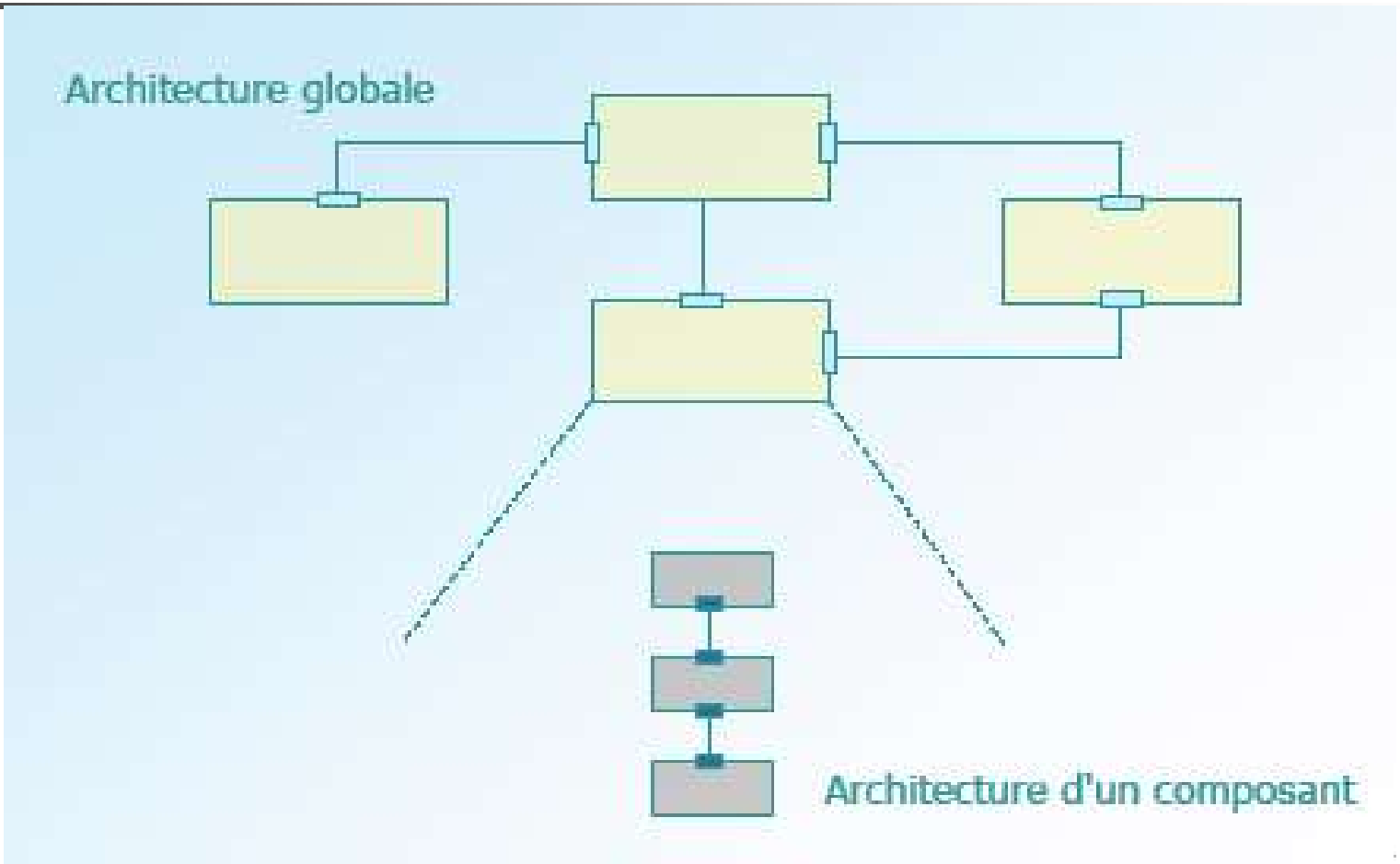
- Assurent les interactions entre composants
- Peuvent être de complexité variable
  - du simple appel de méthode à l'ordonnanceur
- Permettent la flexibilité et l'évolution
- Pas de langage de spécification de connecteurs
- A introduire au fur et à mesure que l'on va détailler l'architecture

L'architecture est une abstraction supplémentaire

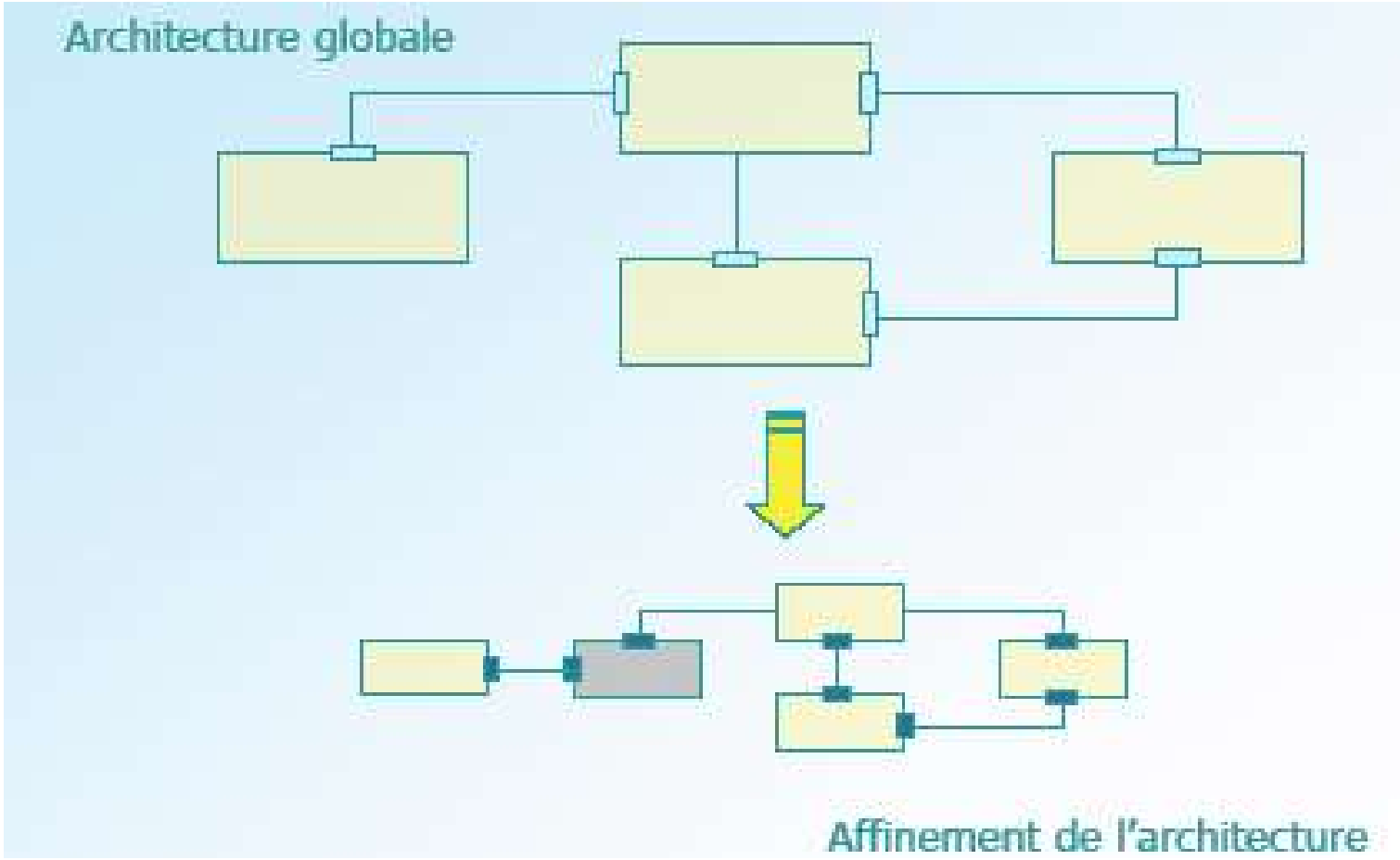
- Ne fournit que les propriétés externes des éléments structurants
- Ne se préoccupe pas des détails d'implantation



Une succession d'abstractions

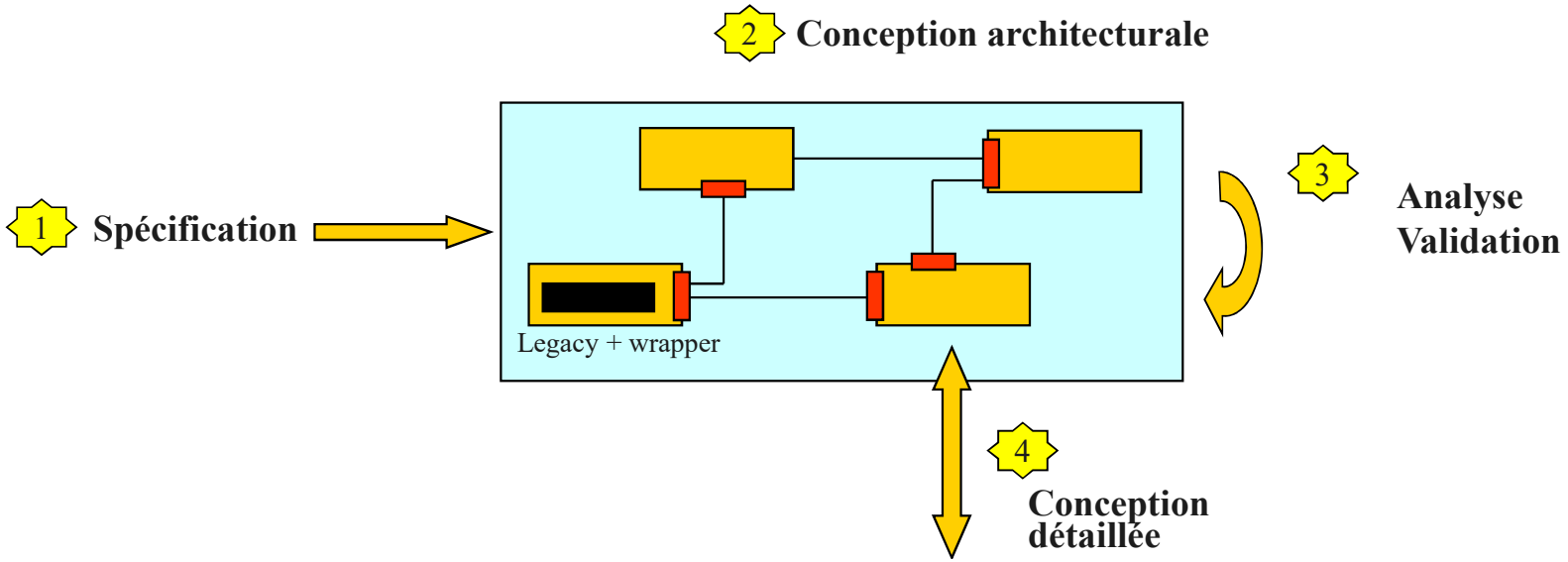


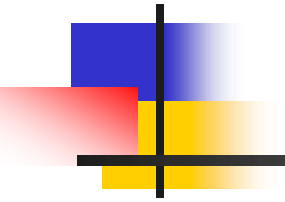
Une succession d'abstractions



Positionnement dans le processus de développement

- L'architecture = première étape de conception
  - réduire la complexité du système abordé en le structurant en composants logiciels



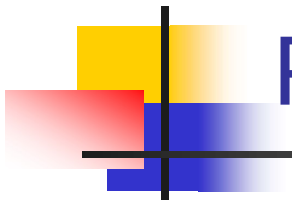


Description architecturale

Lydie du Bousquet & Philippe Lalanda
Université Grenoble Alpes

Qu’est-ce que pourrait-être une « bonne » description architecturale ?

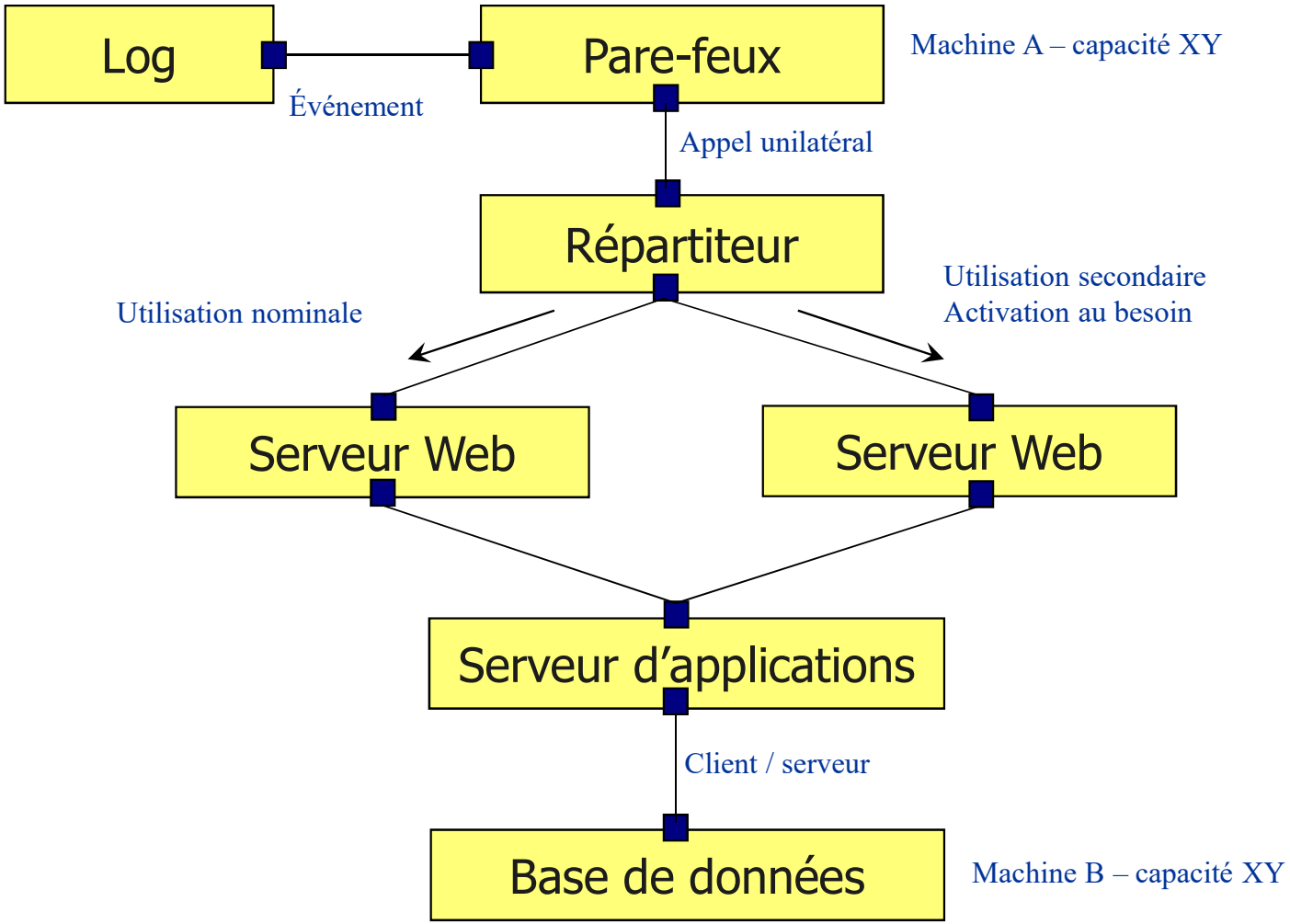
Que faudrait-il décrire ?

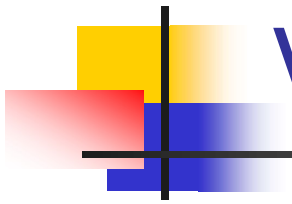


Représentation des architectures

- Le but est de représenter toutes les informations liées aux composants logiciels :
  - leur structure et leurs interfaces
  - Leurs interactions
  - leurs propriétés et les contraintes associées
  - leurs supports d'exécution, …
- L'idéal est de tout représenter
  - sous forme graphique et sur un seul schéma
  - C'est ce qui se fait la plupart du temps

Exemple


Log
Événement
Pare-feux
Machine A – capacité XY
Appel unilatéral
Répartiteur
Utilisation nominale
Utilisation secondaire
Activation au besoin
Serveur Web
Serveur Web
Serveur d'applications
Client / serveur
Base de données
Machine B – capacité XY



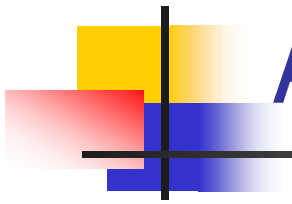
Verdict

- Hétérogène, incomplet, peu lisible, voire ambigu
  - Informations très différentes qui ne s'adressent pas aux même personnes
    - mélange des préoccupations
    - difficile à lire
  - Il manque de nombreuses informations
    - Combien de machines ?
    - Activation du premier serveur web ?
    - Protocole de communication entre le pare-feux et le répartiteur ?
    - etc.
- Besoin d'une approche structurée et répétable

Analogie

- Dans un bâtiment, on doit également manipuler des structures différentes
  - la topologie (pièces, couloirs, portes, etc.)
  - le câblage électrique
  - la plomberie
  - la ventilation, etc.

- Plans spécialisés servant de spécification
  - utilisés par des personnes différentes
    - électriciens, maçons, etc.
  - utilisés pour apporter des propriétés différentes
    - Densité des murs, diamètres des câbles, pression des tuyaux, …

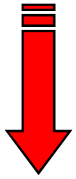


Application au logiciel

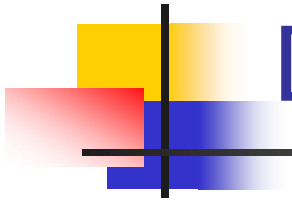
- Un logiciel est également composé de plusieurs structures (Parnas, 74)
  - Représentation indépendante des ces différentes structures au niveau des programmes
- On peut aussi appliquer cette décomposition au niveau architecture
- Notion de vues architecturales

Première conclusion

Architecture = composants + connecteurs

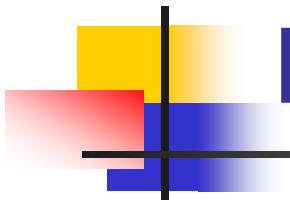


Représentation = ensemble de vues



Définition des vues logicielles

- Une vue offre une perspective spécifique sur un logiciel
  - Séparation des préoccupations
- Une vue définit :
  - Les éléments logiciels représentables sur cette vue
  - Les relations représentables
  - Un formalisme
  - Éventuellement un vocabulaire
  - Éventuellement un langage de contraintes
- On utilise plusieurs types de vues complémentaires
- Elles doivent être complètes, cohérentes

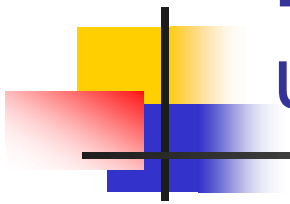


Plusieurs vues

- On peut se focaliser et travailler sur un aspect donné
- On gagne en cohérence locale
- On abaisse significativement la complexité

- L'architecture se dématérialise un peu …
- Cohérence générale ?
- Et comment recoller les morceaux ?

De quelles vues a-t-on besoin pour représenter une architecture ?



Avant toute chose

il faut cerner le système qui doit être représenté.

- les limites et
- les fonctionnalités / contraintes

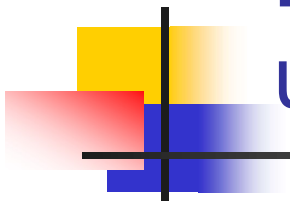
☞ sous forme de vues

De quelles vues a-t-on besoin pour représenter une architecture ?

- Pour identifier les **limites** du système
  - **Diagramme de contexte**
  - Identifie le système comme un tout
  - Et tous les acteurs qui

- Pour identifier les **fonctionnalités** du système et les contraintes
  - **Diagramme de cas d'utilisation**
  - Résumé du cahier des charges

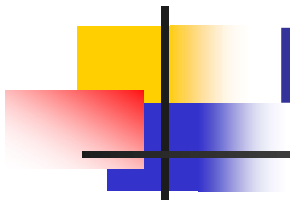
De quelles vues a-t-on besoin pour représenter une architecture ?



- **Structure** de l'application
  - Les composants et la façon dont ils sont liés
  - **Vue(s) logique(s)**
- **Comportement**
  - Les interactions entre les composants au cours du temps
  - **Vue(s) dynamique(s)**
- **Machines**
  - Projection des composants vers un environnement d'exécution
  - **Vue physique** (répartition des composants sur les machines)
  - Vue d'allocation (si focus sur les processus)

Exemple : bornes de péage

1. Le conducteur doit introduire un ticket et sa carte bancaire
2. Le système enregistre la transaction et vérifie la validité de la carte
3. Toutes les transactions journalières sont transmises à la banque en même temps

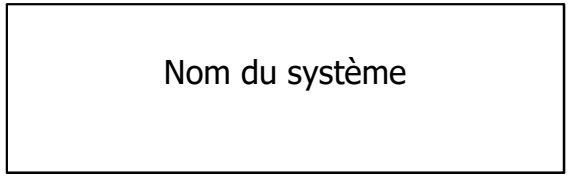


Description architecturale

- Un diagramme de contexte
- Une liste de fonctionnalités
- Des vues logiques (structure)
  - Une vue abstraite qui identifie les principaux composants
  - Des vues plus spécialisées qui décrivent ces composants
- Des vues dynamiques (comportement)
- Une vue physique (support d'exécution)

Diagramme de contexte
Pour identifier les limites du système

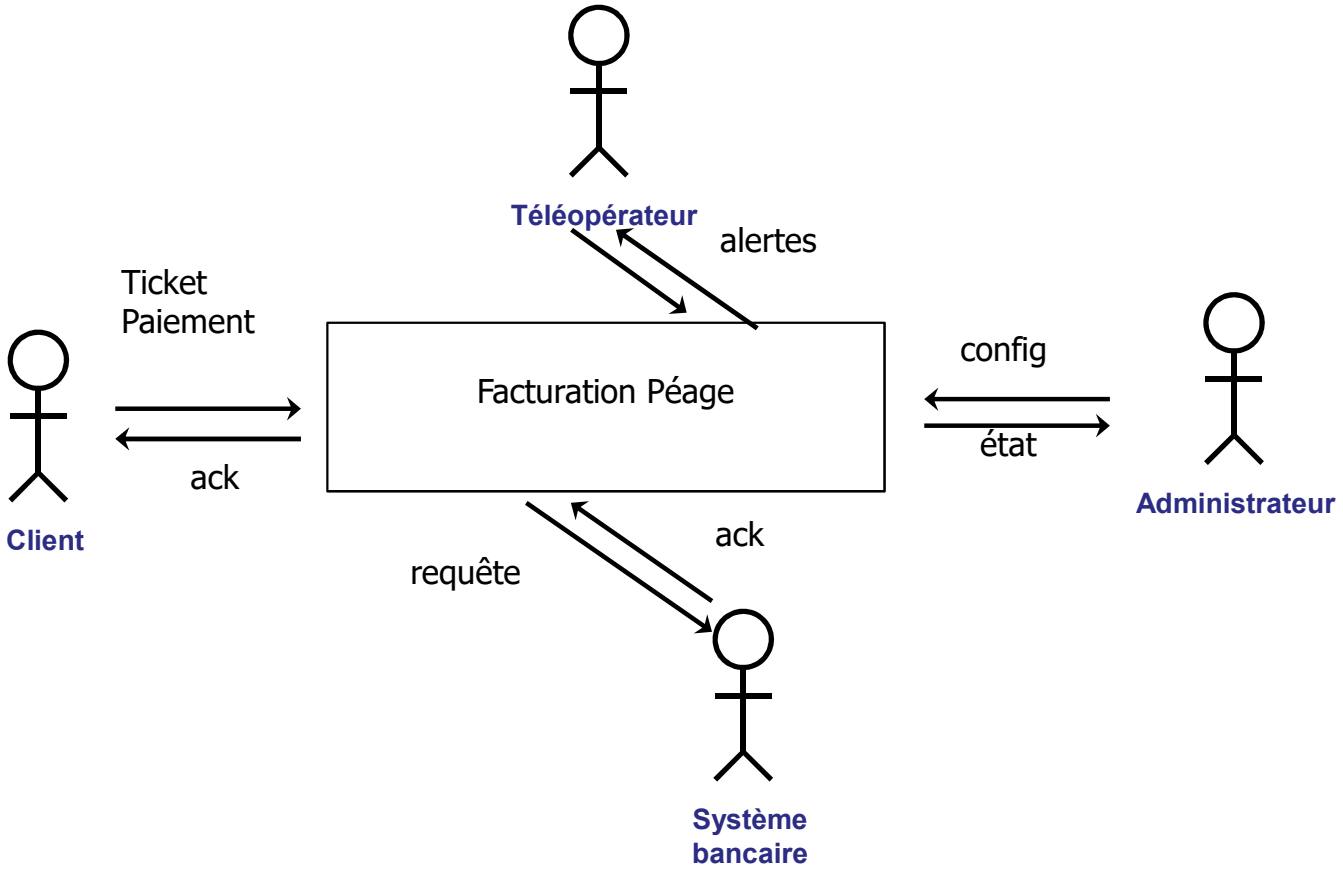
- Le système



- Tous les acteurs
  - humains
  - autres systèmes ou logiciels

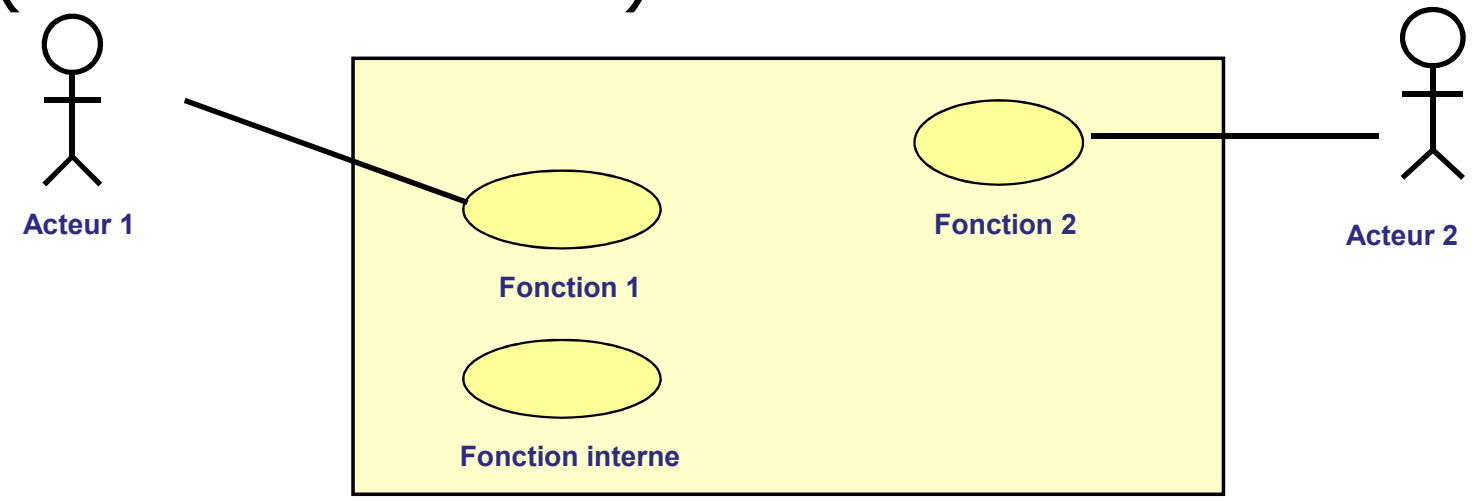
- Nature des interactions

Exemple

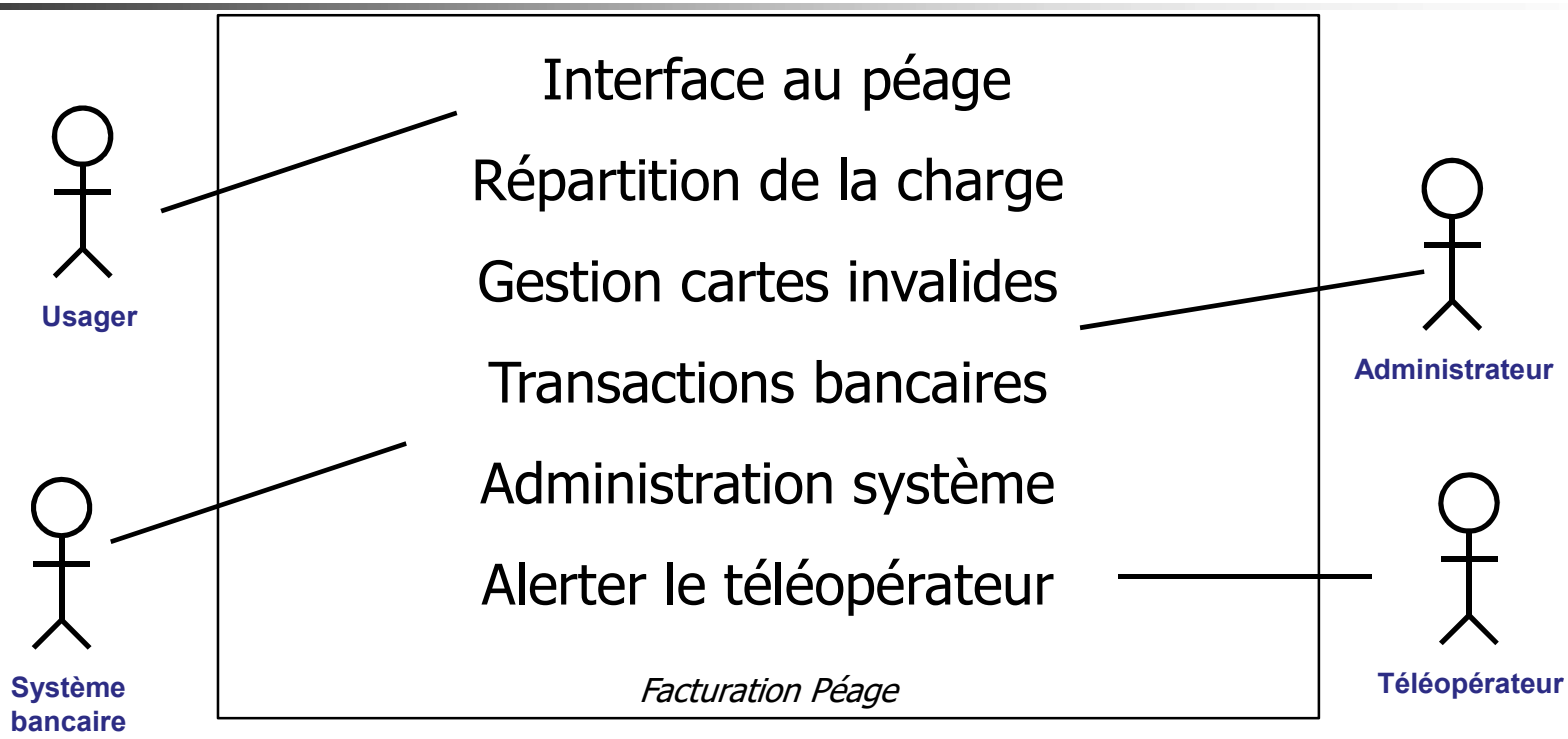

Téléopérateur
alertes
Ticket
Paiement
Facturation Péage
config
état
Administrateur
Client
ack
ack
requête
Système
bancaire

Les fonctionnalités pour indiquer ce qui doit être construit

- Zoom du diagramme de contexte
- Liste de **toutes** les fonctionnalités (externes+internes) + contraintes



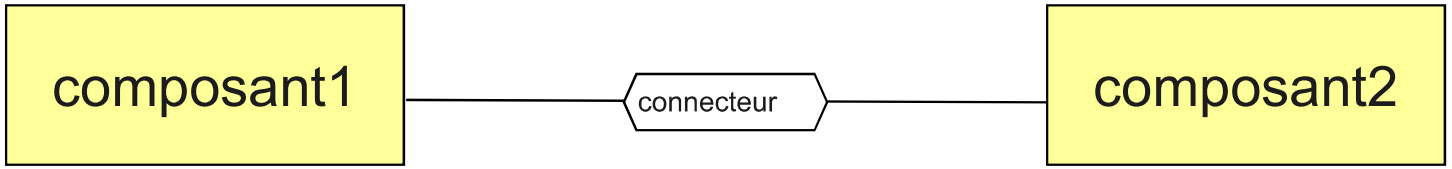
Exemple



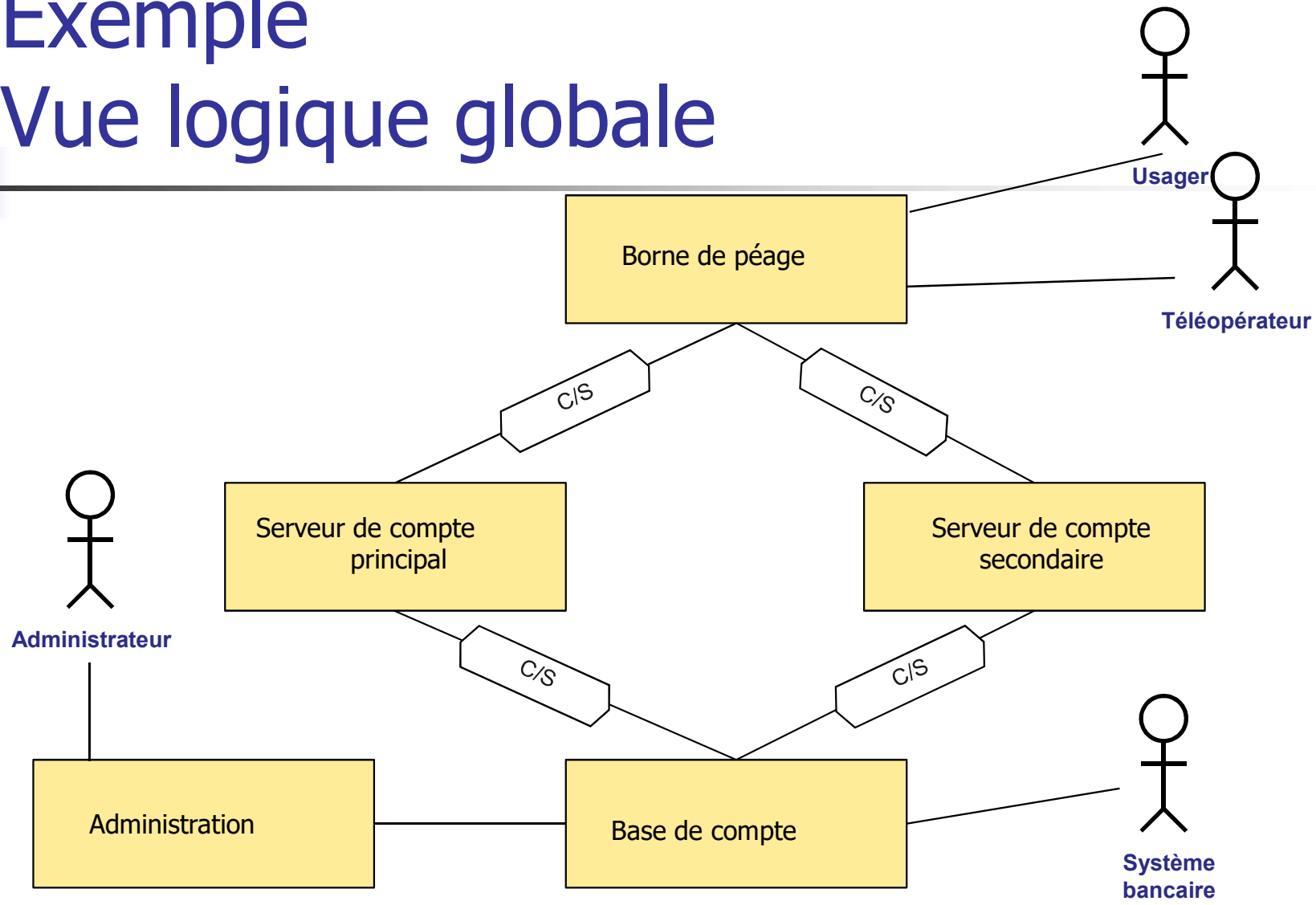
Contraintes : sécurité des transactions; performance du système

Vue(s) logique(s)
Pour décrire la structure de l'application

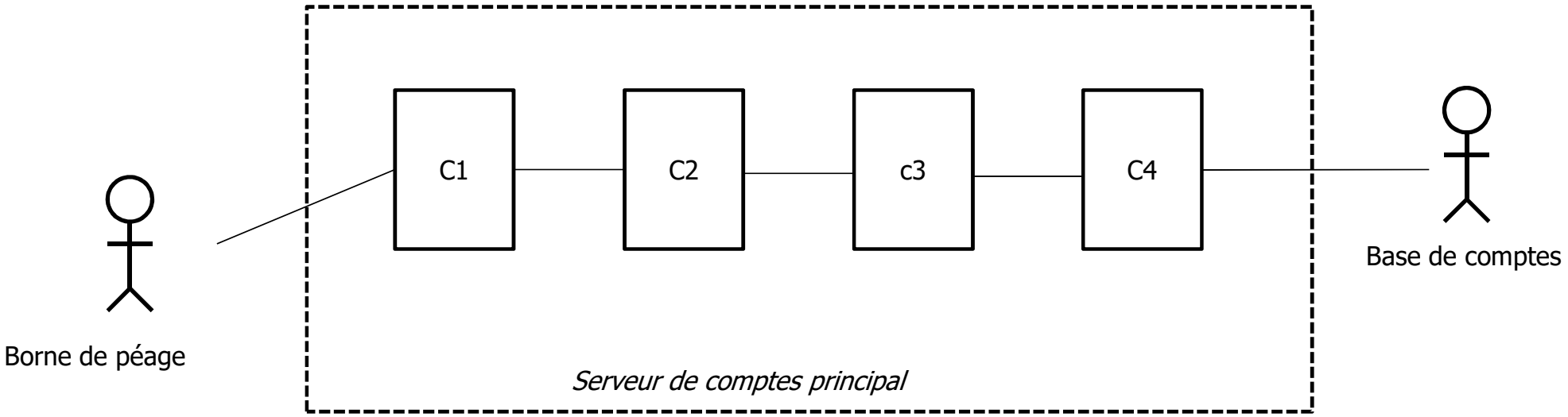
- Structure = l'**ensemble de composants et les connexions**
  - Composant : boite + nom
  - Connexion : simple trait (+ connecteur)



Exemple
Vue logique globale



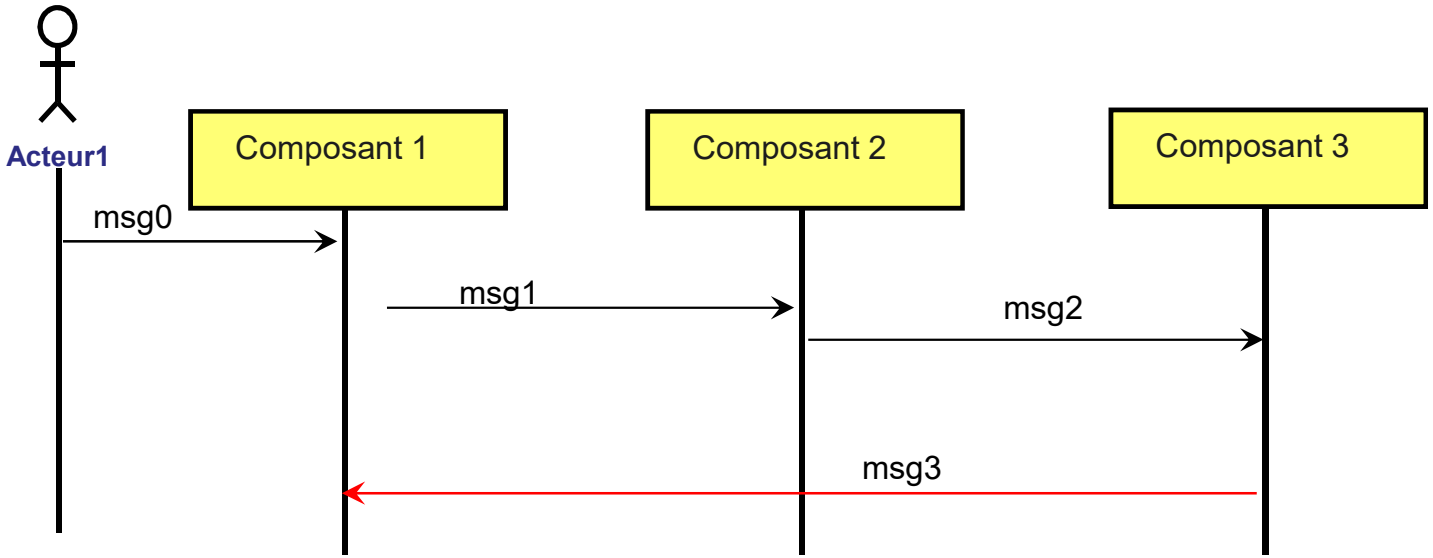
Exemple
Vue logique, raffinement d'une composant



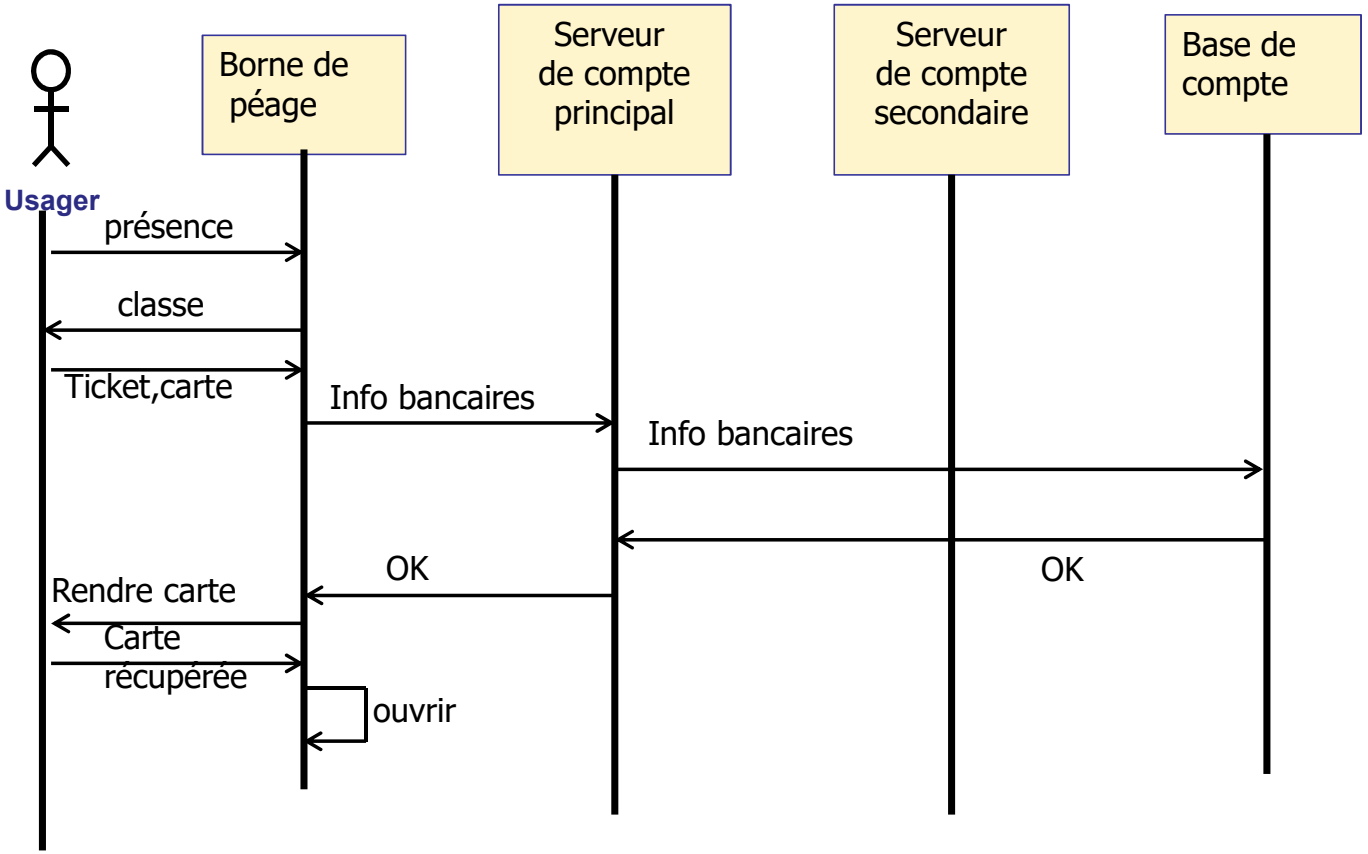
« Borne de péage » et « Base de compte » sont des **acteurs** ici,
parce qu'ils se situent ***en dehors*** du composant « serveur de comptes principal »

Vue(s) dynamique(s)
Pour illustrer la façon dont les composants interagissent

- Définit le comportement dynamique de l'application dans un **contexte** donné
  - Composants
  - Axe temporel
  - Interactions entre les composants (message et données)

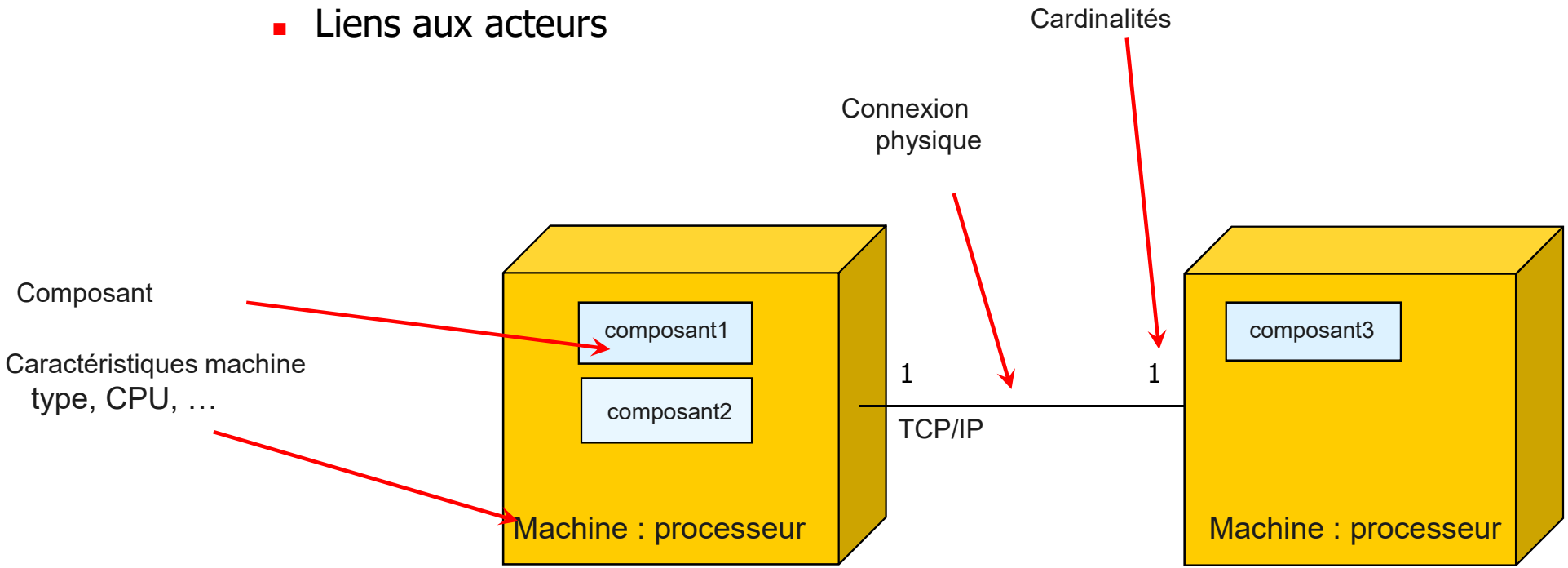


Exemple du péage :
passage d'un véhicule, sans erreurs

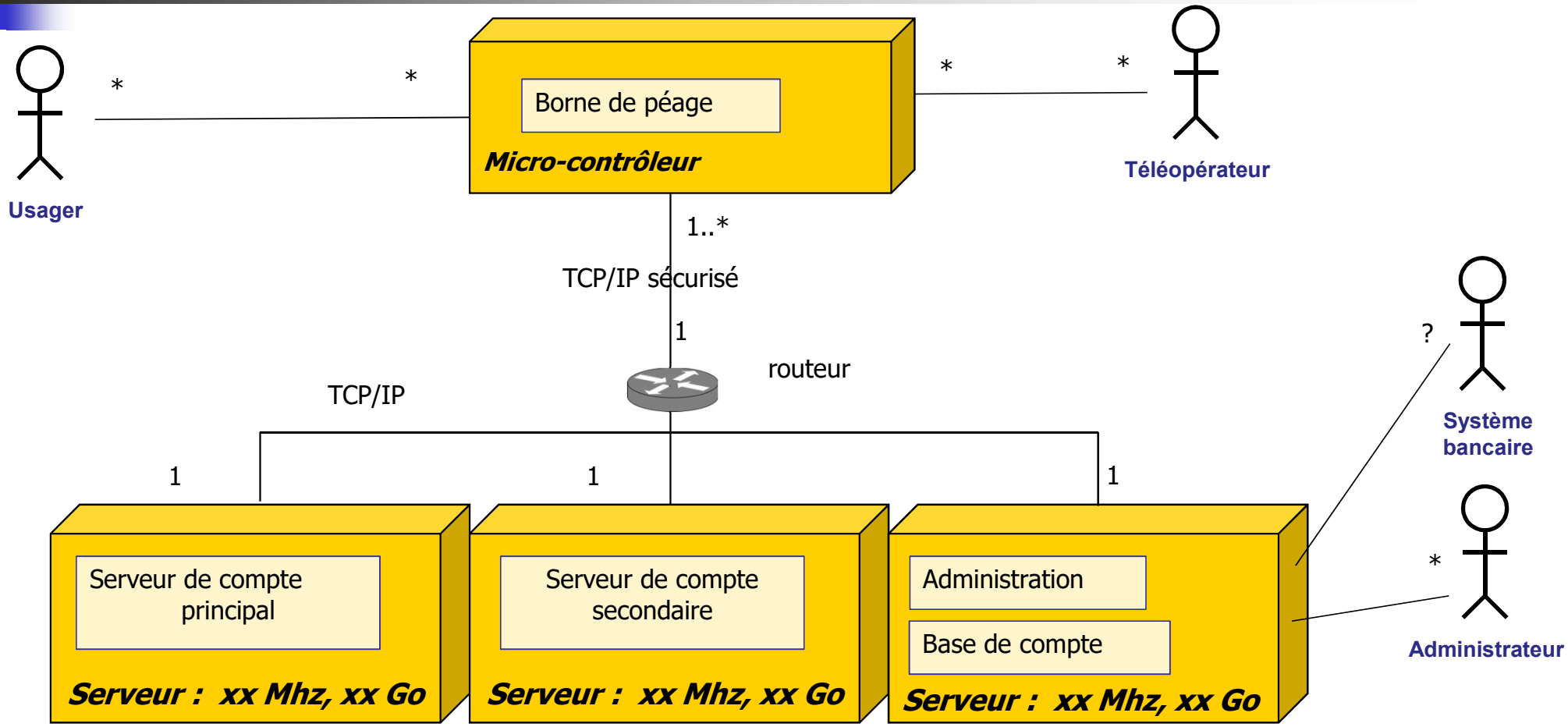


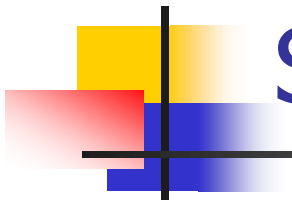
Vue physique
Pour décrire la répartition sur le matériel

- Quels composants sur quelles plate-formes d'exécution
  - Propriétés liées aux éléments hardware (CPU, mémoire, disques, …)
  - Cardinalités
  - Liens aux acteurs



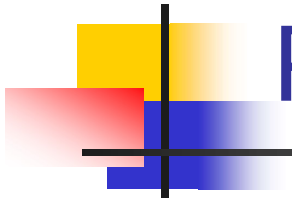
Exemple du péage
Vue physique





Synthèse : une architecture

- Représentée par plusieurs vues complémentaires
  - Séparation des préoccupations, raffinements successifs
  - Attention à assurer la cohérence et la complétude
- Commencer par reformuler la spécification
  - Diagramme de contexte
  - Liste des fonctionnalités
- Décrire l'architecture elle-même :
  - Vues logiques : 1 vue globale + vues spécialisées
  - Vues dynamiques contextualisées
  - Vue physique



Références

- Software architecture in practice - second edition
  Len Bass, Paul Clements, Rick Kazman
  Addison Wesley, 2003

- Pattern-oriented software architecture
  Buschmann, Meunier, Rohnert, Sommerlad, Stal
  Wiley, 1996

- Applied software architecture
  Hofmeister, Nord, Soni
  Addison Wesley, 2000

- Design and use of software architectures
  Jan Bosch
  Addison Wesley, 2000