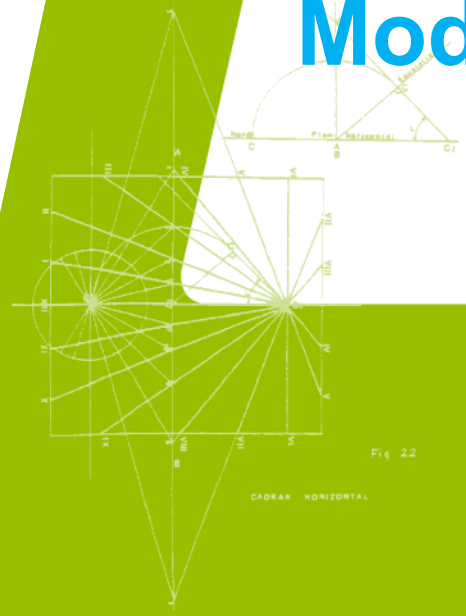
ICT301
2025-2026

Université de Yaoundé I

ICT301 : Architecture logicielle et conception

Modélisation d'architectures



Novembre 2025

Valéry MONTHE

valery.monthe@facsciences-uy1.cm

Bureau R114, Bloc pédagogique 1

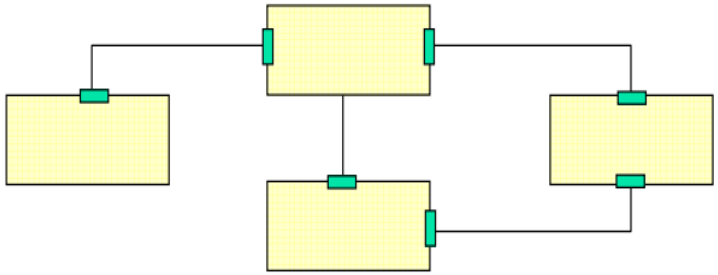
Tiré du cours de Lydie du Bousquet

Plan du cours

1. Architecture technique – Architecture logicielle
2. Composants d'architecture et description
3. Impact de l'architecture sur la qualité du système
4. Description d'architecture logicielle
5. Modéliser l'architecture (avec UML)
6. Exercices d'application

Architecture logicielle

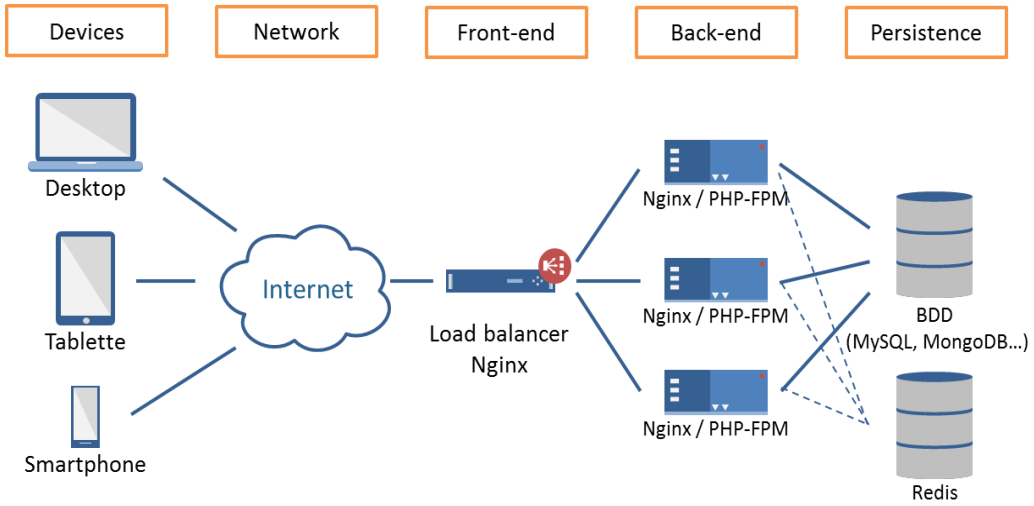
- Une **architecture logicielle** est une représentation abstraite d'un système exprimée essentiellement à l'aide de composants logiciels en interaction via des connecteurs.
- Se définit dans la phase de conception, en termes de :
  - Organisation interne et découpage d'un logiciel en « modules »
  - Description de la nature des modules, leurs responsabilités et fonctionnalités, et la nature de leurs relations
  - Donne des premières réponses sur comment sera structuré le futur logiciel, avant le début du travail de programmation
- Phase d'analyse vs phase de conception
  - « quoi faire » vs « comment faire »

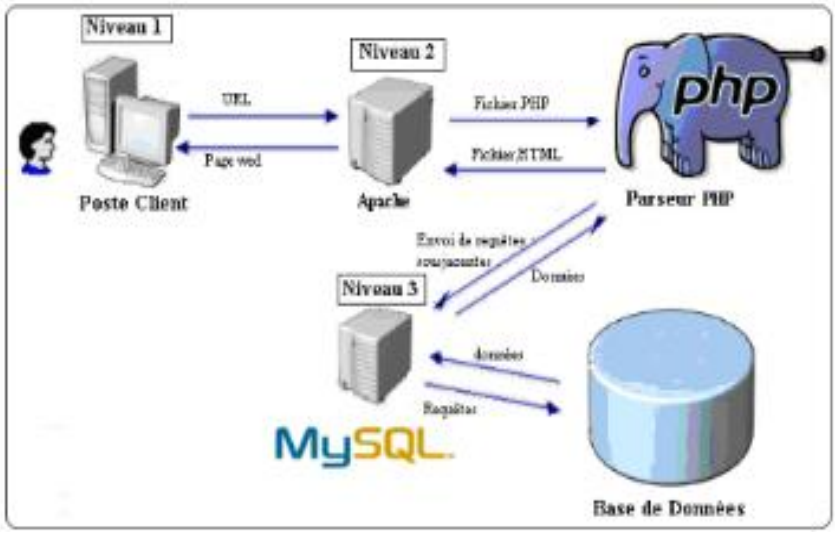


Architecture Technique

- Eléments matériels supportant le logiciel (serveur, poste de travail, équipements réseau, stockage, …)
- Composants logiciels déployés sur les éléments matériels (OS, SGBD, serveur web, serveur DNS, composants spécifiques au logiciels, etc…)

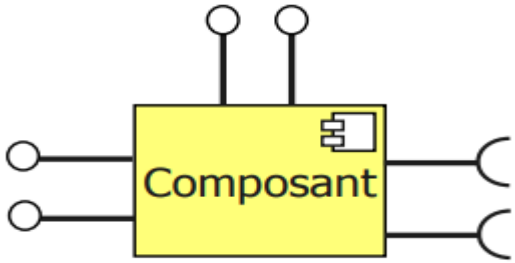
➡ Il s'agit donc de la répartition des éléments logiciels entre les matériels existants ou nécessaires.

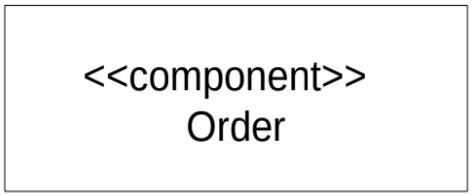


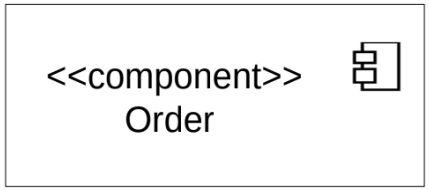


Composants logiciels

- Les composants sont des spécifications d'unités fonctionnelles
  - ✓Clairement définies
  - ✓Sémantiquement cohérent et compréhensible
- Développés ou acquis
- Ne pas confondre
  - ✓Spécification
  - ✓Réalisation







Description des Composants

- Propriétés fonctionnelles
  - ✓Service requis
  - ✓Services fournis
- Contraintes
  - ✓Type de communication
  - ✓Ordonnancement
- Propriétés non fonctionnelles
  - ✓Performance
  - ✓Robustesse
  - ✓Etc.

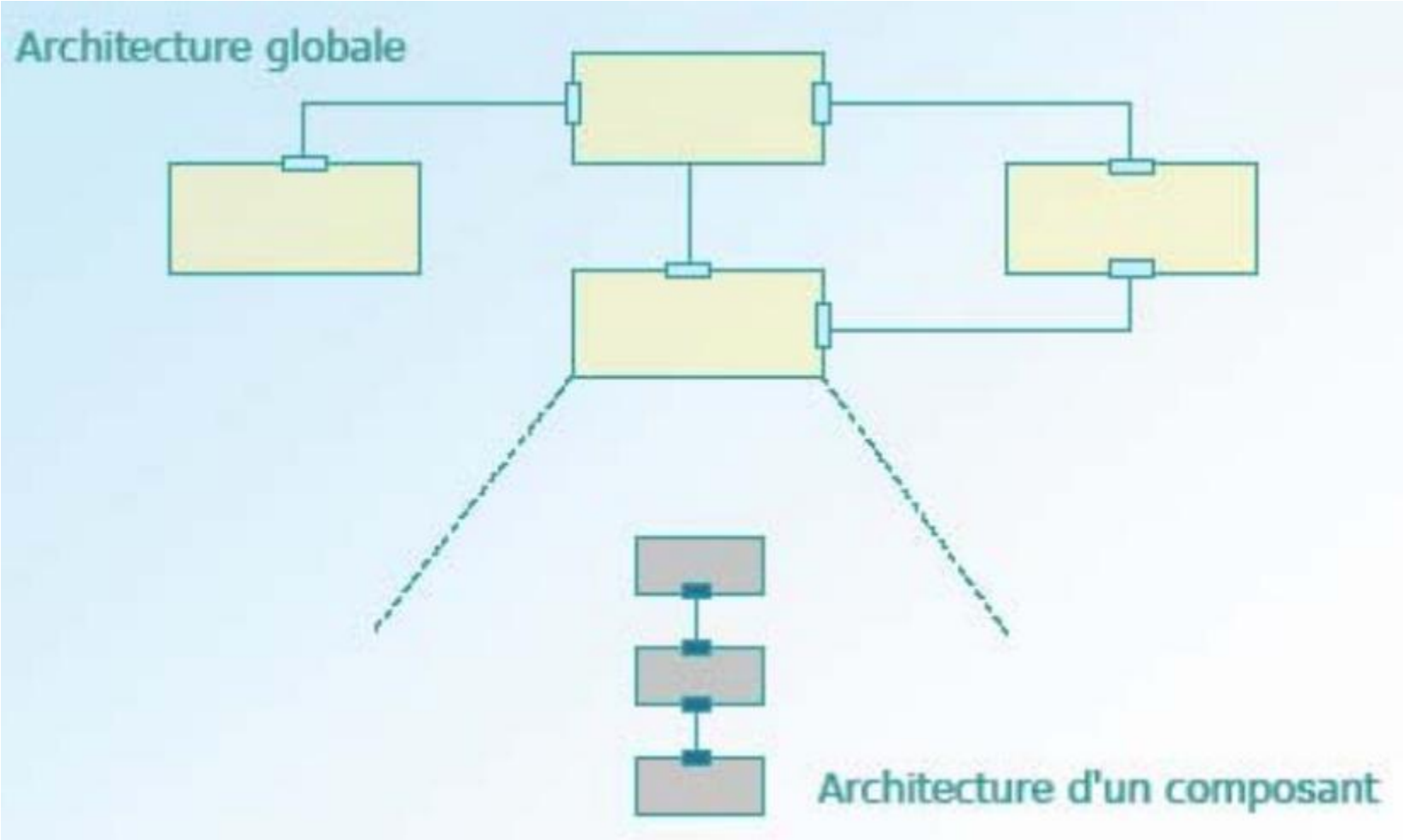
Connecteurs

- 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

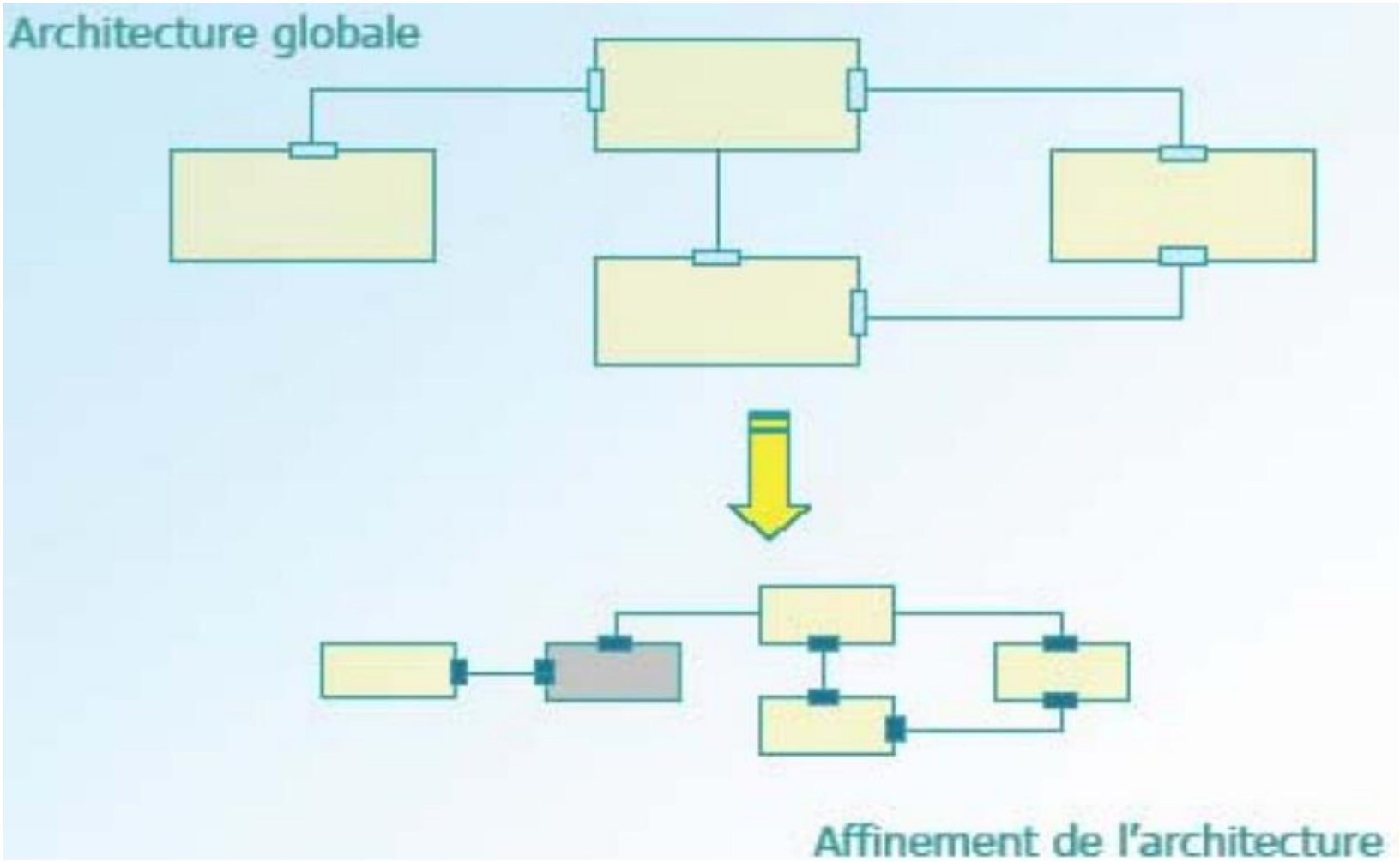
L’architecture : une abstraction supplémentaire

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

Succession d'abstractions

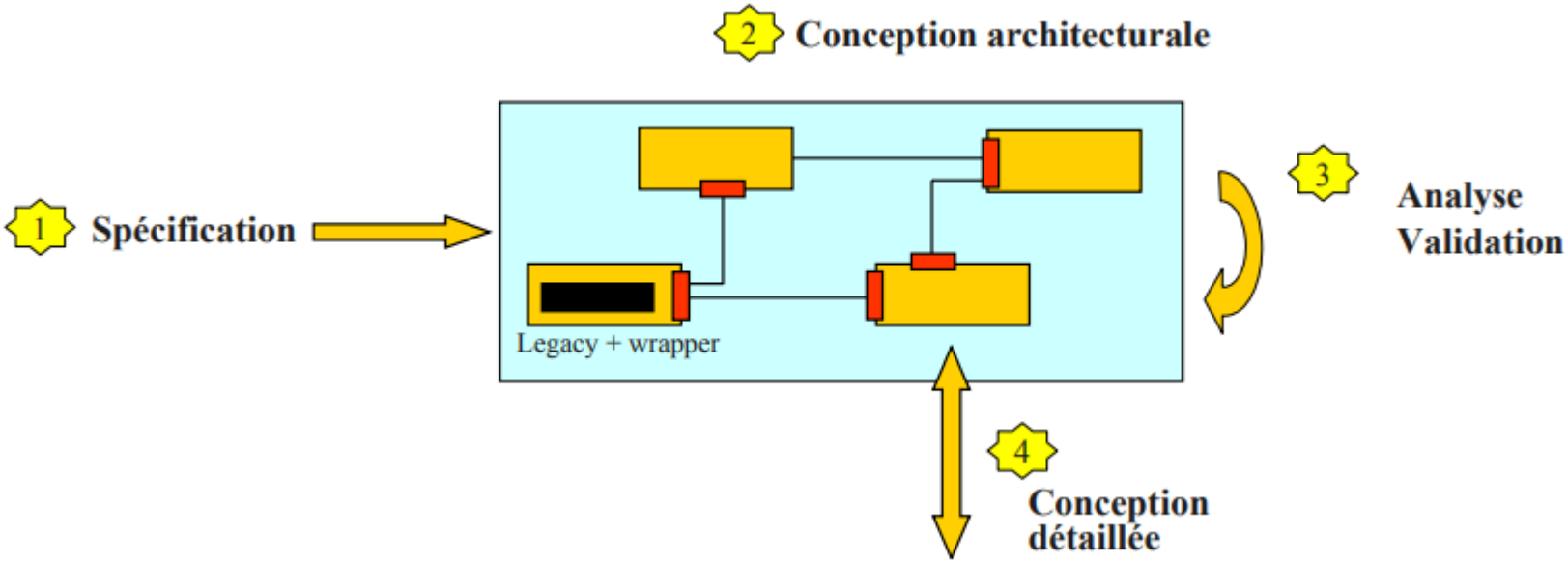


Succession d'abstractions



Etape dans le cycle de développement

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



Exemple : application e-Pharmacie

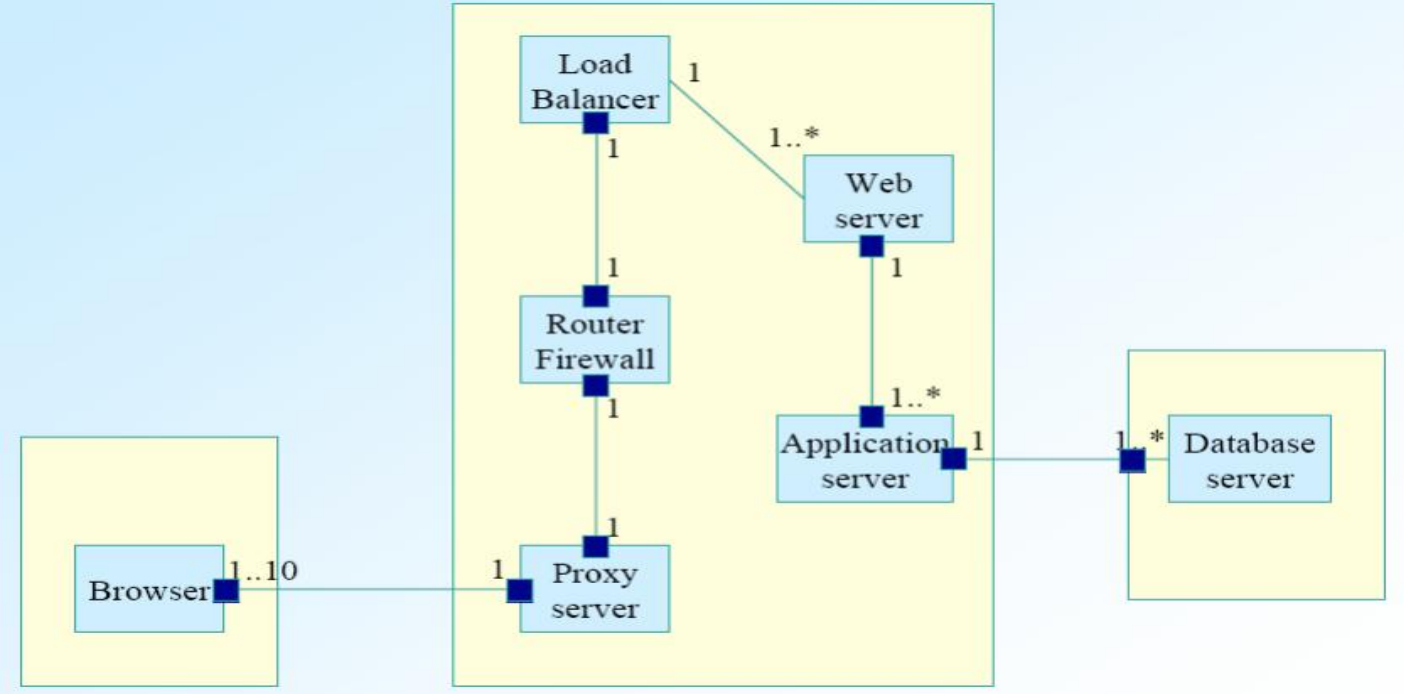
- Une pharmacie souhaite acquérir une application de e-commerce pour la vente de ses produits en ligne.
  - Concevoir l'architecture

- Exemple d' architecture Technique

Exemple : application e-Pharmacie

- Représentation à l'aide des composants

Affinement



Intérêt de l'architecture

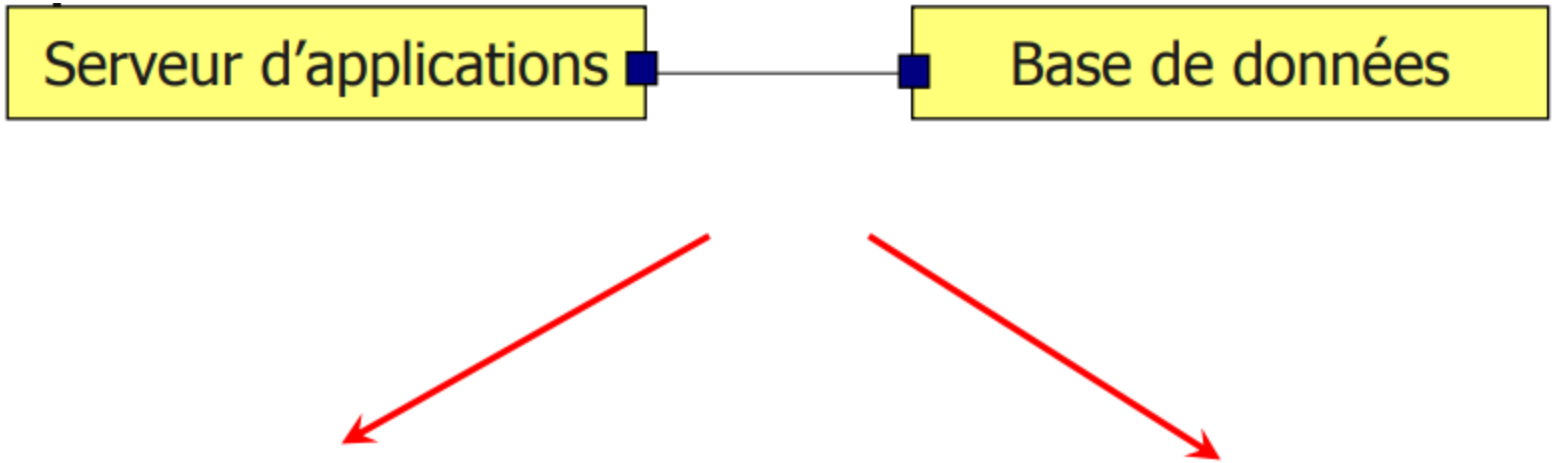
- **Aide à répondre aux questions :**
  - Comment développer ?
  - Quelles équipes, quelles technologies?
  - Quel coût
- **A partir de l'architecture, on peut :**
  - Définir un plan de travail
  - Répartir le travail entre les équipes
  - Allouer les ressources
  - Imposer des contraintes techniques
  - Structurer les différentes étapes
    - ✓ Le développement
    - ✓ Les tests
    - ✓ La documentation
    - ✓ La maintenance

Impact sur la qualité logicielle

- L'architecture a une forte **influence** sur les propriétés finales d'un système

- La structuration architecturale **favorise ou pénalise** les propriétés non fonctionnelles telles que :
  - ✓ Sécurité
  - ✓ Sûreté
  - ✓ Disponibilité
  - ✓ Maintenabilité
  - ✓ etc.

Impact sur la qualité logicielle : Exemple



Sur la même machine

+ sécurité
+ performance (à voir)

Sur deux machines

+ disponibilité (caches, ...)
+ maintenabilité
+ sûreté (réplication possible)

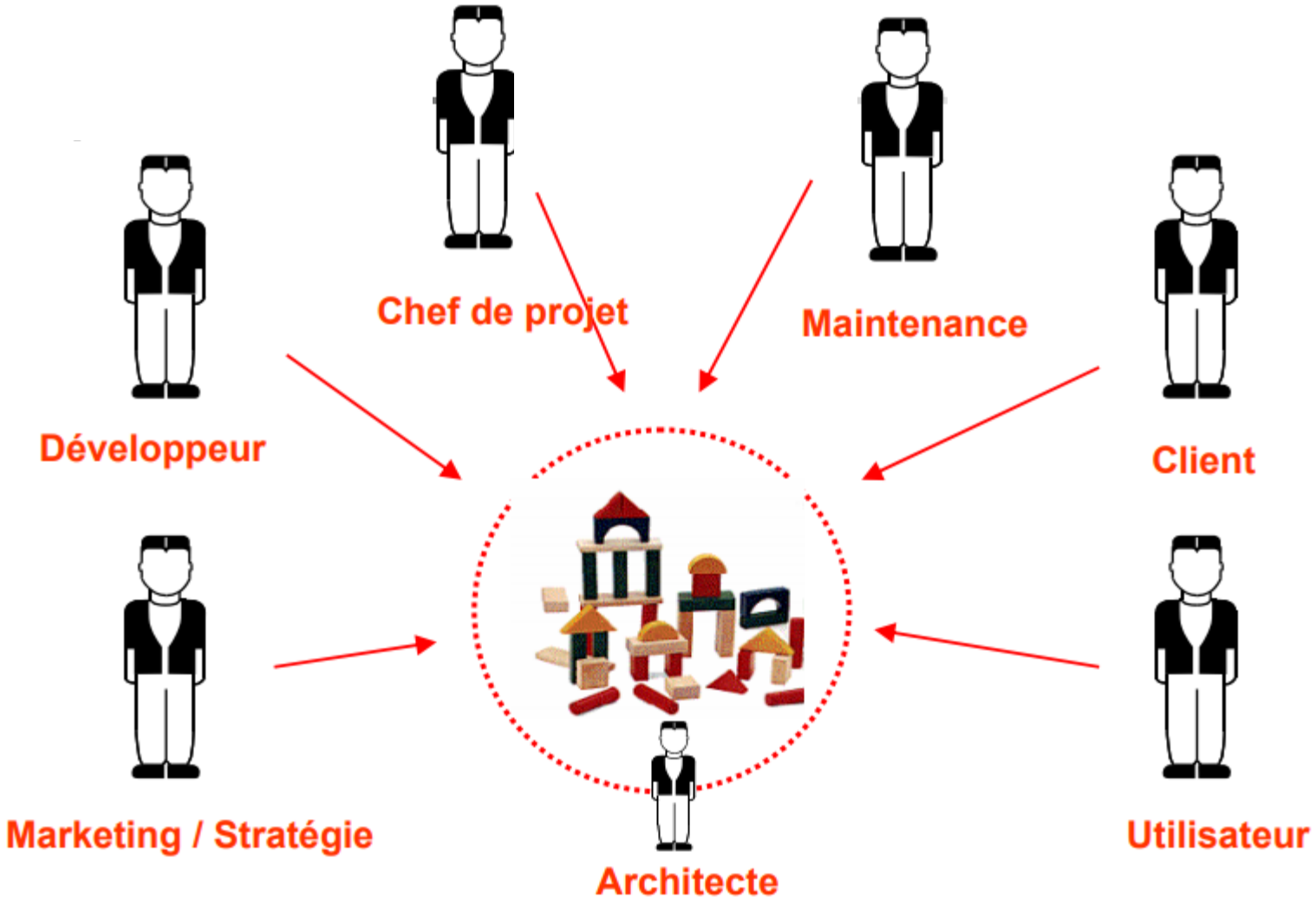
Des éléments à prendre en compte

- Peu de composants favorisera la performance (moins de communication)
- Beaucoup de composants favorisera la maintenance (au détriment de la performance)
- La redondance peut favoriser la sûreté, mais pas forcément la sécurité
- Le casse-tête commence !

Vecteur de communication

- L'architecture fournit un canevas permettant à tous d'exprimer leurs intérêts et de négocier :
  - ✓ réunion de tous les intervenants autour de l'architecture
  - ✓ négociation des exigences avec les utilisateurs
  - ✓ négociation des évolutions à apporter
  - ✓ présentation régulière aux clients et au management des avancées (fonctions / coûts / échéances)
  - ✓ structuration des équipes et allocations des ressources

Un lieu de rencontre



Exercice : Enoncé

Pour chaque besoin décrit ci-dessous, dire s'il s'agit de produire une architecture ou de réaliser une conception

1. On veut une couche de GUI, une couche d'analyse et une couche de stockage des données.
2. Toutes les nouvelles applications doivent étendre l'interface *Application.*
3. L'application sera disponible comme un service déployé dans le nuage.
4. On a besoin d'une base de données NoSQL avec un taux de disponibilité élevé.
5. Une méthode prend le type d'un objet comme paramètre et retourne une instance de ce type en appelant le constructeur privé de la classe correspondante.

Exercice : Solution

Pour chaque besoin décrit ci-dessous, dire s'il s'agit de produire une architecture ou de réaliser une conception

1. On veut une couche de GUI, une couche d'analyse et une couche de stockage des données. **Architecture** **Conception**
2. Toutes les nouvelles applications doivent étendre l'interface *Application*. **Architecture** **Conception**
3. L'application sera disponible comme un service déployé dans le nuage. **Architecture** **Conception**
4. On a besoin d'une base de données NoSQL avec un taux de disponibilité élevé. **Architecture** **Conception**
5. Une méthode prend le type d'un objet comme paramètre et retourne une instance de ce type en appelant le constructeur privé de la classe correspondante. **Architecture** **Conception**

Représentation des architectures

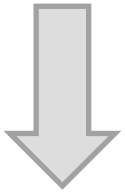
Représentation des architectures

- 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

Représentation des architectures

- Un logiciel est constitué de plusieurs structures
- Représentation indépendante de ces structures au niveau des programmes
- Représentation sous quelle vue ?

- **Architecture = composants + connecteurs**



- **Représentation = ensemble de vues**

Définition des vues logicielles

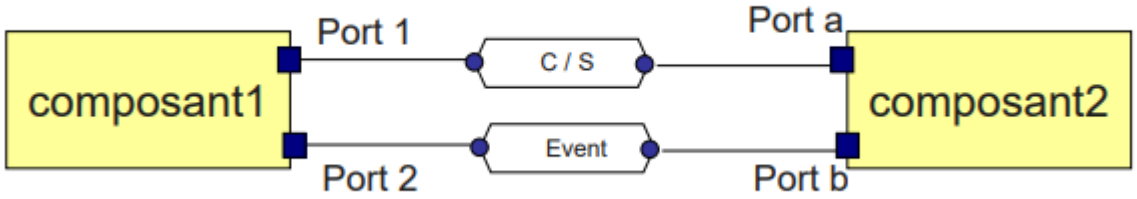
- Une vue offre une perspective spécifique sur un logiciel
  - ✓ Séparation des préoccupations
- Une vue définit :
  - ✓ Les éléments logiciels représentables sur cette vue
  - ✓ Les relations représentables
  - ✓ Un formalisme
  - ✓ Éventuellement un vocabulaire
  - ✓ Éventuellement un langage de contraintes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

Vues (Structurelles) d'une architecture

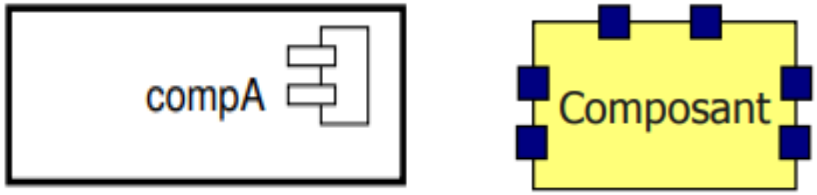
- **Vue logique**
  - ✓ Description logique du système décomposé en sous-systèmes (modules + interface)
  - ✓ Comment le logiciel est structuré en unité d'exécution (les composants)
    - *UML : diagramme de paquetages*
- **Vue d'implémentation ou dynamique.**
  - ✓ Description de l'implémentation du système logiciel en termes de composants et de connecteurs
  - ✓ Comment les composant interagissent au cours du temps
    - *UML : diagramme de composants*
- **Vue de déploiement ou d'allocation.**
  - ✓ Description de l'intégration et de la distribution de la partie logicielle sur la partie matérielle
  - ✓ Projection des composants vers un environnement d'exécution
    - *UML: diagramme combiné de composants et de déploiement*

Description de l'architecture : vue logique

- Cette vue définit la structure de l'architecture : décomposition en éléments logiques
- Tous les composants et leurs connexions sont décrits
- Les ports sont spécifiés
  - ✓ Fonctionnalités fournies par les composants
  - ✓ Fonctionnalités requises par le composant
- Les contraintes
  - ✓ Type de communication à utiliser
  - ✓ Etc.
- Les propriétés non fonctionnelles
  - ✓ Performance, puissance, robustesse, …

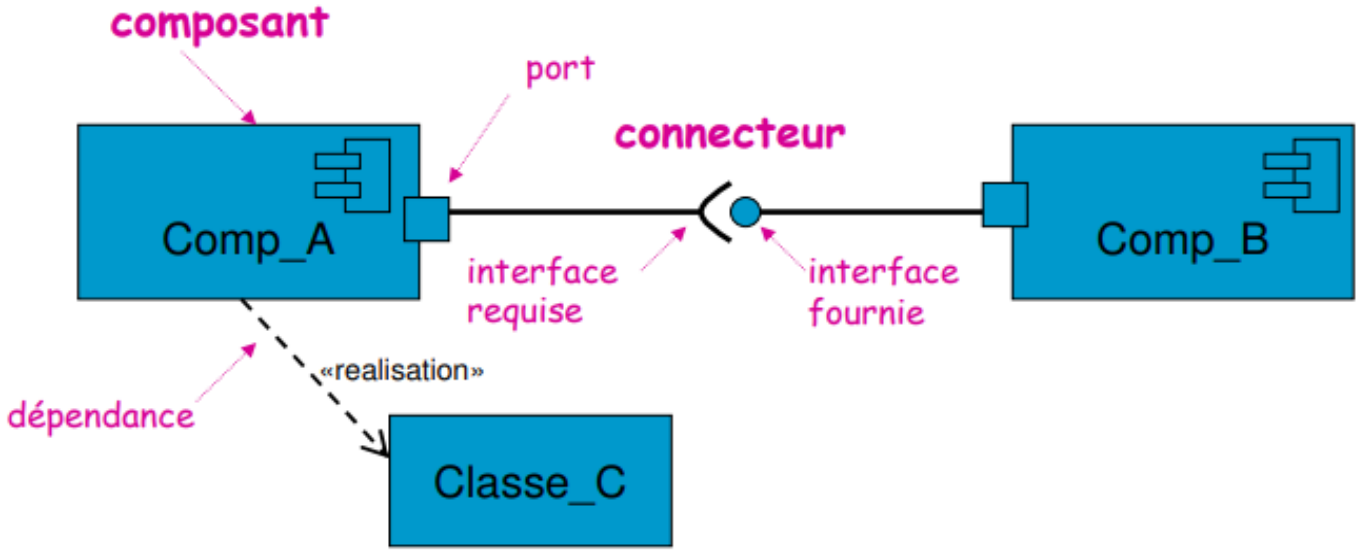


Composant



- Encapsule un traitement et/ou des données
- Encapsule un sous-ensemble de fonctionnalités et/ou de données du système
- Restreint l'accès à ce sous-ensemble au moyen d'une interface définie explicitement
- Possède des dépendances explicitement définies pour exprimer les contraintes requises par son contexte d'exécution ou sa réalisation

Vue logique : éléments architecturaux

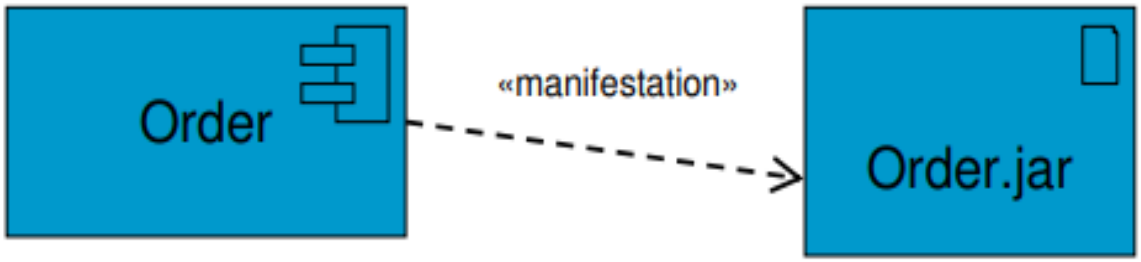


- Deux ou plusieurs composants interagissent via un connecteur
- Chaque élément architectural possède une structure et/ou comportement pouvant être décrit par un modèle UML approprié

Vue logique : éléments architecturaux

Composant

- Unité autonome servant de bloc de construction pour le système
- Les composants implémentent typiquement des services spécifiques à l'application
- La **manifestation concrète d'un composant** est appelée **artéfact** (*instance du composant déployée sur le matériel*)
  - N'importe quel type de code sur n'importe quel support numérique
  - Code source, fichiers binaires, scripts, fichiers exécutables, bases de données, applications, etc.



Vue logique : éléments architecturaux

Interface de composant

- Permet à un composant d'exposer les moyens à utiliser pour communiquer avec lui
- Types d'interfaces
  - ***Interface offerte*** : définit la façon de demander l'accès à un service offert par le composant
  - ***Interface requise*** : définit le type de services (aide) requis(attendu) par le composant
- Une interface est attachée à un port du composant
- Port = point de communication du composant
- Plusieurs interfaces peuvent être attachées à un même port

Vue logique : éléments architecturaux

Dépendances entre composants

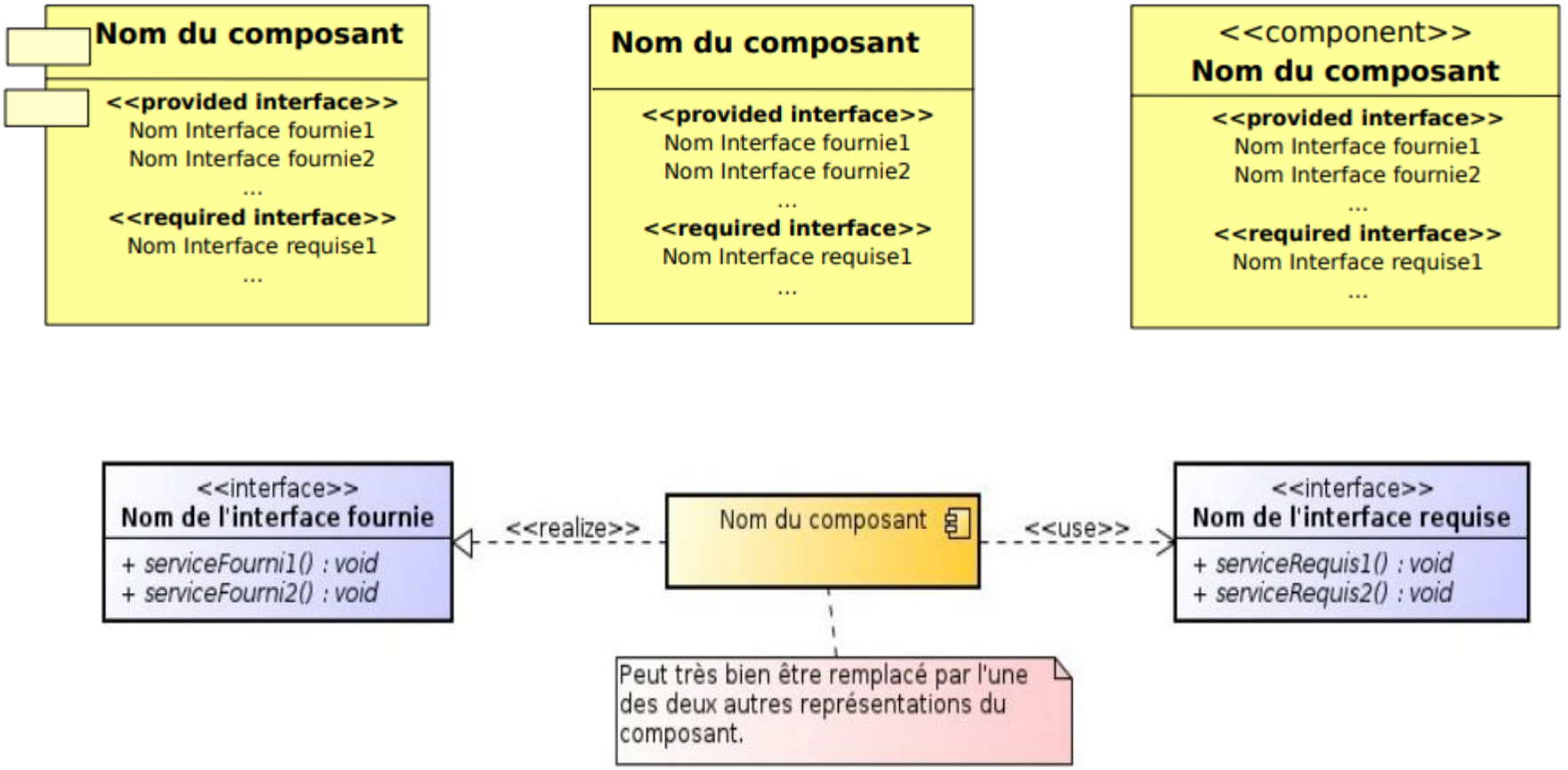
- **Dépendance** = relation entre deux composants
- Types de dépendances
  - ✓ Un composant peut dépendre d'un autre composant qui lui fournit un service ou une information
  - ✓ Un composant peut dépendre d'une classe qui implémente une partie de son comportement.
    - *Dépendance de réalisation*
  - ✓ Un composant peut dépendre d'un artefact (code source, fichier .jar, etc.) qui l'implante concrètement.
    - *Dépendance de manifestation*

Vue logique : éléments architecturaux

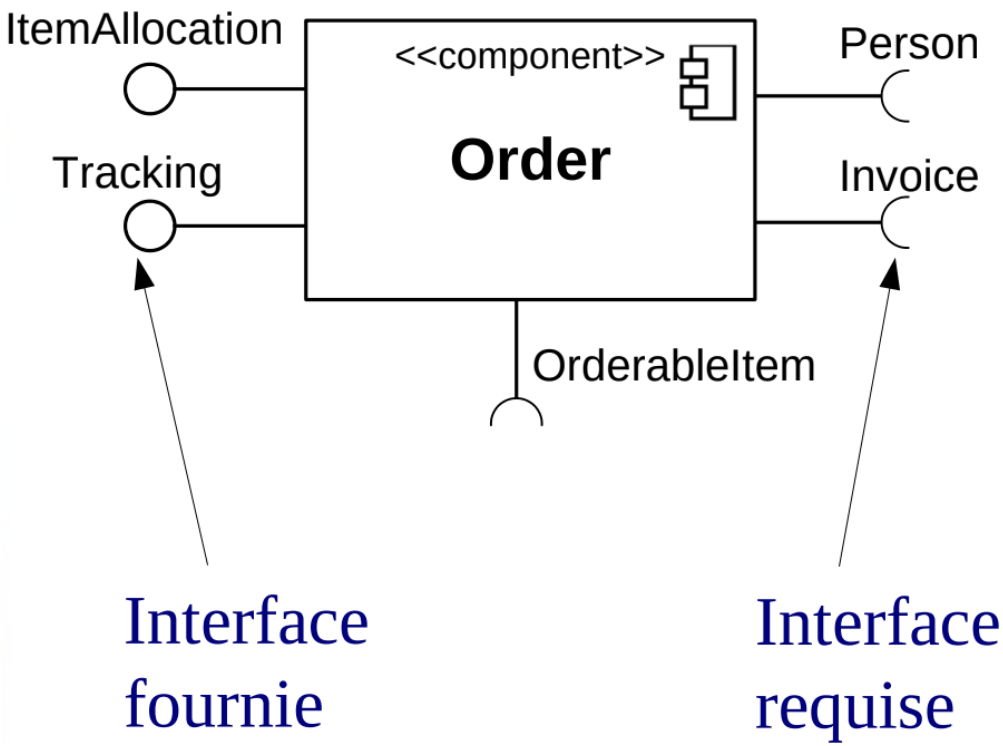
Connecteur

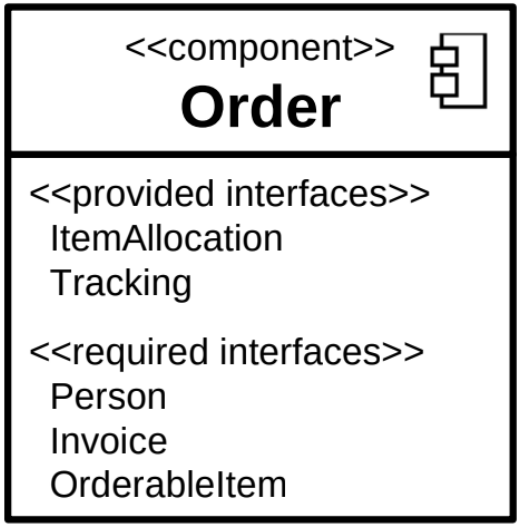
- **Définition** : élément architectural qui définit le type d'interactions entre les composants et les règles gouvernant ces interactions
- Un connecteur **relie les ports de deux ou plusieurs** composants
- Les attributs du connecteur décrivent ses propriétés comportementales
  - ✓ **Exemple** : *sa capacité, le temps de latence, le type d'interaction (binaire/naire, asymétrique/symétrique, détails du protocole), etc*.

Interface de composant



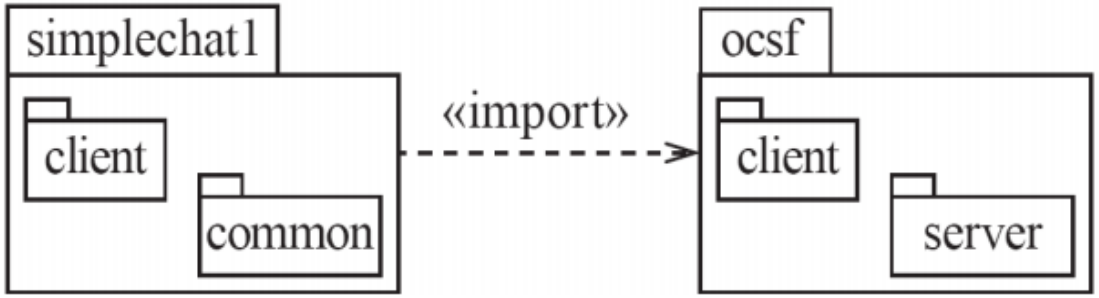
Exemple


ItemAllocation
Tracking
<<component>>
Order
Person
Invoice
OrderableItem
Interface fournie
Interface requise

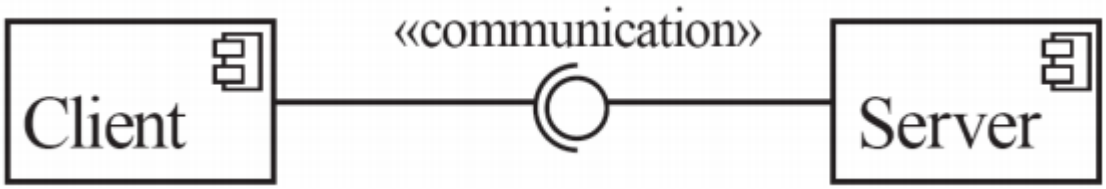

<<component>>
Order
<<provided interfaces>>
ItemAllocation
Tracking
<<required interfaces>>
Person
Invoice
OrderableItem

Modéliser l'architecture avec UML

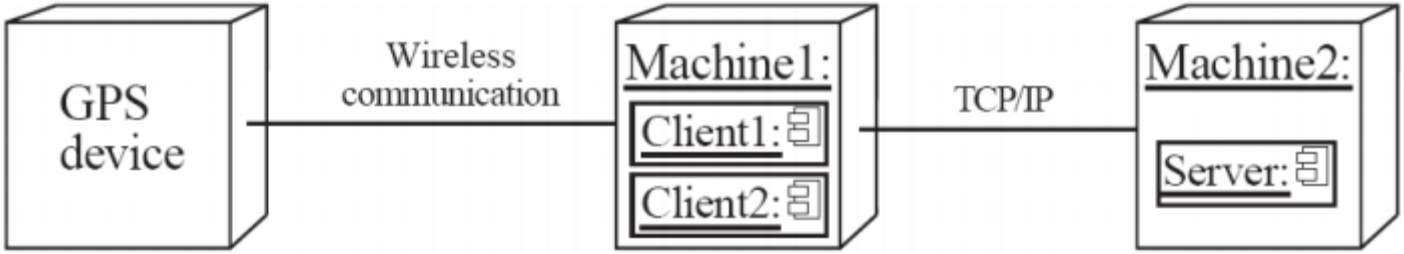
- **Diagramme de paquetage**



- **Diagramme de composants**



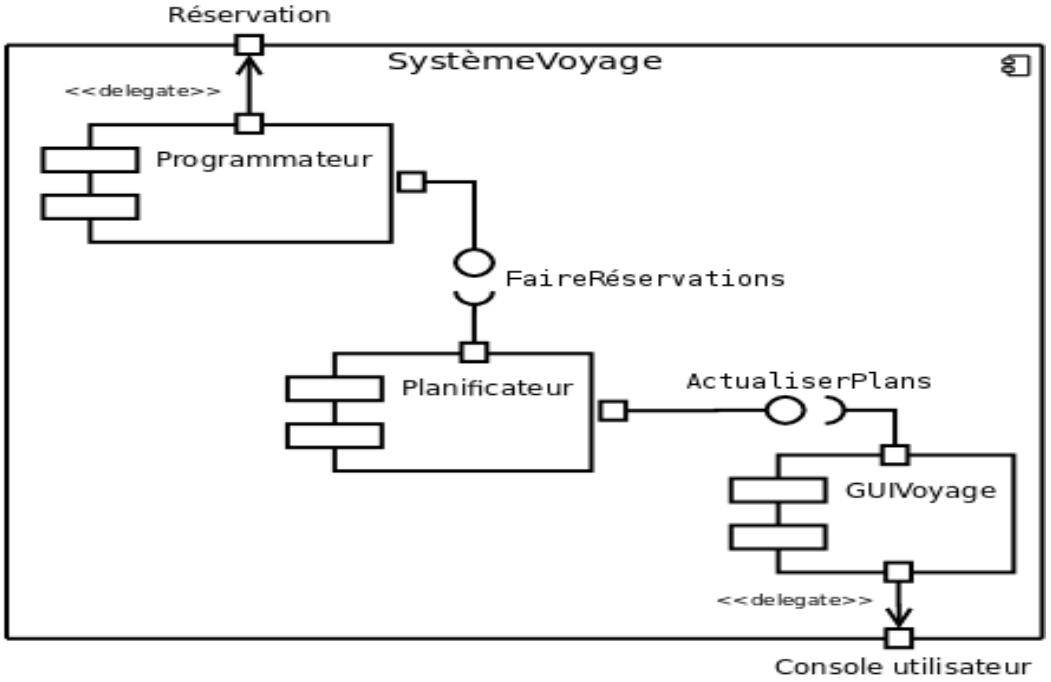
- **Diagramme de déploiement**



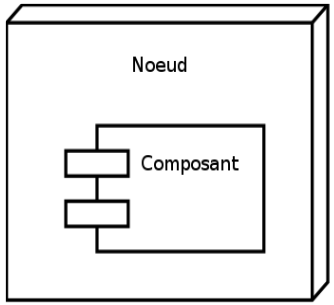
Modéliser l'architecture avec UML

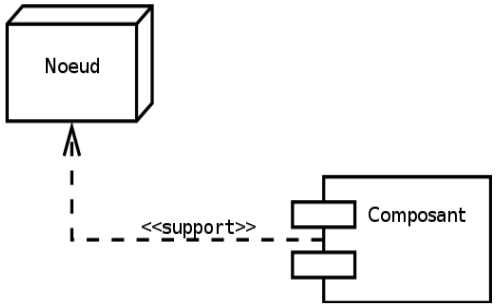
Deux diagrammes UML utilisés dans la modélisation des architectures :

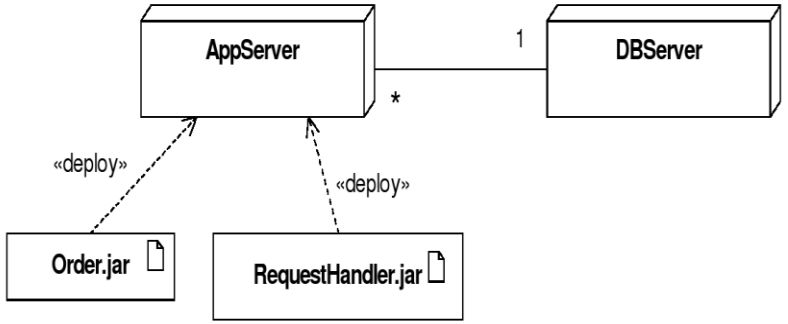
- **Le diagramme de composants**



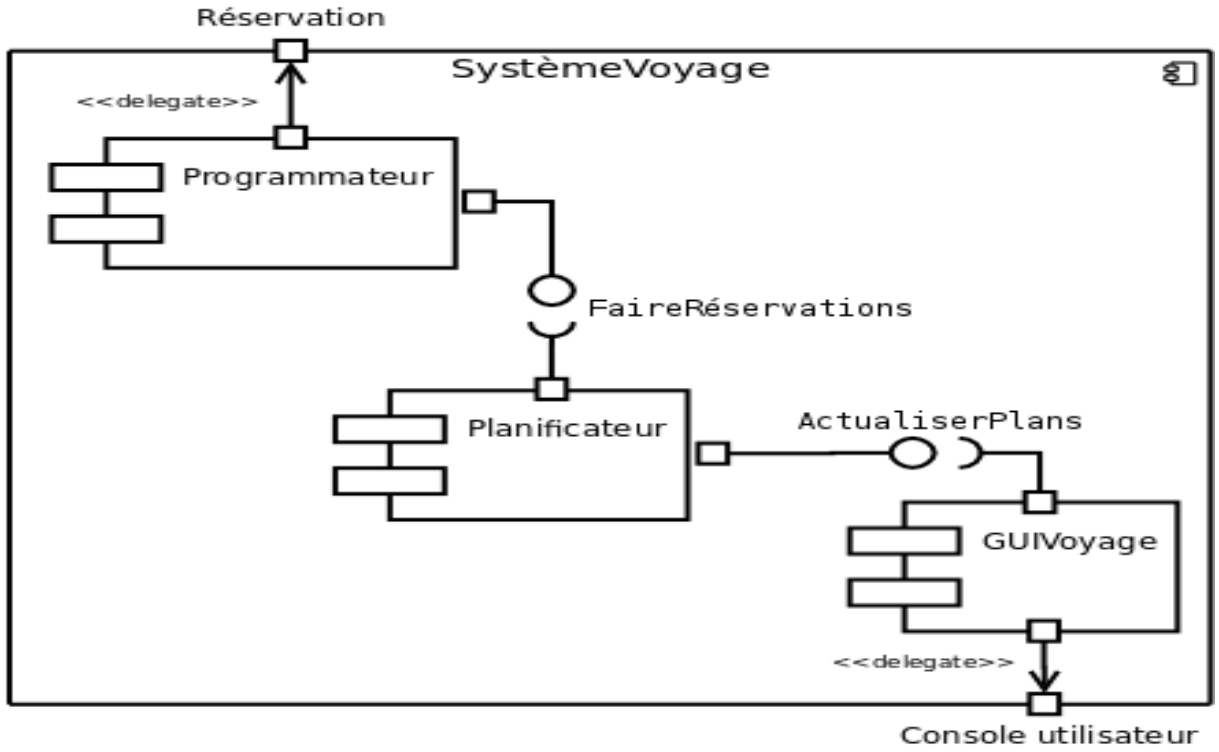
- **Le diagramme de déploiement**







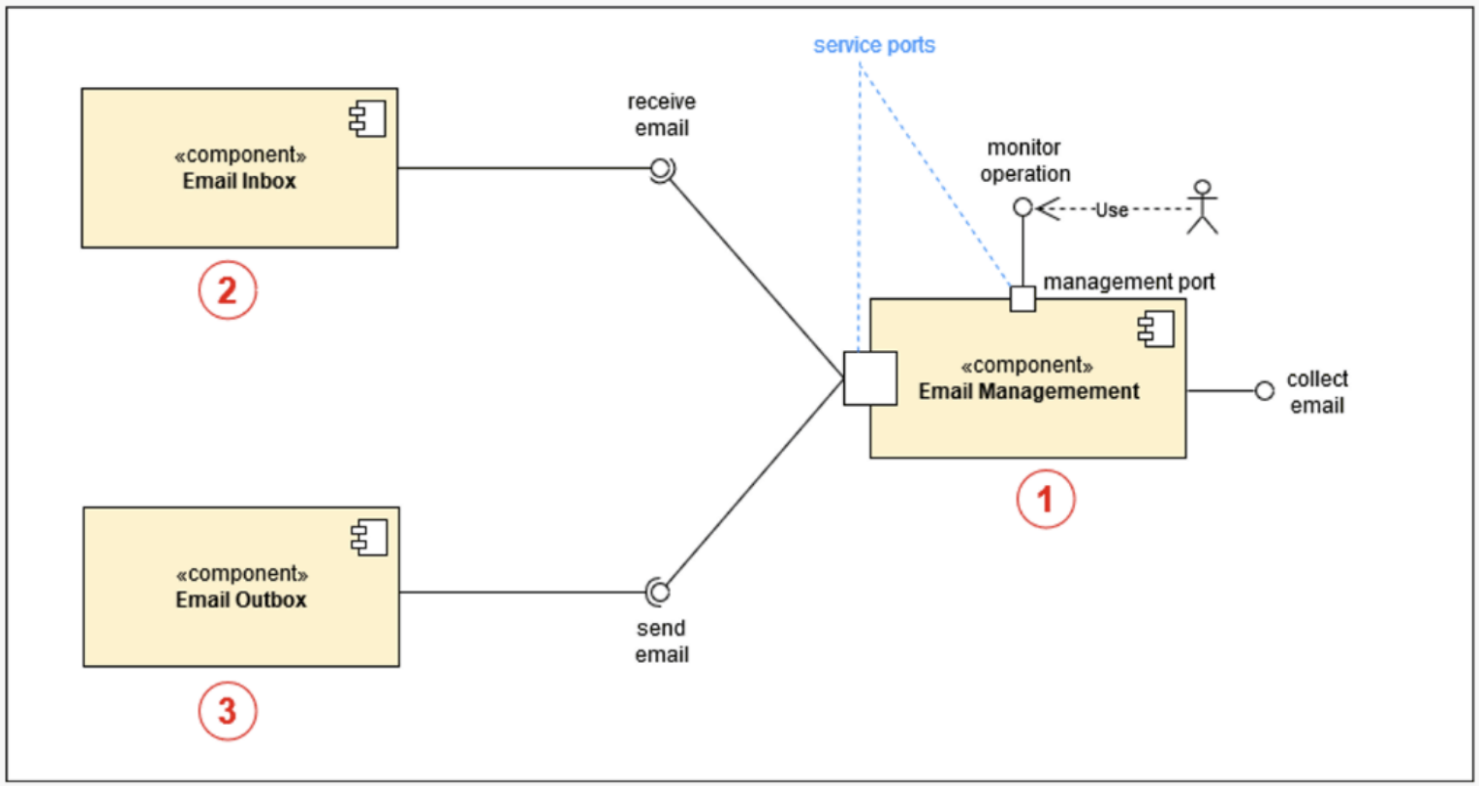
Exemple 1 : un diagramme de composants

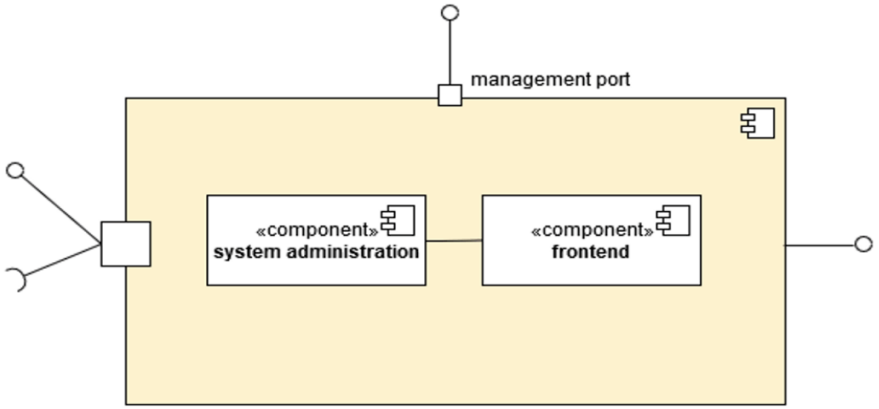


1. **Actualiser les plans** c'est : ajouter et supprimer des évènements; et produire le plan du voyage.
2. **Faire des réservations** c'est : réserver un avion, réserver un hôtel, annuler une réservation.

TAF : Décrire le composant Planificateur, et donner une schématisation avec les détails sur ses interfaces.

Exemple 2 : un diagramme de composants



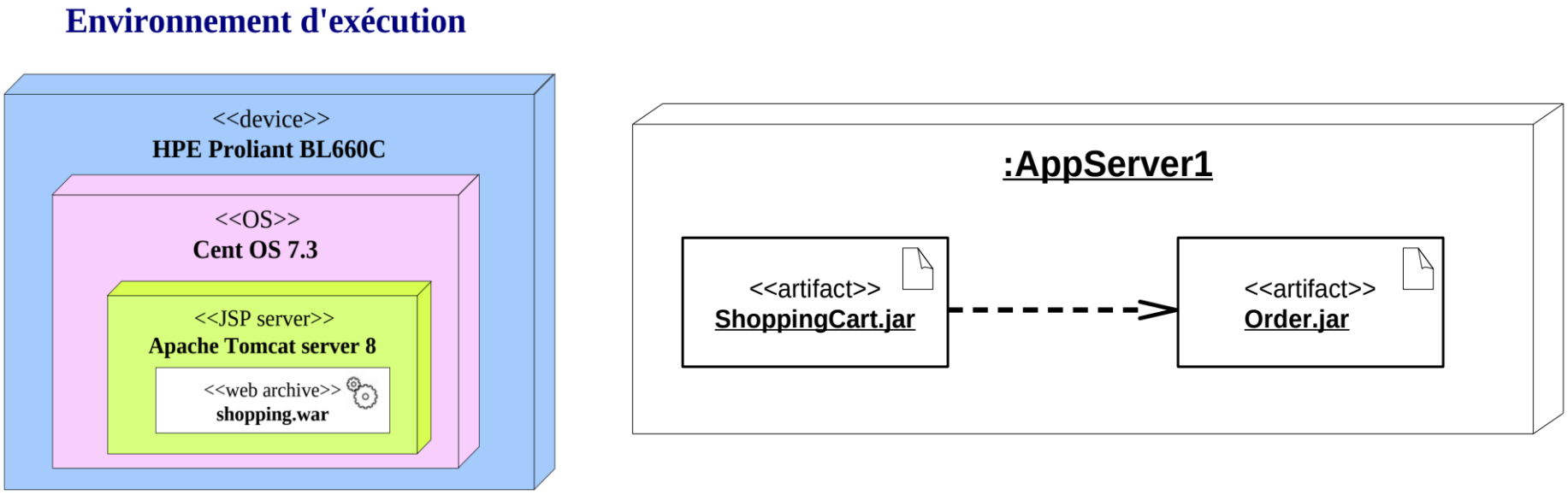


Exercice : un système d'achat en ligne

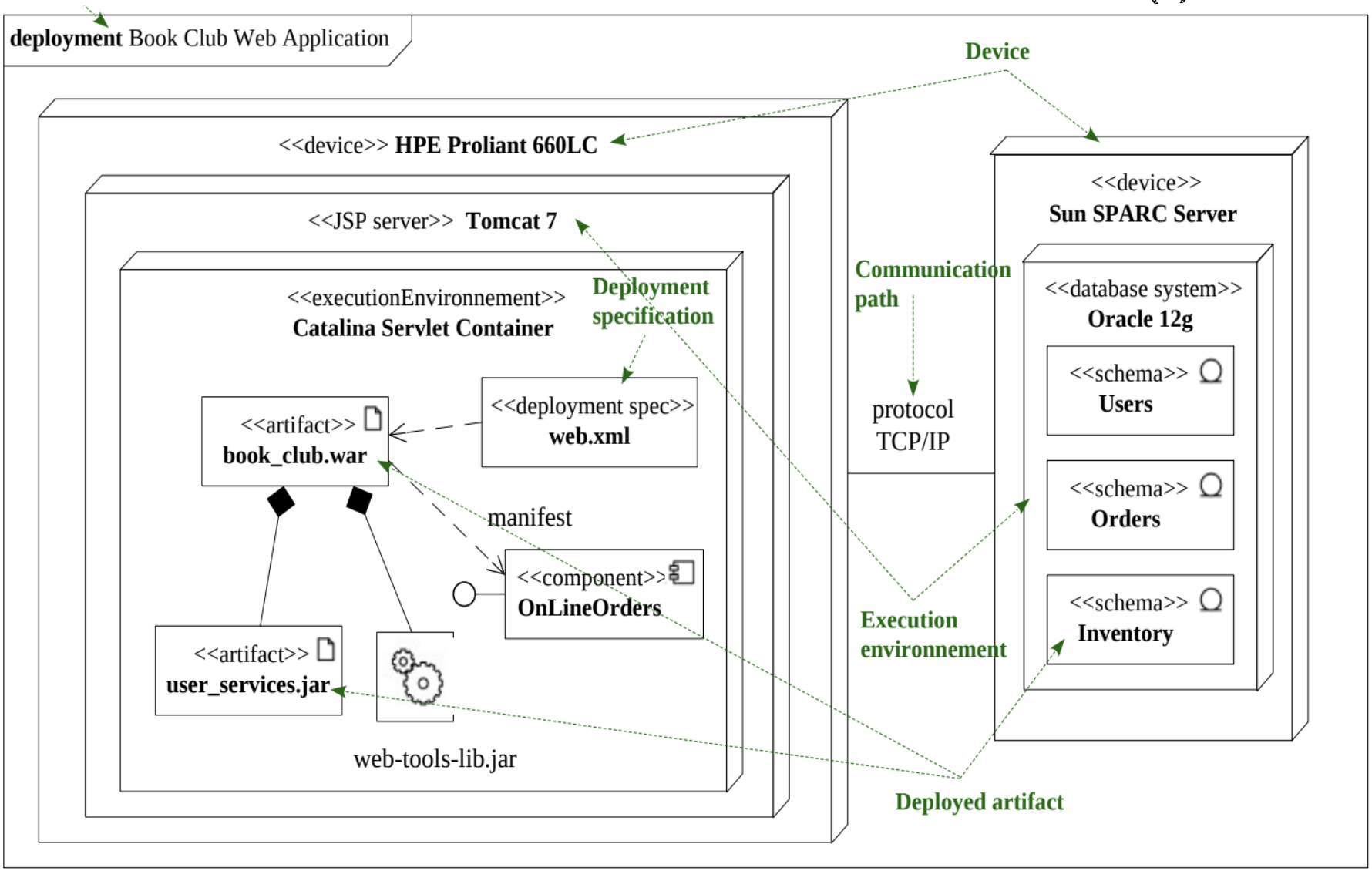
Gestion des comptes clients et gestion des commandes de produits en ligne.

Proposer une architecture logicielle pour le système

Exemple 3 : diagramme de déploiement



Exemple 3 : diagramme de déploiement



Source : support de S.HERAUVILLE, Univ Rouen

Choix d'une architecture

- Dépend des exigences fonctionnelles et non fonctionnelles du logiciel

- Choix favorisant la stabilité : l'ajout de nouveaux éléments sera facile et ne nécessitera en général que des ajustements mineurs à l'architecture

- Influencé par certains « modèles connus » de décomposition en composant (styles architecturaux) et de mode d'interactions (exemple : orienté objet)

Bibliographie et ressources

Ouvrages recommandés

- Software Architecture in Practice, 3e édition, Len Bass, Paul Clements et Rick Kazman, Addison-Wesley, 2012.
- Architecture logicielle : Concevoir des applications simples, sûres et adaptable, 2e edition, Jacques Printz, Dunod.

Notes de cours

- Architecture logicielle et conception avancée, Ecole polytechnique de Montréal, Yann-Gaël Guéhéneuc.
- Architecture logicielle, Université Joseph Fourier, Lydie du Bousquet
- Architectures logicielles : Livre Blace, Vincent Composieux
- [GLO-3001] : Architecture logicielle, cours de Luc Lamontagne