esprit
Se former autrement

UNIFIED MODELING LANGUAGE™

Chapitre IV: Conception Statique (suite et fin)

Diagramme de composants & de déploiement

Références:

- OMG Unified Modeling Language TM (OMG UML) Version 2.5

Cours: Architecture logicielle et conception avancée, Yann-Gaël Guéhéneuc, Architectures Partie 2/2

- http://www.uml-diagrams.org/

DIAGRAMME DE COMPOSANTS

Qu’est ce que le diagramme de composants?

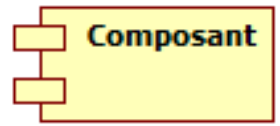
- Offre une vue de haut niveau de l’architecture du système
- Utilisé pour décrire le système d’un point de vue implémentation
- Permet de décrire les composants d’un système et les interactions entre ceux-ci
- Illustre comment grouper concrètement et physiquement les éléments (objets, interfaces, etc.) du système au sein de modules qu’on appelle composants

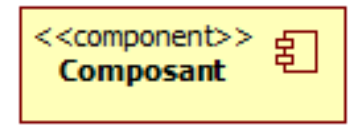
Qu’est ce qu’un composant? (1/3)

- Unité autonome faisant partie d’un système ou d’un sous-système qui encapsule un comportement (i.e., implémentation) et qui offre une ou plusieurs interfaces publiques
- Partie constituante d’un système qui peut être remplacée ou/et réutilisée
- Élément d’implémentation
  - un sous-système, un fichier exécutable, une classe d’implémentation (i.e., non abstraite, etc.) muni d’interface(s)
- Chaque composant est le représentant d’une ou plusieurs classes qui implémentent un service à l’intérieur du système

Qu’est ce qu’un composant? (2/3)

- Caractérisé par :
  - un nom
  - une spécification externe sous forme
    - soit d’une ou plusieurs interfaces requises
      - interfaces nécessaires au bon fonctionnement du composant
    - soit d’une ou plusieurs interfaces fournies
      - interfaces proposées par le composant aux autres composants
- Formalisme:





Qu’est ce qu’un composant? (3/3)

- Granularité:
  - Un composant peut représenter quelque chose d’aussi fin qu’un objet, comme il peut représenter un sous-système complexe
- Différence entre composant et instance de composant
  - Composant : vue de haut niveau d’un élément logiciel qui compose le système (~classe)
  - Instance de composant: le composant effectivement utilisé (~objet)

Composant : Représentation

- Deux types de représentation :
  - « Boîte noire »
    - Vue externe du composant qui présente ses interfaces fournies et requises sans entrer dans le détail de l'implémentation du composant
  - « Boîte blanche »
    - Vue interne du composant qui décrit son implémentation à l'aide de classificateurs (classes, autres composants) qui le composent

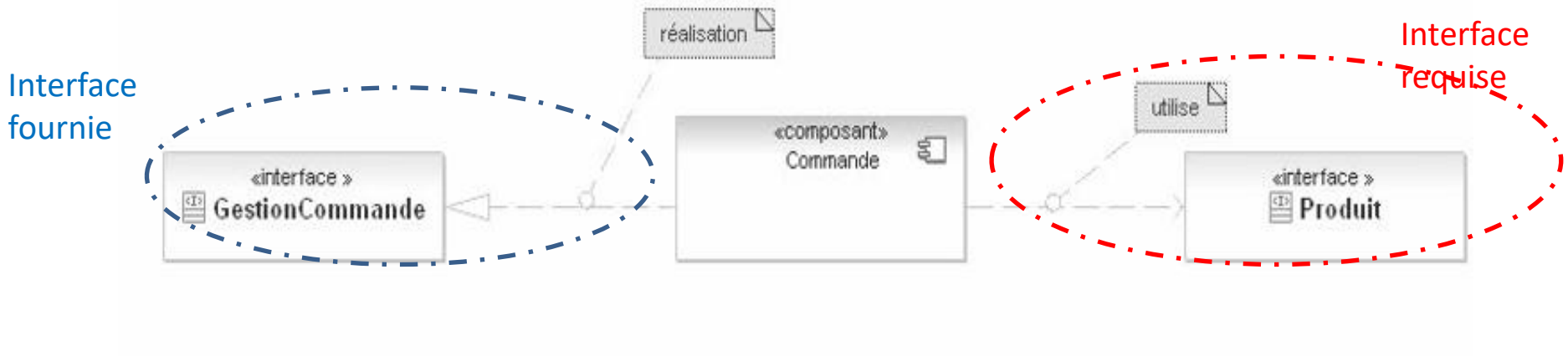
Composant : Boîte noire (1/3)

- « Boîte noire » : 3 représentions
  - Connecteurs d'assemblage :
    - un trait et un petit cercle pour une interface fournie
    - un trait et un demi-cercle pour une interface requise
      - Exemple :

Source : UML2 analyse et conception

Composant : Boîte noire (2/3)

- Connecteur d'interfaces
  - Utilisation des dépendances d'interfaces *use* et *realize* :
    - *use* pour une interface requise
    - *realize* pour une interface fournie
      - » Exemple :

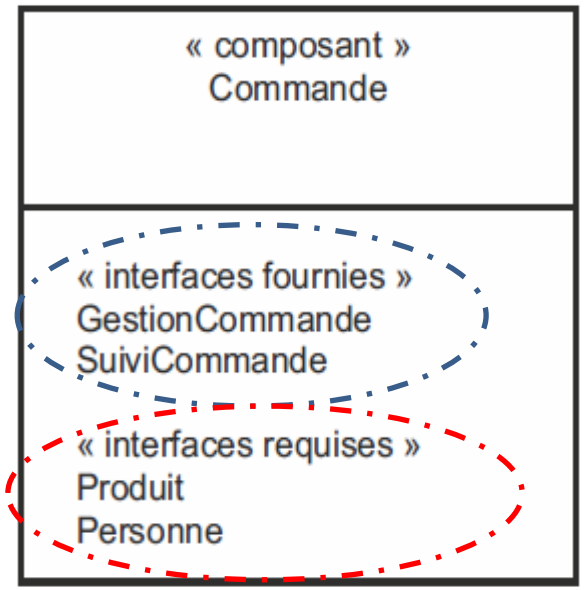


Source : UML2 analyse et conception

Composant : Boîte noire (3/3)

– Compartiment

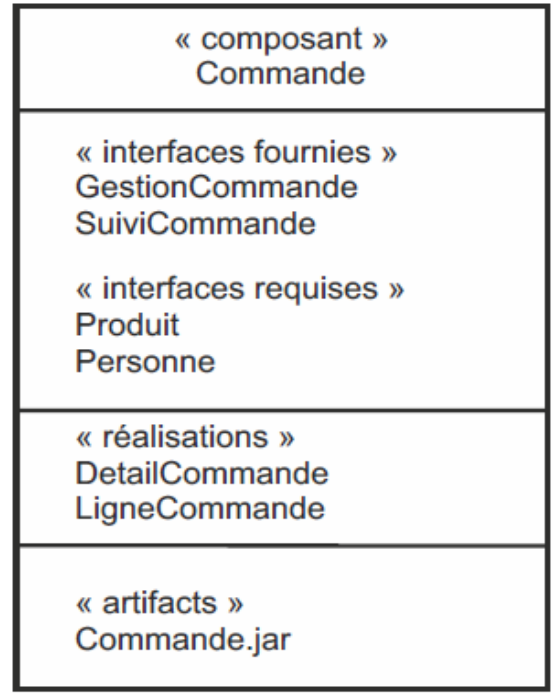
- Décrire sous forme textuelle les interfaces fournies et requises à l'intérieur d'un second compartiment
  - Exemple :



Source : UML2 analyse et conception

Composant : Boîte blanche (1/3)

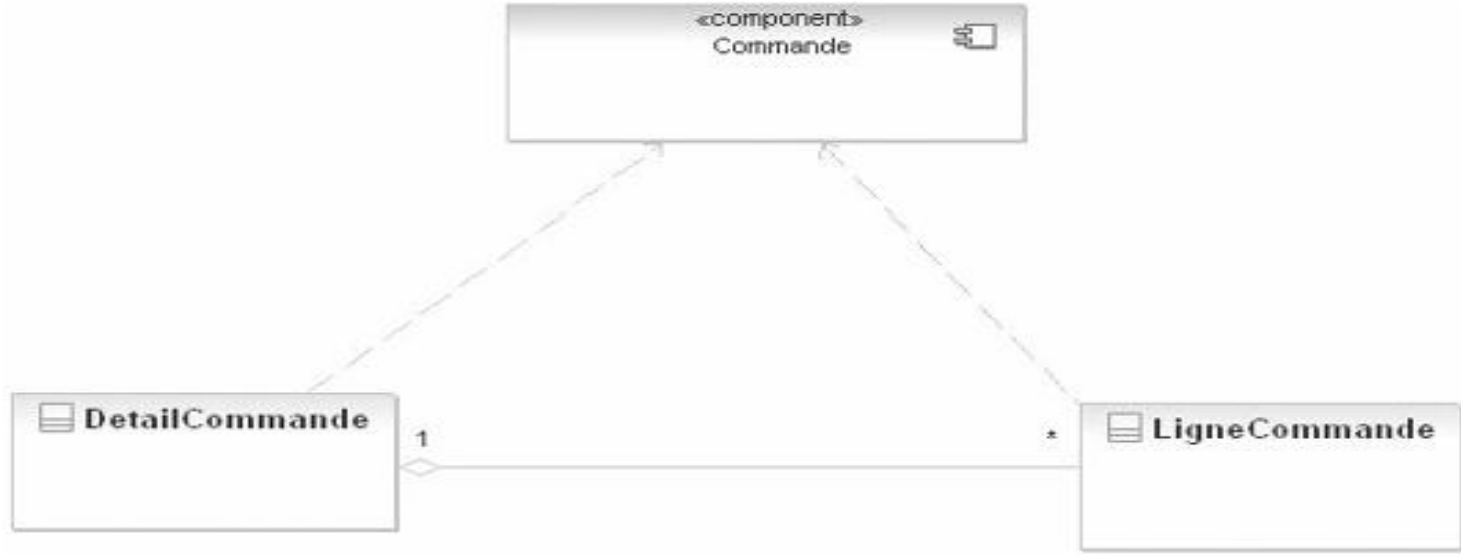
- «Boîte blanche » :
  - Compartiment
    - Trois compartiments:
      - Pour les interfaces fournies et requises
      - Pour les classificateurs (classes, autres composants)
      - Pour les artefacts (élément logiciel : jar, war, ear, dll) qui représentent physiquement le composant
        - » Exemple :



Source : UML2 analyse et conception

Composant : Boîte blanche (2/3)

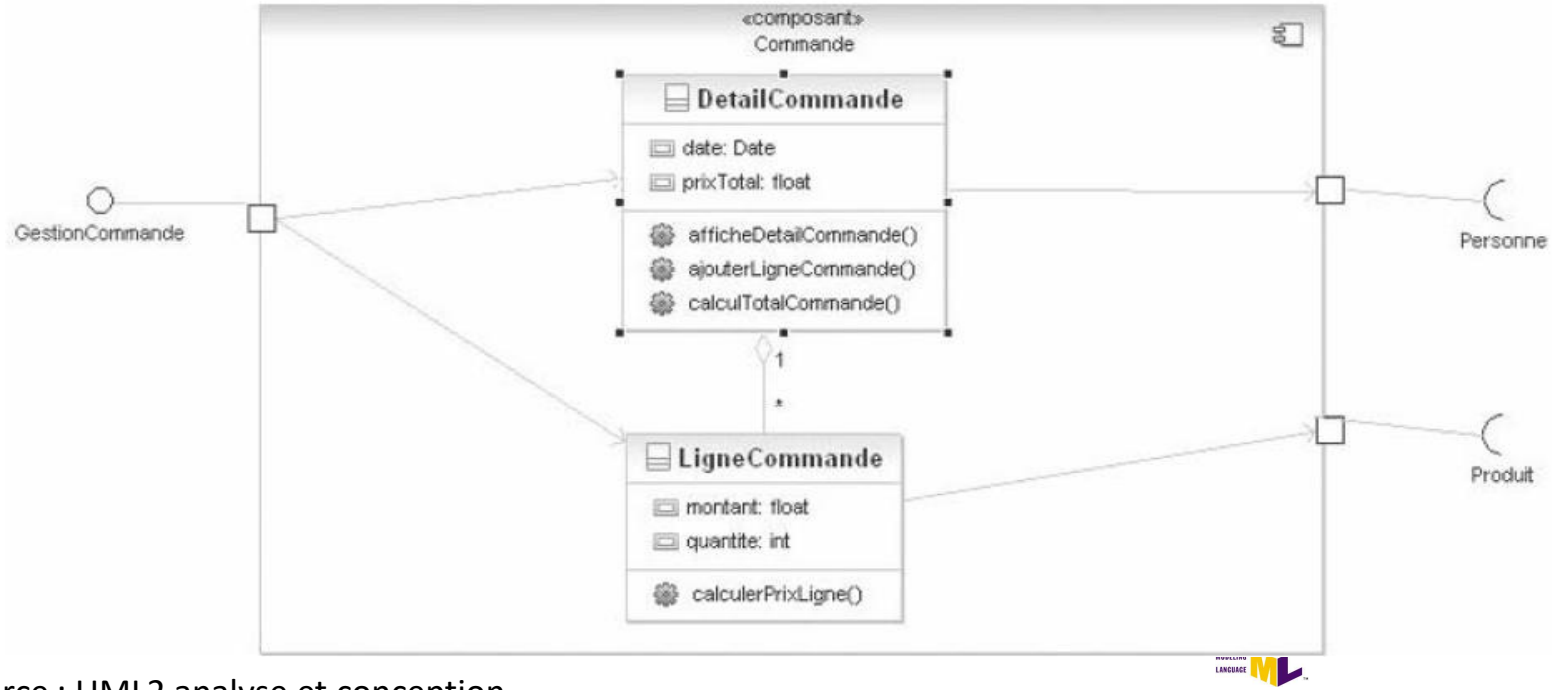
- Dépendance
  - Modéliser
    - Liens de dépendance entre le composant et les classificateurs qui le composent
    - Relations présentes entre les classificateurs (association, composition, agrégation)
      - » Exemple :



Source : UML2 analyse et conception

Composant : Boîte blanche (3/3)

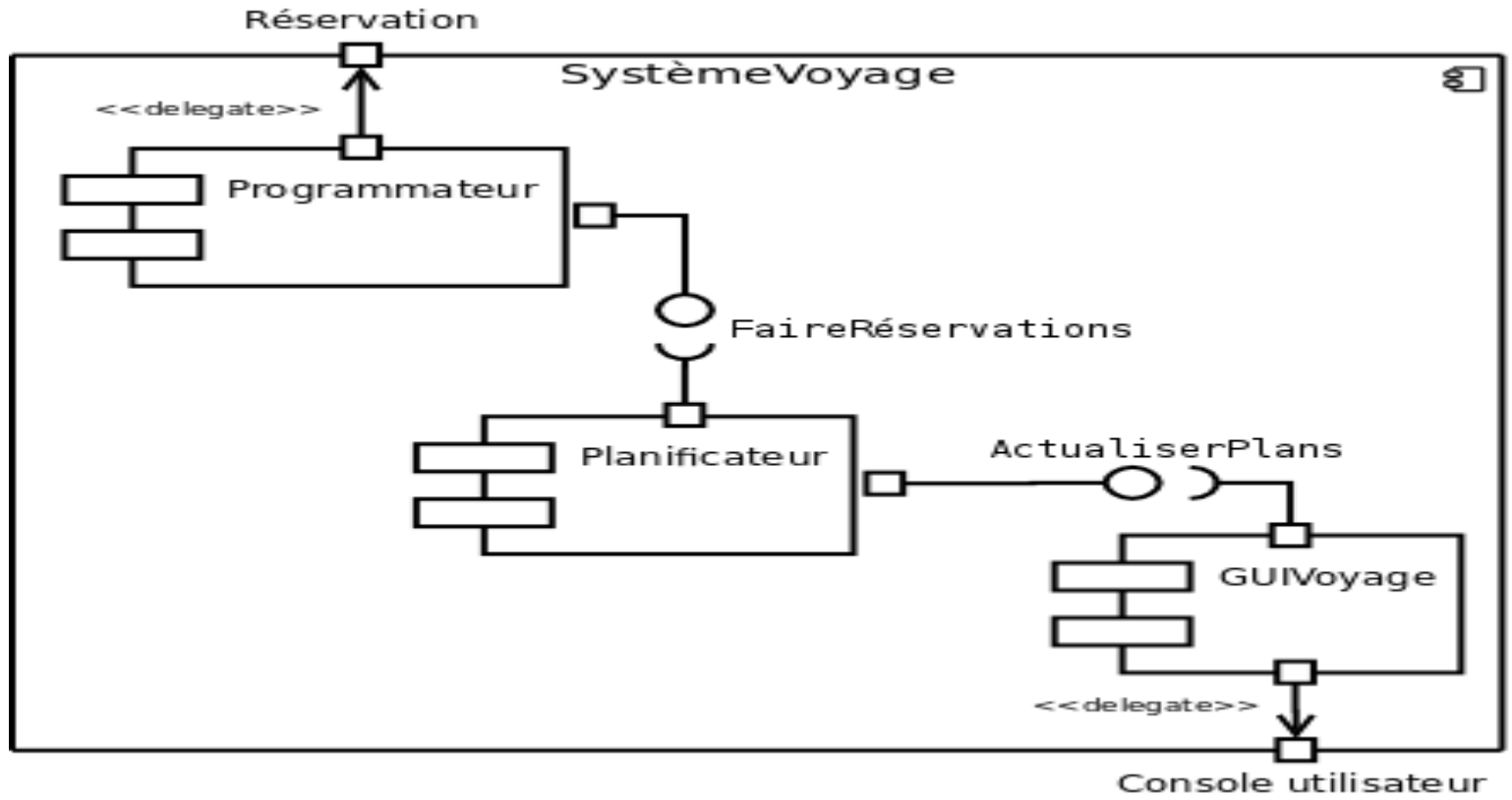
- Ports et connecteurs
  - Le port est un point de connexion entre le composant et une interface et est représenté par un petit carré sur le composant
  - Les connecteurs permettent de relier les ports aux classificateurs
    - Exemple :



Source : UML2 analyse et conception

Composant complexe

- Est constitué de sous-composants
  - Exemple:



Source : UML2 de l'apprentissage à la pratique

Relation avec les autres diagrammes

On peut utiliser un diagramme de composant avec d'autres diagrammes.

Autre diagramme	Permet d'évoquer et de communiquer ces aspects de votre conception
Diagramme de séquence UML	•Interactions entre les différents composants d'un système •Interactions entre les différentes parties d'un composant.
Diagramme de classes UML	•Les interfaces d'un composant: Le diagramme de classes vous permet de détailler les méthodes de l'interface. •Données envoyées dans des paramètres par le biais des interfaces des composants.
Diagrammes d'activités	•Traitement interne exécuté par un composant en réponse à des messages entrants.

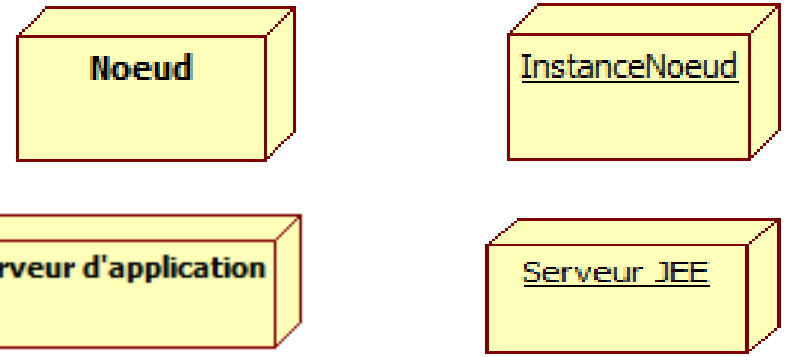
DIAGRAMME DE DÉPLOIEMENT

Qu’est ce qu’un diagramme de déploiement

- Un diagramme de déploiement propose une vision statique de la topologie du matériel sur lequel s’exécute le système
- Représente l’architecture physique du système:
  - Ensemble de nœuds
    - Correspondent aux supports physiques (serveurs, routeurs, environnements…)
  - Imbrication des nœuds
  - Connexions entre les nœuds
  - Répartition des artefacts sur chaque nœud
  - Un diagramme de déploiement ne montre pas les interactions entre les noeuds

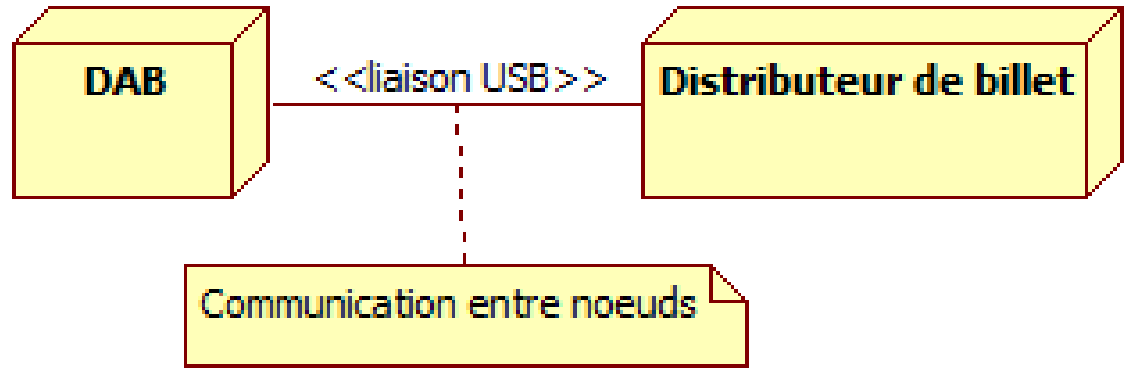
Nœud

- Ressource matérielle (ou environnement) de traitement sur laquelle des artefacts seront mis en œuvre pour l'exploitation du système
- Peut posséder des attributs
- Peut inclure d'autre nœuds
- Formalisme :
  - Nœud et instance d'un nœud
  - Exemple :



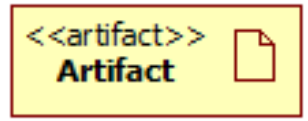
Nœud – Nœud : connexions

- Permettent l'échange d'informations
- Possèdent des stéréotypes prédéfinis pour les protocoles de communication (TCP/IP, Ethernet, X25,…)
- Peuvent se comporter comme une association (avoir un rôle, une direction,…)
- Exemple :

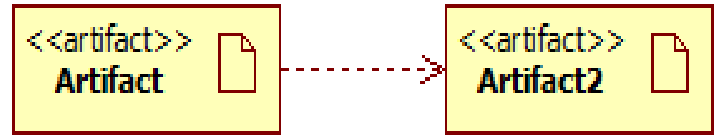


Artefact

- Spécification d'un élément physique utilisé ou produit par le processus de développement du logiciel ou par le déploiement du système
- Est un élément concret
- Exemple :
  - un fichier, un exécutable, une librairie,…
- Formalisme :

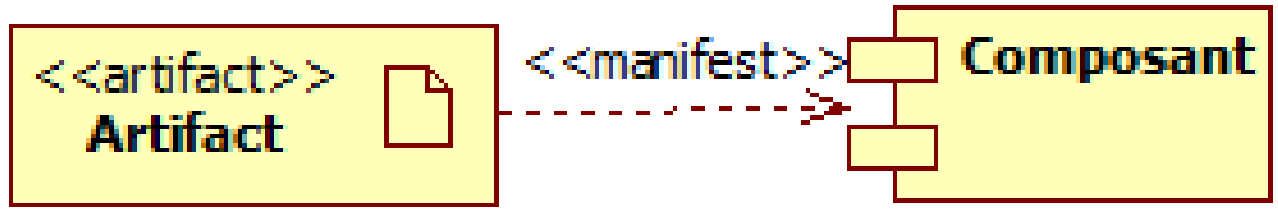


- Peut être relié à d'autres artefacts par des liens de dépendance
  - Exemple :



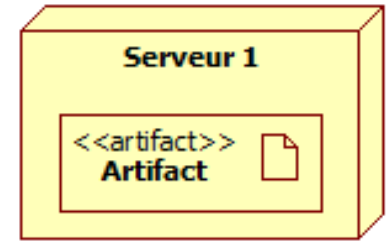
Artefact et composant

- Un artefact peut représenter (implémenter) des composants ou d'autres classificateurs
- Un composant peut être manifesté par plusieurs artefacts déployés dans des nœuds différents
- Formalisme :

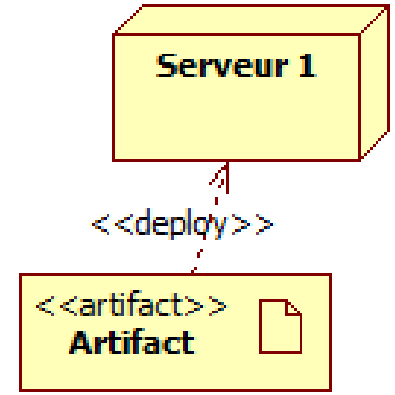


Nœud et artefact : déploiement

- Deux représentations :
  - Représentation inclusive
    - Exemple :

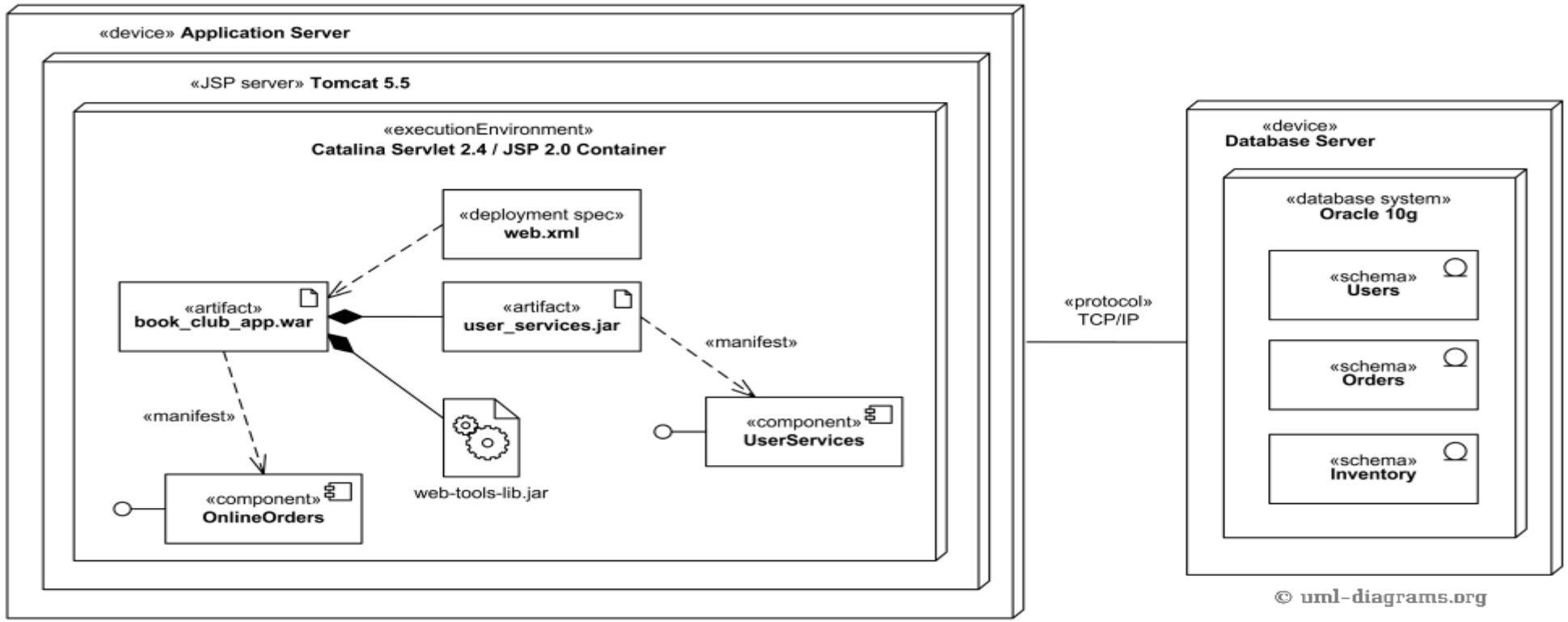


  - Représentation avec un lien de dépendance typé *«deploy»*
    - Exemple :



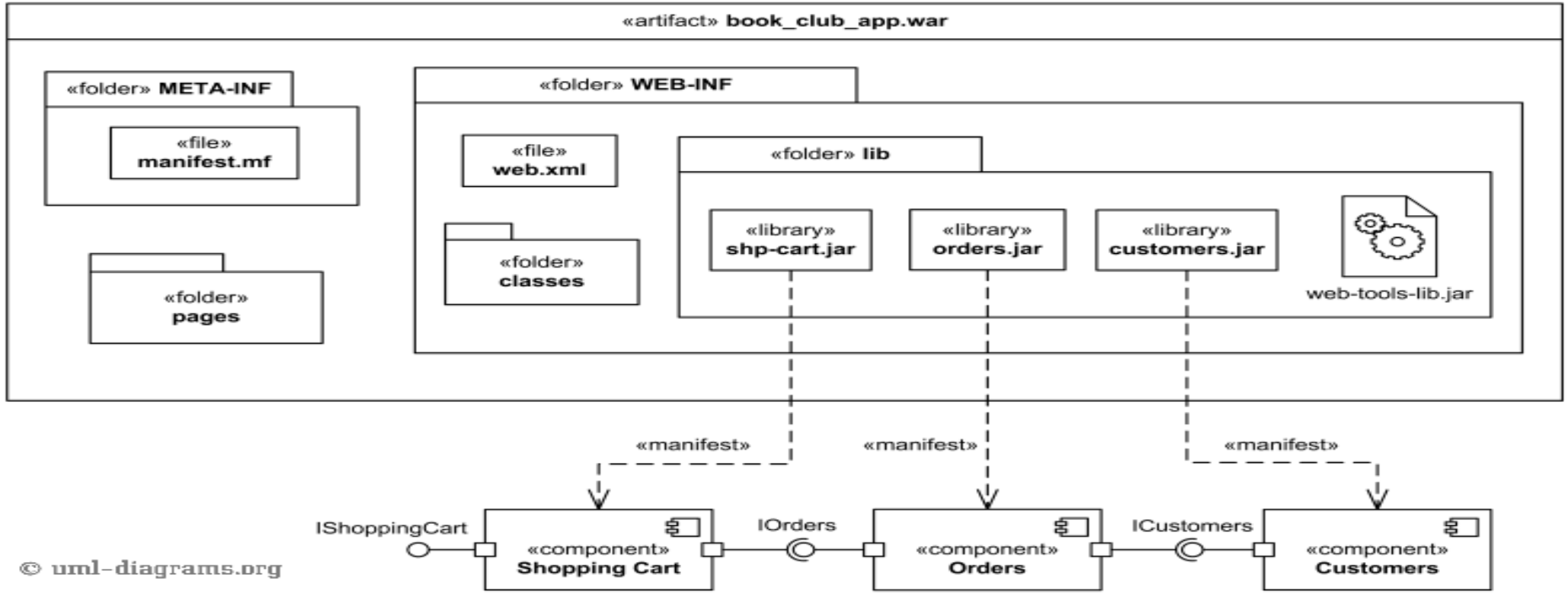
Utilisation du diagramme de déploiement (1/5): Exemple Application Web

- L'artefact **"book_club_app.war"** de l'application "Book club", est déployé sur "Catalina Servlet 2.4 / JSP 2.0 Container" qui est lui même une partie de "Apache Tomcat 5.5 **web server"**.
- L'artefact " **book_club_app.war"** manifeste le composant "**OnlineOrders**". Cet Artefact contient trois autres artefacts qui manifeste le composant "**UserServices**".
- Le serveur d'application **«device»** (computer server) est connecté au serveur de base de donnée «device» (un autre serveur) à travers le "protocol" TCP/IP



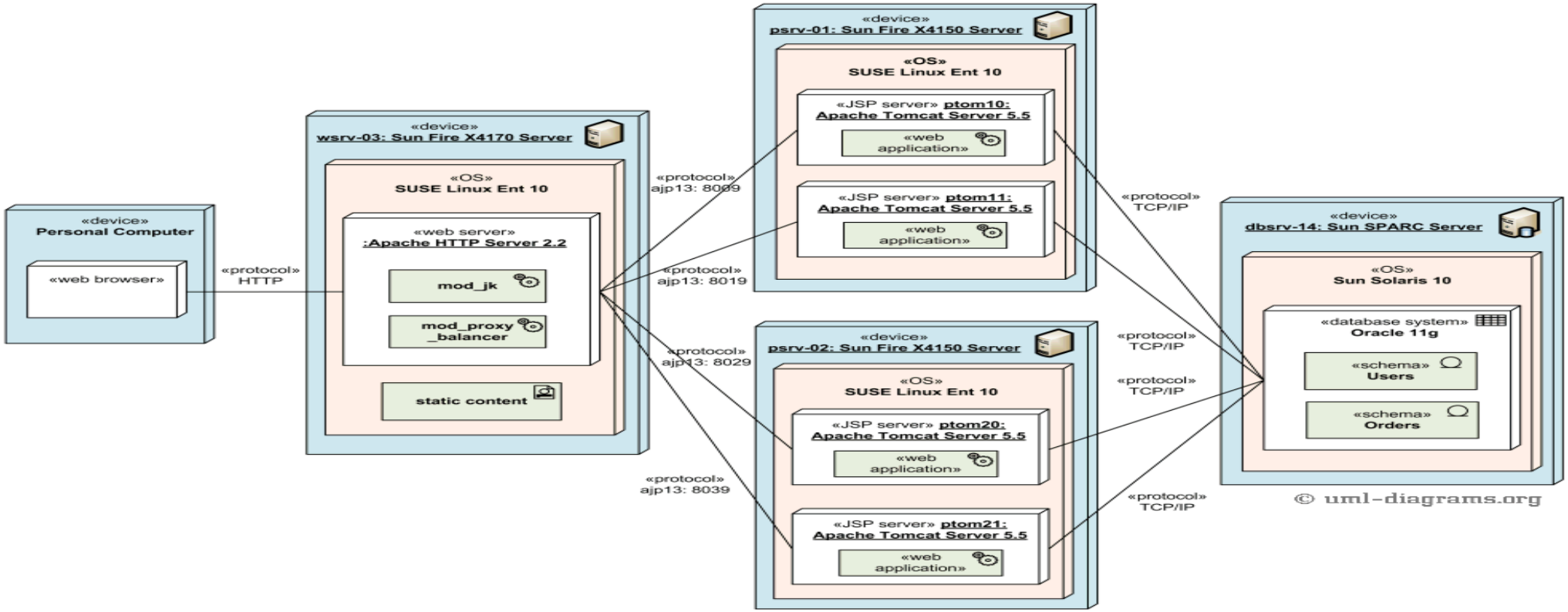
Utilisation du diagramme de déploiement (2/5):
Manifestation des composants

- Le formalisme du diagramme de déploiement peut être utilisé, aussi, afin d'illustrer la manifestation des composants par les artefacts sans pour autant réellement représenter une architecture, ou un déploiement.



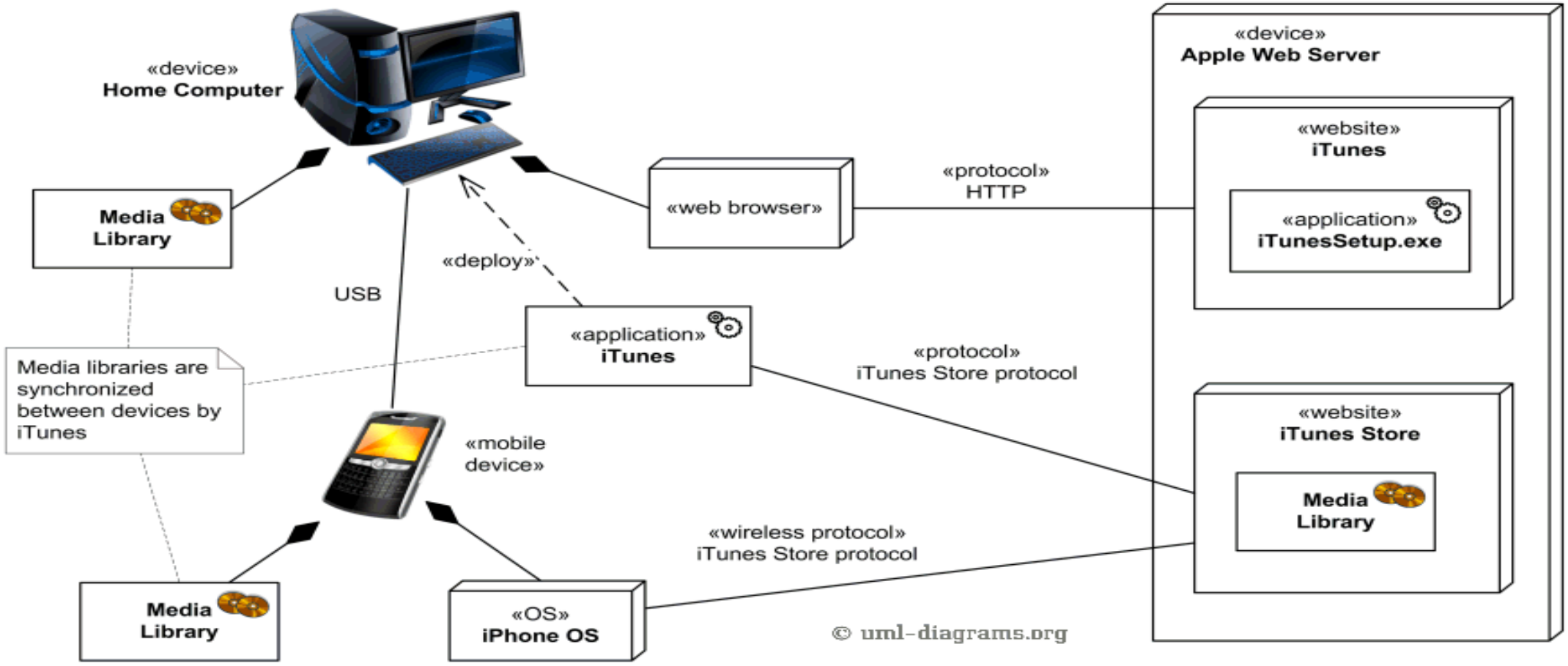
Utilisation du diagramme de déploiement (3/5):
Représentation de l'architecture physique d'une application Web J2EE avec équilibrage de charge

- Le formalisme du diagramme de déploiement est utilisé ici, afin de représenter l'architecture physique et les différentes instances des nœuds d'une application Web J2EE avec un équilibrage de charge.
- Cette illustration mets l'accent sur les nœuds, leurs instances et les liaisons qui existe entre elles plutôt que sur la répartition détaillée des composants et leurs artefacts sur ces nœuds.



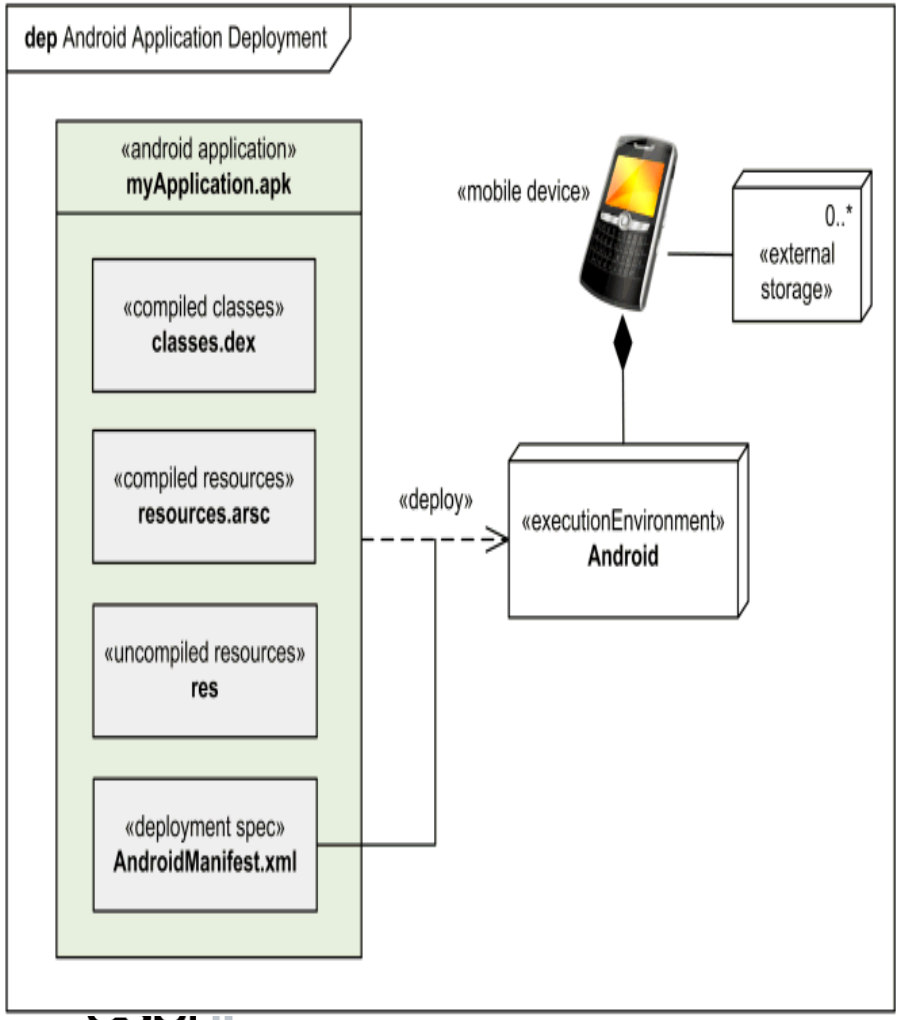
Utilisation du diagramme de déploiement (4/5):
Représentation de l'architecture physique d'une application Apple : ITunes

- Dans cette représentation nous voyons le couplage du vocabulaire UML à des représentations graphique illustratives. Le stéréotypage des nœuds et leurs liaisons reste néanmoins obligatoire.



Utilisation du diagramme de déploiement (5/5):
Représentation de l'architecture physique d'une application android

- Ce diagramme met l'accent sur l'environnement de déploiement, qui est représenté ici par un nœud



- En effet Applications Android sont composées d'un ou plusieurs composants de l'application (activités, services, fournisseurs de contenu, et les récepteurs de radiodiffusion). Chaque élément joue un rôle différent dans le comportement général de l'application, et chacun peut être activé individuellement (même par d'autres applications).
- Le manifeste (spécification de déploiement) déposer AndroidManifest.xml décrit les exigences d'application, tels que la version minimale d'Android nécessaire et toutes les configurations matérielles prises en charge, et il déclare également tous les composants de l'application.
- Avec l'API Android Niveau 8 ou plus tard, l'application pourrait être installé sur le stockage externe (par exemple, sur la carte SD). Ceci est une option qui pourrait être demandée pour l'application spécifique en utilisant un attribut manifeste. Par défaut, l'application est installée sur la mémoire interne de l'appareil mobile et ne peut pas être déplacé vers la mémoire externe.