Pour aller plus loin : Conception détaillée

Denis Conan

CSC4102

Télécom SudParis Janvier 2024

Table des matières

1 Conception détaillée, vues développement et physique

2

Les vues développement et physique sont présentées dans ce complément de cours pour des raisons de complétude du cycle de vie et des diagrammes UML. Cependant, les décisions prises lors de la construction des diagrammes de ces vues dépendent beaucoup des technologies utilisées. Ces sujets sont étudiés plus précisément dans les voies d'approfondissement ASR (« Architecte des Services informatiques en Réseaux ») et DSI (« Intégration et Déploiement de Systèmes d'Information »), par exemple.

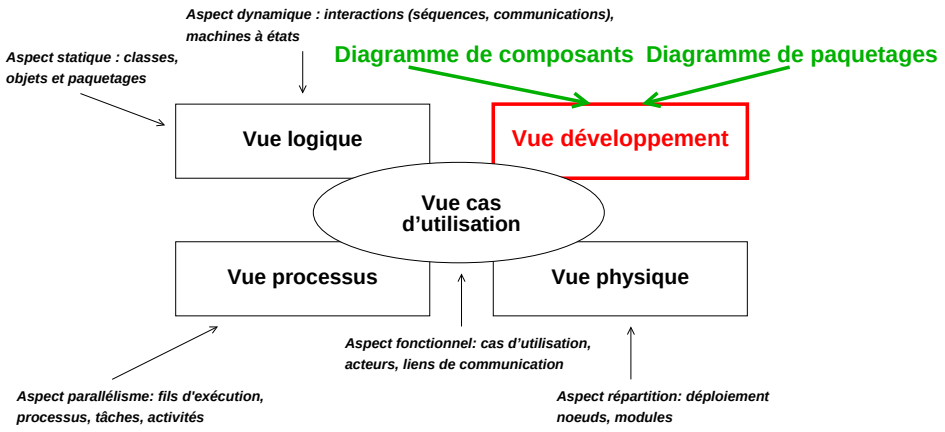
Lors de la conception détaillée, l'architecture logicielle du système comprend l'organisation des parties de logiciels spécifiées avec les termes utilisés par les développeurs : paquetage, composant, connecteur, assemblage, etc. Cette étape permet ensuite à chaque équipe de développement de connaître l'organisation du système et de positionner sa contribution dans l'ensemble. Ce modèle intéresse aussi les personnes qui déploient et administrent le système en production, car, implicitement, la vue développement définit les contributions de chaque équipe de développement, donc les responsabilités, et donne une indication pour la préparation du déploiement.

La seconde partie de cette section présente la vue physique, c'est-à-dire comment les composants de la vue développement sont projetés sur une architecture matérielle. La projection s'appelle le déploiement.

3

1.1 Diagrammes de la vue développement

- Composant = partie de logiciel réutilisable et remplaçable
- Paquetage = espace de nommage d'éléments de modélisation



Lors du développement d'un logiciel, lorsque le nombre de classes à développer devient très important, il est nécessaire d'organiser le développement de groupes de classes par des équipes de développeurs différentes. Les composants et les paquetages sont deux concepts de regroupement. Les composants sont des parties de logiciel réutilisables et remplaçables, selon l'approche COTS (en anglais, *Components Off The Shelf*, pour composants sur étagère). Les paquetages organisent des éléments, pas uniquement des classes, dans des espaces de nommages différents et permettent de gérer l'organisation des artefacts logiciels produits.

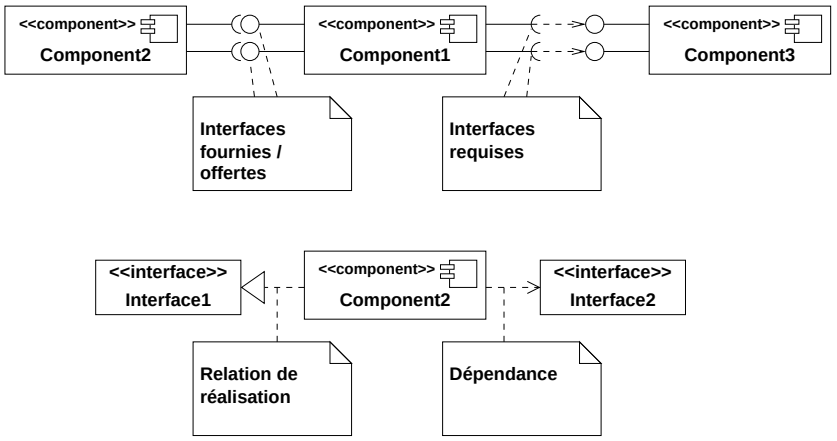
1.2 Diagramme de composants

4

Le diagramme de composants définit l'architecture logicielle du système dans un environnement de développement donné. Il permet de représenter le système et les sous-systèmes du modèle physique de l'architecture logicielle à réaliser. Plus précisément, un système ou un sous-système définit un espace de visibilité et regroupe des classes.

1.2.1 Composant, interfaces offertes et requises

5

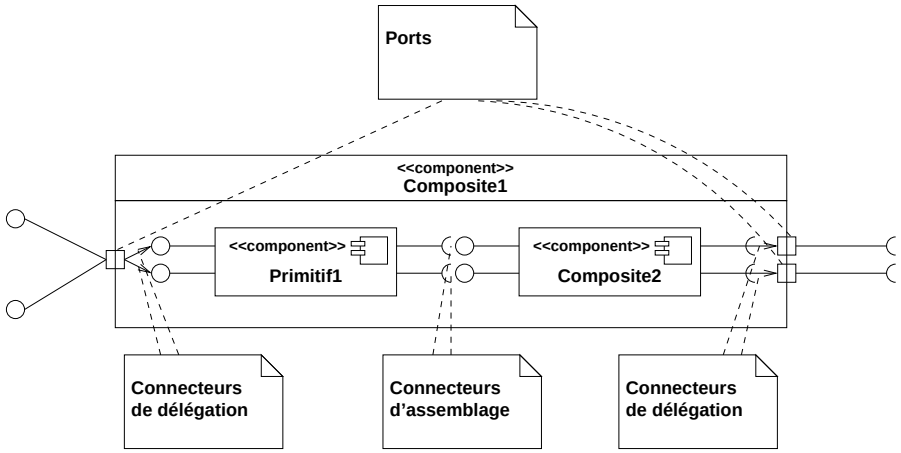


Dans le monde du bâtiment, l'architecture permet de visualiser, spécifier et documenter sur papier les caractéristiques de la future construction : positions des murs, des fenêtres, etc. Lors de la construction, on utilise des composants fenêtres, portes, murs. Ils rendent des services mais définissent leurs exigences : taille, espace, etc. Par analogie, dans un système informatique, le modèle logique d'une application permet de visualiser, spécifier et documenter la structure et le comportement des entités qui collaborent. La construction s'appuie sur des composants logiciels. Un composant logiciel définit les services logiciels qu'il rend et aussi les services dont il dépend pour pouvoir fonctionner. Ainsi, un composant spécifie les interfaces qu'il fournit (qu'il réalise) et les interfaces qu'il requiert (ses dépendances).

Un composant joue le même rôle qu'une classe : généralisation, association avec d'autres classes et composants, réalisation d'interfaces, etc. La principale différence entre une classe et un composant est qu'un composant regroupe des classes. Un composant est dessiné comme un rectangle avec le stéréotype `«component»`. Si le composant est gros alors UML suggère de remplacer le stéréotype `«component»` par `«subsystem»`. La diapositive montre les deux manières de spécifier les interfaces fournies (offertes) et requises d'un composant. La première notation graphique est plus concise et la seconde utilise uniquement des stéréotypes. Notez que le composant est relié à ses interfaces offertes par une association de réalisation alors qu'il est relié à ses interfaces requises par une relation de dépendance.

6

1.2.2 Composite, port, connecteurs de délégation et d'assemblage



La diapositive montre que les composants peuvent être de deux types. Les composants primitifs ne contiennent pas d'autres composants alors que les composites sont composés de composants primitifs et d'autres composites. Les composants à l'intérieur d'un composite sont connectés par des connecteurs dits d'assemblage. Les interfaces offertes et requises sont connectées à des ports du composite et les ports sont à leur tour connectés aux composants de l'intérieur par des connecteurs dits de délégation. La notion de port matérialise le passage du contrôle de l'extérieur vers l'intérieur et *vice versa*. Non montré dans cette diapositive, les composants primitifs sont en général constitués de classes.

Un modèle de composant hiérarchique, c'est-à-dire intégrant la notion de composants primitif et composite, permet d'étudier le système à différents niveaux de granularité : tout le système est un composite, que l'on peut ouvrir récursivement pour en connaître plus de détails, jusqu'à trouver les classes dans les composants primitifs.

Ce module étudie l'orientation objet. Les composants étant « plus que des objets » ne font pas partie du périmétre. Les VAP ASR et DSI proposent des enseignements sur l'orientation composant avec les modèles de composants EJB et SCA par exemple.

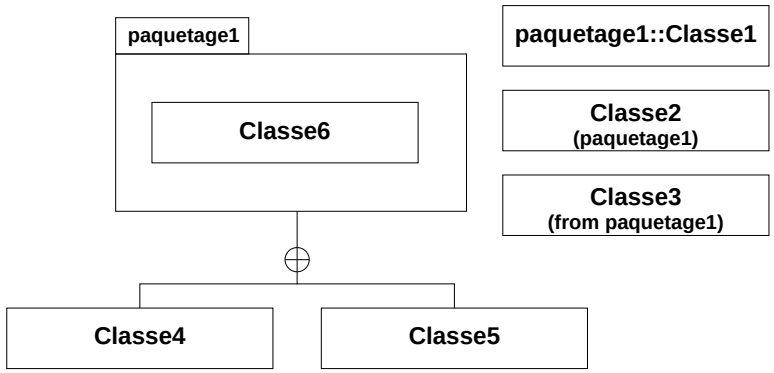
1.3 Diagramme de paquetages

7

Un système informatique peut aisément contenir des centaines de classes ou d'éléments de modélisation. Pour gérer cette complexité, UML fournit le concept de paquetage (en anglais, *package*) qui organise un espace de nommage (en anglais, *name space*).

8

1.3.1 Paquetage, espace de nommage



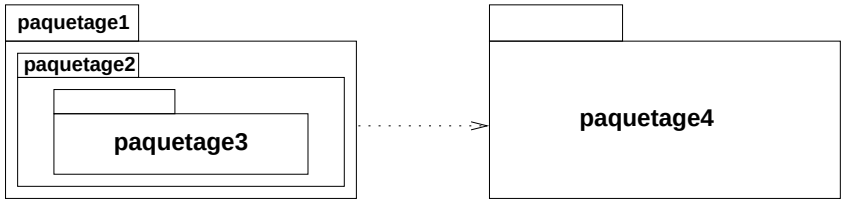
Un paquetage est représenté par un rectangle avec un cartouche rectangulaire en haut à gauche. Lorsque le diagramme montre des éléments de modélisation (dans la diapositive, une classe) dans le paquetage, le nom du paquetage est indiqué dans le cartouche. Le diagramme de la diapositive montre les différentes notations pour indiquer qu'une classe est contenue dans un paquetage :

- `Classe1` possède dans son nom le nom du paquetage, la hiérarchie des paquetages étant indiquée par le symbole « `::` » ;
- la notation pour `Classe2` possède en dessous de son nom le nom du paquetage entre parenthèses ;
- la notation pour `Classe3` utilise une notation proche de celle utilisée pour `Classe2` ;
- les classes `Classe4` et `Classe5` sont reliées au paquetage ;
- `Classe6` est directement visualisée dans le paquetage.

1.3.2 Relation entre paquetages

9

- Imbrication de paquetages = imbrication d'espaces de nommage
- Dépendance entre paquetages

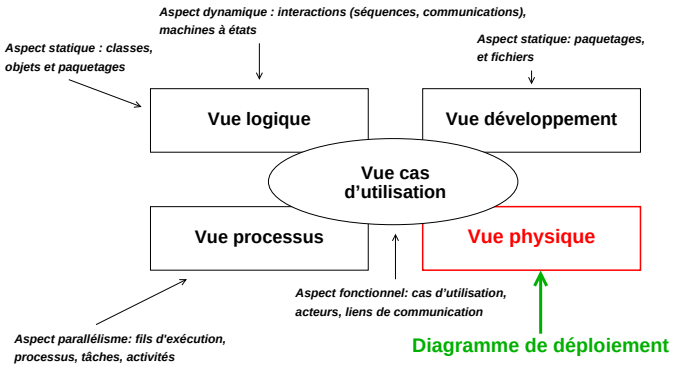


Les paquetages peuvent contenir d'autres paquetages pour construire une imbrication d'espaces de nommage. Si un élément d'un paquetage, par exemple `paquetage3` dans la diapositive, utilise un élément d'un autre paquetage, par exemple du `paquetage4`, alors le premier paquetage dépend du second. La dépendance est notée par une flèche pointillée allant du paquetage utilisateur vers le paquetage contenant l'élément utilisé.

10

1.4 Diagramme de la vue physique

- Placement des logiciels sur les matériels



La vue physique modélise les éléments physiques (le matériel) avec le placement des logiciels sur le matériel pour l'exécution du système. Il n'y a qu'un type de diagramme dans la vue physique, le diagramme de déploiement.

1.5 Diagramme de déploiement

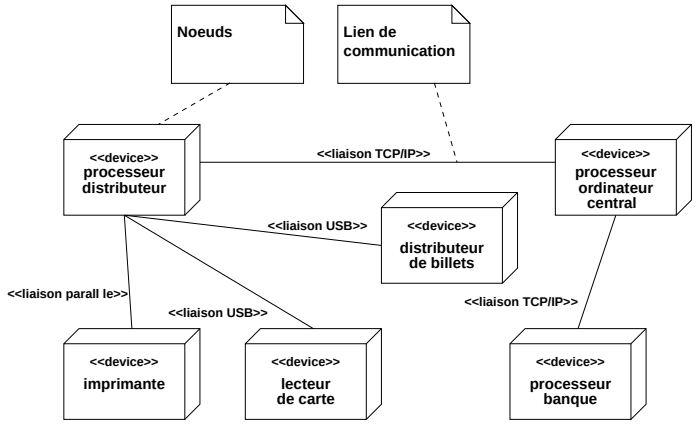
11

Dans ce module, nous ne nous intéressons qu'aux systèmes dans lesquels la partie logicielle est prépondérante. Autrement dit, nous ne discutons pas de la co-conception matérielle logicielle. Le diagramme de déploiement permet de spécifier la projection des artefacts logiciels exécutables sur le matériel, ainsi que les liens logiques et physiques via le réseau utilisé.

12

1.5.1 Nœud et lien de communication

- Ce modèle définit le diagramme de l'architecture matérielle du système
- Il représente les différentes machines et les logiciels
- Il montre les liens de communication entre ces différentes entités

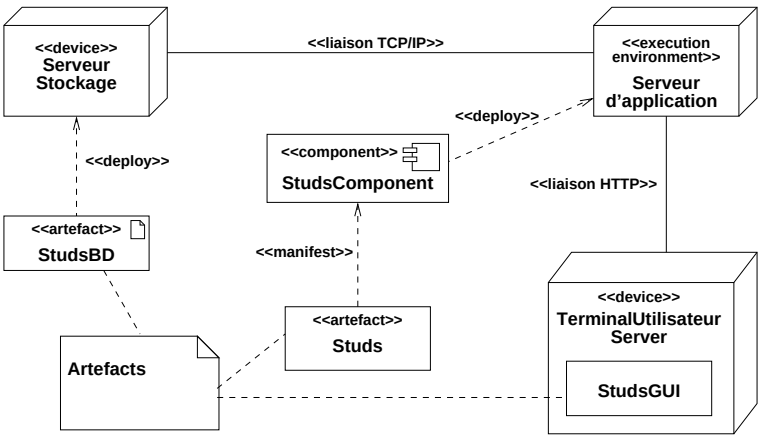


Un « nœud » est un matériel qui peut abriter des logiciels et des fichiers : hôte, disque, etc. UML étend la notion de nœud aux logiciels qui jouent le rôle d'environnement d'exécution tels que les systèmes d'exploitation, les serveurs Web et les serveurs d'application. Par opposition, des logiciels tels que les bibliothèques, les fichiers de propriétés et les fichiers exécutables ne peuvent pas être des nœuds, mais restent des « artefacts ».

Le diagramme de déploiement spécifie les liens de communication entre nœuds avec des associations stéréotypées par les noms des protocoles de communication utilisés.

13

1.5.2 Artefact et composant



La diapositive montre trois façons de représenter un artefact : avec le stéréotype `«artefact»` et un icone, avec seulement le stéréotype `«artefact»`, et sans stéréotype et sans icone.

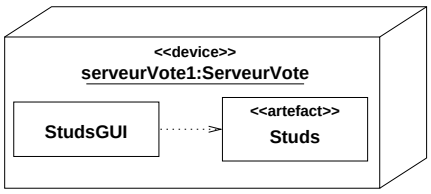
Le déploiement d'un artefact peut être montré de deux manières différentes : soit l'artefact est placé dans le nœud, soit l'artefact et le nœud sont reliés par une association stéréotypée `«deploy»`.

Enfin, le diagramme de déploiement peut modéliser par une association stéréotypée `«manifest»` le fait qu'un artefact est « empaqueté » dans un composant lui-même déployé sur un nœud.

14

1.5.3 Dépendance entre artefacts

- Dépendance entre artefacts
- Instance de nœud



Cette diapositive montre d'une part comment les dépendances entre artefacts sont spécifiées et d'autre part que les diagrammes de déploiement peuvent aussi contenir des instances de nœuds plutôt que des « types » de nœuds.