Chapitre 7 Diagrammes de composants (Component diagram) et Diagrammes de déploiement (Deployment diagram) et diagrammes de structure composite.

1. Introduction

Les diagrammes de composants et les diagrammes de déploiement sont des diagrammes de vues statiques en UML. Les premiers décrivent le système modélisé sous forme de composants réutilisables et mettent en évidence leurs relations de dépendance. Les seconds se rapprochent encore plus de la réalité physique, puisqu'ils identifient les éléments matériels (PC, Modem, Station de travail, Serveur, etc.), leur disposition physique (connexions) et la disposition des exécutables (représentés par des composants) sur ces éléments matériels. Un diagramme de structure composite est un diagramme UML qui joue le même rôle qu'un diagramme de classes, mais permet d'approfondir la description de la structure interne de plusieurs classes.

2. Diagrammes de composants

2.1. Pourquoi des composants ?

Parmi tous les facteurs qui concourent à la qualité d'un logiciel, nous avons introduit la notion de réutilisabilité comme étant l'aptitude d'un logiciel à être réutilisé, en tout ou en partie, dans de nouvelles applications. Or, la notion de classe, de par sa faible granularité et ses connexions figées (les associations avec les autres classes matérialisent des liens structurels), ne constitue pas une réponse adaptée à la problématique de la réutilisation.

Pour faire face à ce problème, les notions de patrons et de canevas d'applications ont percé dans les années 1990 pour ensuite laisser la place à un concept plus générique et fédérateur : celui de composant. La programmation par composants constitue une évolution technologique soutenue par de nombreuses plateformes (composants EJB, CORBA, .Net, WSDL…). Ce type de programmation met l'accent sur la réutilisation du composant et l'indépendance de son évolution vis-à-vis des applications qui l'utilisent.

2.2 Composant

- Un composant est une unité autonome représentée par un classeur structuré, stéréotypé « component », comportant une ou plusieurs interfaces requises ou offertes.
- Son comportement interne, généralement réalisé par un ensemble de classes, est totalement masqué : seules ses interfaces sont visibles. La seule contrainte pour pouvoir substituer un composant par un autre est de respecter les interfaces requises et offertes. *Les plu-gins, les bibliothèques sont des composants.*
- La notion de composant est proche de celle d'objet, dans le sens de la modularité et de réutilisation avec toute fois une granularité qui peut être différente. *Le composant est à l'architecture du logiciel ce que l'objet est à l'architecture du code.*
- Un composant peut évoluer indépendamment des applications ou des autres composants qui l'utilise à partir du moment ou les interfaces sont respectées.

- Un composant peut être vu de 2 manières :
  - Comme une boite noire dont nous ne connaissons pas le contenu et auquel nous accédons via les interfaces qui sont la seule partie visible.
  - Comme une boite blanche en spécifiant les objets qui constituent le composant et en indiquant leurs relations.

2.3 Représentation graphique d'un composant

Il existe plusieurs possibilités pour représenter un composant

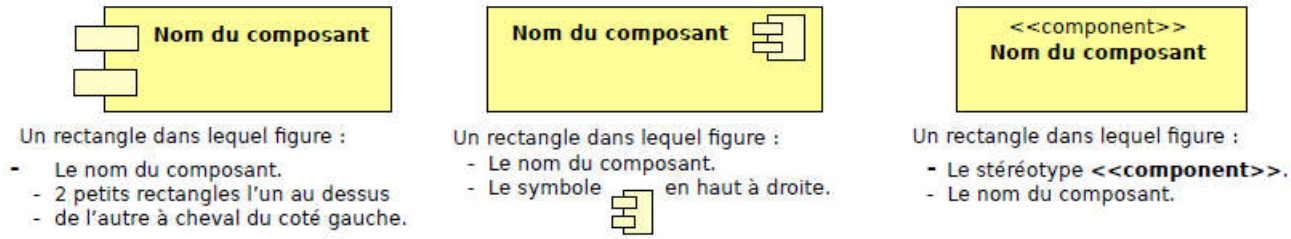


Figure 1 Différentes Représentation graphique d'un composant.

2.4 Les interfaces et représentation graphique de ses interfaces

Il existe deux types d'interface :

✔ *Les interfaces requise* : Ce sont des interfaces qui fournissent un service au composant et dont il a besoin pour fonctionner.

✔ *Les interfaces fournies* : Ce sont des interfaces par lesquels le composant fourni lui-même un service.

Il existe plusieurs possibilités pour représenter les interfaces :

1. Intégrées dans la représentation du composant :

Un compartiment dans lequel nous listons les interfaces requises et fournies (grâce aux Stereotypes <<required interface>> et <<provided interface>>).

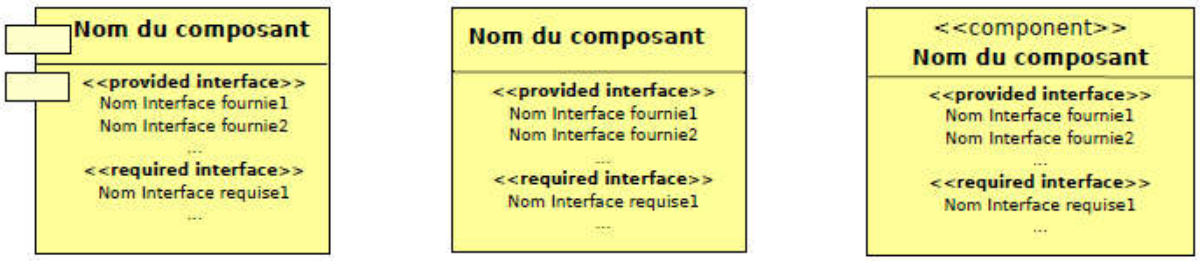


Figure 2 représentations de l'interface intègre dans le composant.

2. Dans un classeur séparé du composant dans lequel sont listés les différents services :

- Les interfaces requises sont reliées au composant par une flèche en pointillées sur laquelle figure le stéréotype <<use>>.
- Les interfaces fournies sont reliées au composant par une flèche en pointillées sur laquelle figure le stéréotype <<realize>> (le bout de la flèche est un triangle vide).

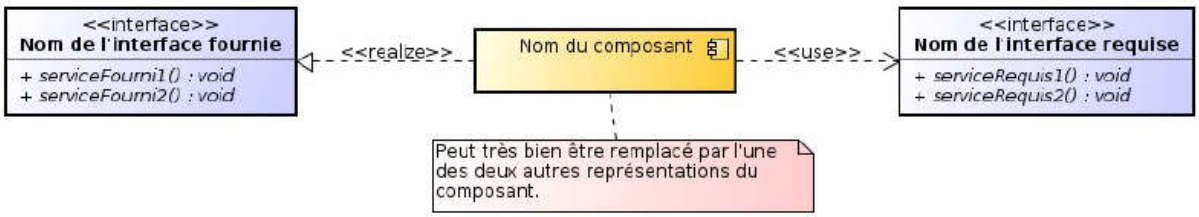


Figure 3 Représentation des interfaces dans des composants séparés.

3. Avec des connecteurs d'assemblage :

- Les interfaces requises (représentées par un demi-cercle)
- les interfaces fournies (représentéespar un cercle) sont raccordées au composant par un trait.

Figure 4 Représentation des interfaces avec des connecteurs d'assemblages

2.5. Les ports :

Un port est un point de connexion entre un classeur et son environnement.
Graphiquement, un port est représenté par un petit carré à cheval sur la bordure du contour du classeur. On peut faire figurer le nom du port à proximité de sa représentation. Il est la matérialisation de l'interface.

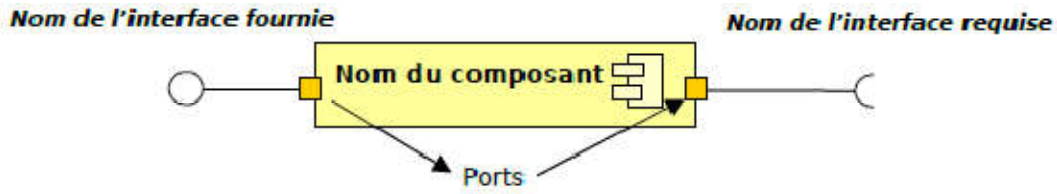


Figure 5 Représentation des ports de connexion.

3. Diagramme de déploiement

3.1 Objectif du diagramme de déploiement

Un diagramme de déploiement décrit la disposition physique des ressources matérielles qui composent le système et montre la répartition des composants sur ces matériels. Chaque ressource étant matérialisée par un nœud, le diagramme de déploiement précise La nature des connexions de communication entre les différentes ressources matérielles.

3.2 Les éléments du diagramme de déploiement:

3.2.1 Les noeuds :

Un nœud est une ressource matérielle du système. En général, cette ressource possède au minimum de la mémoire et parfois aussi des capacités de calcul (des ressources humaines ou des périphériques sont des ressources modélisées par des nœuds). Les ressources matérielles sont quelquefois

représentées avec le stéréotype ***<<device>>*** (généralement plutôt les périphériques mais également tout systeme informatique comme par exemple les ordinateurs de bureau).

Un nœud est représente par un parallélépipède rectangle dans lequel figure son nom.

Figure 6 Représentation d'un nœud.

Un nœud possède des attributs (quantité de mémoire, vitesse de processeur, marque, type…) que nous pouvons spécifier a l'intérieur du parallélépipède.

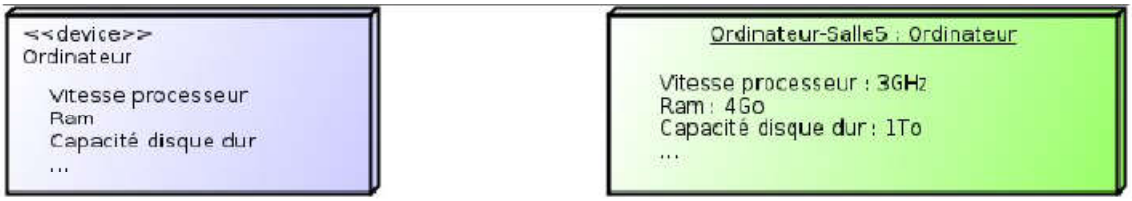


Figure 7 Représentation d'un nœud et ses attributs.

Pour montrer qu'un composant est affecté sur un noeud, il faut :*

- Soit placer le composant dans le noeud.
- Soit en le reliant a l'aide d'une relation de dépendance (flèche en pointillées) stéréotypée ***«support»*** orientée du composant vers le noeud .

Figure 8 Représentation d'un composant affecté sur un noeud.

3.2.2 Les chemins de communications

Le chemin de communication est donc un lien qui permet de modéliser de façon simpliste la communication entre 2 noeuds (liaison Ethernet, USB, série…).

Il est possible de faire figurer sur ce lien :

- les cardinalités ;
- des contraintes entre accolades (pour indiquer par exemple qu'un accès est sécurise) ;
- le type de réseau et/ou son débit en l'indiquant comme un stéréotype…

Figure 9 Représentation des liaisons entre nœud.

3.2.3. Les artefacts :

Un artefact correspond à un élément concret existant dans le monde réel (document, exécutable, fichier, tables de bases de données, script…). Il se représente comme un classeur par un rectangle contenant le mot-clef « artifact » suivi du nom de l'artefact.

On appelle manifestation la relation entre un élément de modèle et l'artefact qui l'implémente. Graphiquement, une manifestation se représente par une relation de dépendance stéréotypée « manifest ».

Une instance d'un artefact se déploie sur une instance de nœud. Graphiquement, on utilise une relation de dépendance (flèche en trait pointillé) stéréotypée « deploy » pointant vers le nœud en question. Ce sont les artefacts qui sont déployés sur les noeuds et nom pas les composants.

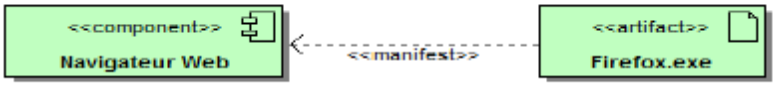


Figure 10 Représentation de la liaison entre un artefact un composant.

L'artefact est place soit :

- ✓ A l'intérieur du noeud dans lequel il est déployé.

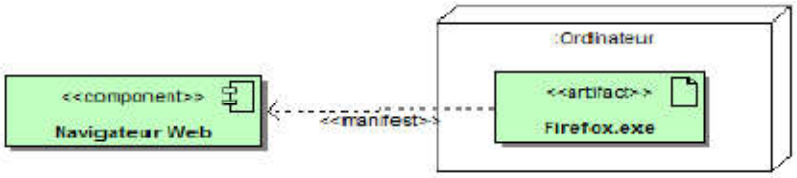


Figure11 Représentation d'un artefact à l'intérieur d'un nœud et de sa liaison avec un composant.

- ✓ A l'extérieur du noeud dans lequel il est déployé, mais relie a celui-ci par une relation de dépendance stereotype ***<<deploy>>***.

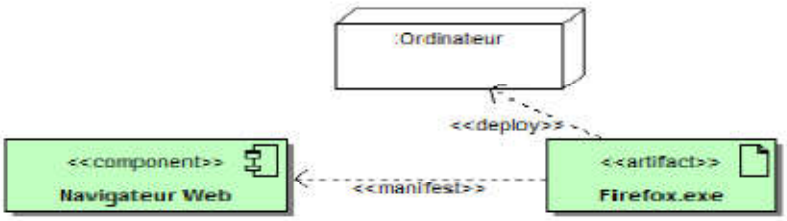


Figure12 Représentation d'un artefact à l'extérieur et ses liaisons avec un composant et un noeud.

4. Diagramme de structure composite

4.1. Définition du diagramme de structure composite

Un diagramme de structure composite est un diagramme UML qui joue le même rôle qu'un diagramme de classes, mais permet d'approfondir la description de la structure interne de plusieurs classes, à l'intérieur de celles-ci des instances collaborent par le biais de liens de communication afin de parachever des objectifs communs.

Le diagramme de structure composite permet d'afficher :

- La structure interne d'un classificateur
- Les interactions avec l'environnement par le biais des ports
- Un comportement d'une collaboration

4.2. Représentation de diagramme de structure composite :

Le diagramme de structure composite ne remplace pas un diagramme de classe mais le complète. Dans le diagramme de structure composite l'objet composé est décrit par un classifieur tans dit que ses composants sont décrits par des parties.

L'objet composé décrit par le diagramme de classe possède un composant issu d'une composition forte et un autre issu d'une agrégation.Dans le diagramme de structure composite les composants sont intégré au sein du classifieur qui décrit l'objet composé, la cardinalité est indiquée entre crocets. Un composant issue d'une agrégation est représenté par une ligne en points tillés, un composant issu d'une composition forte est représenté par une ligne continue.

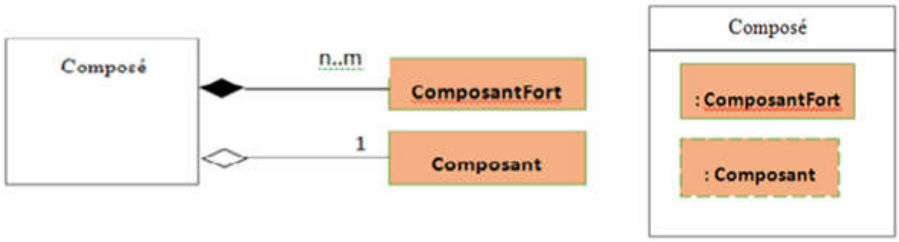
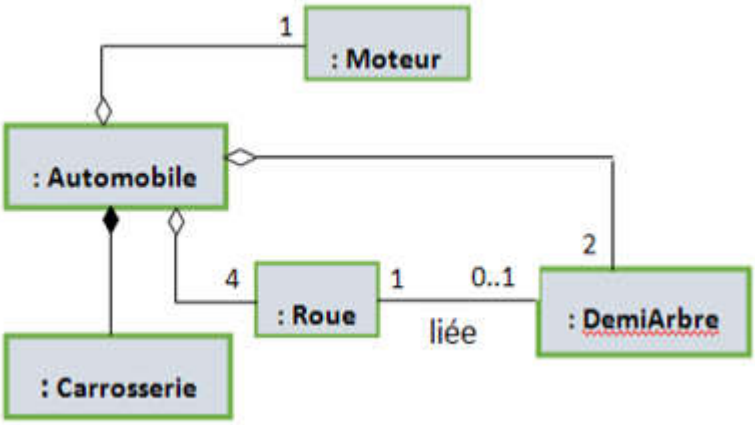
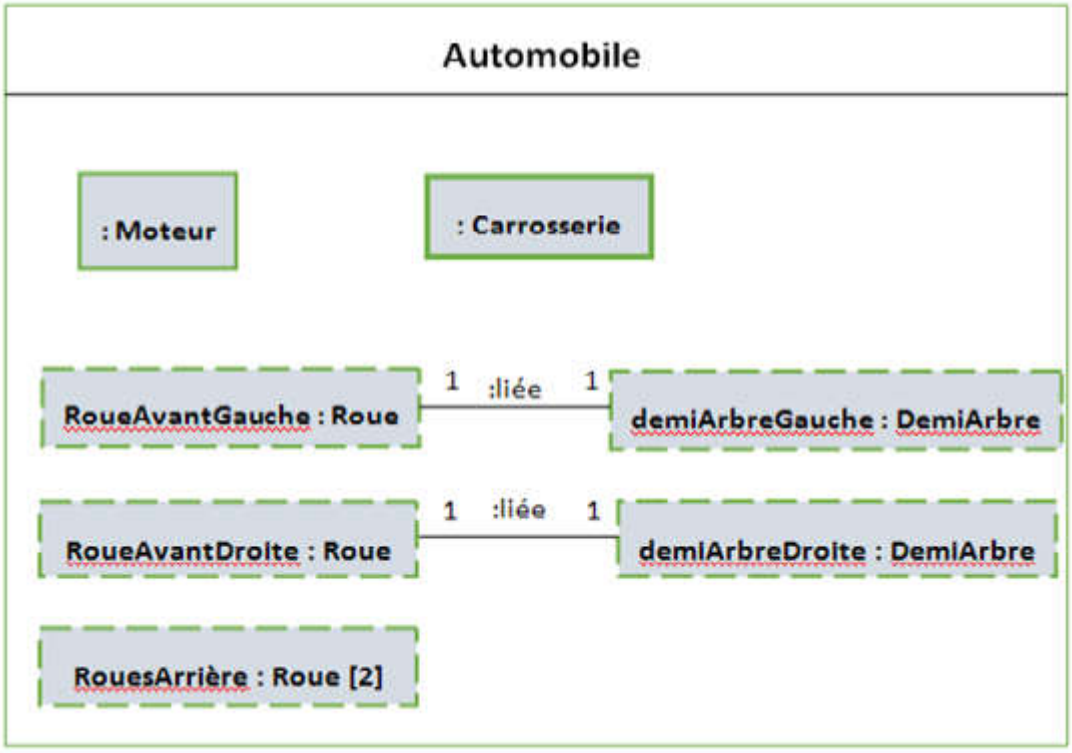


Figure 13 Représentation d'un diagramme de structure composite.

Exemple :



Ce diagramme de classes illustre une automobile en tant qu'objet composé il introduit l'association liée entre les roues et les demi arbres qui assurent la transmission entre le moteur et les roues avant qui sont les roues motrices. La cardinalité de l'association liée est 0..1 un pour les roues avant et zéro pour les roues arrière. Cette information ne peut pas être introduite dans le diagramme de classes à moins d'introduire deux sous classes de roue : roue avant et roue arrière mais cela alourdira le diagramme de classes.



Références :

1. Roques, P. (2009). UML 2 par la pratique : Etudes de cas et exercices corrigés, Eyrolles.
2. Charroux, B. Osmani, A. Thierry, Y. (2010). UML 2 pratique de la modélisation, Pearson.
3. Vallée, F. (2005). UML pour les décideurs. Paris, Editions Eyrolles
4. Chantal, M. Hugues, J. Leblanc, B. (2000). UML pour l'analyse d'un système d'information, Dunod.
5. Debrauwer, L. Van der heyde, F. (2008). UML 2 initiation, exemples et exercices corrigés, Editions ENI.