Claude Belleil
Université de Nantes

Le langage UML 2.0
Diagramme de Composants

1 Introduction

En UML, le diagramme de composant et le diagramme de déploiement correspondent à des vues statiques. Le premier décrit le système modélisé sous forme de composants réutilisables et met en évidence leurs relations de dépendance. Le second 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.
En modélisation de système d'information, il est rare de passer directement des besoins à la définition des classes. Dans la plupart des cas, il est plus simple de commencer par décrire les éléments de haut niveau du système afin d'établir son architecture, puis de gérer la complexité et les dépendances entre ces différentes parties. Les composants permettent d'organiser un système en morceaux logiciels gérables, réutilisables et échangeables.

2 Notion de composant

Un composant est un élément encapsulé, réutilisable et remplaçable du logiciel. On peut voir les composants comme des briques de construction: on les combine (créant peut-être des composants plus gros) afin de construire l'application. Ainsi, les composants peuvent avoir une taille relativement petite, comme une classe, ou importante, comme un gros sous-système
Un composant représente donc la partie modulaire d'un système qui encapsule son contenu et qui est remplaçable au sein de son environnement. Il définit un comportement en terme d'interfaces fournies ou requises. Il se définit donc par ses interfaces requises et fournies. Des fonctionnalités plus importantes peuvent être mises en oeuvre par la réutilisation de composants au sein d'un composant englobant ou d'un ensemble de composants, et en connectant leurs interfaces requises ou fournies entre elles. La définition d'interfaces requises ou fournies d'un composant peut mettre en jeu des ports. Un composant peut être substitué à un autre composant en représentant une relation de dépendance.

Les bons candidats à devenir des composants sont les éléments qui mettent en oeuvre une fonctionnalité clé et qui sont employés fréquemment dans le système. Dans un système, on peut créer un composant qui fournit des services ou accède à des données. En UML, un composant peut réaliser les mêmes choses qu'une classe :

- généraliser et s'associer à d'autres classes et composants,
- implémenter des interfaces,
- avoir des opérations,
- etc.

De plus, comme les structures composites, ils peuvent avoir des ports et révéler une structure interne. La principale différence entre une classe et un composant réside dans le fait qu'un composant a généralement de plus grandes responsabilités qu'une classe. De plus, il contient et utilise fréquemment d'autres classes ou composants pour mener à bien sa tâche.
Les composants sont des acteurs majeurs de la conception de système. Il est important qu'ils soient faiblement couplés afin que toute modification de l'un d'entre eux n'affecte pas le reste du système. Le faible couplage a pour objectif de faciliter la maintenance en minimisant les dépendances entre éléments. Le couplage exprime la relation étroite qu'un élément (Classe, Système ou sous système) entretien avec un ou plusieurs autres éléments. Un élément faiblement couplé ne possède pas ou peu de dépendances vis a vis d'autres éléments. Un couplage trop fort entre deux éléments peu entraîner lors de la modification d'un de ses éléments une modification d'un élément lié, ou bien une mauvaise compréhension des éléments pris séparément, ou encore une réutilisation contraignante du fait de la quantité d'éléments a intégrer pour en utiliser un seul.
Bien entendu, il ne faut pas tomber dans le piège qui consiste à concevoir des éléments tous indépendant et faiblement couplés, car cela irait à l'encontre du paradigme objet définissant un système comme un ensemble d'objets connectés les uns aux autres et communiquant entre eux.
Pour favoriser le couplage faible et l'encapsulation, l'accès aux composants se fait au travers d'interfaces. En permettant aux composants d'accéder les uns aux autres au travers d'interfaces, on réduit les risques de dégradation en chaîne dans le système.

Un composant[1] se représente par un rectangle à onglets facultatif placé dans le coin supérieur droit et possédant le stéréotype `<<component>>`

Un composant peut également représenter un sous-système appartenant à un système de dimension supérieure en remplaçant `<<component>>` par «`subsystem`». UML considère qu'un sous-système est un type de composant spécial et autorise donc une certaine souplesse quant à l'emploi de ce stéréotype. Cependant, il est préférable de le réserver aux éléments les plus gros du système général.

[1] Dans les versions précédentes d'UML, le symbole d'un composant était réalisé avec une représentation plus large de l'icône du rectangle à onglets. De nombreux AGL UML utilisent encore ce symbole.

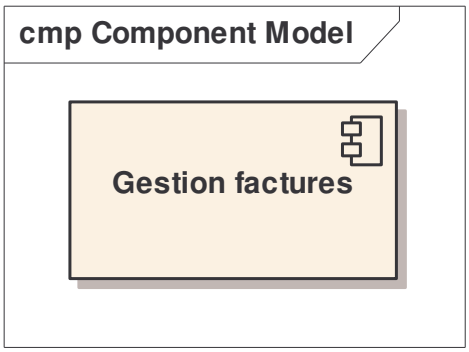

Figure 1:un composant simple (formalisme Enterprise Architect)

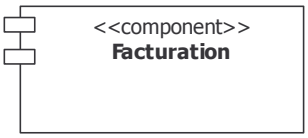

Figure 2: idem avec le formalisme StarUML

```
==========================================================================
«Composant[2]»

Règles de cohérence extraites du document de l'OMG « UML2SuperStructure[3]»


• Un composant peut être substitué à un autre composant seulement s'il est
de même type, c'est-à-dire si les interfaces fournies et requises par le
nouveau composant sont conformes aux interfaces de l'ancien composant.
[p.136]
• Une interface fournie par un composant doit être soit implantée par le
composant lui-même, soit par un des classificateurs qui le réalisent, ou
être le type d'un port fourni du composant. [p.136]
• Si cela est précisé, un composant ne peut être réalisé que par des classes
ou des collaborations.[Nouvelle règle]
• Les éléments possédés par un composant doivent être des éléments
empaquetables. [Règle dérivée du méta-modèle]
• Les artifacts qui implantent des composants doivent leur être connectés
soit par contenance physique soit par une relation < implement >, qui est
une instance de la méta-association entre composant et artifact. [p.141]
• Tout composant a au moins une interface fournie. [Nouvelle règle]
==========================================================================
```

3 Interfaces de composant

Les interfaces sont nécessaires car les composants doivent être faiblement couplés afin qu'ils puissent évoluer sans imposer des modifications aux autres parties du système. Les composants interagissent les uns avec les autres au travers des interfaces fournies et requises. Cela permet de contrôler les dépendances entre composants et de les rendre interchangeables.
L'interface fournie d'un composant est une interface qu'il est capable de mettre en œuvre. Les autres

[2] Les paragraphes avec ce style (Courier New 8) sont des traductions d'extraits du document officiel de définition du langage UML 2 dont vous trouverez la référence web ci-dessous.
[3] http://www.omg.org/technology/documents/formal/uml.htm

composants et classes interagissent avec ce composant au travers de ses interfaces fournies. Elles décrivent les services qu'il offre.
L'interface requise d'un composant est une interface dont il a besoin pour fonctionner. Plus précisément, il a besoin d'une autre classe ou d'un autre composant qui réalise cette interface. Pour conserver l'objectif de couplage faible, il accède à la classe ou au composant au travers de l'interface requise. Une interface requise déclare les services dont le composant a besoin.

En UML, il existe trois manières standard de représenter les interfaces fournies et requises :

- la notation à rotule avec parties fixe et articulaire,
- la notation de stéréotype,
- l'énumération textuelle. (non représenté ici)

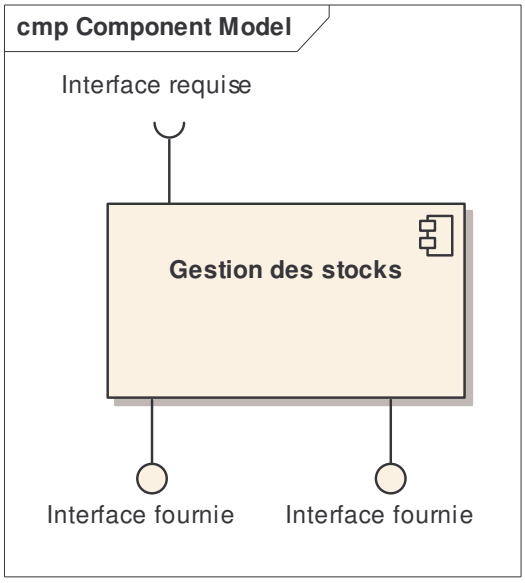

Figure 3:Interfaces "fournies" et "requises"(EA[4]) notation à rotules

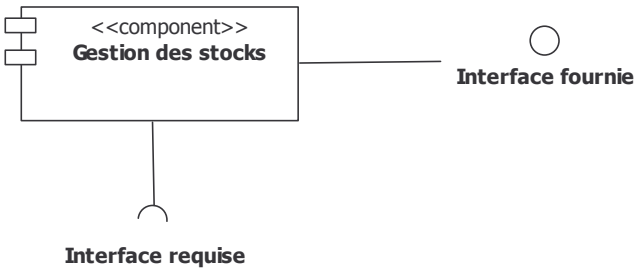

Figure 4: Formalisme « rotules » StarUml

[4] Entreprise Architect

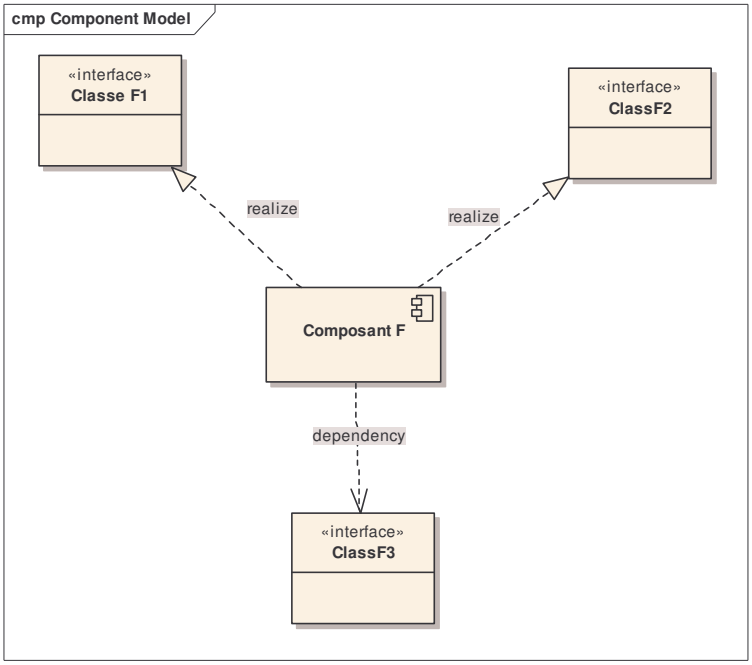

Figure 5 : Notation de stéréotype

4 Relations entre composants

Si un composant possède une interface requise, il a besoin d'une autre classe ou d'un autre composant du système pour la lui fournir. Pour représenter cette dépendance entre un composant et un autre composant fournissant l'interface requise, on trace une flèche de dépendance entre la partie articulaire du composant dépendant vers la partie fixe du composant fournisseur.

Certain AGL UML permettent de regrouper les deux parties (en omettant la flèche de dépendance), comme le montre la figure suivante. Il s'agit en fait de la notation de connecteur d'assemblage qui sera développée un peu plus loin.

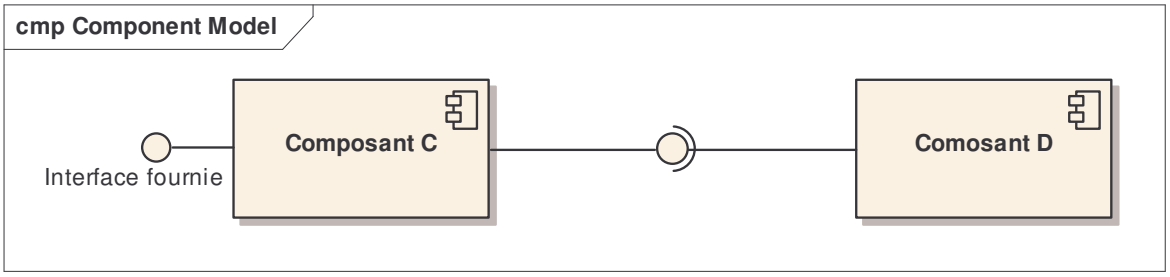

Figure 6: Regroupement de deux composants par un connecteur d'assemblage (EA)

Il est également possible d'omettre l'interface et de tracer la relation de dépendance directement entre les composants, comme le montre la figure ci-dessous.

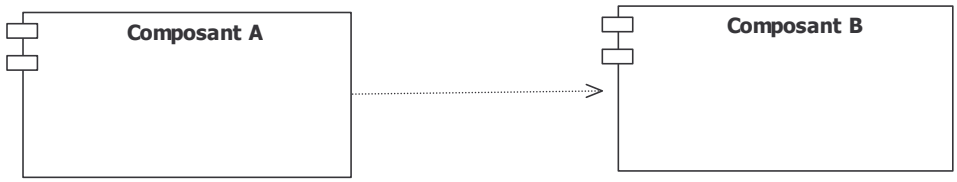

Figure 7: Regroupement par flèche de dépendance (StarUml)

Cette notation (omettre l'interface) est plus simple que la précédente. Cependant, quand on l'utilise, il ne faut pas oublier quelques éléments importants liés au formalisme :

Les interfaces facilitent le couplage faible des composants. Elles sont donc un facteur important de l'architecture des composants. Montrer les composants clés du système et leurs interconnexions au travers des interfaces constitue une bonne manière de décrire l'architecture du système et la première notation s'en charge parfaitement.

La seconde notation autorise des vues de dépendances entre les composants avec une granularité moins fine. Elle est utile pour exposer les problèmes de gestion ou de déploiement d'une configuration du système. En mettant l'accent sur les dépendances entre les composants et en donnant la liste des artefacts, il est plus facile d'identifier les composants et les fichiers requis pour le déploiement.

5 Ports et structure interne

Un composant doit fournir un service bien précis. Les fonctionnalités qu'il encapsule doivent être cohérentes entre elles et génériques (par opposition à spécialisées) puisque sa vocation est d'être réutilisable.
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 notions de ports et de structure interne ont déjà été évoquées au sujet des classes. Les composants ont également ces deux notions. On peut utiliser des ports pour modéliser différentes manières d'utiliser un composant en connectant les interfaces adéquates aux ports.

5.1 Définition

Un port est l'élément d'un composant qui sert à relier entre eux des composants via des liaisons entre ports. Un port réalise une interface de services. Un composant possède au moins un port. Il existe deux types de ports :

– Port de services requis : port qui définit des services qui sont requis sur d'autres composants par le composant pour son fonctionnement.
– Port de services offerts : port qui définit des services réalisés par le composant et offerts aux autres composants.

La connexion d'un port peut être optionnelle ou obligatoire. Un port de services requis obligatoire doit forcément être lié, soit directement, soit via un connecteur, à un port de services offerts compatible d'un autre composant. Pour le cas des composants composites, un port obligatoire de services offerts doit forcément être réalisé par un de ses composants internes.

Figure 8: Deux ports associés à des interfaces

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.
Généralement, un port est associé à une interface requise ou offerte. Parfois, il est relié directement à un autre port situé sur la limite du composant englobant par une flèche en trait plein, pouvant être stéréotypée «delegate», et appelée connecteur de délégation.
L'utilisation des ports permet de modifier la structure interne d'un classeur sans affecter les clients externes.

5.2 Connecteur de délégation

5.2.1 Présentation

L'interface fournie d'un composant peut être réalisée par l'une de ses parties internes. De manière similaire, son interface requise peut être imposée par l'une de ses parties. Dans ce cas, il est possible d'utiliser des connecteurs de délégation pour montrer que ces parties internes réalisent ou utilisent les interfaces du composant.
Les connecteurs de délégation se représentent par des flèches qui pointent dans la « direction du trafic », connectant le port attaché à l'interface à la partie interne. Si celle-ci réalise une interface fournie, la flèche pointe alors du port vers la partie interne. Si la partie utilise une interface requise, la flèche pointe de la partie interne vers le port.

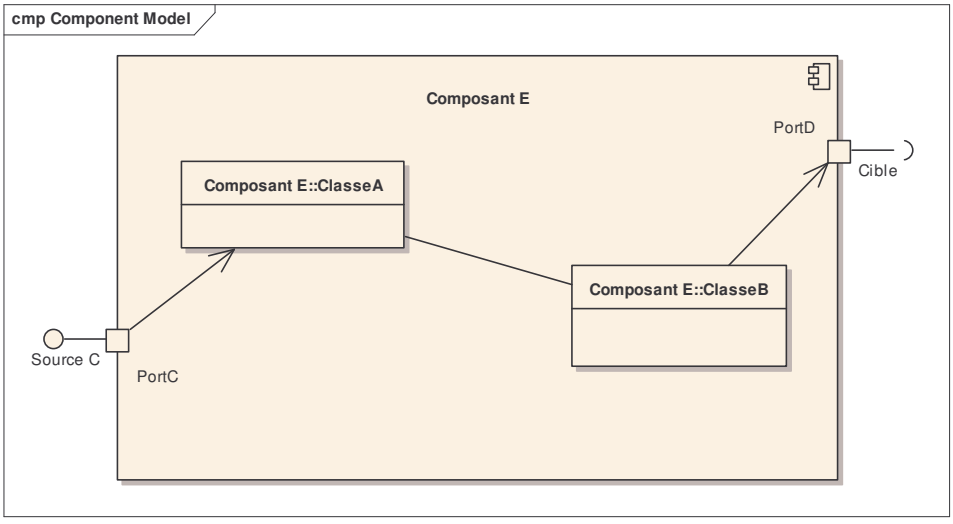

Figure 9: Structure interne de composant avec connecteurs de délégation

Interprétation du connecteurs de délégation : le port représente une entrée dans un composant au travers de laquelle passent les communications et les connecteurs de délégation pointent dans le sens de communication. Ainsi, un connecteur de délégation qui va d'un port à une partie interne représente des messages traités par celle-ci.

5.2.2 Définition et règles de cohérence

Un connecteur de délégation est un connecteur qui relie le contrat externe d'un composant (spécifié par ses ports) à la réalisation de ce comportement par les parties internes du composant.
Remarque La délégation suggère que les messages concrets et le flot de signaux apparaîtra entre les ports connectés, possiblement sur plusieurs niveaux de décomposition hiérarchique. Il est à noter que ce flot de données n'est pas toujours réalisé dans tous les environnements ou toutes les implantations (c'est-à-dire qu'il ne peut être considéré que seulement en phase de conception).

```
==========================================================================
«Connecteur de délégation[5]»

Règles de cohérence extraites du document de l'OMG « UML2SuperStructure[6]»


• Un connecteur de délégation doit uniquement être défini entre des
interfaces utilisées ou des ports de même type, c'est-à-dire entre deux
ports ou interfaces fournis (provided en anglais) par le composant ou entre
deux ports ou interfaces requis (required en anglais) par le composant.
[p.143]
• Si un connecteur de délégation est défini entre une interface fournie ou
un port et un classificateur interne au composant, alors ce classificateur
doit avoir une relation <implement> avec l'interface de ce port.[ p.143] 1
• Si un connecteur de délégation est défini entre une interface source ou un
port et une interface cible ou un port, alors l'interface cible doit
supporter un sous-ensemble des signatures compatible des opérations de
l'interface source ou du port.
• Dans un modèle complet, si un port source possède des connecteurs de
délégation vers un ensemble de ports cibles délégués, alors l'union des
interfaces de ces ports cibles doit être compatible (au niveau des
signatures) avec l'interface qui type le port source. [p.144]
• Un connecteur de délégation doit relier le port d'un composant à une
partie interne du composant.[Nouvelle règle]

==========================================================================
```

5.3 Connecteur d'assemblage

5.3.1 Présentation

Les connecteurs d'assemblage montrent qu'un composant requiert une interface fournie par un autre composant. Ils réunissent les parties fixe et articulaire d'une rotule qui représentent les interfaces fournies et requises.
Les connecteurs d'assemblage sont une forme spéciale de connecteurs utilisée pour montrer la structure composite des composants. Les connecteurs d'assemblage sont aussi parfois utilisés comme option de présentation pour la dépendance d'un composant au travers des interfaces

[5] Les paragraphes avec ce style (Courier New 8) sont des traductions d'extraits du document officiel de définition du langage UML 2 dont vous trouverez la référence web ci-dessous.
[6] http://www.omg.org/technology/documents/formal/uml.htm

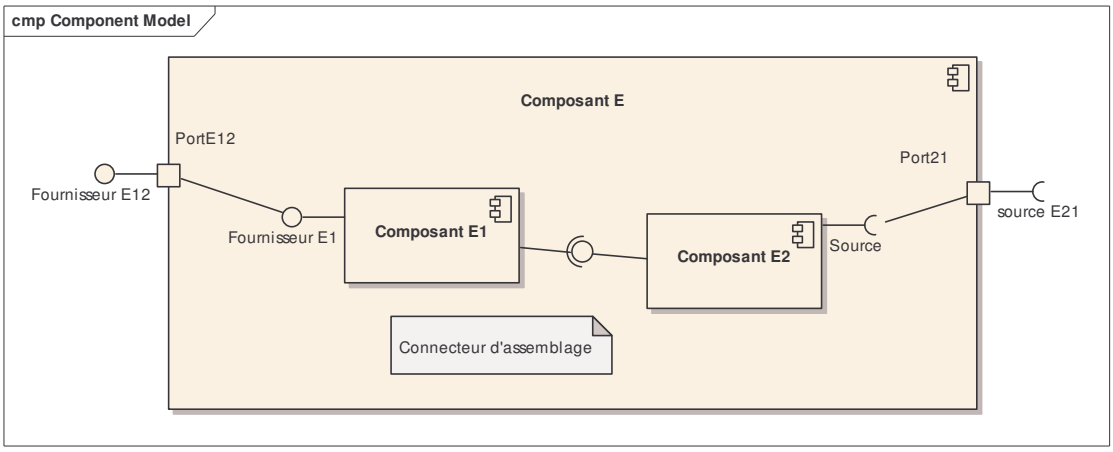

Figure 10: Structure interne avec connecteur d'assemblage

5.3.2 Définition et règles de cohérence

Un connecteur d'assemblage est un connecteur entre deux composants qui définit qu'un composant fournit le service qu'un autre composant requiert.

```
================================================================================
«Connecteur d'assemblage[7]»

Règles de cohérence extraites du document de l'OMG « UML2SuperStructure[8]»


• Un connecteur d'assemblage doit uniquement être défini à partir d'une
interface requise ou d'un port vers une interface fournie ou un port.
[p.144]
• L'ensemble des services fournis par le port ou l'interface doit être un
sur-ensemble de l'ensemble des services requis. [Nouvelle règle]. Les
services requis ou fournis sont la liste des opérations présentes dans
Les interfaces.
================================================================================
```

6 Conclusion

Le diagramme de composant spécifie un ensemble de constructions qui peuvent être utilisées pour définir des systèmes logiciels de taille et de complexité arbitraire.
Un composant est une unité modulaire dont les interfaces sont bien définies et qui est remplaçable dans son environnement. Les concepts mis en jeu dans ce diagramme traitent du domaine du développement basé composant (component-based development) et de la structuration de système basée composant. Le composant est modélisé au travers du cycle de vie et est successivement raffiné.
Un aspect important du développement basé composant est la réutilisation de composant construits précédemment. Un composant peut être considéré comme une unité autonome à l'intérieur d'un système ou d'un sous-système. Il a une ou plusieurs interfaces requises ou fournies (potentiellement exposée au travers de ports), et son intérieur est caché et inaccessible autrement qu'au travers de ses interfaces.

[7] Les paragraphes avec ce style (Courier New 8) sont des traductions d'extraits du document officiel de définition du langage UML 2 dont vous trouverez la référence web ci-dessous.
[8] http://www.omg.org/technology/documents/formal/uml.htm

```
================================================================================
«Diagramme de composant»

Règles de cohérence extraites du document de l'OMG « UML2SuperStructure»


• Les nœuds contenus dans un diagramme de composants ne peuvent être que :

- des composants,
- des interfaces,
- des classes,
- des artefacts,
- des ports. [p.149]


• Les chemins graphiques qui peuvent apparaître dans un diagramme de
composants sont : [p.149]

- des connecteurs d'assemblage,
- des connecteurs de délégation,
- des réalisations,
- des relations de d'implantation,
- des dépendances d'utilisation,
- des dépendances.
================================================================================
```

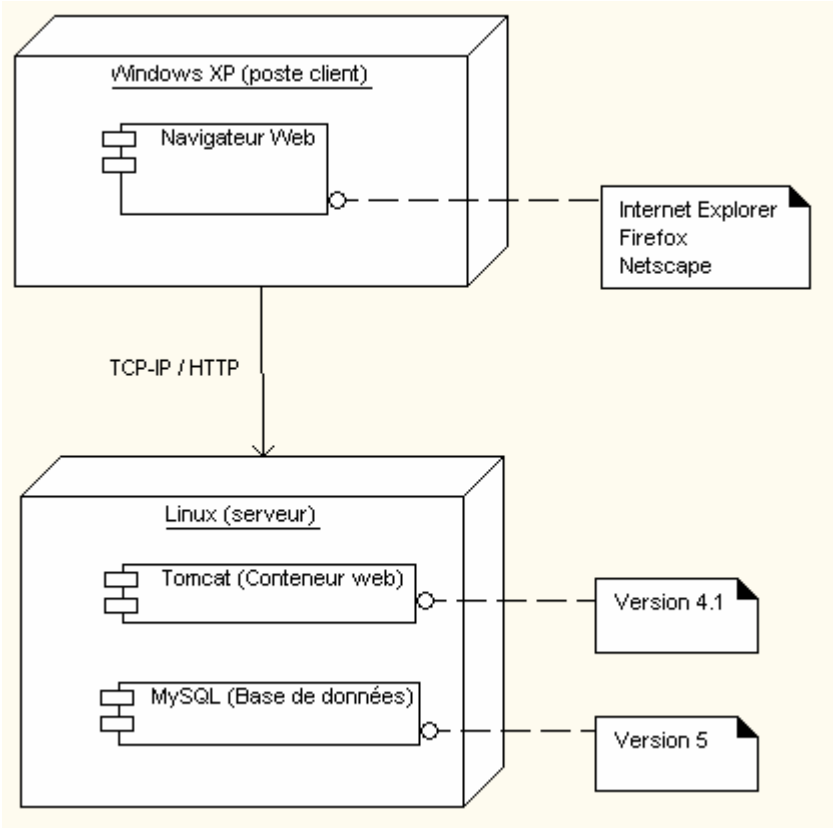

Figure 11: Exemple de composants répartis sur un déploiement

Index du texte :

Index des figures :