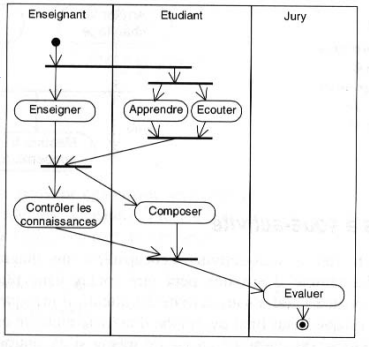
Les diagrammes UML (suite)

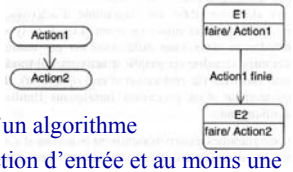
- Les vues statiques
  - *Diagrammes de classes* : classes et associations
  - *Diagrammes d'objets* : liens et objets, diag. de collaborations simples
  - *Diagrammes de cas d'utilisation* : fct du système du point de vue de l'utilisateur
  - *Diagrammes de composants* : composants physiques d'une application
  - *Diagrammes de déploiement* : déploiement des composants sur les matériels
- Les vues dynamiques
  - *Diagrammes de séquences* : représentation temporelle des objets et des interactions
  - *Diagrammes de collaborations* : représentation spatiale des objets, des liens et des interactions
  - ***Diagrammes d'états-transitions*** : comportement d'une classe ou d'une méthode en terme d'états
  - *Diagrammes d'activités* : **comportement d'une méthode, d'un cas d'utilisation ou d'un processus métier**

Les diagrammes d'activités

- Un diagramme d'activité :
  - Met en avant les activités et les transitions
  - Modélise le comportement interne d'une méthode, d'un cas d'utilisation
- Une vue différente sur des automates donnés
- Utilise des éléments des diagrammes d'états-transitions



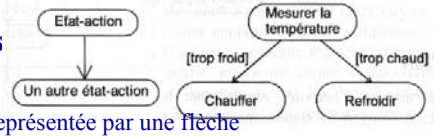
Les états-actions



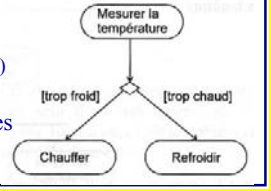
- Une étape dans l'exécution d'un algorithme
- Un état simplifié, avec une action d'entrée et au moins une transition automatique vers un autre état
- Symbole : un rectangle arrondi
- L'action peut-être définie en langage naturelle, en pseudo-code ou en langage de programmation

Ouvrir la porte Dossier.Ouvrir(Fichier) i := i +1

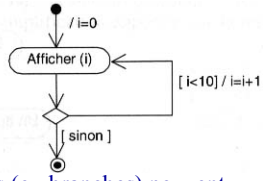
Les transitions



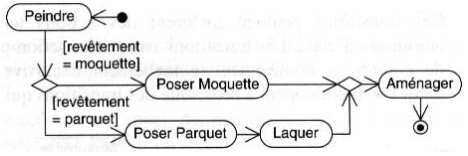
- Une transition représentée par une flèche
- Elle est automatique après la fin de l'action
- Pas la peine de faire figurer un nom d'évènement
- Les transitions peuvent également être déclenchées par :
  - un signal
  - La réalisation d'une condition
  - Des conditions booléennes (décisions) mutuellement exclusives (les gardes)
- Point de jonction (les 2 diagrammes sont équivalents)



Les transitions (suite)

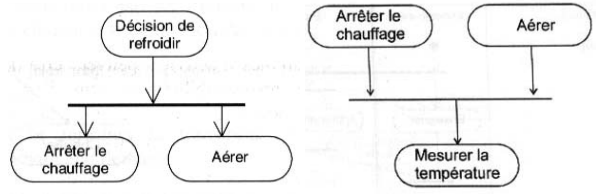


- Une garde prédéfinie : **Sinon**
- Au point de jonction, des chemins (ou branches) peuvent se rejoindre
- Plusieurs branches arrivent au point de jonction (losange)



La synchronisation

- Une barre de synchronisation permet d'ouvrir ou de fermer une activité concurrente



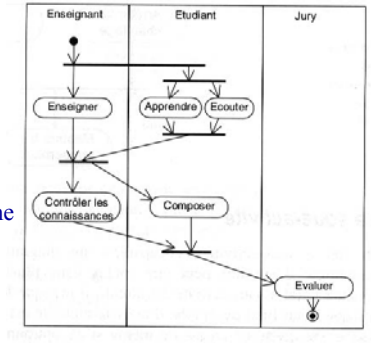
La barre ne peut être franchit que lorsque toutes les transition en entrée sur la barre sont déclenchées

Les états à sous-activité

- Encapsule un diagramme d'activité
- Un diagramme d'activité peut-être inclus dans plusieurs états à sous-activité
- Lorsqu'un tel état est atteint il exécute le graphe d'activité emboîté
- Il en ressort quand celui-ci atteint l'état final du sous-graphe d'activité

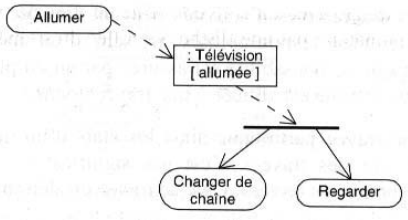
Les travées

- Découpage qui fait ressortir les responsabilités au sein d'un mécanisme ou d'une organisation
- Une travée partitionne les états d'un diagramme d'activité

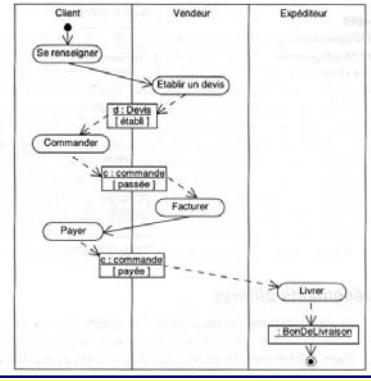


Flots entre actions et objets

- L'objet apparaît clairement dans le diagramme d'activité, éventuellement au sein d'une travée
- L'objet utilisé par une action est visualisé en connectant celui-ci à l'état action par une flèche en pointillée
- Cet objet peut apparaître plusieurs fois (lisibilité)
- On précise son état

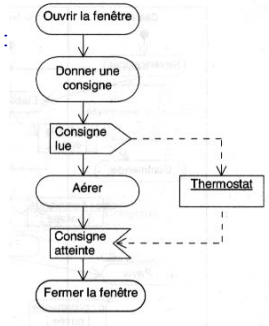


Flots entre actions et objets (suite)



Icônes associées aux transitions

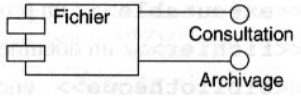
- UML définit les symboles suivants :
  - L'envoi d'un signal : un pentagone convexe (rectangle + triangle)
  - L'attente d'un signal : un pentagone concave (rectangle enfoncé par un triangle). Dans les 2 cas (émission et réception de signal) la signature du signal est indiquée dans le symbole
  - Il est inséré entre 2 états-action



Les diagrammes UML (suite)

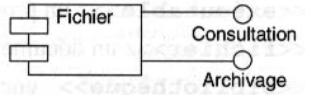
- Les vues statiques
  - *Diagrammes de classes* : classes et associations
  - *Diagrammes d'objets* : liens et objets, diag. de collaborations simples
  - *Diagrammes de cas d'utilisation* : fct du système du point de vue de l'utilisateur
  - ***Diagrammes de composants* : composants physiques d'une application**
  - *Diagrammes de déploiement* : déploiement des composants sur les matériels
- Les vues dynamiques
  - *Diagrammes de séquences* : représentation temporelle des objets et des interactions
  - *Diagrammes de collaborations* : représentation spatiale des objets, des liens et des interactions
  - *Diagrammes d'états-transitions* : comportement d'une classe ou d'une méthode en terme d'états
  - *Diagrammes d'activités* : comportement d'une méthode, d'un cas d'utilisatio ou d'un processus métier

Les diagrammes de composants

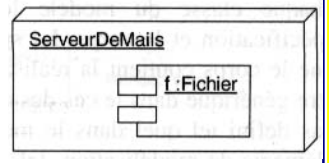


- Ils décrivent les composants et leurs dépendances dans l'environnement de réalisation
- Un composant :
  - un éléments physique (du code source, binaire, exécutable), un script un fichier de commandes, un fichier de données, une table, …
  - peut réaliser un ensemble d'interfaces : le comportement offert à d'autres composants
  - Comprend des éléments qui implémentent les services
  - Peut posséder des attributs et des opérations
  - Peut être connecté à d'autres composants
  - Réside dans un ou plusieurs nœuds (diagramme de déploiement)
  - Ex : un composant Fichier avec deux de ses interfaces

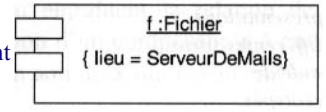
Les composants (suites)



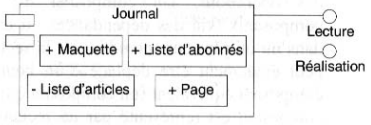
- Une instance de composant
  - instance:nom_du_composant
- Comme pour les objets, le nom de l'instance ou du composant peut-être omis
- Ex : instance d'un composant **Fichier** dans un nœud



- On peut préciser le lieu où les instances d'un composant vont résider (valeur marquée)

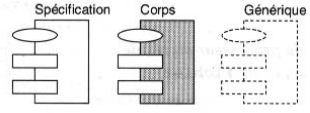


Les composants (suite)



- Les composants d'un composant plus global sont représentés dans le symbole de ce dernier
- Il contient également les classes implémentées (mais non possédées)
- Indication de visibilité au niveau des classes
- Stéréotypes (on peut définir des icônes, UML n'en prévoit pas):
  - « document » : quelconque
  - « exécutable » : un programme qui peut s'exécuter
  - « fichier » : du code source ou des données
  - « bibliothèque » : statique ou dynamique
  - « table » : un base de donnée

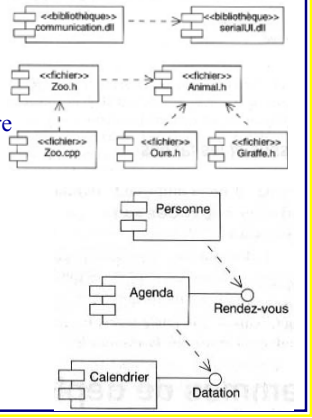
Les modules



- Unité pour la manipulation et le stockage des éléments physiques qui entrent dans la fabrication des applications informatiques
- Ex :
  - De simples fichiers
  - Des paquetages du langage ADA
  - Des bibliothèques
- Par défaut chaque classe du modèle logique est réalisé par deux composants :
  - La spécifications : l'interface de la classe
  - Le corps : la réalisation de cette classe
- En java les modules existent sous l'appellation de paquetage

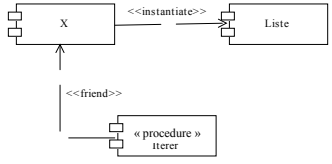
Les dépendances entre composants

- Une dépendance : un élément d'implémentation fait appel à un autre élément d'implémentation dans un autre composant
- Dépendance : flèche en pointillé
- Si le composant utilise l'interface la relation de dépendance connecte le composant utilisateur à l'interface du composant fournisseur

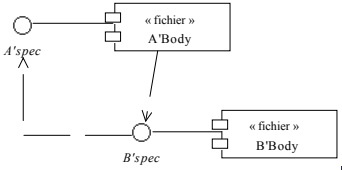


Stéréotypes et dépendance

- La relation de dépendance peut-être spécialisée par un stéréotype
  - Préciser la nature de la relation

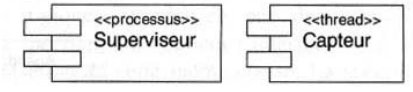


- Relation de dépendance de compilation, les composants sont alors du code source
- L'ordre de compilation est donné par les relations de dépendance du graphe



Les processus et tâches

- Les tâches ou thread : composants qui possèdent leur propre flot de contrôle
- Ces tâches peuvent être contenues dans d'autres composants
- Les processus ne peuvent pas contenir d'autres processus
- 2 stéréotypes : « processus » et « thread »

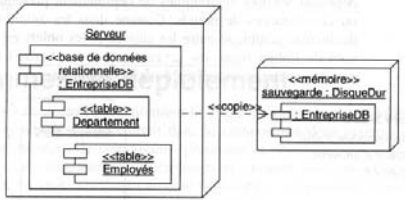


Les diagrammes UML (suite)

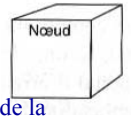
- Les vues statiques
  - *Diagrammes de classes* : classes et associations
  - *Diagrammes d'objets* : liens et objets, diag. de collaborations simples
  - *Diagrammes de cas d'utilisation* : fct du système du point de vue de l'utilisateur
  - *Diagrammes de composants* : composants physiques d'une application
  - ***Diagrammes de déploiement* : déploiement des composants sur les matériels**
- Les vues dynamiques
  - *Diagrammes de séquences* : représentation temporelle des objets et des interactions
  - *Diagrammes de collaborations* : représentation spatiale des objets, des liens et des interactions
  - *Diagrammes d'états-transitions* : comportement d'une classe ou d'une méthode en terme d'états
  - *Diagrammes d'activités* : comportement d'une méthode, d'un cas d'utilisatio ou d'un processus métier

Les diagrammes de déploiement

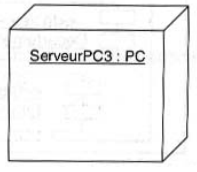
- Un graphe composé de nœuds interconnectés par des liens de communication
- Montrent la disposition physique des différents matériels : les nœuds qui entrent dans la composition d'un système et la répartition des instances de composants, de processus et objets qui « vivent » sur ces matériels:
- Sous 2 formes :
  - Spécification
  - Instance



Les noeuds

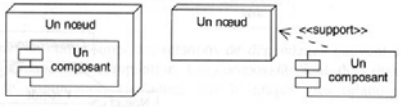


- Une ressource matériel, représenté par un nœud
- Une ressource possède en général au minimum de la mémoire et des fois des capacités de calcul
- Un nœud : un cube 3D
  - Un calculateur
  - Des ressources humaines
  - Un périphérique

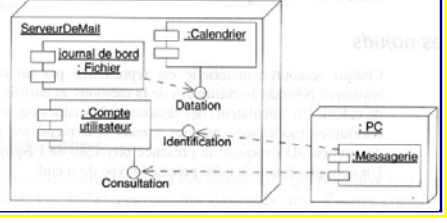


- Il peut avoir des attributs :
  - vitesse du processeur
  - Espace mémoire
- Des classes de nœuds (notation comme avec classe/objet)

Les nœuds (suite)

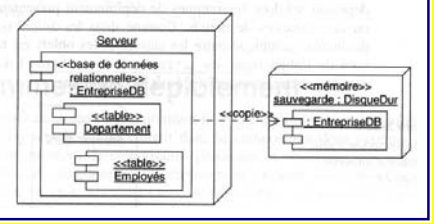


- 2 manières de montrer qu'un composant réside sur un nœud :
  - Le composant emboîté
  - Mot clé : « support » sur le lien de dépendance orienté du composant vers le nœud
- A l'exécution les instances résident sur le noeud



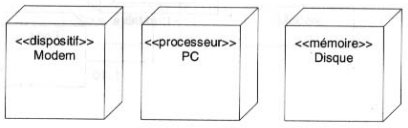
Les nœuds (suite)

- 2 stéréotypes additionnels :
  - « Devient » : migration d'un composant d'un nœud vers autre nœud
  - « Copie » : la réplication d'un composant. Une dépendance de copie de A vers B signifie que B est une copie exacte de A
- La migration d'un composant ou d'un objet

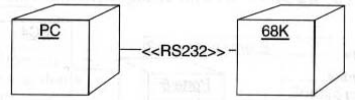


Les nœuds : préciser la nature de l'équipement

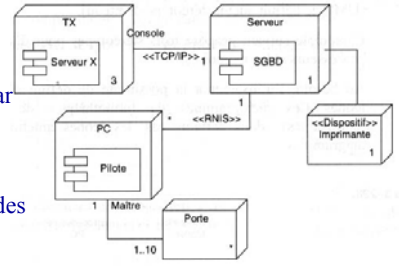
- Au moyen de stéréotypes (aucun par défaut défini par UML)
- Exemple de 3 stéréotypes pour distinguer les dispositifs, les processeurs et les mémoires
- L'utilisateur est libre d'en définir



Les supports de communication

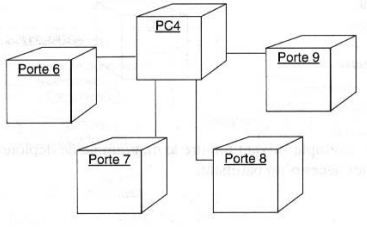


- Les divers nœuds sont connectés par des lignes qui symbolisent un support de communication *a priori* bidirectionnel
- La nature de ce lien peut-être précisée par un stéréotype
- Ex : diagramme de déploiement d'un système de gestion des accès d'un bâtiment



Diagrammes de déploiement et instances de noeuds

- Instances : noms soulignés
- Les portes 6, 7, 8 et 9 sont pilotées par le PC4



Les diagrammes UML (fin)

- Les vues statiques
  - *Diagrammes de classes* : classes et relations
  - *Diagrammes d'objets* : liens et objets, diag. de collaborations simples
  - *Diagrammes de cas d'utilisation* : fct du système du point de vue de l'utilisateur
  - *Diagrammes de composants* : composants physiques d'une application
  - *Diagrammes de déploiement* : déploiement des composants sur les matériels
- Les vues dynamiques
  - *Diagrammes de séquences* : représentation temporelle des objets et des interactions
  - *Diagrammes de collaborations* : représentation spatiale des objets, des liens et des interactions
  - *Diagrammes d'états-transitions* : comportement d'une classe ou d'une méthode en terme d'états
  - *Diagrammes d'activités* : comportement d'une méthode, d'un cas d'utilisatio ou d'un processus métier

Résumé

Diagrammes	Vue des cas d'utilisation	Vue logique	Vue de réalisation	Vue des processus	Vue de déploiement
Cas d'utilisation	Acteurs, cas d'utilisation				
De classes		Classes Associations			
D'objets	Objets liens	Classes objets liens			
De séquence	Acteurs objets messages	Acteurs, objets, messages		Objets messages	
De collaboration	Acteurs objets liens messages	Acteurs objets liens messages		Objets liens messages	
D'états-transitions	Etats transitions	Etats transitions		Etats transitions	
D'activités	Activités transitions	Activités transitions		Activités transitions	
De composants			Composants	Composants	Composants
De déploiement					Nœuds liens

Exemple de diagramme d'activité

Exemple de diagramme de déploiement

Exemple de diagramme de composant