Documentation d’architecture logicielle

GLO-3001

Architecture logicielle

Luc Lamontagne

Hiver 2010

Introduction

- L’architecture sert de devis pour le système :
  - Elle est la principale description des attributs de qualité du système.
  - C’est un excellent artefact pour l’analyse préliminaire.
  - Elle défini l’affectation du travail en équipe.
  - C’est un élément important pour faire de la maintenance après le déploiement.
- La documentation décrit les décisions de l’architecte logicielle.

Matériel extrait du livre *Documenting Software Architecture: Views and Beyond* (2003) par Clements et al.

La documentation est importante?

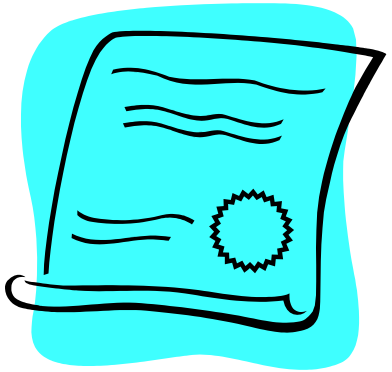
- La documentation est importante
  - si et seulement si communiquer les détails de cette architecture est important.
- Comment utiliser une architecture si on ne peut pas la comprendre?
  - Et comment la comprendre si elle ne peut pas être communiquée efficacement?

Comment documenter une architecture?

- Dans l'industrie, la réponse semble :
  - "Dessine des boîtes et des lignes !"
  - "Utilise UML."
  - "On a besoin d'autre chose que des diagrammes de classes?"
  - "Comment on documente une *quoi*?"
- On peut suivre une approche structurée pour y arriver.

Sept principes de documentation

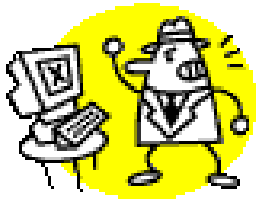
1. Rédiger du point de vue du lecteur.
2. Éviter les répétitions inutiles.
3. Éviter les ambiguïtés.
4. Utiliser une organisation standardisée.
5. Prendre en note les motivations.
6. Maintenir la documentation à jour… mais pas trop!
7. Réviser la documentation pour les besoins de la cause.



1. Rédiger du point de vue du lecteur

- Qu'est ce que le lecteur veut savoir ?
  - Rendre l'information facile à trouver !
  - Le lecteur appréciera…
    - Et sera mieux disposé pour lire le document.

- Une documentation rédigée en fonction du rédacteur :
  - Séquence d'idées: l'ordre de présentation correspond à ce qui passait par la tête du rédacteur
  - Séquence d'exécution: présentation suivant l'ordre que les choses se produisent dans l'ordinateur



2. Éviter les répétitions inutiles

- Chaque sorte d'information devrait être consignée à exactement un seul endroit.
- Cela rend le document plus facile à utiliser et à modifier.
- Les répétitions sèment la confusion
  - Parce que l'information peut être répétée légèrement différemment à chaque fois.
  - Laquelle est correcte?

3. Éviter les ambiguïtés

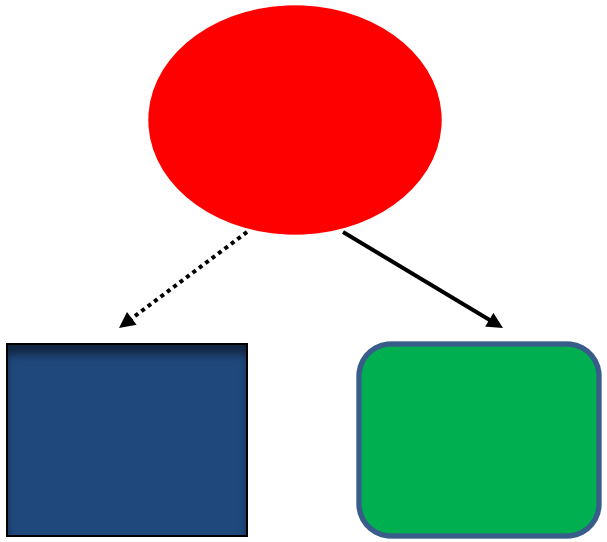
- Documentation → communiquer idées et informations.
  - Trop de détails → le lecteur est confus et la documentation devient inutile.
- Utilisation de notations bien définies
  - permet d'éviter toutes sortes d'ambiguïté.
- Si on utilise une représentation graphique
  - On inclut une légende
  - Ou on indique la signification de chaque symbole.
    - Ne pas oublier les lignes et les flèches!

3. Éviter les ambiguïtés (suite)

- Les diagrammes boîtes et lignes sont une forme très répandue.
  - Mais qu'est-ce qu'ils veulent dire?
  - Ne désigne pas une architecture, mais seulement un début de description.

- Si vous en utiliser un :
  - toujours définir précisément ce que les boîtes et les lignes représentent.

- Si on vous présente un tel diagramme :
  - demandez ce qu'il désigne.
  - Le résultat est parfois divertissant.



4. Utiliser une organisation standardisée

- Établir l'organisation du document
  - S'assurer que le document la respecte.
  - S'assurer que le lecteur en est conscient.
- Une organisation standardisée
  - Aide le lecteur à naviguer et trouver l'information
  - Aide le rédacteur à placer l'information et estimer le travail qui reste à accomplir.
  - Permet de vérifier la complétude du travail.
- Organiser la documentation pour faciliter les références.
  - Un document pourrait n'être lu qu'une fois… s'il est lu.
  - Un bon document fera l'objet de références plusieurs centaines ou milliers de fois.
- Ne pas laisser de section vide !
  - Mettre une indication du genre *"à être complété"*

5. Prendre en note les motivations

- Pourquoi est-ce que nous avons pris certaines décisions de cette manière?
  - Comment s'en souvenir dans une semaine, un mois, une année?
  - Comment le petit nouveau sur le mandat en prendra connaissance?
- Exige de la discipline, mais sauve énormément de temps à la longue.
- Documentez également les options qui ont été étudiées mais rejetées.

6. Maintenir la documentation à jour… mais pas trop!

- Mettre à jour :
  - Une documentation incomplète ou périmée, ne représente pas l'existant
    - Ne permet pas de rencontrer ses objectifs (communiquer efficacement avec les intervenants du projet) .
  - Une documentation à jour a plus de chance d'être utilisée.
  - Avec une documentation à jour, il est plus facile de répondre aux questions en référant la personne à la documentation.
  - Si une question ne peut pas être répondue avec le document :
    - Modifier le document et référer la personne à la nouvelle documentation.

6. Maintenir la documentation à jour… mais pas trop!

- Mais pas trop!
  - Durant le processus de conception, on reconsidère souvent nos décisions.
  - Faire une révision à tous les 10 minutes occasionne des dépenses inutiles.
  - Choisir des jalons dans le plan de développement pour faire les mises à jour.
  - Faire des livraisons de documentation qui correspondent au déroulement du projet.

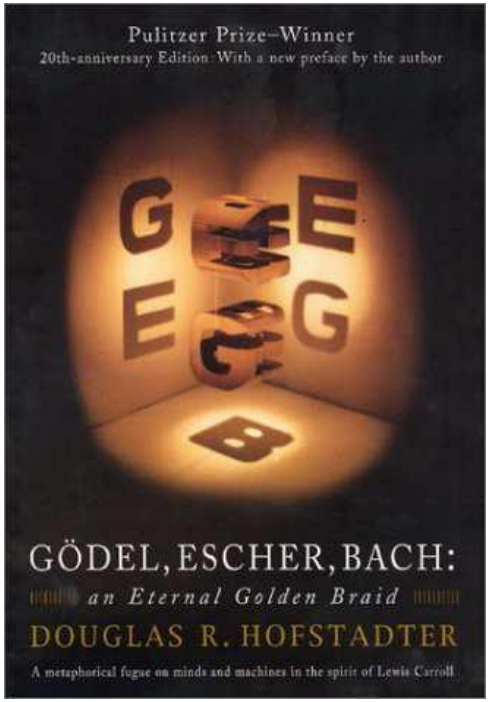
Quelle information doit être documentée?

- *Documenter les vues pertinentes, et ajouter l'information qui s'applique à plus d'une vue, ce qui permet de les lier ensemble.*

- L'architecture a trait à la structure. Mais quelle structure ?
- Le logiciel a plus d'une dimension:
  - Programmes, objets, classes, modules, processus, *frameworks*, tâches, *threads*…
  - Chacune défini une structure différente.
  - Laquelle est l'architecture ?
    - Réponse : Elles peuvent toute l'être.

Utiliser les structures et les vues

- Chaque structure fournie à l'architecture des points pour manipuler des aspects du système.
  - Ils consignent leur conception en utilisant les vues correspondantes.
- *Structure*
  - Une ensemble d'éléments architecturaux tels qu'ils existent dans le logiciel
- *Vue*
  - Ensemble d'éléments architecturaux, tels que rédigés et lus par les intervenants du projet.
  - Une vue représente un ensemble d'éléments et les relations entre ces éléments.
  - Utilisée pour répondre à différentes questions à propos de l'architecture
    - Quels sont les principales unités d'exécution et les principaux dépôts de données?
    - Quels autres logiciels ce système peut utiliser?
    - Quel est le flot de données dans le système?
    - Comment est-ce que le logiciel est déployé sur les machines?



Structures et vues

- Les structures architecturales et les vues peuvent être divisées en 3 types :

1. Structure de "module" –

- Les éléments qui sont des unités d'implémentation appelés *modules.*

2. Structure "composant-connecteur"

- Les composants exécutés (unités de traitement) et les connecteurs (liens de communication) entre eux.

3. Structure "allocation"

- Les éléments logiciels et leur relations avec les éléments externes de l'environnement sur lequel il s'exécute.

Vues de modules

- **Vue de décomposition**
  - Illustre les modules qui sont reliés par une relation de type *“est un sous-module de”.*
- **Vues d’utilisation**
  - Illustre les modules qui sont reliés par une relation de type *“uses”*.
  - C.-à-d. un module utilise les services fournis par un autre module.
- **Vue par couche**
  - Illustre les modules qui sont partitionnés en regroupement de fonctionnalités cohérentes qui sont reliées entre elles.
  - Chaque groupe représente une couche dans la structure globale.
- **Vue de classe et généralisation**
  - Illustre les modules de classes qui sont reliés par une relation d’héritage et de type *“est une instance de”* .

Vues composant-connecteur

- **Vue de processus**
  - Illustre les processus ou *threads* qui sont connectés par communication, synchronisation, ou opérations d'exclusion
- **Vue de concurrence**
  - Désigne les composants et connecteurs où les connecteurs représentent des "threads" logiques
- **Vue de partage de données (dépôt)**
  - Désigne les composants et connecteurs qui crée, emmagasine, et accède à des données persistantes
- **Vue client-serveur**
  - Illustre la coopération entre des clients et des serveurs, ainsi que les connecteurs entre eux
  - Les connecteurs indiquent les protocoles et les messages qu'ils s'échangent

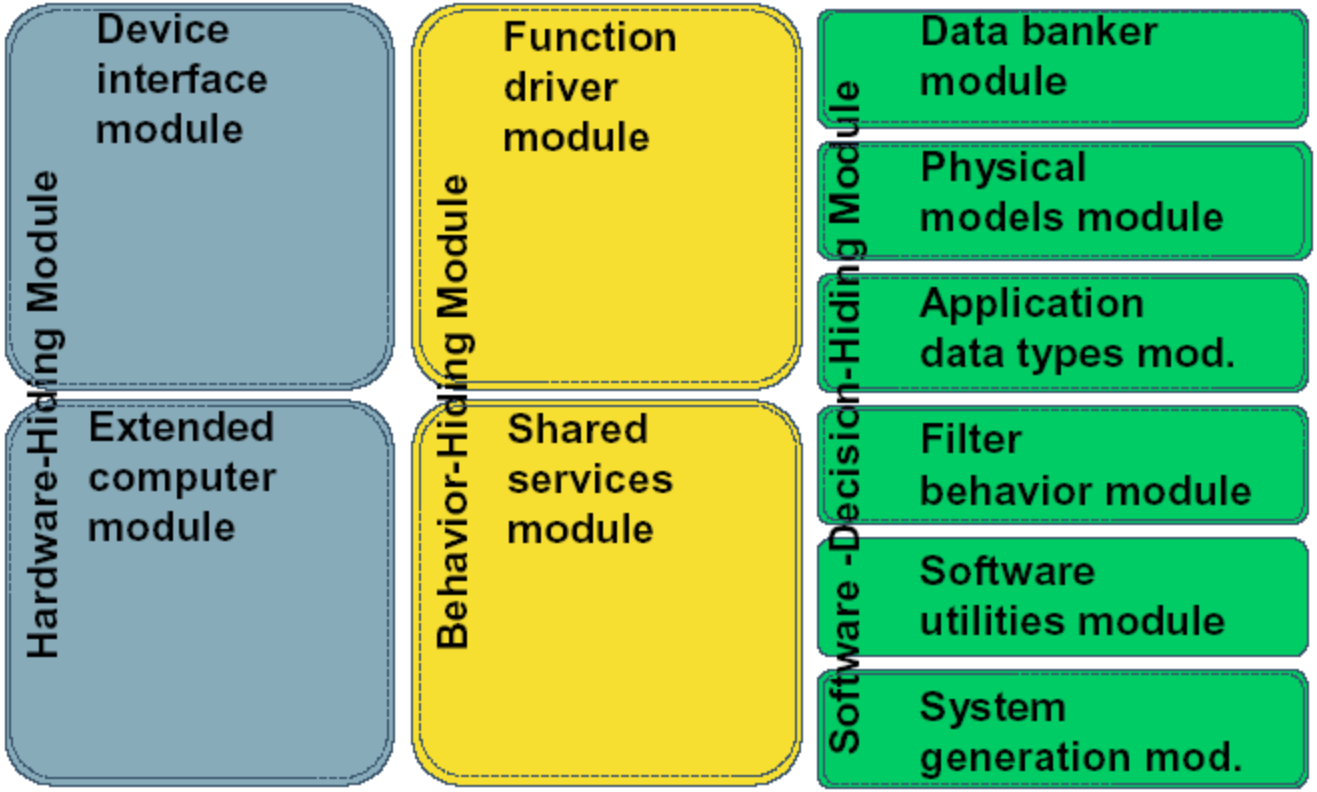
Vues d’allocation

- **Vue de déploiement**
  - Illustre les éléments logiciels et leur allocation aux composantes *hardware* et éléments de communication.
- **Vue d’implémentation**
  - Illustre les éléments logiciels et leur affectation à des structures de fichiers dans le développement, l’intégration et l’environnement de contrôle de configuration.
- **Vue de l’affectation de travail**
  - Désigne les modules et comment ils ont été affectés à l’équipe responsable de les implanter et de les intégrer.

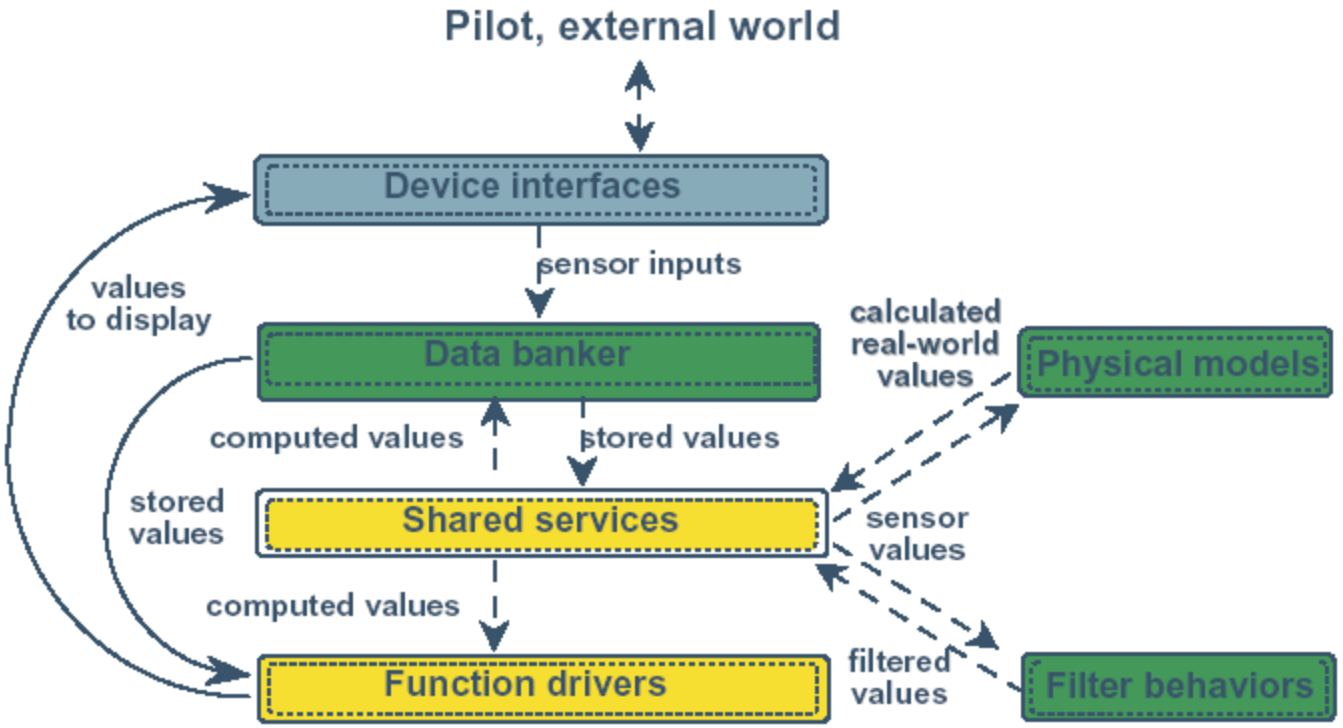
Exemples de vues multiples

- Architecture logicielle pour l'avion A-7E Corsair II
  - Un chasseur léger, basé sur porte-avion américain
  - Utilisé des années 1960s aux années 1980s
  - Petit ordinateur à bord pour la navigation et contrôler les armes

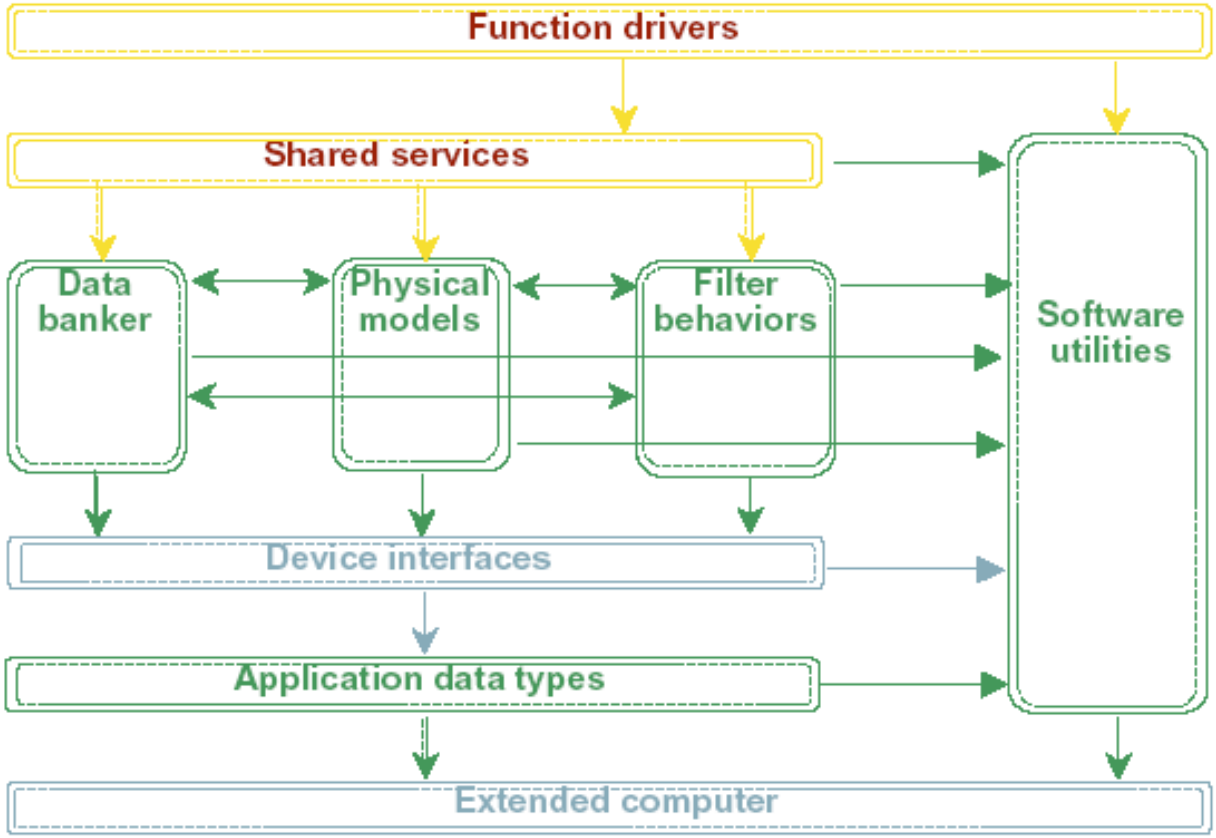
Vue de décomposition de modules



Vue de flot de données



Vue par niveau



Quels langages ou notations utilisés?

- UML
  - Pas spécifiquement conçu pour la documentation d'information architecturale
  - Mais c'est le langage de facto.
  - UML 2.0 est adéquat
    - notions d'architecture comme "composant" et "connecteur".

- Notation informelle "boîte et flèche"
  - Pas toujours facile à déchiffrer!
  - Avantage : flexibilité.
  - Désavantage : Vague, pas d'outil de support.

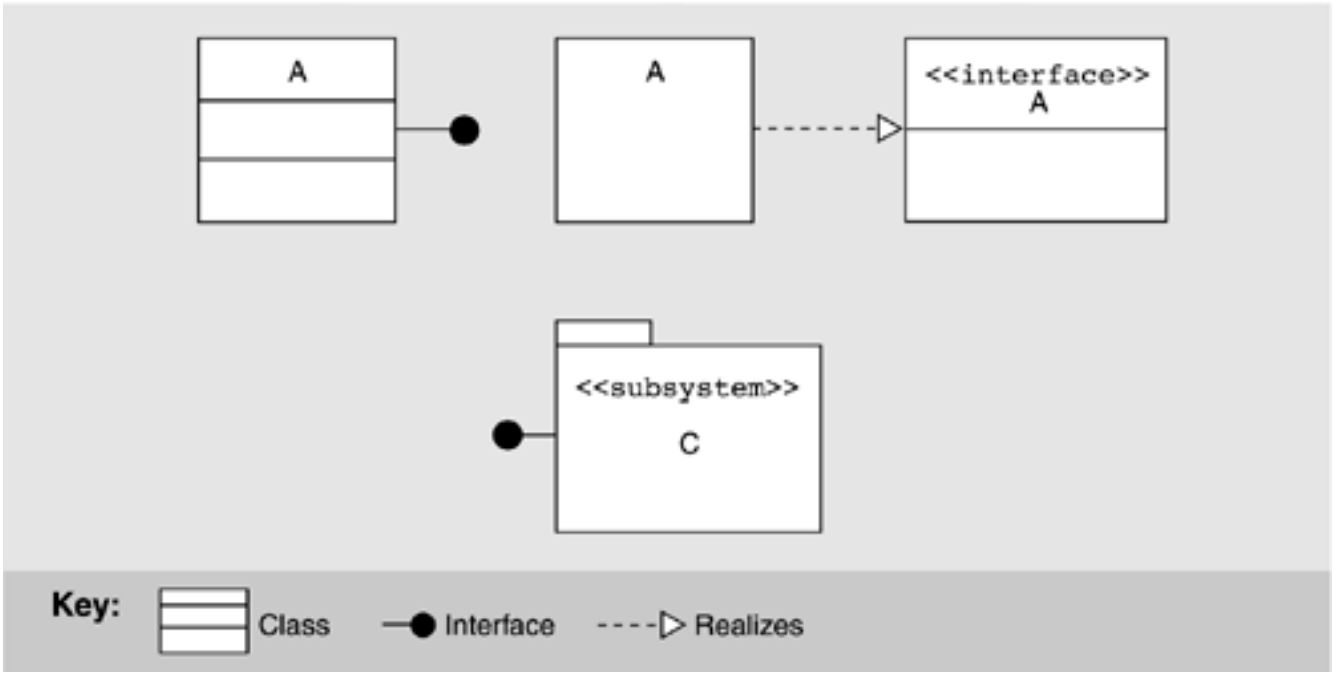
- Langages de description d'architecture (ADLs)
  - Sujet de recherche dans les années 1990s
  - Pas tellement utilisé en pratique: Rapide, Wright, UniCon, ACME, …
  - AADL a été adopté récemment comme un standard IEEE

Si vous utilisez UML…

- Ne pas se laisser *séduire* par l'expressivité des diagrammes UML.
  - Par exemple, un diagramme de classes UML est une notation qui illustre une vue de modules.
  - Ces diagrammes sont tellement versatiles qu'on peut même indiquer des informations propre à l'exécution des classes.
- Adopter une discipline pour utiliser les diagrammes UML.
  - C'est également un standard de description pour les programmeurs.

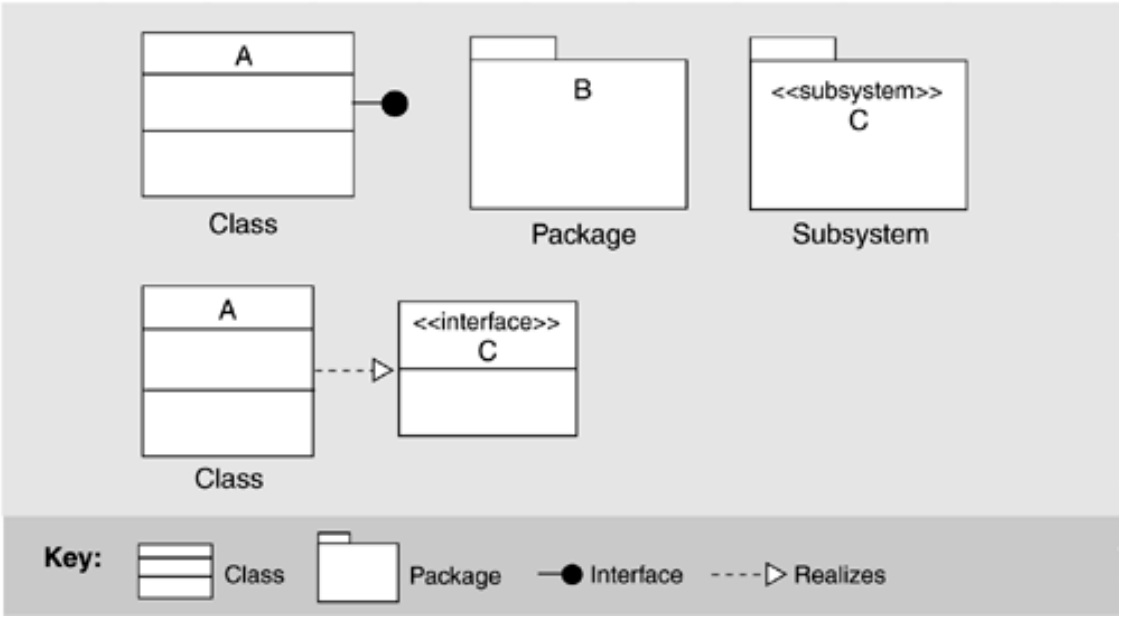
Utilisation d'UML

- Interfaces en UML (vue de module)



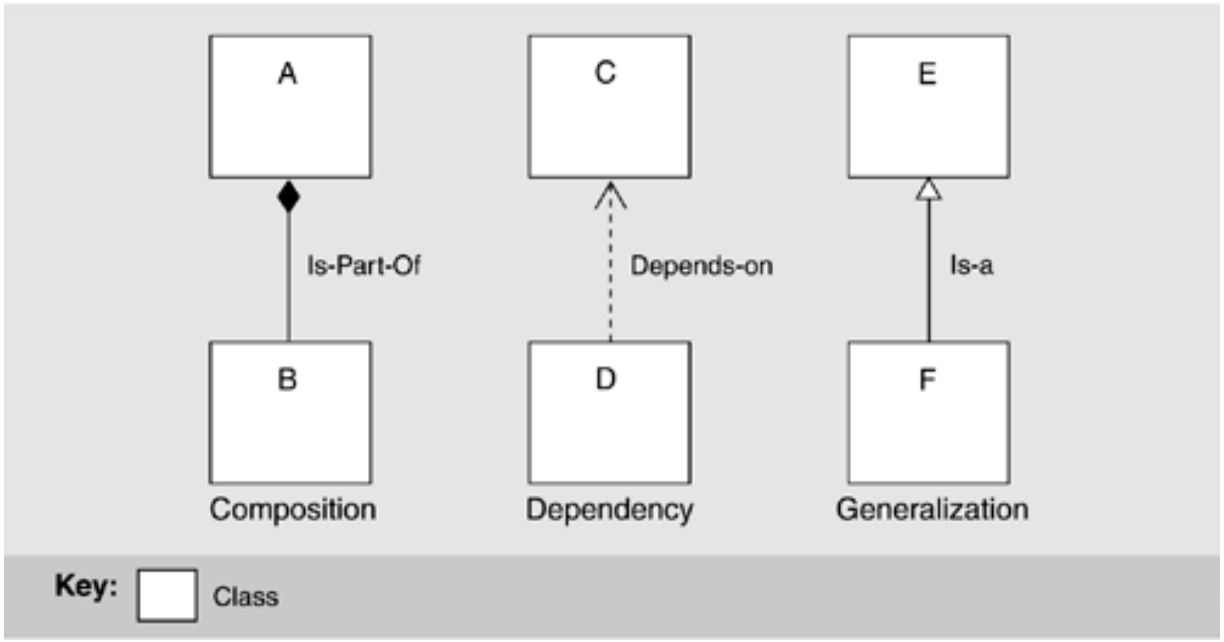
Utilisation d'UML

- Notations de modules en UML (vue de module)



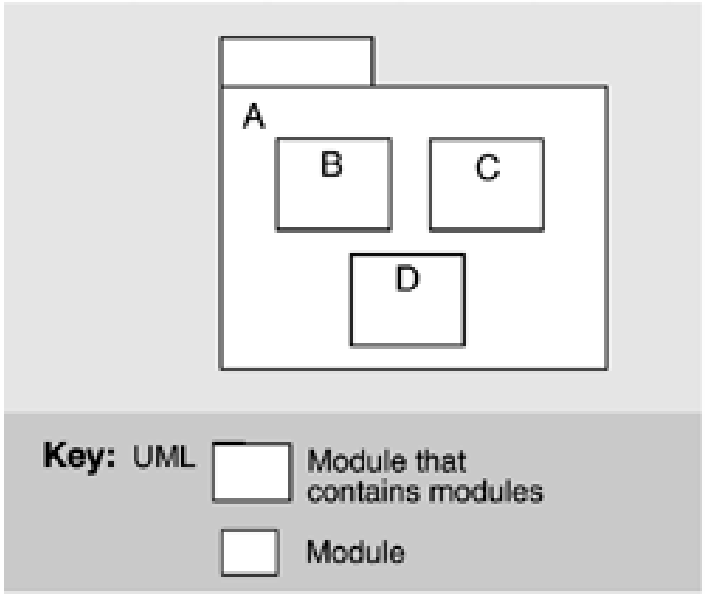
Utilisation d'UML

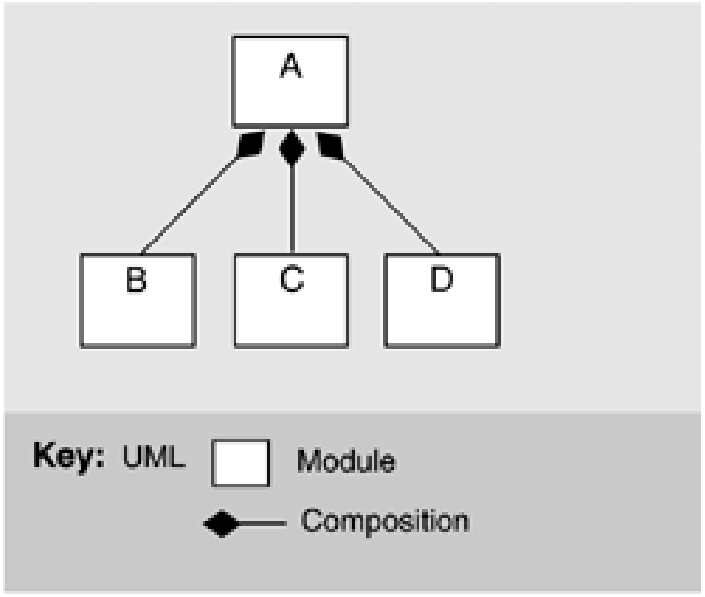
– Exemples de relations en UML (vue de module)



Utilisation d'UML

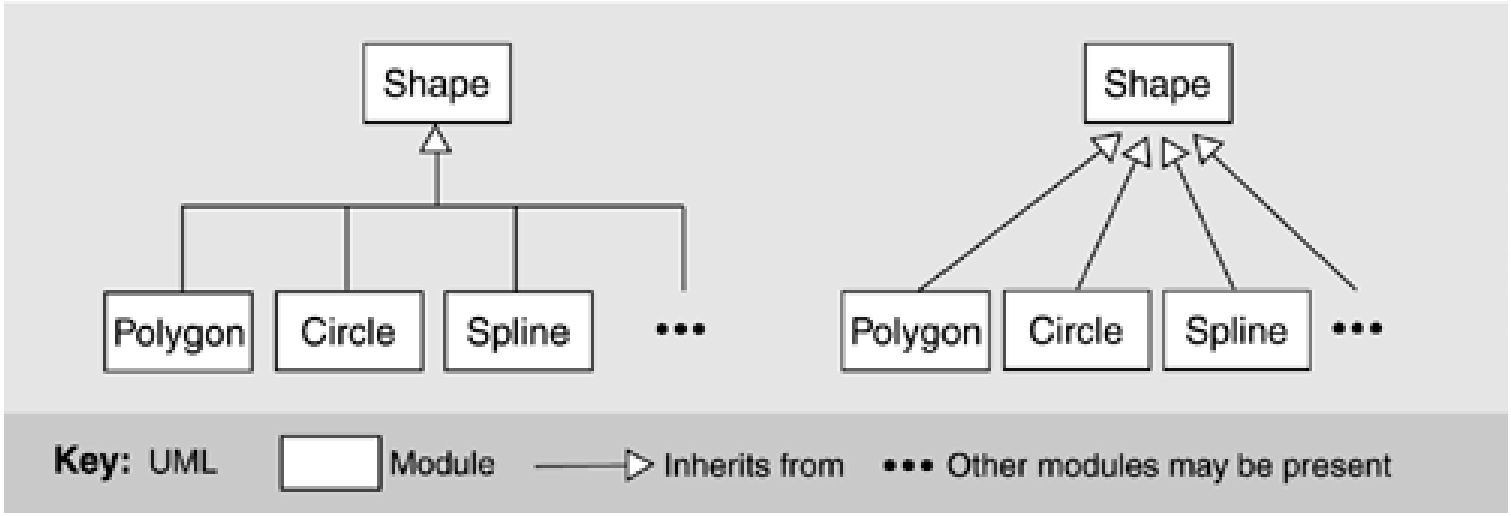
– Décomposition en UML (vue de module)





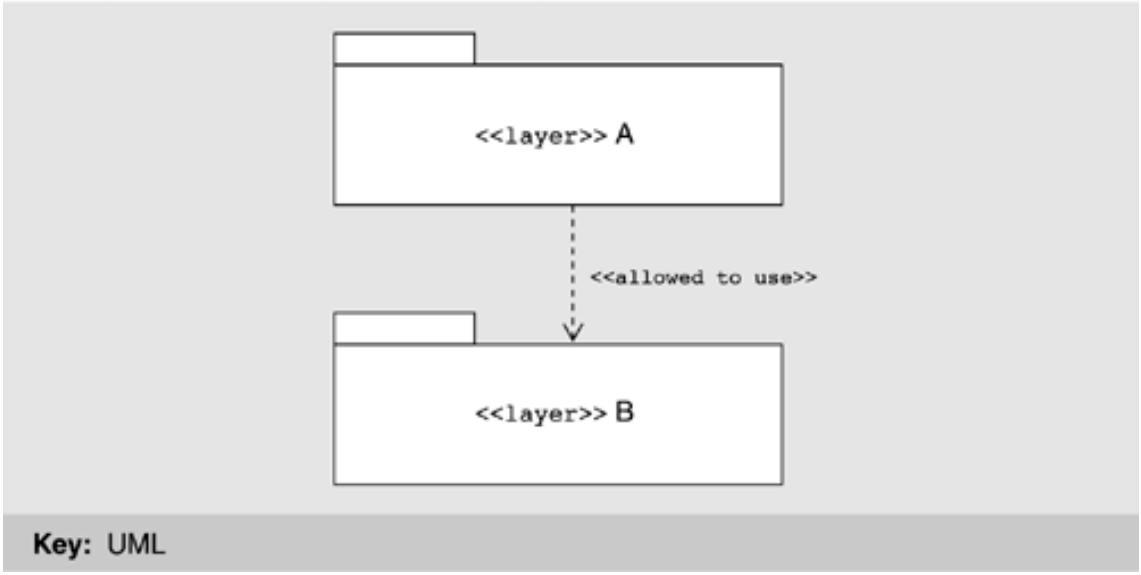
Utilisation d'UML

– Généralisation en UML (vue de module)



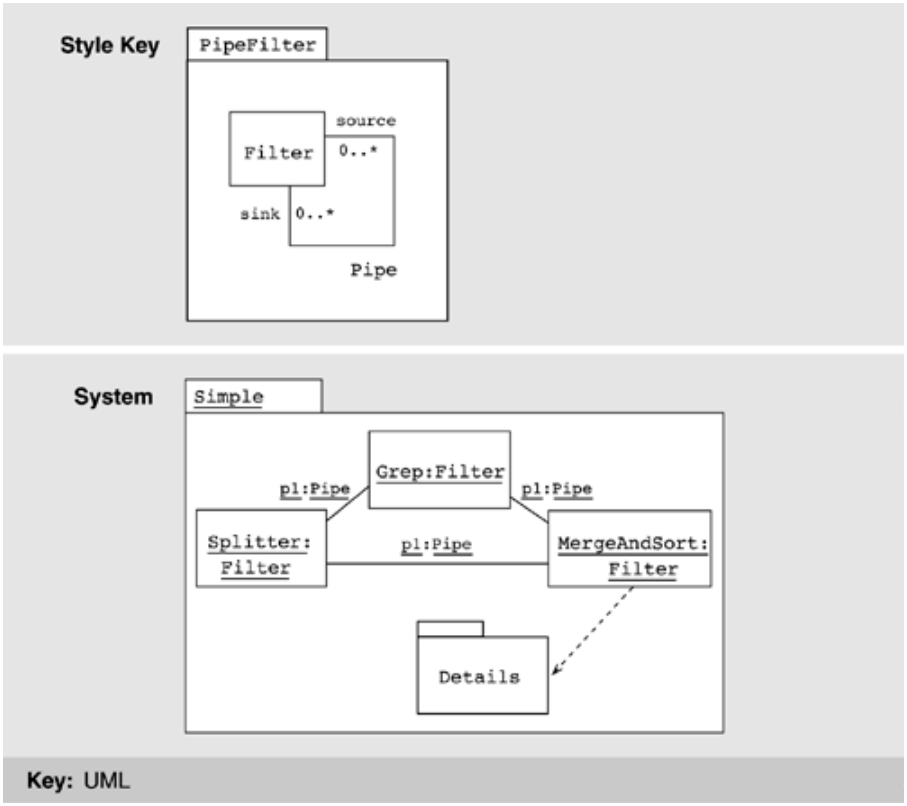
Utilisation d'UML

– Couches en UML – *layers* - (vue de module)



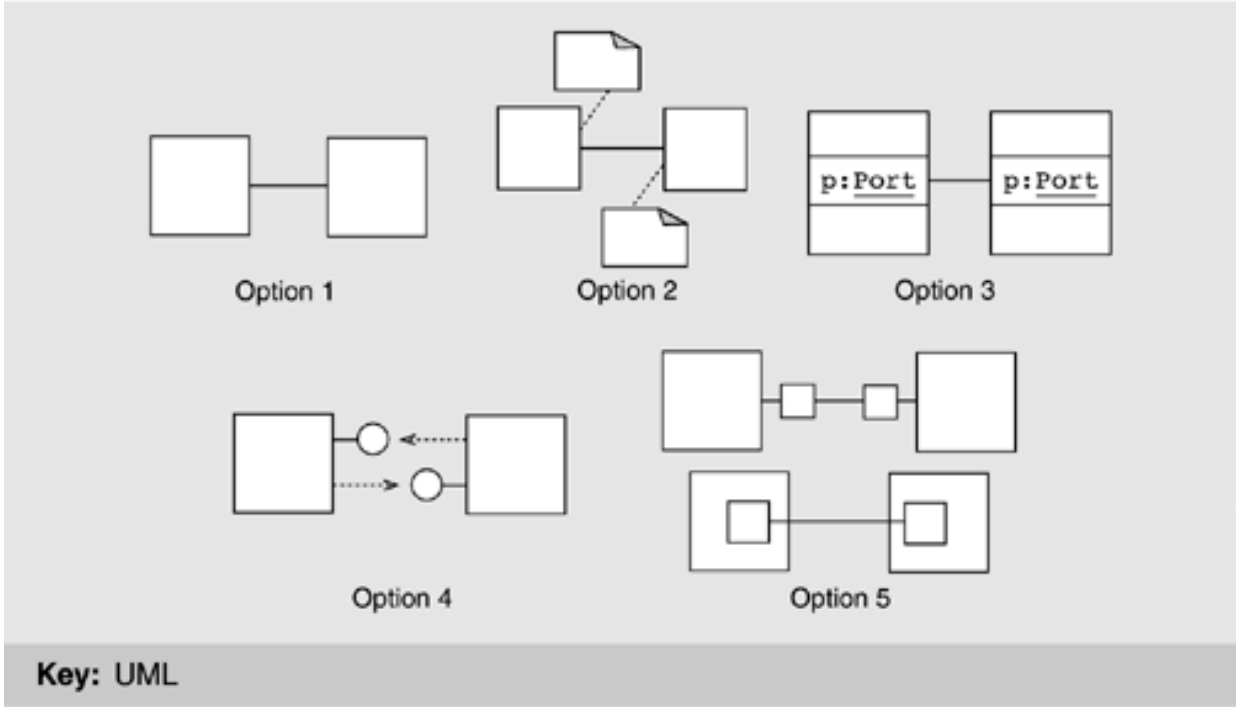
Utilisation d'UML

- Vues composants - connecteurs: exemple du Pipes & filters



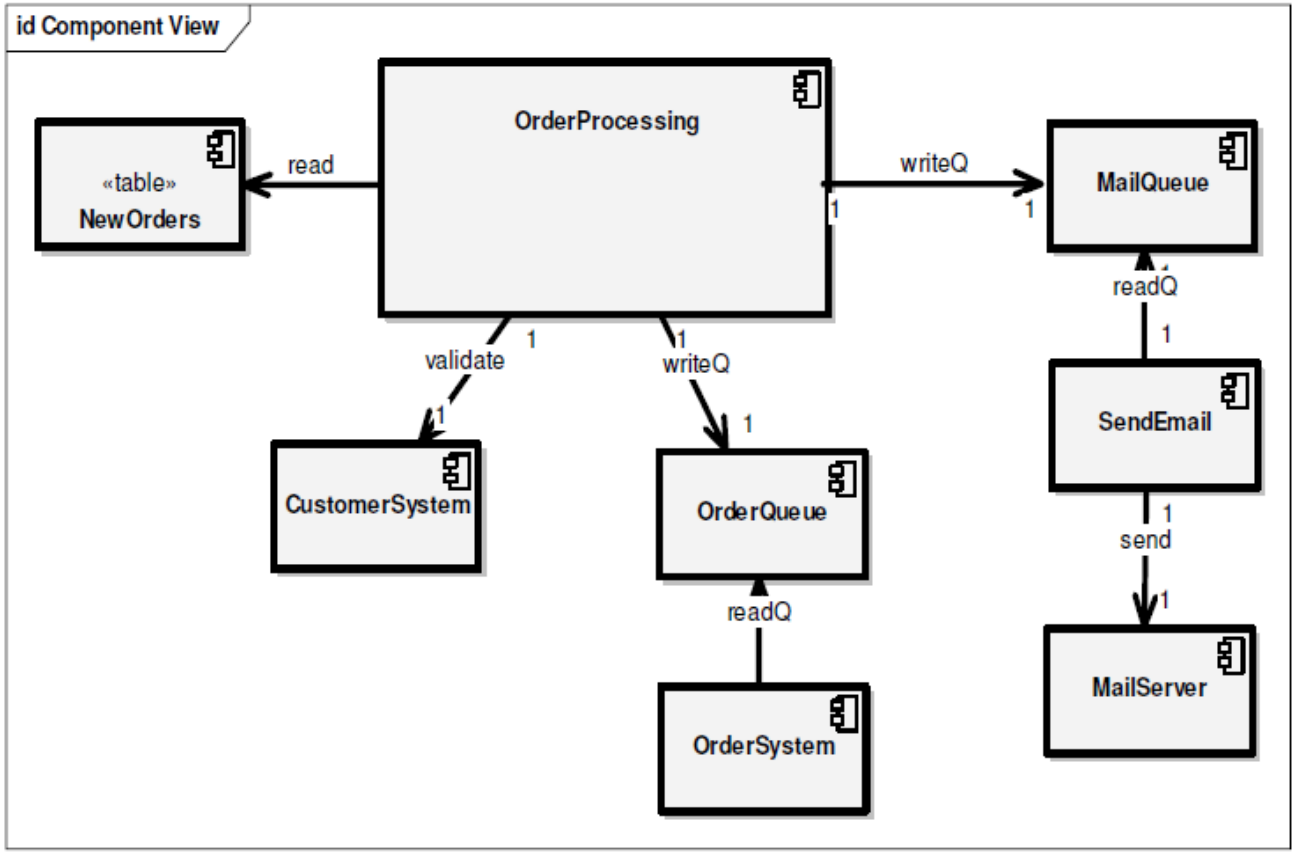
Utilisation d'UML

- Interfaces aux composants (vue composant-connecteur)



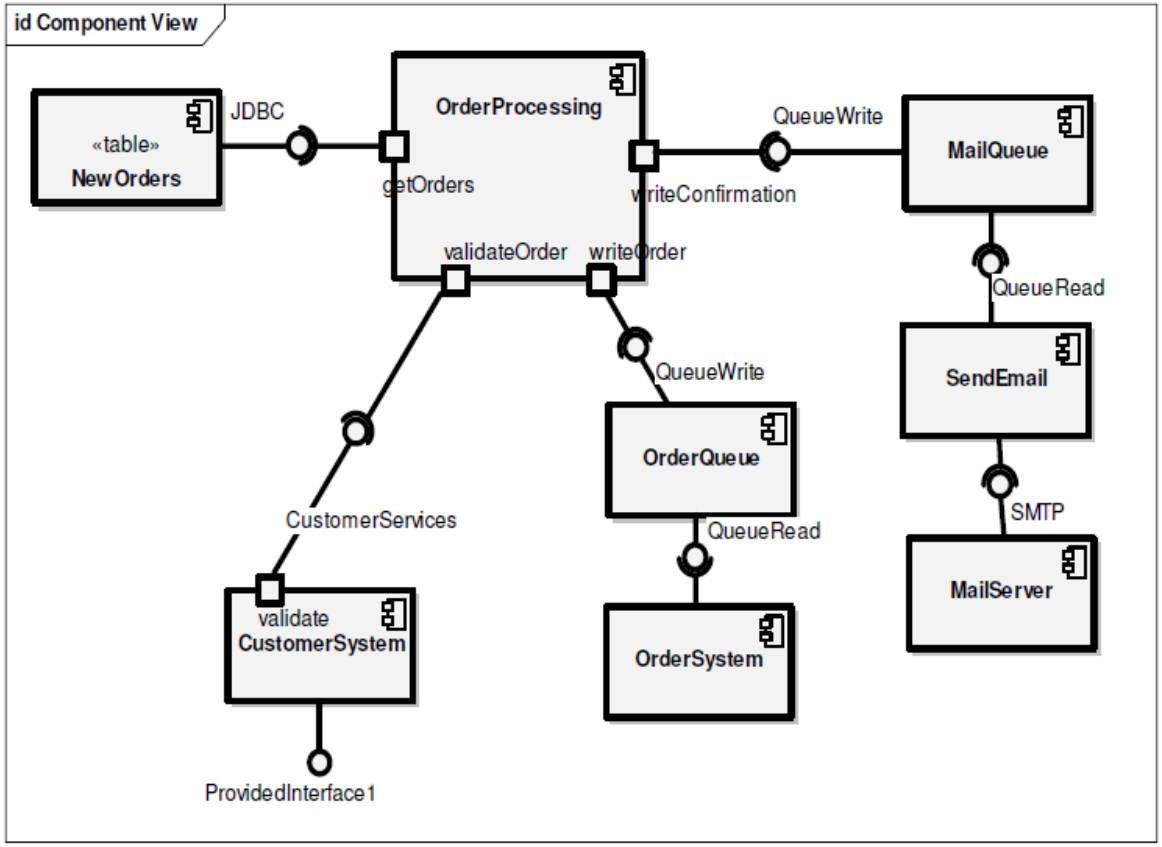
Utilisation d'UML

- Exemple de vue composant-connecteur



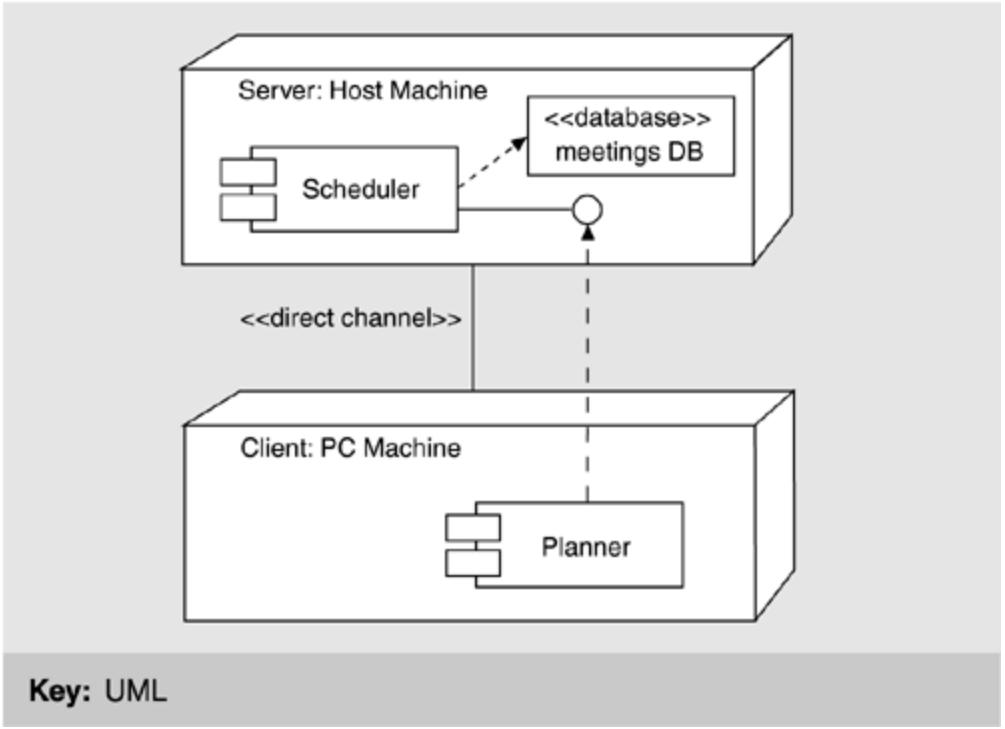
Utilisation d'UML

- Exemple de vue composant-connecteur



Utilisation d'UML

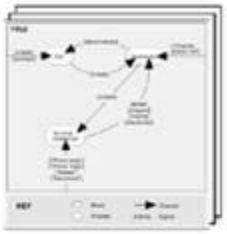
- Vue de déploiement (vue d'allocation)



Comment organiser l'information?

Views

Section 1. Primary Presentation of the View



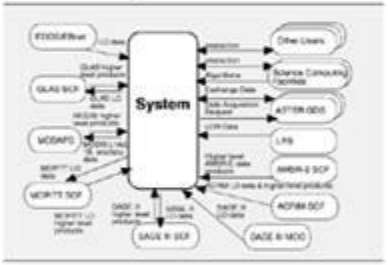
OR

Textual version of the primary presentation

Section 2. Element Catalog

Section 2.A Elements and their properties
Section 2.B Relations and their properties
Section 2.C Element interfaces
Section 2.D Element behavior

Section 3. Context Diagram



Section 4. Variability Guide

Section 5. Architecture Background

Section 5.A Design rationale
Section 5.B Analysis of results
Section 5.C Assumptions

Section 6. Glossary of Terms

Section 7. Other Information

Source: Adapted from [Clements 03].

Documentation d'une vue

- 1. Un présentation initiale
  - Habituellement graphique
  - Peut être textuelle – par ex. une table
  - Si graphique, inclure des explications de la notation (ou une référence aux explications)
  - Illustrer les éléments et leur relations
  - Illustrer premièrement les informations que l'on veut communiquer sur les vues
- Plusieurs fois, la représentation initiale est tout ce qui est fourni. Ce n'est pas suffisant.

Documentation d’une vue

- 2. Un catalogue d’éléments
  - Explique les éléments décrits par la représentation initiale
  - Listes des éléments et leur propriétés
  - Explique les relations, et les exceptions et ajouts aux relations présentées dans la représentation primaire
  - L’interfaces des éléments
- 3. Un diagramme de contexte
  - Illustre comment le système interagit avec son environnement.

Documentation d'une vue

- 4. Un guide de variabilité
  - Indique les mécanismes pour changer les éléments
- 5. Information contextuelle (*background*)
  - Motivation des décisions, incluant les options rejetées et les facteurs qui ont contraint la conception.
  - Les résultats d'analyse qui valident les décisions de conception.
  - Les hypothèses à propos de l'environnement et les besoins que le système tente de satisfaire

Documentation d’une vue

- 6. Autres informations
  - Spécifique au système et au projet.
  - Gestion de configuration, information sur la propriété.
  - Lien avec les besoins.
  - Pas vraiment architectural, mais utile en accompagnement.
- 7. Les vues reliées
  - Pointeurs vers des vues enfants ou parents

Format de documentation

- Un canevas Word est disponible sur le site du SEI

  http://www.sei.cmu.edu/architecture/arch_doc.html

- Contient différentes sections qui donnent le contexte autre que les vues dont:
  - Explication sur l'organisation du document et le choix des vues
  - Survol du système
  - Liens entre les vues
  - Index des relations, éléments et propriétés

Conclusion

- Documentation communique les choix de conception à différents intervenants (décideur, programmeur, usager…)
  - Sinon la documentation n'est pas utile
- S'appuie sur le principe de vues
- Plusieurs types de vues possibles
- Le choix des vues est effectué en fonction du projet