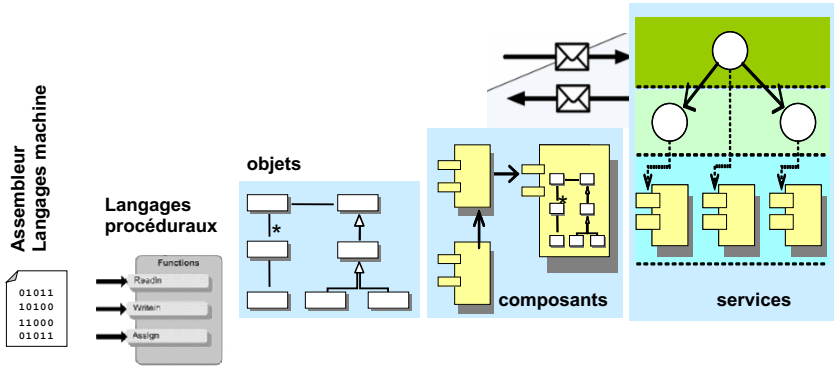
Ingénierie des Composants
de l'**objet** au **composant**



1

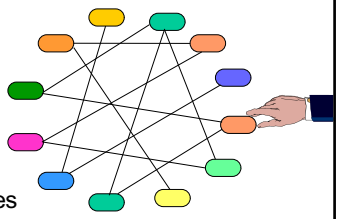
Objectifs du cours

- Introduction et définitions
- L'environnement du composant « industriel » : interfaces, connecteurs, conteneurs et structures d'accueil
- Architecture de composant et langage de description
- Exemples de modèles industriels de composants (monde java)
  - **OSGi**
  - Fractal
  - EJB
  - Etc.

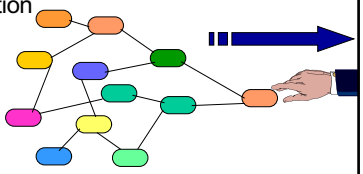
2

De l'objet … « programming in the small »

- Tout est à la charge du programmeur
  - Construction des différents modules
  - Définition des instances
  - Interconnections de modules



- Structure de l'application peu visible
  - Ensembles de fichiers source nécessaires
- Évolution et modification difficile
  - Changement du mode de communication
  - Évolution, ajout, suppressions de fonctionnalités
  - Modifications du placement
  - **Couplage fort**

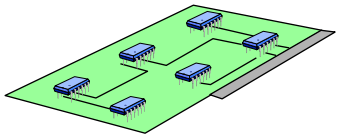


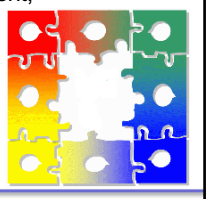
- Développement, génération des exécutables, déploiement
  - Pas ou peu d'outils pour les applications réparties

3

… au composant « programming in the large »

- Programmation constructive ou par composition
- Motivation : réutilisation du logiciel
  - Intégration de modules déjà existants
  - Construction d'applications réparties par assemblage de modules logiciels existants
  - Programmation à gros grain
- Approche descriptive de l'architecture de l'application à l'aide d'un langage déclaratif (ADL)
  - Modèle de construction des composants (interfaces, attributs, implémentation)
  - Description des interactions entre composants (connecteurs)
  - Description des variables d'environnements (placement, regroupement, sécurité, etc.)





4

De l'objet au composant, une évolution naturelle

- Paradigme objet et composants portent sur l'intégralité du développement de systèmes
  - De la spécification à l'implémentation
- Mais ils portent sur des aspects différents
  - Pas le même niveau d'abstraction
  - Pas la même granularité
  - Pas le même mode opératoire
- En général, on dira d'un composant qu'il a
  - Un plus haut niveau abstraction
  - Une meilleure encapsulation, protection, autonomie
  - Des moyens de communications plus explicites (ports, interfaces, connecteurs)
  - Une séparation « métier » technique
  - Une meilleure couverture du cycle de vie de l'objet « informatique »
    - conception, implémentation, *packaging, déploiement, exécution*
    - *Surtout vrai pour l'après « codage »*

5

De l'objet au composant, une évolution naturelle

- Les composants répondent à un problème d'ingénierie du système/intergiciel (middleware)
- En terme de génie logiciel, à la fin des années 90, des besoins spécifiques se sont fait ressentir avec l'usage des langages objets (on parle ici d'implémentation)
  - Configuration
  - Déploiement
  - Empaquetage (packaging)
  - Assemblage
  - Dynamicité
  - Gestion des interactions et des dépendances
- Le paradigme « Composant » dans son ensemble est une réponse à ces questions (au même titre que le paradigme « Aspect »)

6

Qu'est-ce qu'un composant ? (1/2)

- Une première définition d'un composant

> A component is a unit of composition with **contractually specified interfaces** and **context dependencies** only. A software component can be **deployed** independently and is subject to **composition** by third parties.
>
> Szyperski C., Component Software: Beyond Oriented-Object Programming, Addison-Wesley, 1998.

- Une seconde définition

> Un composant est un morceau de logiciel assez petit pour que l'on puisse le créer et le maintenir et assez grand pour que l'on puisse l'installer et en assurer le support. De plus il est doté d'interfaces standards pour pouvoir interopérer.
>
> Harris J. and Henderson A., « A Better Mythology for System Design », in Proceedings of CHI 99, Pittsburgh, USA, 1999.

7

Qu'est-ce qu'un composant ? (2/2)

- De manière pragmatique et usuelle, un composant est
  - Un module logiciel autonome pouvant être installé sur différentes plateformes
  - Qui exporte différents attributs, propriétés ou opérations
  - Qui peut être configuré
  - Qui est capable de s'auto-décrire
- Ainsi, il forme une brique de base configurable pour permettre la construction d'applications par composition
  - donc pas uniquement des applications réparties

8

Caractéristiques d'un composant

- Des différentes définitions existantes, on déduit qu'un composant doit, entre autres, être
  - Auto-descriptif — introspection, modification dynamique du comportement
  - Composable — connectable avec d'autres composants
  - Configurable — adaptable à l'environnement d'exécution
  - Réutilisable — il doit être une unité réutilisable même s'il faut le reconfigurer
  - Autonome — Il peut être déployé et exécuté indépendamment

9

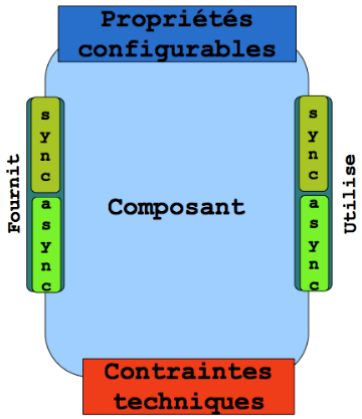
Systèmes à composants

- Quelques notions clés
  - Le composant (interfaces, attributs)
  - Les connecteurs
  - Les conteneurs
  - Les structures d'accueil

10

Un composant coopère…

- Un composant définit comment il coopère
  - Ce qu'il fournit : interfaces, opérations, propriétés
  - Ce qu'il utilise : composition et références aux autres composants
  - Potentiellement, la façon dont il coopère (synchrone, asynchrone)
- Un composant est configurable
- Un composant fonctionne sous certaines contraintes techniques
  - Middleware : placement, sécurité, transaction
  - Interne : cycle de vie, persistance
  - Implantation : OS, bibliothèques, versions



11

Représentation graphique d'un composant logiciel

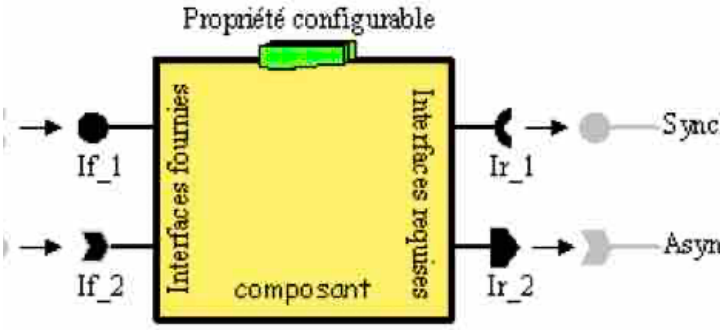
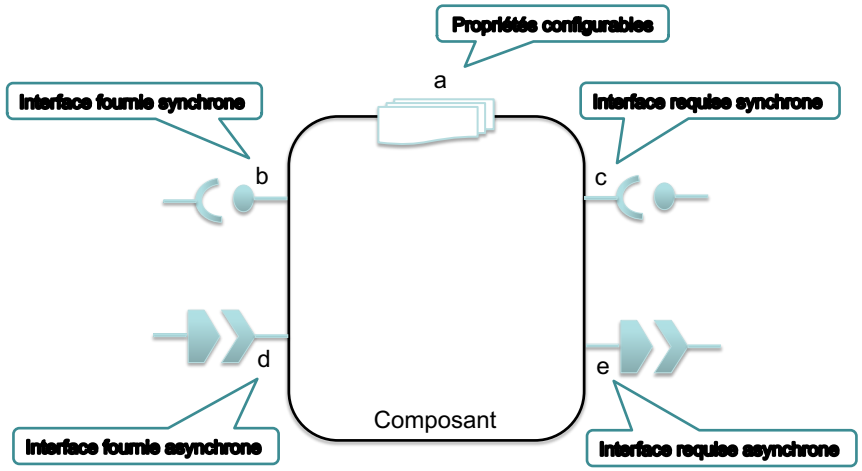


Figure 2.3 : Composant logiciel

12

Représentation graphique d'un composant logiciel

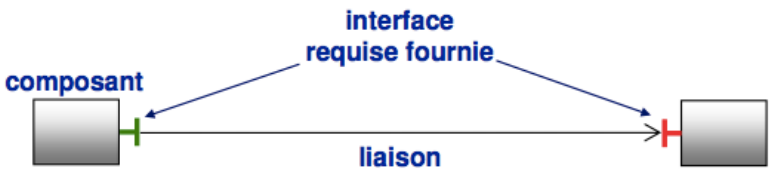


13

Un composant communique (1/2)

- Deux grandes catégories de modèles de composants
  - Différents par le modèle de communication
- Première catégorie

  I triptyque : composant, interface, liaison
  - un *composant* fourni et/ou requiert une ou plusieurs *interfaces*
  - une *liaison* est un chemin de communication entre une interface requise et une interface fournie
  - *Ex :* éditeur de texte (C1) requiert un service d'impression fournit par un composant (C2) décrit comment il doit être appelé pour imprimer



14

Un composant communique (2/2)

- Seconde grande catégorie de modèle de composants

  Un triptyque : composant, port, **connecteur**
  - un *composant* fourni et/ou requiert un ou plusieurs ports
  - un *connecteur* implémente un schéma de communication entre des composants (client/serveur, diffusion, etc.)
  - un *composant* est relié à un connecteur via un ou plusieurs ports

Ex : connecteur → gestionnaire de connexions aux imprimantes

- connecteur ≈ liaison avec comportement
- ➢ on peut considérer connecteur = composant (de communication)
- ➢ composant, interface, liaison

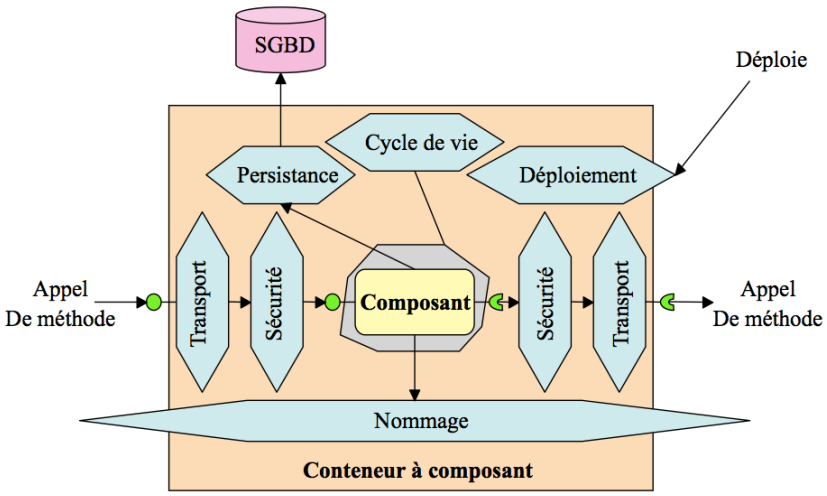
15

Un composant vit dans un environnement dédié (1/3)

- Conteneur
  - Un conteneur encapsule **un composant**
  - Prend en charge pour lui les services systèmes (nommage, persistance, transaction, sécurité)
  - Prend partiellement en charge les connecteurs
  - Techniquement utilise la délégation
- L'un des points forts des modèles à base de composants comparés aux modèles objets est la prise en charge des aspects techniques
  - Rôle d'un « middleware » en somme

16

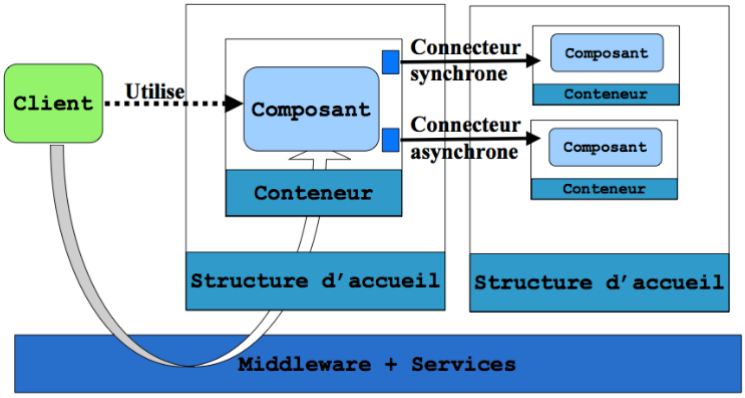
Un composant vit dans un environnement dédié (2/3)



17

Un composant vit dans un environnement dédié (3/3)

- Structures d'accueil
  - Les conteneurs sont eux-mêmes exécutés dans des structures d'accueil
  - Joue le rôle de médiateur entre les conteneurs et les services systèmes
  - Des + comme le téléchargement de code (navigateur)



18

Cycle de développement d'un composant

- La gestion de la vie d'une application est l'une des forces des modèles à composants
- Installer les composants
  - technologie de packaging
  - production des conteneurs
- Créer les composants
  - par des fabriques (maisons / « home »)
  - configuration des valeurs initiales
- Retrouver les composants
  - services de désignation (Nommage ou Vendeur) ou maisons
- Utiliser
  - invocation synchrone et événements (asynchrone)
- Introspection
  - découvrir leurs APIs (fonctionnelle)
  - découvrir les connecteurs (structurelle)

19

Construction par assemblage de composants

- Construction par assemblage plutôt que ingénierie de développement
  - réduire les besoins en compétences techniques
  - focaliser l'expertise sur les problèmes du domaine
- Besoin de décrire les architectures à l'aide de langages (ADL)
  - Capturer les composants
    - fonctionnalités et besoins
  - Capturer les connecteurs (liens)
    - composition et modes de communication
    - impédance entre composants → adaptateurs
- C'est le point faible des solutions industrielles !

20

Architecture de composants

- Comme le terme de composant, le terme d'architecture logicielle a une définition assez floue
- Une définition

> A software architecture of a program or computing system is the structure or **structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them. [Bass 98]**

- Un modèle d'architecture logicielle
  - Fournit un niveau d'abstraction élevé pour les différents utilisateurs
  - Permet de raisonner sur les propriétés fonctionnelles et non fonctionnelles du système
  - Facilite l'extension et la réutilisation des composants manipulés

Architecture de composants

- Description nécessaire à la bonne utilisation des composants
  - Comme le sont les modèles utilisés pour la conception de systèmes à base d'objets
- Inexistante dans les approches dîtes « industrielles » qui se concentrent sur l'activité d'implémentation
- Une architecture logicielle à base de composants peut être décrite à l'aide de
  - Modèles de composants UML
  - Langages de description d'architecture, ou Architecture Description Language (ADL)
    - SOFA, Unicon, Fractal, C2, Acme, Wright, ArchJava, …

Utilité des architectures

- Synthétise la complexité d'un système
  - Fournit des vues partielles du système
  - Mais aussi une vue complète
- Facilite l'installation et l'évolution de l'application
  - automatise le processus d'installation de l'application
    - car on dispose de la description de l'architecture
  - puis permet l'évolution de l'application
    - la modification des constituants d'un système
    - la modification des interconnexions entre ces constituants
    - la modification du placement

23

De nombreux modèles de composants

Enterprise Java Bean (JEE de Sun)
- ✓ Composants répartis en Java
- ✓ Basé sur RMI

Corba Component Model (CCM de l'OMG)
- ✓ Composants répartis multi-langage
- ✓ Basé sur Corba

Web component - servlet/jsp (JEE de Sun)
- ✓ Composants centralisé en Java
- ✓ Basé sur HTTP/HTML, spécialisé dans la production de code HTML

.NET (COM+, Microsoft)
- ✓ Composants répartis avec la CLR (VM de C#,J#…)
- ✓ S'intègre naturellement à windows

OSGi (OSGi Alliance)
- ✓ Composants centralisés en Java
- ✓ Spécialisé pour l'embarqué, minimal, largement utilisé comme moteur à plugins

Fractal (ObjectWeb)
- ✓ Architecture à composant multi-langage (Julia = implémentation en Java)
- ✓ Spécialisé dans les description d'architecture (ADL)

24

Attention à la multiplicité des modèles!

Conséquence de la multiplicité des modèles

- Multiplicité du vocabulaire
  - **composant**, bean, bundle
  - **interface/liaison**, port/connecteur, facette, puits, source
  - **requis/fourni**, client/serveur, export/import, service/référence
  - conteneur, membrane, services techniques, contrôleur
  - **framework**, serveur d'applications
- Grande variabilité dans les propriétés associées aux notions
  - **OSGi : bundle, package importé/exporté, service/référence**
  - Fractal : composant, interface, liaison, client/serveur
  - CCM : composant, facette, port, puits, source
  - UML 2 : composant, fragment, port, interface
- Un même terme peut avoir des acceptations différentes selon les modèles
  - pas toujours facile de définir les équivalences

En résumé, points clés sur les composants (1/2)

- Décomposition structurelle a peu près commune à tous les modèles de composants
  - Un composant est une unité de composition qui
    - décrit et/ou assure des fonctions spécifiques
    - Possède des interfaces de besoins (requises) et des interfaces de services (fournies)
    - Un contexte particulier d'exécution
  - Un composant peut être déployé indépendamment et composé avec d'autres composants
  - Les interactions ou connecteurs représentent les moyens de communication entre composants
    - Des plus simples : appels de méthodes, évènements (OSGI)
    - Au plus complexes : protocoles client-serveur ou lien SQL entre une BD et une application
  - Les interfaces sont des points de communication qui permettent d'interagir avec l'environnement (en particulier les autres composants)
  - Les propriétés représentent les informations sémantiques des composants
- Avec deux grandes catégories de modèles de composants (selon le rôle accordé au connecteur)

En résumé, points clés sur les composants (2/2)

- Quelques avantages des composants
  - Une meilleure réutilisation due à un couplage plus faible entre composants qu'entre objets
  - Ils fournissent le support de propriétés non fonctionnelles
  - Ils spécifient les besoins requis en plus des services fournis
- Mais vu que l'on va parler de modèles industriels, concrètement le paradigme composant est actuellement supporté au travers de langages de programmation objet où un composant =
  - Une ou plusieurs classes
  - Les interfaces du composants sont représentées par de interfaces (IDL, interface Java, etc.)
  - Les interactions entre composants peuvent être définies en terme d'association et d'appel d'opérations