Introduction au module de **GLA**
« *Génie Logiciel Avancé* »

M. Belguidoum

Université Mentouri de Constantine

Master2 Académique

Plan

Introduction au génie logiciel

Génie logiciel

un ensemble des **méthodes** de **techniques** et **d'outils** pour la production de logiciel de **qualité** avec maîtrise des **coûts** et **délais**.

Logiciel

un ensemble de **programmes** et de **documents** nécessaires à leur installation, utilisation, développement et maintenance

Qualités d'un logiciel

validité, fiabilité, extensibilité, réutilisabilité, compatibilité, efficacité, portabilité, intégrité, facilité, maintenabilité.

Comparaison avec le génie civil

- ***génie*** fait directement référence à génie civil = l'art de la construction.
- Construire un ouvrage architecturale $\neq$ poser briques
- Construire un bâtiment :

$\sum_{activités}$ **(conception architecturale, maçonnerie, plomberie, électricité,...)**$_{coordonnées}$

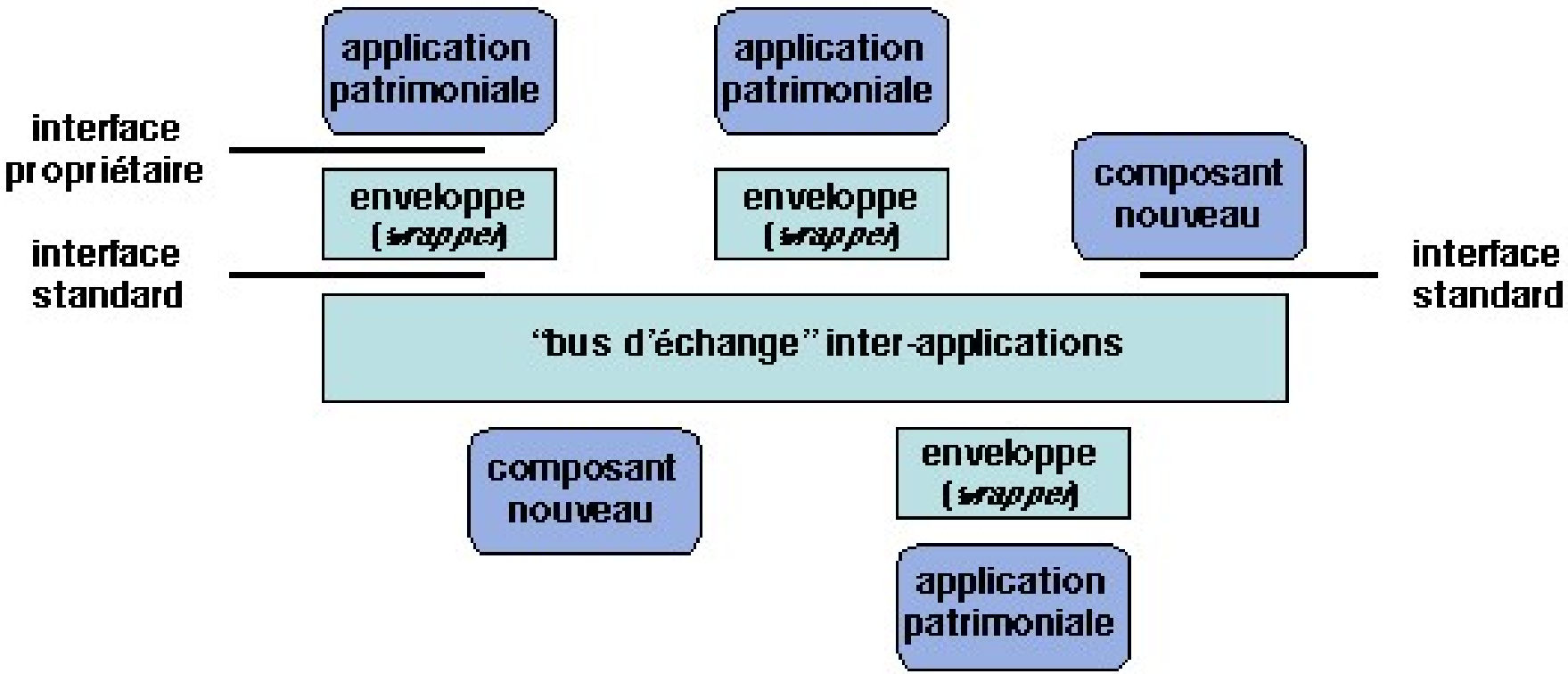
Objectifs du génie logiciel

- Produire des logiciels **adaptés aux besoins** des utilisateurs.
- **Réduire** le **coût** de la production et de la maintenance des logiciels en développant des composants **réutilisables**.
- **Augmenter** la **performance**, la **portabilité**, et la **fiabilité** des logiciels.
- **Augmenter la durée de vie** des logiciels.
- Produire des logiciels **efficaces** dans un **délai** raisonnable.

Un exemple de problématiques

- **Réutiliser un logiciel patrimonial**
  - utilise des outils propriétaires
  - nécessite des environnements spécifiques
  - doit être repris car le coût de la réécriture est important
- Résolution de ces problèmes
  - adopter une norme commune (non liée à un langage particulier) pour spécifier des **interfaces** et des **protocoles d'échange** pour la communication entre applications.
  - les protocoles sont réalisés par une couche logicielle qui fonctionne comme un **bus d'échanges** entre applications (**courtier** ou *broker*).
  - pour intégrer une application patrimoniale, il faut développer une couche logicielle (**enveloppe** ou *wrapper*) qui fait le **pont** entre l'interface originelle de l'application et une nouvelle interface conforme à la norme choisie.
- *Quelques exemples de courtiers* : CORBA, file de messages, les systèmes à publication et abonnement (*publish-subscribe*).

Intégration d'applications patrimoniales



Introduction aux intergiciels

Intergiciel (*Middleware*)

L'intergiciel est la **couche logicielle** située entre les couches basses (systèmes d'exploitation, protocoles de communication) et les couches hautes (applications) dans un système informatique.

Objectifs de l'intergiciel

- Développement, évolution, réutilisation des applications
- Portabilité des applications entre plates-formes
- Interoperabilité d'applications hétérogènes
- Faciliter la programmation répartie
- **Comment ?**
  - fournissant une interface ou API (Applications Programming Interface) de haut niveau aux applications
  - masquant l'hétérogénéité et la répartition des systèmes matériels et logiciels sous-jacents

Pré-requis du module

- Génie logiciel
- Programmation orientée objet
- Conception UML
- Anglais scientifique

Objectifs et contenu du module

- **Objectifs**
  - Ce module est la continuité des modules relatifs au génie logiciel
  - L'objectif de ce module de mettre en relief les concepts de base et les techniques applicables aux aspects avancés du middleware
  - Le module s'appuie sur quelques systèmes avancés issus de la recherche et de l'industrie (distribués par le consortium objectweb).
- **Contenu (40h)**
  1. Introduction aux intergiciels
  2. Le système de composant Fractal
     - Concepts de base
     - Les plates-formes
     - Fractal ADL
  3. Les Patterns
     - les patrons architecturaux
     - les patrons de conception (*design pattern*)
     - les patrons pour intergiciel à objets répartis
  4. Quelques exposés de recherche (présentés par les étudiants)

Références Bibliographiques

1. **Fractal** :
   - Szyperski, C. Component Software - Beyond Object-Oriented Programming. Addison-Wesley. 2nd ed. 2002.
   - Fractal objectweb : `http://fractal.ow2.org/`
2. **Les Design patterns**
   - Gamma E., Helm, R., Johnson, R., and Vlissides, J. Design Patterns : Elements of Reusable Object Oriented Software. Addison-Wesley (1994).
   - Larman C. *UML2 et les design patterns* (2005).
   - Metsker S. J et Wake W. Les design Patterns en Java : les 23 modèles de conception fondamentaux. CompusPress (2006)