Année universitaire : 2015-2016 **Date 4/2/2016**

Filière : Ingénieur **Durée : 1h 15mn**

Semestre : S5

Elément de module : Ingénierie à base de composant

Professeur : Mme B. EL ASRI

Consignes aux élèves ingénieurs : documents papiers autorisés

Questions

1. Qu'est ce que la description d'une architecture logicielle ? 2 points

2. Pourquoi développer une architecture logicielle à base de composants? 2 points

3. Le choix d'une architecture dépend des … ? 2 points

Exercice (14 points)

Vous devez assister une équipe sur un projet de partenariat industriel relatif à la conception d'une gamme de voitures intelligentes.
Une première partie du projet est relative à la définition de l'architecture du système. Ce système devra assurer (FT1) des fonctions bas niveau liées à la conduite-maîtrise des véhicules, (FT2) des fonctions de communication avec l'environnement extérieur et (FT3) des fonctions d'aide au conducteur, notamment celles relatives à l'interface homme-machine.
Pour des raisons de coût des produits finaux, on souhaite produire un ensemble d'assets réutilisables pour la gamme de ces produits.
En ce qui concerne le développement du système, de multiples composants existants développés par des tiers vont devoir être intégrés sans avoir à être modifiées. Notamment, on veut pouvoir intégrer des applications existantes de type réseaux sociaux ou cartes géographiques s'exécutant sur les systèmes d'exploitation patrimoniaux.

Q1. Décrire les contraintes de qualité liées à la mise en oeuvre de ce système ;

Q2. Décrire les défis liés à l'utilisation d'une architecture à base de composants et au besoin d'intégration transparente des composants existants.

Q3. Proposez une architecture en couche en précisant les styles d'architecture et les patrons utilisés pour répondre aux contraintes de qualités définies dans Q1. La solution développée devra pouvoir passer dynamiquement à l'échelle avec le nombre de véhicules déployés.

- Bien expliciter la couche « Business » en identifiant les composants métiers essentiels répondant aux FT1, FT2 et FT3.

Q4. Un ensemble de composants sera développés en interne : Proposer trois composants que vous développerez sous forme d'EJBs. Décrire et donner les interfaces de chaque composant.

AL BC 1

ALBC

Asr.

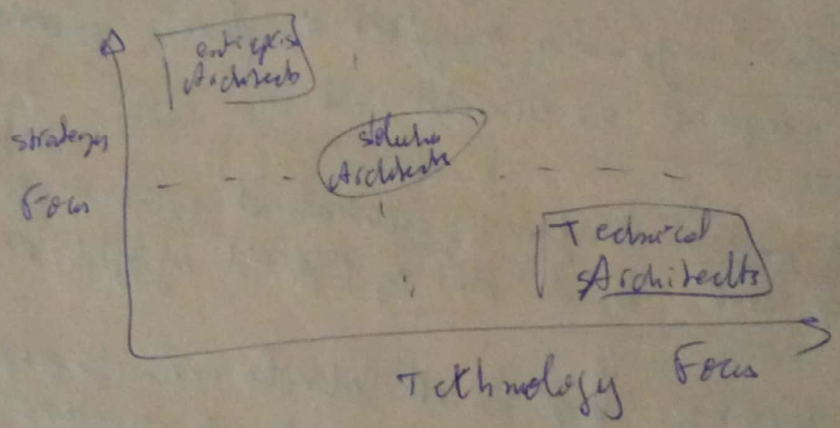
IBC : une discipline d'ingénierie qui consiste à assurer la disponibilité des composants appropriés et nécessaires pour fabriquer un produit conséquent

↳ consiste à { la maintenance / création / construction } de composants. (industrie des automobiles → l'aéronautique)

IBC requirement

① Architecture.

- Architecte c'est la personne qui a la vision d'agrandir les petits systèmes vers les grands syst. Il trouve la solution qui répond au mieux aux exigences de l'entreprise
- Aussi résoudre problématiques Architecture en IT



⊕ 3 types d'architecte

↳ Enterprise Architect : aide le PDG dans tout ce qui est stratégique

↳ Solution Architect : Mise en œuvre de la méthode

↳ Technical Architect : spécialiste de la technologie expérimentée, tendances de marché, choix des solutions technologiques

⊙ Les Compétences d'un Architecte SKILLS

⊕ Communication ⊕ Stakeholder Relationship Management
⊕ Listening skills

⊕ Conceptual thinking ⊕ Technical Acumen

⊕ **Architecture logicielle** : Une spécification abstraite du système (?) : structure des composants fonctionnels décrit par l'interaction de leur composant et de leurs interfaces et de leur assemblage

élément ABC :
- Données
- connecteurs
- composants
- contraintes, propriétés

⊕ **Components** : is a unit of software that performs some function at run time (objects, filters, programs)

→ **Connectors** : abstract mechanism that mediates communication, coordination, cooperation among components.

→ **Data** : information transferred from a component, or received by a component via connector

→ **Configurations** : structure of architectural relationships among comp. conn. Data during a period of system run time.

→ **Properties** : propriétés architecturales ∑ des propr. qui décrivent la sélection ou l'agencement des composants, des connecteurs et des données au sein du système

→ **Views** : ∑ problèmes : issues ; temporal issues, security

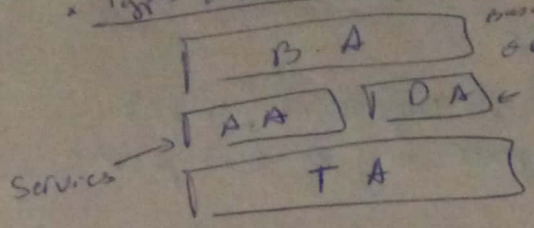
⊕ **Style d'architecture** : ∑ coordonnée de contraintes ~~sur les relations~~ autorisées entre les élts de une archi qui conforme à ce style

⊕ **Entreprise Architecture** : Un cadre de référence proposant
- ∩∩ des concepts
- Règles
- Archi fonctionnelle
- Démarche méthodologique

↳ ? pourquoi → Aligner TI avec les besoins d'affaires
→ promouvoir la réutilisation et l'intégration
→ Mettre en œuvre la planification et la gouvernance des TI à l'échelle de l'entreprise

Scanned by CamScanner

* Types d'architectures



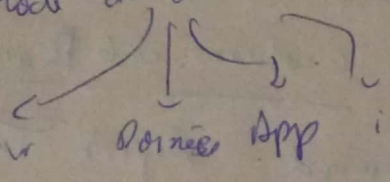
⊕ Archi Frame Work :

→ Un outil qui peut être utilisé pour le dev d'un large éventail d'architectures différentes { doit contenir Σ d'outils, de fournir un vocabulaire commun } { inclure une liste des normes recommandées et des produits conformes qui peuvent être utilisés pour mettre en œuvre les blocs de construction }.

⊕ TOGAF 9 : The open Group Architecture Framework.

Cadre de référence générique pour le développement d'architectures afin de réaliser différents besoins d'affaires.

ADM : method de dev de l'archi



EA → Business Arch: tecture : → S.B Process Model
→ S.B Functions
→ S.B entities

⊕ What can be shared

→ Organization Architecture → S.O Entities : Σ Actors, structure
→ S.O Functions : Security & Activity
→ Decomposition of Process & Activity domain

→ IT Architecture → S. Data
→ Dev. Architecture : languages, Dev Tools
→ Operation Architecture

schéma

⊕ DoDAF : Department of Defense Architecture Framework

Cadre d'architecture (Framework).

Permet de décrire l'architecture d'un système complexe s'appuie sur le langage UML pour présenter la structure d'un système sous forme Visuelle, Textuelle, Tabulaire

→ les perspectives sont "Produits"

DoDAF ——> classe 3 Vues

→ OV Vue opérationnelle : focalise sur les comportements, les fct qui décrivent le Σ des aspects mission Entreprise

Vue Syst → SV décrit la structure interne, les ressources des systèmes et leurs relations avec la VO

Vue Tech → TV : définit les standards et les règles qui régissent l'organisation des systèmes décrits par la SV. leurs états et les encours et à venir

All Views → AV : fournissent Σ des infos sur Σ de la description de l'architecture d'un environnement opérationnel

④ Zachman :

→ Framework Archi d'entre., c'est une matrice représentant le SI d'une entreprise à travers 6 colonnes et 6 Rangs (Architectes)

Avantage ? Quand Aspect Temporel

Scheme

⑤ Design Patterns

provides a description of a problem and the essence of its solution

→ Patron d'architecture : x DP. MVP (MVC)

Patron de conception

→ DAO, DTO

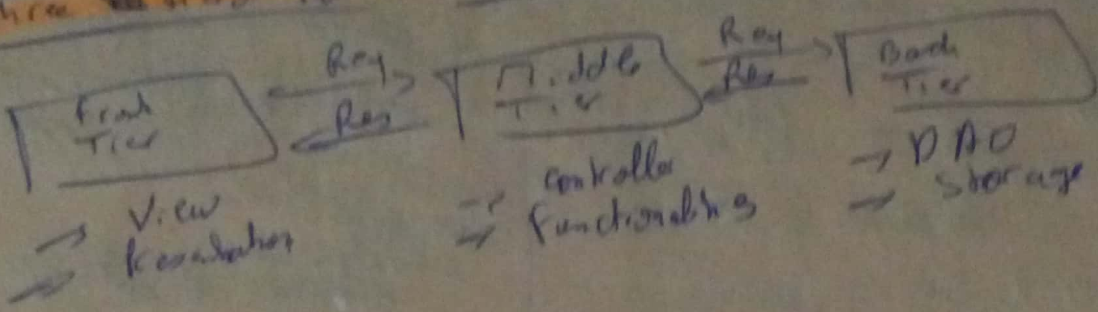
→ Identity Map

Déf un modèle de référence qui sert à source d'inspiration lors de la conception de l'architecture d'un sys, Logiciel informatique en sous elt simples.

ALM 2

AL BC

Three tiered Pattern



L'intégration des Patterns

- 2 Méthodes : Refinement , Combinaison
  - Refinement : 1 Patron affine la structure d'un autre
  - Combinaison : 2..n Patrons forment une structure plus vaste & complexe

afoor Design

- Les DP se complètent vers Design Complet et Équilibré

CH2 ALBC D Reuse

Dev :

① Fait Maison : Dev à zero :
+ adopté parfaitement aux besoins clients
+ profite des pratiques des formations de solutions
- couteux
- maintenance
- Risque d'interopérabilité

② Sous traitance = Externalisation : (Achats des outils standards / besoins clients)
+ Prix fixe
+ interopérabilité, maintenabilité … garanties par le vendeur de solution
- Réorganisation du processus métiers
- Difficile en adaptation aux nouveaux besoins

⚠ Software REUSE = Réutilisation Logicielle
→ l'utilisation d'un asset comme solution de plusieurs problèmes

?? ASSET : { code source, conception, spécifications, serie de test, ⋮ } qui est conçues pour être utilisé dans plusieurs Contexte

② 2 Types :
1/ Réutilisation Planifiée : on fait dev des composants logiciels dans le but de les utiliser dans plusieurs solut
2/ " Accidentale : suivit sans attention et qui implique à rendre un composant réutilisable à l'improviste

ex Reuse libraries

→ Les benefits of Reuse
① fiabilité, Réduction de Risque, Dev accéléré, Deploiement efficace, coût économique

→ Les Prérequis
1/ La Recherche des composants réutilisable qui correspondent le mieux
2/ Les composants réutilisés doivent être bien documenté pour la maintenance et l'adaptation

→ Problème de la Reuse
- Manque de support et d'outils
- coût de maintenance élevé
- Coût d'adaptation des composants réutilisables

Def Composants : un morceau logiciel assez petit pour l'on puisse 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 par pouvoir interopérer

composant	Objet
→ entité indépendante de déploiement	→ Unité d'instanciation
→ N'a pas d'état observable (non distinguable)	→ Encapsule son état et son comportement
→ Unité par la composition par des tiers.	→ Peut avoir un état observable de l'ext.

- Composant a des propriétés :

① Nom ()

- Composant doit avoir :
  - ① Unité de composition
  - ② connecteurs : (Relation entre composants)
  - ③ interfaces (pts de communication)
  - ④ Propriétés (les infos sémantiques de composants et de leurs interactions)
  - ⑤ Contraintes / contrats : critères sur lesquels un modèle d'archi reste valide durant sa durée de vie (prise en compte l'évolution des composants logiciels)
  - ⑥ Architecture
  - ⑦ Composition / Assemblage

→ les moyens par lesquels les composants sont connectés
3 types d'interfaces → I : Notion configuration
J / I : maintenance

→ CBSE : component-based Software Engineering : Enhances the reuse process by offering support for various appropriate component.

→ Framework : Sub-System design containing a collection of abstract and concrete classes along with interfaces between. Each class

CH3 : SPL : Software Product Line

→ Chaîne de production en industrie d'automobile

SPL : ensemble de produits logiciels qui partagent et gèrent un ensemble de caractéristiques satisfaisant les besoins spécifiques d'un segment de marché particulier et qui sont développés à partir d'un ensemble d'actifs communs de manière prescrite.

• Intérêts :
① Le coût des codes
② Augmentation de la qualité
③ Réduction du temps de dév

• Limitations :
① Mode d'équipes de projet
② Une plateforme unique
③ Inexpressivité des domaines

SPL → Domain Platform → Domain Engineers
→ Descriptive Applications → Application Engineers

SPL → SPL Engineering

Exam Classi :

① L'architecture logicielle décrit d'une manière symbolique et schématique les différents éléments d'un ou plusieurs systèmes informatiques, leurs interactions et leurs interrelations.

② Pourquoi ? → Faciliter la réutilisation et la réutilisabilité
→ Pour permettre aux développeurs de travailler sur des parties individuelles du système en isolation (principe de la composition)
→ Evolution + Scalabilité

③ Le choix d'une architecture dépend :
→ Dépend des exigences fonctionnelles et non fonctionnelles du logiciel
→ Choix favorisant la stabilité (
→ influencé par certains modèles connus de décomposition en composants et de mode d'interaction.

Q1 : ① l'implémentation de la chaine de production dans une architecture industrielle purement avec l'info embarquée

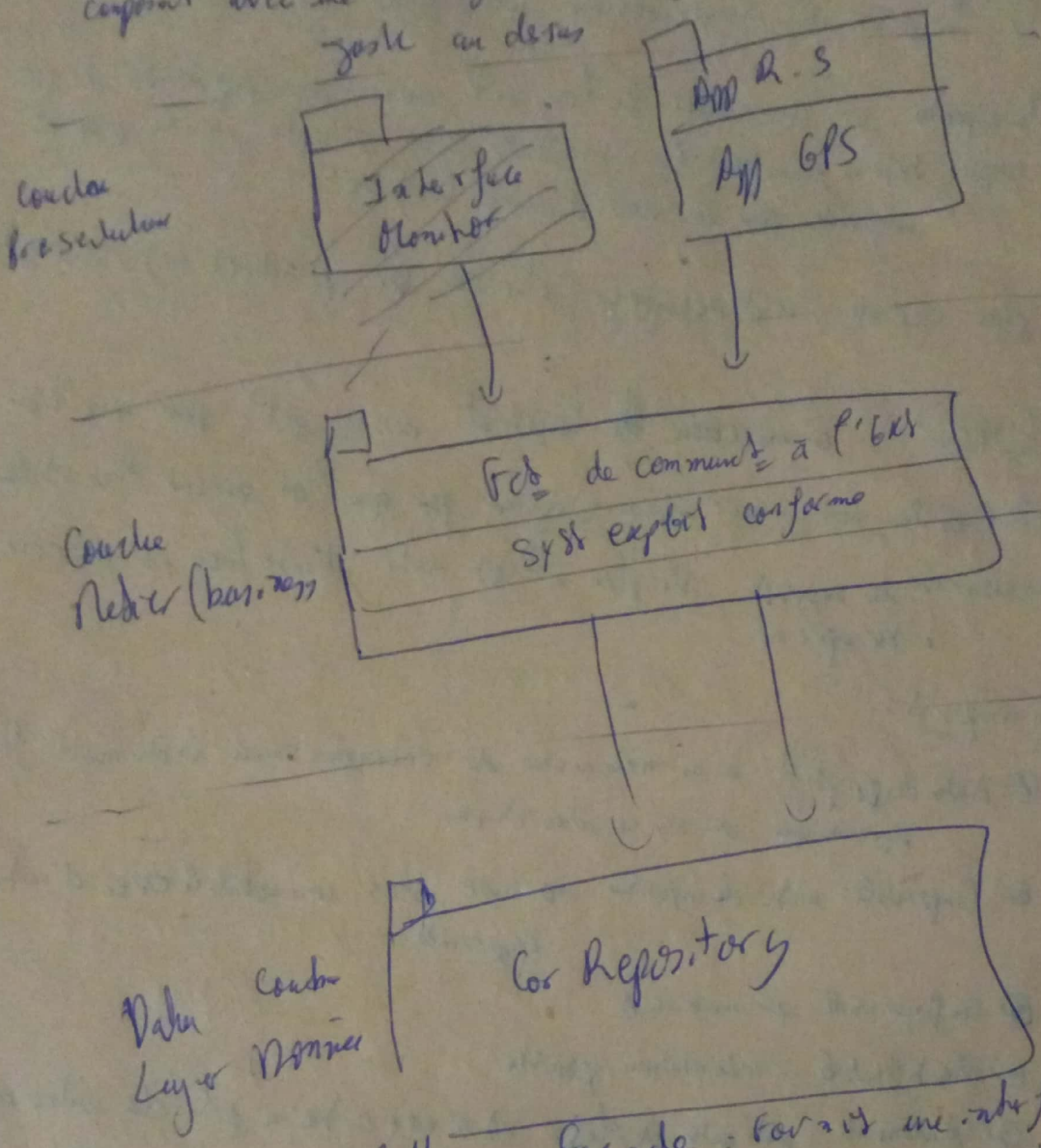
② Un système d'exploitation conformé pour que les app puissent exécuté sans problème

Q2 ① interopérabilité un défi pour que les nouveaux composants puissent → avec les existants

② Définition des modèles de composants avec des standards et de fournir des implémentations à des modèles de composants associés pour permettre à des composants ~~à base de comp~~ d'être conçus d'être mis en application, et d'être déployés

Q3

Architecture multi-couche : Dans laquelle chaque couche n'est en contact avec une interface définie utilisée par la couche juste en dessus



Design Pattern : Façade : Fournit une interface simplifiée à un gros ensemble de code, Flyweight : réduit le coût de construction/manipulation d'un grand nombre d'objets similaires

Q4. Composant de Géolocation, composant de stationnement automatique, composant d'air bag ...

```
public interface ... extends EJBObject {
    ... signatures ...
}
```