R5.A.08 :
Qualité de développement

Chouki TIBERMACINE
Chouki.Tibermacine@univ-ubs.fr

Plan prévisionnel de la ressource

8 séances de CM de 45 min + 7 séances de TP d'1h30

1. Intro aux architectures logicielles
2. Documentation d'architectures en UML
3. Styles et patrons d'architectures
4. Architectures à microservices
5. Design & Implem de frameworks
6. Patrons de conception : Façade, Bridge, MVC et MVVM
7. Patrons de conception : Builder, Proxy et Visitor
8. Autres patrons

Plan prévisionnel de la ressource

8 séances de CM de 45 min + 7 séances de TP d'1h30

1. Intro aux architectures logicielles
2. Documentation d'architectures en UML
3. Styles et patrons d'architectures
4. Architectures à microservices
5. Design & Implem de frameworks
6. Patrons de conception : Façade, Bridge, MVC et MVVM
7. Patrons de conception : Builder, Proxy et Visitor
8. Autres patrons

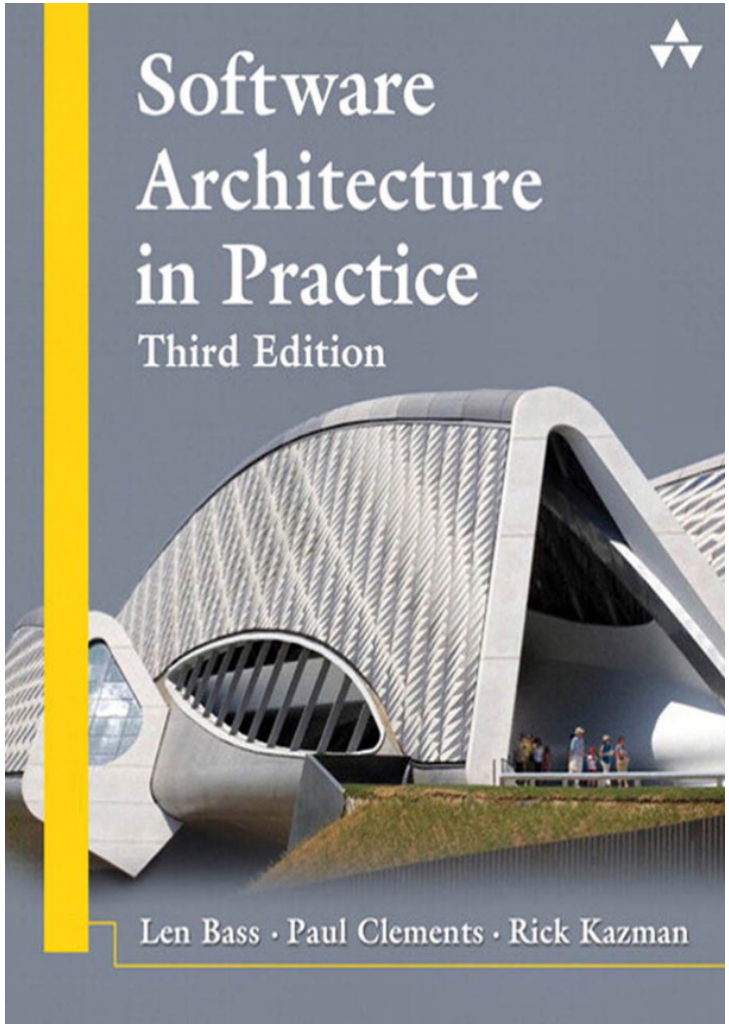
Évaluation de la ressource

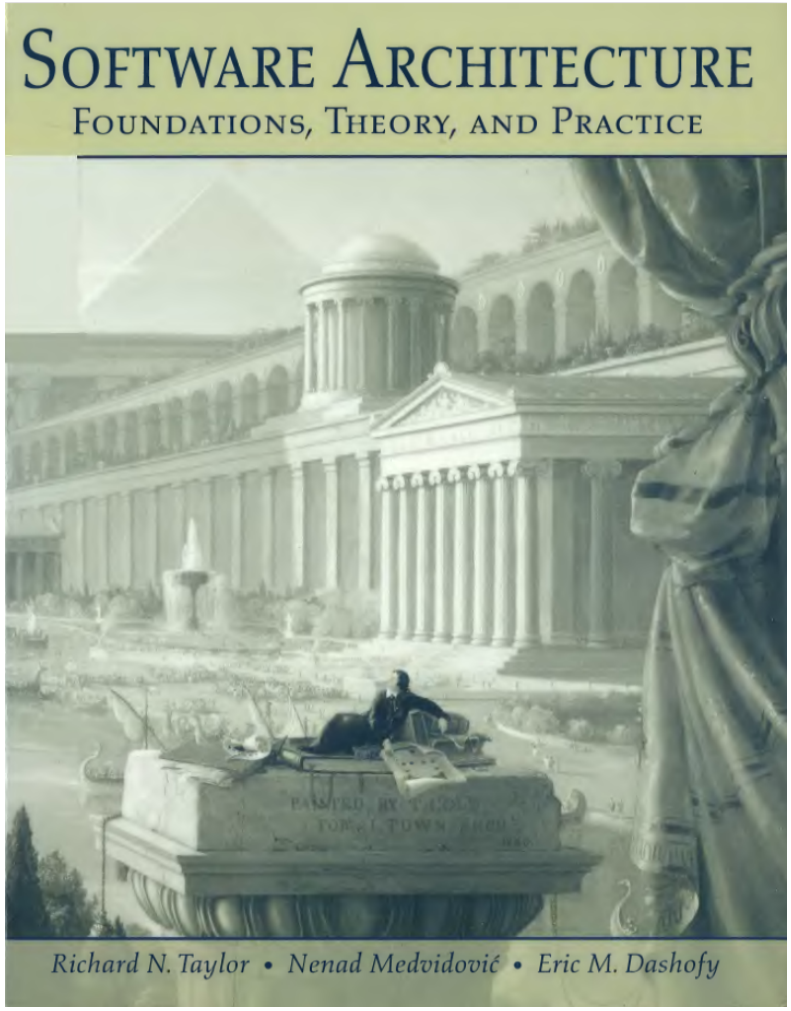
Contrôle continu :

1. Deux quiz :
   - après le 4ème CM
   - après le 8ème
2. 1 TP choisi aléatoirement (tous les TP doivent être rendus)

Note finale = moyenne des deux notes à coef égal

Livres de référence





Qu'est-ce qu'une architecture logicielle ?

Définition

L'ensemble des **structures** nécessaires au **raisonnement** sur un système

- Elle inclut :
  - les composants du système
  - les propriétés visibles de ces composants
  - les dépendances entre ces composants

Autre définition

L'ensemble des décisions "importantes" (*major*) et "précoces" (*early*) de conception

Définition moins généralisable, car on ne peut pas tout anticiper

Quelles structures ?

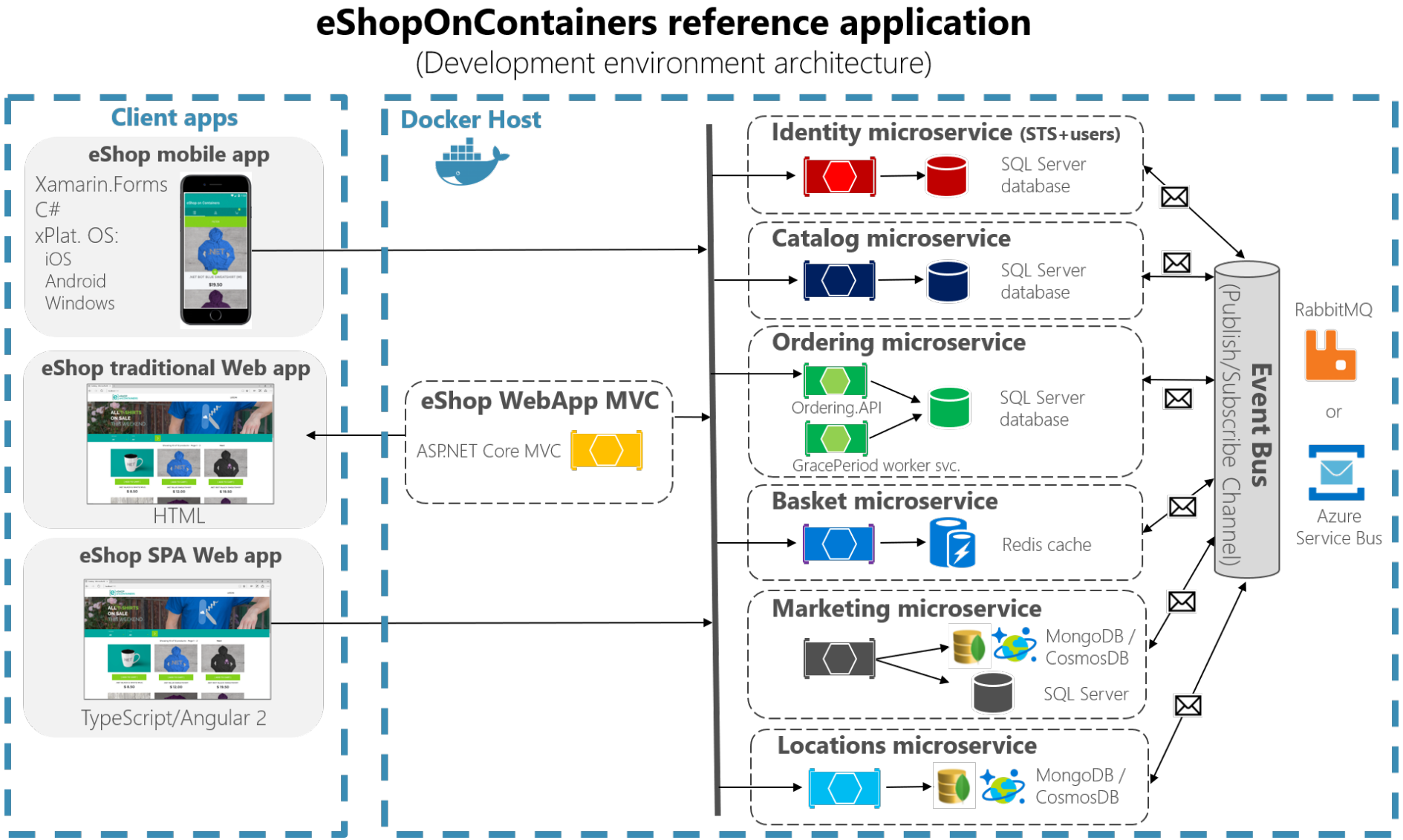
1. Structure statique d'implémentation : composants = **modules** à développer et à répartir sur les membres de l'équipe-projet
2. Structure dynamique : composants = éléments qui interagissent à l'exécution (services, ensemble d'objets, connecteurs, ...)
3. Structure d'allocation : composants = unités de déploiement sur des conteneurs, serveurs Web, d'application ou de bases de données, PaaS (*Platform as a Service*) sur le Cloud, ...

Une structure d'un système est architecturale si elle permet de raisonner sur les propriétés du système

Quel raisonnement sur le système ?

- Raisonnement sur une propriété (ou un **attribut**) du système, qui est important pour une partie-prenante du projet (un *stakeholder*)
- Cela inclut :
  - la fonctionnalité d'un système
  - la disponibilité d'un système face à une charge soutenue
  - la facilité à faire des changements spécifiques sur le système
  - la réactivité du système aux requêtes des utilisateurs
  - … et plusieurs autres **attributs de qualité**
- L'architecture est une abstraction qui réduit la complexité et simplifie le raisonnement sur le système, en masquant les détails inutiles (non-publics/non-visibles)

Exemple d'une architecture



Une architecture n'est pas que structurelle

- Elle peut inclure le comportement inter-éléments qui composent la structure
- Le comportement d'un élément, qui influence un autre élément, doit apparaître dans une architecture
- Le comportement d'un élément, qui ne permet pas de raisonner sur un attribut particulier du système, n'est pas essentiel dans une architecture (il ne doit pas apparaître)

Une architecture peut être bonne ou mauvaise $\Rightarrow$ permet ou empêche de raisonner sur les propriétés du système

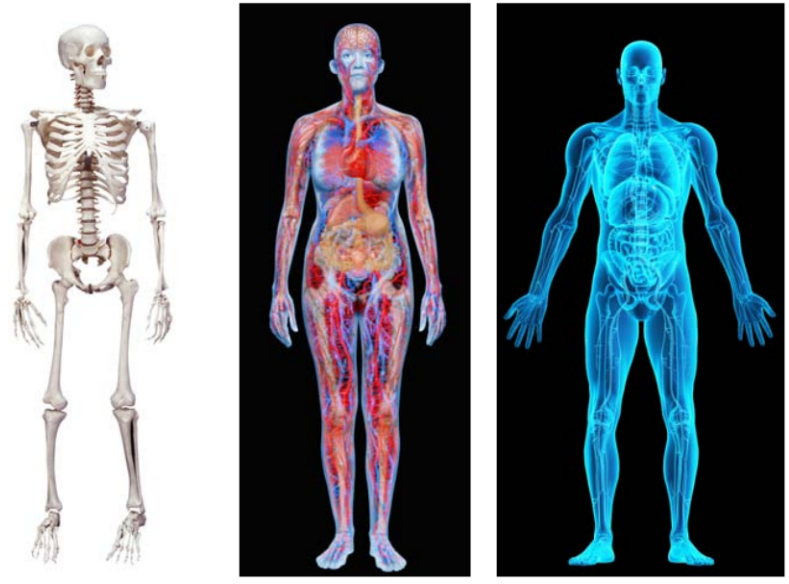
Architecture logicielle (AL) vs Architecture-système (AS) ou d'entreprise (AE)

- AS et AE couvrent un champ plus large
- AS inclut des composants logiciels, matériels et leurs interactions avec les humains
- AE traite de la structure et le comportement des processus d'une organisation, le flot d'informations, le personnel, les objectifs, la stratégie, ...
- Le logiciel est un seul aspect dans l'architecture d'une entreprise (support des processus métier + l'infrastructure) : autres aspects ⟶ comment le logiciel est utilisé par les humains pour exécuter les processus métier ? quel environnement ?

Architecture logicielle (AL) vs Architecture-système (AS) ou d'entreprise (AE) -suite-

- Elles partagent le fait que ce sont des structures qui représentent les éléments importants (leurs propriétés et interdépendances) pour raisonner sur quelque chose (logiciel, système informatique ou entreprise)
- Toutes peuvent être conçues, évaluées et documentées
- Toutes répondent à des besoins pour satisfaire des stakeholders
- Elles ont des langages, des standards, des catalogues de patrons, ...
- Focus dans ce cours sur les archi logicielles

Vues et structures



- Différentes vues pour simplifier le raisonnement sur un système complexe
- Une vue : la représentation graphique/textuelle d'un ensemble cohérent d'éléments architecturaux faite/lue par un stakeholder
- Une structure : l'ensemble de ces éléments, sans représentation
- Équivalent à l'analogie "Modèle vs Diagramme" en UML
- Les architectes "conçoivent" des structures et "documentent" des vues de ces structures

1. Quelques structures statiques (en modules)

Structure de décomposition

- Modules à implémenter et relations de type “est sous module de”
- Les modules ici sont le point de départ de la conception et le développement
- Chacun a comme productions : spec. d’interfaces, code, plans de test, …
- Attribut visé : capacité de changement (*modifiability*) – les changements probables sont localisés dans peu de modules

1. Quelques structures statiques (en modules) -suite-

Structure de dépendances (de type “utilisation”)

- Modules à implémenter et relations de type “utilise” (d’autres modules)
- Attribut visé : capacité d’extension (*extensibility*)

Modèle de données

- Modules : entités du domaine métier, leurs propriétés et les relations entre elles
- Attributs visés : capacité de changement et performance

2. Quelques structures dynamiques

Structure de services

- Les unités sont des services qui interagissent en utilisant des protocoles précis
- Attributs visés : interopérabilité et performance

Structure de concurrence

- Composants et connecteurs (mécanismes de communication)
- Composants organisés dans des threads/processus logiques
- Offre la possibilité de voir ce qui est parallélisable (perf. et disponibilité)

3. Quelques structures d'allocation

Structure de déploiement

- Montre comment le logiciel est affecté aux éléments matériels et de communication
- Utile pour raisonner sur les performances, la disponibilité et la sécurité

Structure d'implémentation

- Montre comment les modules de la structure statique sont affectés au système de fichier, à l'environnement de contrôle de configuration (screenshot de l'organisation des modules dans un IDE ou dépôts Git)
- Attribut visé : efficacité dans le développement

3. Quelques structures d'allocation -suite-

Structure d'affectation du travail

- Montre comment les modules sont affectés aux membres de l'équipe
- Surtout utile dans de grands projets logiciels multi-sources (équipe de dév répartie sur plusieurs sites)
- Attribut visé : efficacité dans le développement

Bonnes pratiques pour définir une architecture

Recommandations liées au processus

- L'architecture doit être l'oeuvre d'un seul architecte ou un petit groupe d'architectes ayant un leader technique identifié : pour garantir l'intégrité conceptuelle et la cohérence technique
- Fonder l'architecture sur la base d'une liste hiérarchisée de besoins d'attributs qualité (ça informe des compromis produits)
- L'architecture doit être documentée en utilisant différentes vues ($\neq$ stakeholders) : documentation minimaliste pour commencer
- L'architecture doit être évaluée (tôt) pour sa capacité à fournir les attributs de qualité importants du système
- L'architecture doit se prêter à une implémentation progressive, pour éviter de tout intégrer en même temps

Bonnes pratiques pour définir une architecture

Recommandations liées au produit (à la structure)

- L'architecture doit comporter des modules bien définis dont les responsabilités fonctionnelles sont assignées selon les principes d'encapsulation de l'information et de séparation des préoccupations (interfaces bien définies et dév indépendant)
- Les besoins en attributs qualités devront être satisfaits en utilisant des styles d'architecture connus/confirmés
- L'architecture initiale ne doit jamais dépendre d'une version particulière d'un produit commercial (supporter le changement)
- Les modules producteurs et consommateurs de données doivent être séparées (facilité de changement)

Bonnes pratiques pour définir une architecture

- Ne pas confondre la structure statique de modules (descripteurs) et la structure dynamiques de composants exécutables (instances) : plusieurs instances d'un même descripteur peuvent coexister et déployés différemment
- L'architecture doit comporter un nombre limité de moyens de communication entre composants (mêmes interactions de la même manière)
- L'architecture doit contenir un ensemble spécifique (et réduit) de zones de conflit de ressources, dont la résolution est clairement spécifiée et maintenue

Pourquoi l'architecture est importante ?

- Met en évidence les points où un système peut être modifié et favorise le développement par la réutilisation
- Bien documentée, elle améliore la communication entre stakeholders
- Est un moyen pour véhiculer les premières (et plus importantes) décisions de conception (les plus difficiles à changer)
- Contraint l'implémentation et peut dicter l'organisation de l'équipe de dév
- En limitant les alternatives de conception, elle canalise la créativité des développeurs, et réduit ainsi la complexité (facilite la compréhension)
- Peut servir comme base pour la formation de nouveaux membres de l'équipe