LOG430 – Architecture logicielle-revision-1

Scénarios de qualité, tactiques et justifications des attributs produits par les équipes en classe

[Le lecteur de carte doit être changé pour accommoder une technologie supplémentaire de vérification sur la carte bancaire. (Par exemple: Puce RFID, Infrarouge…)](#_gjdgxs)

[Le guichet doit prendre en charge un nouvel algorithme de chiffrement des données de carte bancaire.](#_1fob9te)

[Implémenter une gestion de configuration pour les règles d’affaires](#_2s8eyo1)

[Changer la configuration des règles d’affaires](#_3rdcrjn)

[Une faille permet l’accès aux données du serveur](#_1ksv4uv)

[Ajouter un nouveau mode d’authentification](#_44sinio)

[Changement d’un composant physique de l’ATM](#_4i7ojhp)

[Ajout d’une méthode d’authentification](#_2xcytpi)

[Ajout de nouvel type de transaction](#_qsh70q)

[Retirer de l’argent avec un téléphone intelligent par code QR](#_3as4poj)

[Ajout d’une nouvelle fonctionnalité (Fingerprint)](#_ihv636)

[Changement de règle d’affaire sur le serveur bancaire](#_41mghml)

Modifiabilité – Justification

Un guichet automatique est un outil du domaine bancaire; il constitue un enjeu économique. Garder le système à la pointe de la technologie est une source de motivation importante pour fidéliser les clients. Cependant, cela demande de modifier le système de façon fréquente ou en profondeur.

Pour les clients des banques, il est agréable d’avoir accès aux nouvelles technologies qui rendent plus rapide leur utilisation du système. Pensons par exemple au récent Spot Pass permettant de payer en quelques secondes.

Pour l’administration des banques, offrir un service à jour est une principale façon de faire face à la compétition. De plus, il faut que le système puisse s’adapter aux changements critiques dans l’industrie, par exemple il faut s’assurer que le système puisse recevoir des ajouts ou remplacement de nouveaux Sous ­systèmes de sécurité.

Pour les développeurs du système, il faut que le système soit suffisamment facilement modifiable pour répondre aux demandes de l’administration dans un temps raisonnable.

# Le lecteur de carte doit être changé pour accommoder une technologie supplémentaire de vérification sur la carte bancaire. (Par exemple: Puce RFID, Infrarouge…)

| Scénario | Le lecteur de carte doit être changé pour accommoder une technologie supplémentaire de | |
| --- | --- | --- |
|  | vérification sur la carte bancaire. (Par exemple: Puce RFID, Infrarouge…) | |
| Objectifs d'affaires | Ajout ou remplacement des protocoles de sécurité en 160 heures de travail ou moins | |
| Attribut de qualité | Modifiabilité | |
| Source | Administrateur système | |
| Stimulus | Demande de prendre en charge une nouvelle technologie de sécurité de carte bancaire | |
| Artéfact | Module de code faisant interface avec le lecteur physique | |
| Environnement | Période de conception | |
| Réponse | Modifier le module en question et tester les changements | |
| Mesure de la | Temps de développement total de l’ajout d’une nouvelle mesure de sécurité (mesuré par relevés d’heures travaillées) | |
| réponse |  | |
|  | Temps calendrier nécessaire pour accomplir le changement | |
|  | Prix pour effectuer le changement | |
|  | Nouveaux défauts introduits | |
| Questions | 1. | Quel patrons de conceptions permettent d’ajouter rapidement une nouvelle |
|  |  | implémentation d’une fonctionalité similaire? Stratégie, Composite, Médiateur, |
|  |  | Adaptateur |
|  | 2. | Quels IDE seront utilisés? Combien de licenses seront disponibles/nécessaires? |
|  | 3. | Quels langages de programmation et écrit seront utilisés pour implémenter et |
|  |  | documenter? |
|  | 4. | Est­ ce que le couplage des modules du code est minime pour |
|  |  | l’addition/modification des algorithmes existants. |
|  | 5. | Est-­ce que la modification du code respecte toujours la cohésion du code? |

Modifiabilité – Scénario 2

# Le guichet doit prendre en charge un nouvel algorithme de chiffrement des données de carte bancaire.

| Scénario | Le guichet doit prendre en charge un nouvel algorithme de chiffrement des données de carte bancaire. | |
| --- | --- | --- |
|  |  | |
| Objectifs d'affaires | Remplacement de l’algorithme de chiffrement en 10 heures de travail ou moins. | |
| Attribut de qualité | Modifiabilité | |
| Source | Développeurs | |
| Stimulus | Demande de prendre en charge le nouvel algorithme dans les différents guichets | |
|  | automatiques | |
| Artéfact | Module interne du guichet responsable de l’encryption et de la décryption des données bancaires | |
|  |
| Environnement | Période de conception | |
| Réponse | Modifier le module en question et exécuter les tests unitaires | |
| Mesure de la | Temps de développement total de l’algorithme (mesuré par relevés d’heures travaillées) | |
| réponse |  |  |
| Questions | 1. | Le langage utilisé dans le logiciel du guichet supporte ­t-­il le nouvel algorithme de chiffrement? |
|  |  |
|  | 2. | Est­-ce que le nouveau changement aura des impacts sur les ressources humaines utilisant le système après le changement? |
|  |  |
|  | 3. | Est­-ce que ce changement est critique et est-­ce que ce changement affecte le développement de nouvelles fonctions? |
|  |  |
|  | 4. | Comment s’assurer que le couplage des modules du code est minime pour  l’addition/modification des algorithmes existants? |
|  |  |
|  | 5. | Est-­ce que la modification du code respecte toujours la cohésion du code? |

Modifiabilité – Tactiques pour le scénario 1

Description: ​*Use an intermediary*

Justification: L’utilisation d’un intermédiare brise la dépendance entre deux modules. Dans le cas des mesures de sécurité du lecteur de carte qui changeraient, un intermédiaire pourrait standardiser les communications entre le module du lecteur de carte et les composantes qui ont besoin de ce module.

L’utilisation du patron composite permettrait de découpler les détails de la validation d’une carte de celui qui l’utilise, par exemple. Dans le futur, si d’autres changements sont nécessaires, les dépendances ajoutées seront minime car le couplage a diminué entre ces composantes et le reste du système.

Description: ​*Restrict dependencies*

Justification: Limiter les dépendances réduit les modules qui peuvent utiliser ou même voir un module qui risque de changer souvent. L’architecture en couche facilite beaucoup la limitation des dépendances. Dans le contexte des technologies utilisées par le lecteur de carte qui changent, réduire le nombre de dépendances à ce composant facilitera les changements futurs.

Modifiabilité – Tactiques pour le scénario 2

Description: ​*Encapsulate*

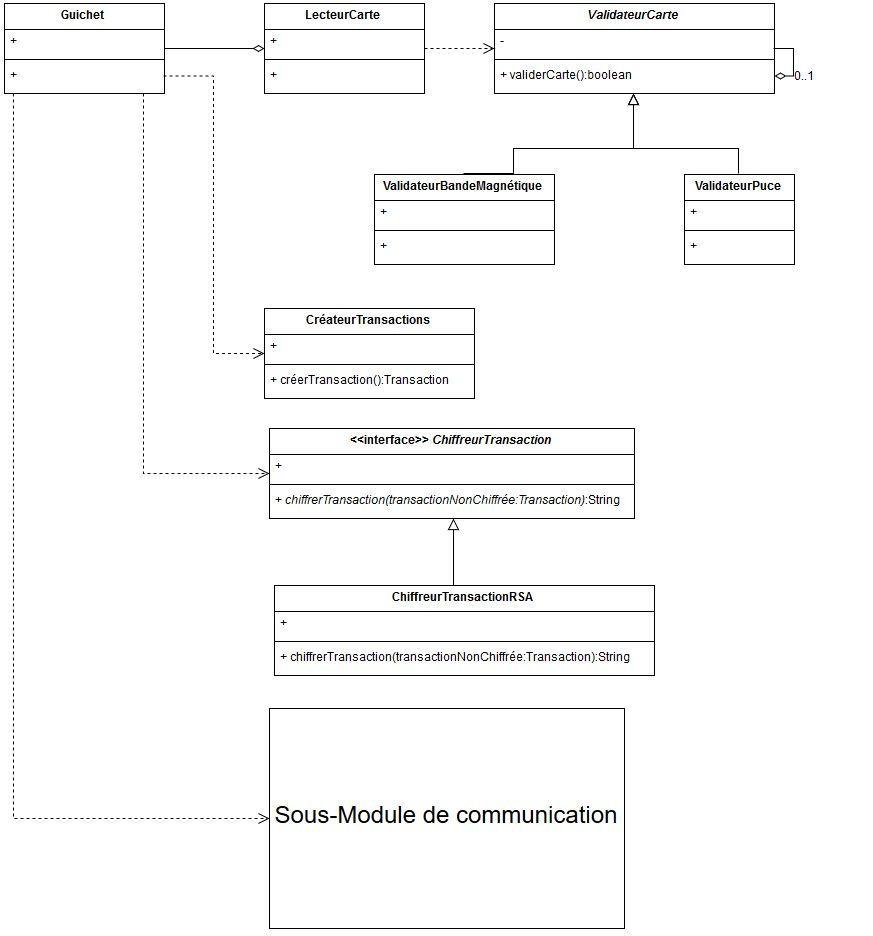
Justification: L’encapsulation introduit une interface explicite pour un module. Une interface externe devient disponible aux autres modules du système sans exposer le fonctionnement interne du module implémentant l’API. Pour le contexte de l’encryption, les modules externes n’ont pas besoin de savoir comment les données sont encryptées. Les modules externes envoient les données brutes à l’API et reçoivent le résultat encrypté. Ceci facilite grandement les changements futures, car la nouvelle méthode d’encryption n’affecte pas le fonctionnement des autres modules.

Description: ​*Use an intermediary*

Justification: Utiliser un intermédiare pour l’encryption a un effet similaire à l’utilisation d’un API par encapsulation. L’intermédiaire serait l’interface qui serait utilisée par les modules ayant besoin du module d’encryption. Les classes utilisant le module d’encryption ne communiquera avec lui que par l’intermédiaire de l’interface, limitant ainsi la dépendance et le couplage avec la classe concrète. Si le module d’encryption est changé, seulement l’implémentation concrète de l’interface aura à changer pour implémenter le nouveau fonctionnement.

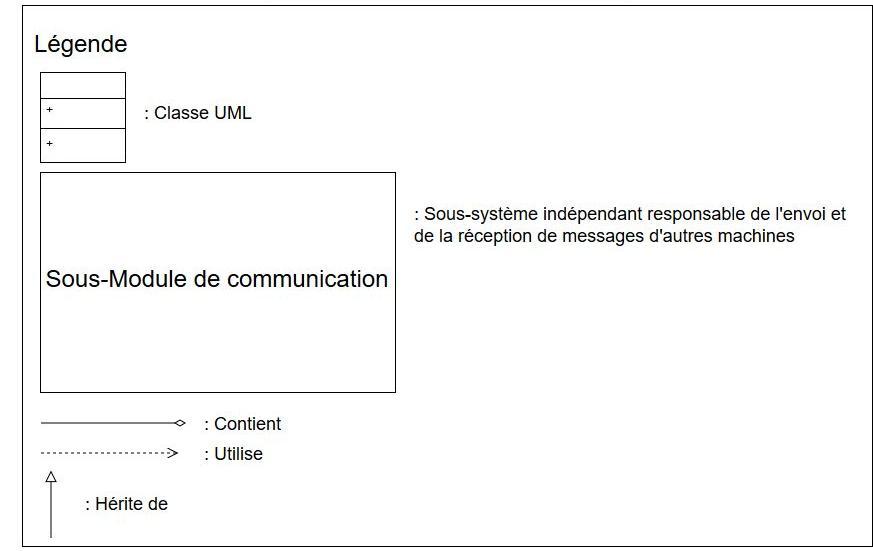
Modifiabilité – Vue architecturale

Diagramme



*(Voir la légende à la page suivante)*

Diagramme (suite : Légende)



Description du diagramme:

La vue architecturale module ci­-haut donne une vue d’ensemble des composantes sujettes à changements mentionnées par les scénarios précédents de l’infrastructure d’un guichet automatique. Le diagramme permet de comprendre comment maximiser la modifiabilité du logiciel propre au guichet en utilisant les patrons composite (ValidateurCarte) et stratégie (ChiffreurTransaction). Il suffira en effet de changer le composant utilisé par le guichet dans le cas d’un changement d’algorithme de chiffrement ou de changer le composite utilisé pour la validation de la carte lue.

Description des éléments

| Élément | Description |
| --- | --- |
|  |  |
| Guichet | Représente une classe contenant l’état et  informations éphémères contenues sur le guichet.  Contrôleur de session. |
|  |
|  |
|  |  |
| LecteurCarte | Représente le lecteur de carte, est responsable  d’effectuer toutes les validations automatiques  effectuées lors de l’insertion de la carte dans le  guichet. |
|  |
|  |
|  |
|  |  |
| ValidateurCarte | Classe abstraite effectuant une vérification sur la carte insérée et pouvant demander à un autre VérificateurCarte de faire de même et ainsi de suite (patron composite). |
|  |
|  |
|  |
|  |  |

| ValidateurBandeMagnétique | Classe concrète effectuant des vérifications  sur la carte à l’aide de la bande magnétique. |
| --- | --- |
|  |
|  |  |
| ValidateurPuce | Classe concrète effectuant des vérifications sur la carte à l’aide de la puce. |
|  |
|  |
| CréateurTransaction | Classe responsable de créer les transactions que le client veut réaliser en utilisant l’ATM. |
|  |
|  |  |
| ChiffreurTransaction | Une interface permettant de chiffrer une  transaction de façon quelconque. Elle permet de changer le module ou classe chiffrant la transaction de façon transparente au guichet. |
|  |
|  |
|  |
|  |  |
| ChiffreurTransactionRSA | Une classe concrète permettant de chiffrer une  transaction grâce à l’algorithme RSA. (Patron  stratégie) |
|  |
|  |
|  |  |

Relation entre les éléments et les tactiques

Scénario 1:

**Intermediary**: L’utilisation d’un composite permet de valider la carte avec tous les systèmes ou de l’invalider dès qu’un la rejette sans avoir à modifier la logique du guichet ou d’éventuelles classes intermédiaires utilisant le composite de validation.

**Restrict dependency:** Composite s’assure que l’ajout d’un protocole sur la carte n’affectera aucune classe et se traduira simplement par l’ajout d’une feuille au composite.

L’utilisation du patron composite assure une grande cohésion et plus faible couplage. Il permet aussi d’ajouter des mesures additionnelles à celles présentes ou alternatives en limitant au maximum les modifications causées sur les autres classes.

Scénario 2:

**Encapsulate**: L’utilisation du patron stratégie s’assure qu’uniquement les fonctions publiques seront disponibles, cachant toute l’implémentation du fonctionnement de la classe ou du module responsable du chiffrement des transactions.

**Use an intermediary:** L’interface ChiffreurTransaction sert ici d’intermédiaire. Qu’importe la façon dont il fonctionne, c’est l’interface commune que le guichet utilisera.

Relation entre les tactiques et les scénarios

Pour les scénarios relevés, les tactiques optées facilitent grandement la modification du programme. Dans le cas du scénario 1 (Ajout d’une mesure de sécurité sur la carte bancaire), le fait d’utiliser un composite permet de conserver les mesures précédentes de sécurité sans les modifier et d’ajouter à une liste de complexité quelconque la nouvelle technologie simplement en profitant du polymorphisme. Le retour de la fonction validerCarte n’a qu’à être le résultat du validateur ET le résultat de son prochain. Les opérations à faire en cas d’invalidité peuvent être laissées spécifique à chaque vérificateur, mais le composite agissant comme une chaîne renverra toujours simplement si la carte est valide ou non au lecteur de carte lui demandant.

Dans le cas du scénario 2 (Remplacement de l’algorithme de chiffrement de Transactions), l’utilisation du patron Stratégie permet encore une fois de cacher les détails d’implémentation du chiffrement aux classes les utilisant, ce qui permet de remplacer l’algorithme au complet sans avoir un impact sur les autres modules. Cela permet aussi de s’assurer que toute la logique de chiffrement est regroupée à un seul endroit dans le code, le rendant plus facile à maintenir.

LOG430 ­ Architecture logicielle

Scénarios de qualité, tactiques et justifications des attributs produits par les équipes en classe

Scénario 1

# Implémenter une gestion de configuration pour les règles d’affaires

| Scénario | Implémenter une gestion de configuration pour les règles |
| --- | --- |
|  | d’affaires |
|  |  |
| Objectifs d’affaires | Moduler les offres disponibles aux clients selon les besoins |
|  | d’entreprise |
|  |  |
| Attribut de qualité | Modificabilité |
|  |  |
| Source | Developpeur |
|  |  |
| Stimulus | Adopter les règles d’affaires d’un client |
|  |  |
| Artéfact | Code |
|  |  |
| Environnement | Design Time |
|  |  |
| Réponse | Effectuer des modifications |
|  |  |
| Mesure de la réponse | La mesure dans laquelle cette modification affecte d'autres |
|  | fonctions ou attributs de qualité |
|  |  |
| Questions | ● Quelles sont les dépendances actuelles? |
|  |  |

Scénario 2

# 

# Changer la configuration des règles d’affaires

| Scénario | Changer la configuration des règles d’affaires |
| --- | --- |
|  |  |
| Objectifs d’affaires | Moduler les offres disponibles aux clients selon les besoins |
|  | d’entreprise |
|  |  |
| Attribut de qualité | Modificabilité |
|  |  |
| Source | Administrateur système |
|  |  |
| Stimulus | La configuration des règles d’affaires doit être changée. |
|  |  |
| Artéfact | Configurations |
|  |  |
| Environnement | Runtime |
|  |  |
| Réponse | Les changements sont pris en compte en temps réel. |
|  |  |
| Mesure de la réponse | Temps requis pour modifier le code. |
|  |  |
| Questions | ● |
|  |  |

Tactique 1

**Description**:​*Encapsulate*​

**Justification**:​Encapsuler la logique d’affaire dans un seul module. Cacher l’implémentationde ce module derrière une interface qui sera utilisé par les autres modules. Cela permet d’éviter la propagation des changements aux autres modules.

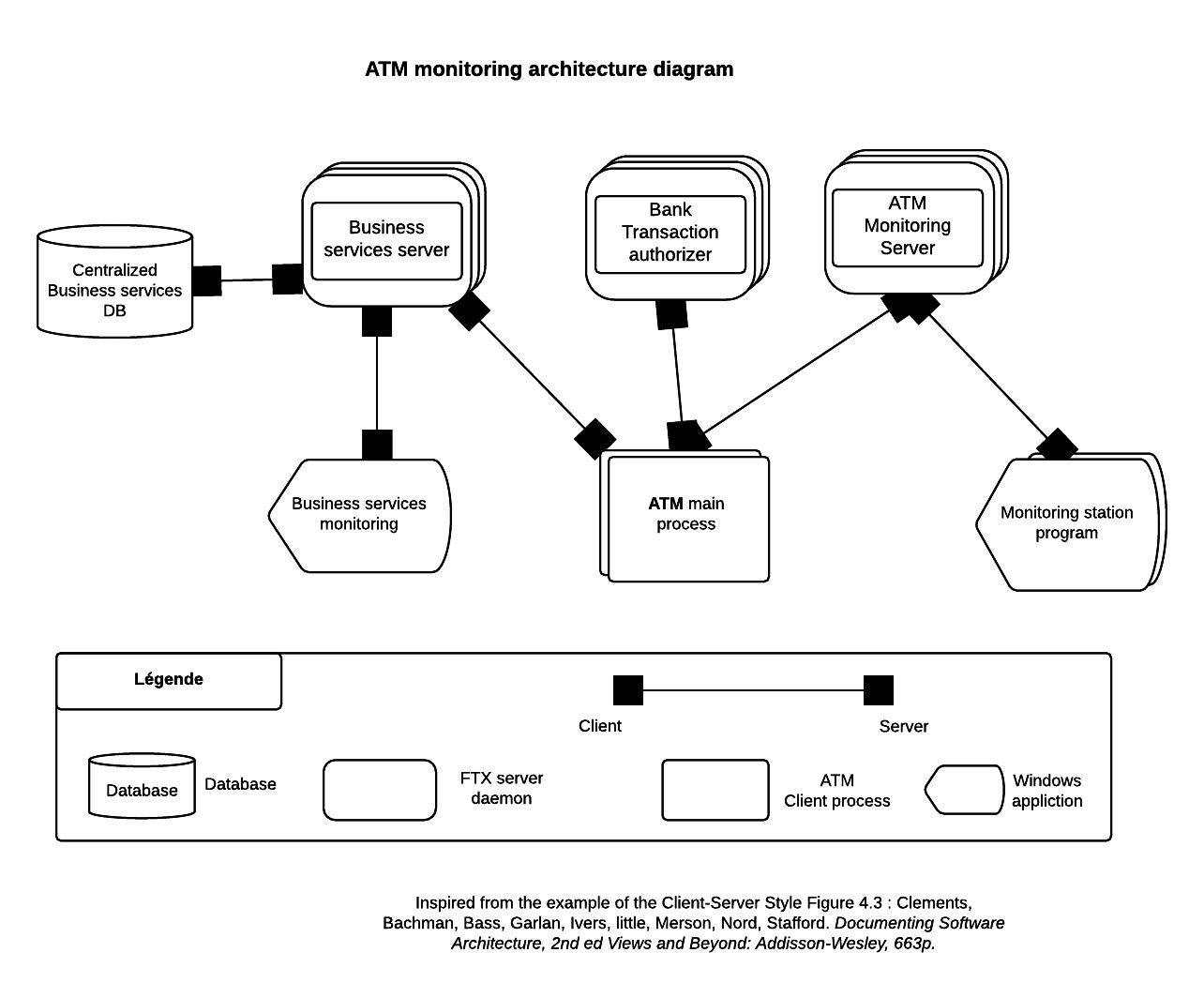
Tactique 2

**Description**:​*Shared*​*repositories*

**Justification**:​Le gestionnaire du service peut, via une interface dédiée, mettre à jour lesdonnées utilisées par le service client, ce qui permet aux systèmes d’utiliser de nouvelles données directement lors de l’exécution. Par exemple, les gestionnaires d’une banque peuvent mettre à jour les taux d’intérêt des produits d’épargne qui seront reflétés directement lors des demandes de services des usagers.

Vue architecturale

Diagramme et légende



Description du diagramme et des relations

L’ajout du *Business*​ *Services Server​*gère la logique d’affaires, le Business Services Server communique avec une base de données qui agrège les données relatives aux variables d’affaires (telles que les taux d’intérêts). Un gestionnaire peur gérer et altérer les variables via une application le Business services monitoring. Par la suite les usagers des ATM accèdent à leurs services et visualisent les données à jour (taux d’intérêt) parce que le ATM communique avec le Business Services Server et obtient les données à jour.

Description des éléments du diagramme

| ATM main process | Représente un appareil physique et son système  embarqué sur le site d’opération (banque) |
| --- | --- |
|  |
|  |
| Bank Transaction | Serveur d’authentification et de validation des  transactions appartenant à la banque qui opère les ATM |
| authorizer |
|  |  |
| ATM Monitoring Server | Serveur qui gère les états et les évènements dans les ATM (rupture de connexion réseau / panne matérielle /niveaux de liquidités…) |
|  |
|  |
|  |  |
| Monitoring station program | Service applicatif dans les bureaux de support de la  banque, panneau de gestion opéré par les employés de support |
|  |
|  |
|  |  |
| Business Service Server | Service chargé de la gestion des règles d’affaires |
|  |  |
| Business Services | Service applicatif dans les bureaux de support de la  banque, panneau de gestion opéré par les gestionnaire des regles d’affaires. |
| Monitoring |
|  |
|  |  |
| Centralized Business | Contients les données des règles d’affaires |
| services DB |  |
|  |  |
| Tactiques répond aux scénarios | |

La tactique Shared repository a été implémentée en utilisant un serveur hébergeant un service de gestion des règles d’affaires (Business services server). Les guichets utilisent ce serveur pour accéder à leur logique d’affaires. Les règles peuvent être modifiées en temps réel à l’aide d’une application Windows accessible par des gestionnaires.

**LOG430 – Architecture logicielle**

**Scénarios de qualité, tactiques et justifications des attributs produits par** **les équipes en classe**

**Modifiabilité – Justification**

La modifiabilité est importante pour assurer la sécurité du système. En effet, si une faille est détectée, un correctif pourra être déployé.

**Modifiabilité – Scénario 1**

# Une faille permet l’accès aux données du serveur

| **Scénario** | Une faille permet l’accès aux données du serveur | |
| --- | --- | --- |
| **Objectifs d'affaires** | Ne pas compromettre la sécurité des données des clients ou et autres données importantes sur la logique de l’entreprise et ses opérations | |
|  |
| **Attribut de qualité** | Modifiabilité | |
| **Source** | Équipe responsable de développement des correctifs | |
| **Stimulus** | Modifie une partie de l’application où se trouve une faille | |
| **Artéfact** | Code source de l’application | |
| **Environnement** | Développé sur l’environnement de test en opération interrompue. | |
| **Réponse** | Un correctif est développé et prêt à être déployé | |
| **Mesure de la** | Temps et effort requis pour développer un correctif à implanter dans l’environnement en | |
| **réponse** | production. | |
| **Questions** | 1. | Quelle est la criticité de la faille? |
|  | 2. | La faille doit-elle être corrigée immédiatement? |
|  | 3. | Quels sont les impacts potentiels du correctif? |
|  | 4. | Quelle est l’ampleur visée du correctif? |
|  | 5. | Est-ce que la gravité de la faille nécessite que l’environnement en production soit stoppé en attente du correctif? |
|  |  |

**Modifiabilité – Scénario 2**

# Ajouter un nouveau mode d’authentification

| **Scénario** | Ajouter un nouveau mode d’authentification | |
| --- | --- | --- |
| **Objectifs d'affaires** | Être capable de supporter de nouveaux modes d’authentification (reconnaissance faciale, lecteur d’empreinte, reconnaissance oculaire, etc.) | |
|  |
| **Attribut de qualité** | Modifiabilité | |
| **Source** | Équipe de développement de nouvelles fonctionnalités | |
| **Stimulus** | Supporter les dernières technologies d’authentification | |
| **Artéfact** | Code source de l’application | |
| **Environnement** | Développé sur l’environnement de test en opération interrompue. | |
| **Réponse** | Un nouveau système d’authentification est développé et prêt à être déployé | |
| **Mesure de la** | Temps du développement de la nouvelle fonctionnalité. | |
| **réponse** | Temps de test de la nouvelle fonctionnalité. | |
|  | Temps de déploiement de la nouvelle fonctionnalité. | |
|  | Temps requis pour performer une authentification avec le nouveau système. | |
|  | Nombre de modules connexes de l’application affectée. | |
|  | Nombre de développeurs requis pour implémenter la fonctionnalité. | |
|  | Nombre de bogues introduit. | |
| **Questions** | 1. Est-ce qu’ajouter un nouveau système d’authentification est prioritaire? 2. Faut-il embaucher un expert de la technologie? 3. Laisse-t-on l’utilisateur choisir son mode d’authentification? 4. Faut t-i l former le personnel de support sur les nouvelles fonctionnalités? 5. Est-ce que des méthodes d’authenfication ne sont pas fiables à 100%? | |
|  |
|  |
|  |

**Modifiabilité – Tactique 1**

**Description**: Restrict dependencies

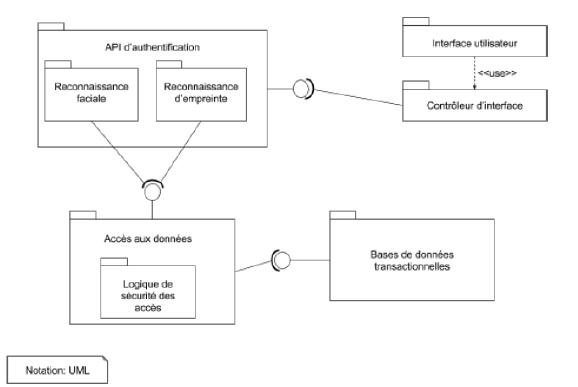
**Justification**: La tactique que l’on pourrait adopter pour le scénario1 est *Restrict dependencies*. En effet cela rendrait maintenable et donc le développement de correctifs de sécurité sera moins laborieux. Réduire la durée du développement des correctif est justement une exigence d’affaire dans ce scénario.

**Modifiabilité – Tactique 2**

**Description**: La tactique employée pour le scénario 2 est *Encapsulate*, de la catégorie *Reduce coupling*. Pour implémenter cette solution, un API d’authentification sera développé. Les modules faisant affaire à l’API d’authentification n’auront pas besoin de connaître ses détails d'implémentation .

**Justification**: Utiliser la tactique *Encapsulate* réduira le couplage entre le système d’authentification et le reste de l’application. Il sera donc possible de modifier la logique d’authentification sans affecter les modules connexes. La définition d’un API rendra aussi plus facile la création de tests de type boite noire.

**Modifiabilité – Vue architecturale**



**Description du diagramme**

Le diagramme représente une architecture de type *Module* utilisant les styles *Décomposition* et *Utilise*. Le diagramme ne montre pas l’architecture complète du projet, mais seulement les modules reliés aux tactiques décrites plus haut.

Le module *Logique de sécurité* est une visualisation de la partie de code où se trouveraient potentiellement des failles de sécurité en lien avec le scénario 1. Nous pouvons constater qu’il s’agit ici d’une architecture dont les dépendances semblent minimales et cela rendra le développement des correctifs moins laborieux.

Le module *API d’authentification* gère l’implémentation de la tactique 2. C’est ce module qui définit une interface publique pour accéder aux diverses fonctionnalités d’authentification. Tous les modules représentant une implémentation spécifique d’un système d’authentification seront placés à l’intérieur de ce module. Tous les modules pouvant faire partie d’*API d’authentification* ne sont pas représentés dans ce diagramme.

**Table de description des éléments du diagramme**

Voici un tableau qui décrit les caractéristiques des éléments des vues architecturales présentées à la page précédente.

| **Élément** | **Description** |
| --- | --- |
|  |  |
| API d’authentification | Sers d’interface pour l’accès aux fonctionnalités d’authentification. Les |
|  | autres modules de l’application ne peuvent pas accéder aux divers |
|  | modules d’authentification directement, ils doivent passer par l’API. |
|  | Cet élément contient tous les modules reliés à l’authentification. |
|  |  |
| Reconnaissance faciale | Gère la reconnaissance faciale. |
|  |  |
| Reconnaissance d’empreinte | Gère la reconnaissance d’empreinte. |
|  |  |
| Accès aux données | Gère l’accès aux données. |
|  |  |
| Interface utilisateur | Gère l’affichage de l’interface utilisateur |
|  |  |
| Contrôleur d’interface | Fait la correspondance entre la logique système et l’affichage des |
|  | données |
|  |  |
| Logique de sécurité des accès | Vérifie que l’authentification a été valide et gère quelles données seront |
|  | accessibles |
|  |  |
| Bases de données | Stocke les données confidentielles des transactions des clients de la |
| transactionnelles | banque |
|  |  |
| Connecteurs requête | Représentation d’un appel aux fonctions d’une interface |
|  |  |
| Connecteurs réponse | Représentation d’une interface qui rend disponibles les données à partir |
|  | d’appels d’autres modules |
|  |  |

LOG430 – Architecture logicielle

Modifiabilité – Justification (en classe)

Réduire les coûts et le temps de développement pour changer/ajouter des composants.

Modifiabilité – Scénario 1 (en classe)

# Changement d’un composant physique de l’ATM

| **Scénario** | Changement d’un composant physique de l’ATM | |
| --- | --- | --- |
| **objectifs d'affaires** | Réduire le temps de développement. | |
| **Attribut de qualité** | Modifiabilité | |
| **Source** | Développeur | |
| **Stimulus** | Vouloir changer un ou plusieurs composants | |
| **Artéfact** | Code et interface | |
| **Environnement** | Design time | |
| **Réponse** | Faire la modification/tester/déployer | |
| **Mesure de la** | Le temps et le coût du développement.. | |
| **réponse** | Le nombre de bogue | |
|  | L’effort. | |
| **Questions** | 1. | Quel composant on change ? |
|  | 2. |  |
|  | 3. | À quel degré les composants peuvent être différents des précédents ? Il manque des questions. La deuxième question n’est pas sur la bonne ligne. |
|  | 4. |  |

Modifiabilité – Scénario 2 (en classe)

# Ajout d’une méthode d’authentification

| **Scénario** | Ajout d’une méthode d’authentification | | |  |
| --- | --- | --- | --- | --- |
| **objectifs d'affaires** | Réduire le temps de développement. | | |  |
| **Attribut de qualité** | Modifiabilité | | |  |
| **Source** | Développeur | | |  |
| **Stimulus** | Ajout d’une nouvelle méthode d’authentification | | |  |
| **Artéfact** | Code | |  |  |
| **Environnement** | Design time | | |  |
| **Réponse** | Faire la modification/tester/déployer | | |  |
| **Mesure de la** | Le temps et le coût du développement.. | | |  |
| **réponse** | Le nombre de bogue | | |  |
|  | L’effort | |  |  |
| **Questions** | 1. | | À quelle fréquence une nouvelle méthode d’authentification est |  |
|  | 2. | | nécessaire ? |  |
|  |  |  |
|  |  |
|  |  |

Modifiabilité – Tactique 1 (en classe)

**Description**:​**Reduce Coupling ­> Abstract Common Services**

**Justification**: Pour simplifier la modification du matériel de l’ATM (changement d’un composant physique de l’ATM, par exemple), les services communs à une famille de drivers seraient fournis à l’intérieur d’une façade abstraite. Cette façade serait ensuite complétée par le biais de spécialisation pour chaque composant matériel potentiellement utilisé par le système.

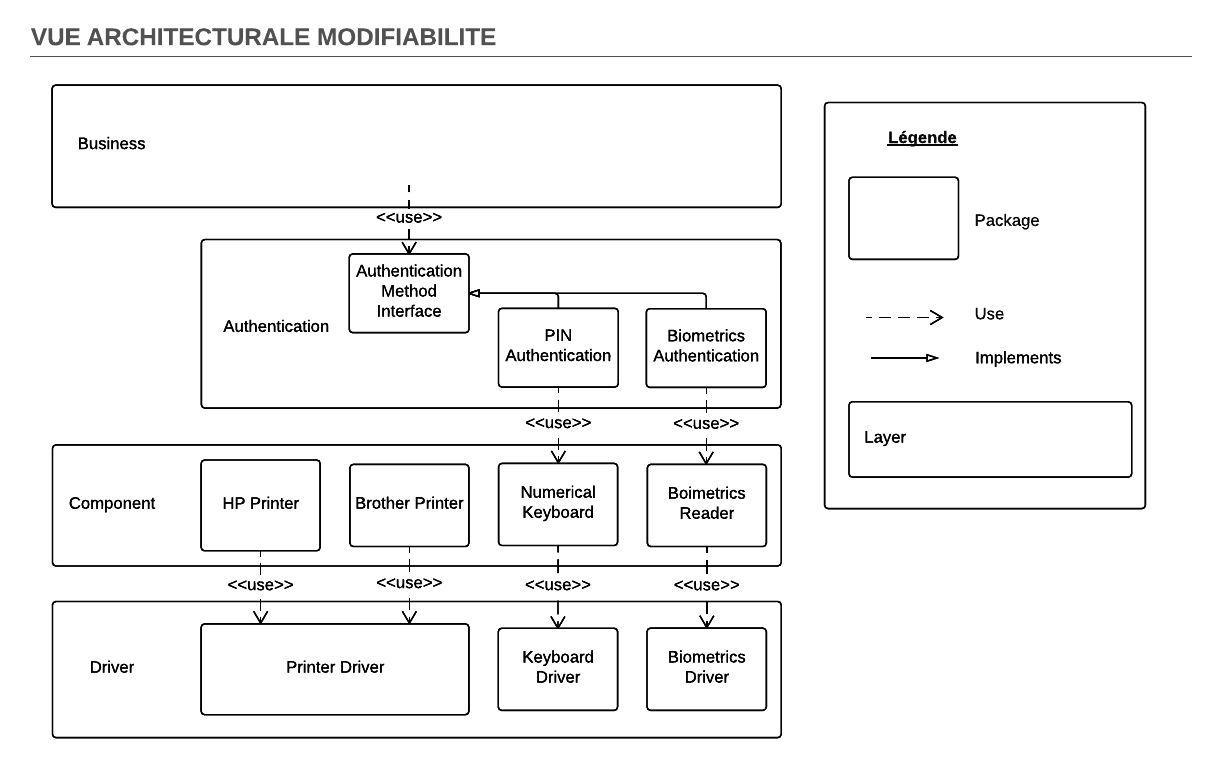
Modifiabilité – Tactique 2 (en classe)

**Description**: **Reduce**​ **coupling ­> Encapsulate**

**Justification**: Dans le but d’éviter d’avoir à modifier le code utilisant les méthodes d’authentification actuelles, celles­-ci seraient encapsulées à l’intérieur d’une interface unique et commune à toutes les méthodes d’authentification. En utilisant cette tactique, l’ajout d’une méthode d’authentification se ferait en implémentant l’interface commune et en ajoutant une référence à cette méthode dans le module de gestion de l’authentification.

Modifiabilité – Vue Architecturale (à l’extérieur de la classe)

L'architecture propose un modèle en couche qui permet d’interfacer les différents composants physiques et leur implémentation particulière en leur faisant respecter une façade (driver). Un nouvel élément matériel ne demandera alors qu’une nouvelle implémentation de l’interface façade.



**Tableau des éléments**

| **Éléments** | **Description** |
| --- | --- |
|  |  |
| Business | Package et couche représentant la logique |
|  | d’affaire, ayant un droit d’utilisation sur les |
|  | packages Authentication et Component |
|  |  |
| Authentication | Package et couche regroupant les éléments en |
|  | lien avec l’authentification. |
|  |  |
| Component | Package et couche regroupant les |
|  | représentations des éléments physiques de |
|  | l’ATM. |
|  |  |
| Driver | Package et couche façade avec les éléments |
|  | physiques de l’ATM. |
|  |  |

**Description de la relation entre les éléments et les tactiques**

* L’élément “Driver” représente la façade abstraite permettant l’abstraction des services communs, pour la tactique 1.
* L’élément “Component” donne un exemple d’implémentations concrètes pour des composants physiques spécifiques, pour la tactique 1.
* L’élément “Authentication” et son contenu représente la totalité de la tactique 2. Plus particulièrement:
  + L’élément “Authentication Method Interface” est l’interface unique et commune à toutes les méthodes d’authentification, permettant aux packages le connaissant d’utiliser toutes les méthodes d’authentification l’implémentant.
  + Les éléments “PIN Authentication” et “Biometrics Authentication” sont des exemples de méthode d’authentification concrète, utilisée par le package “Business” sans dépendance directe.

|  | LOG430 – Architecture logicielle |
| --- | --- |
|  | Modifiabilité |
| Ajout de nouvel type de transaction | |
|  |  |
| Scénario | Ajout d’un type de transaction |
| objectifs d'affaires | Satisfaction client, compétitivité |
| Attribut de qualité | Modifiabilité |
| Source | L’usager, l’administration, développeur |
| Stimulus | Ajout d’un nouveau type de transaction |
| Artéfact | Interface d’utilisateur, système de transaction (logiciel) |
| Environnement | Design |
| Réponse | Effectuer une série de tests pour valider l’ajout du nouveau type de transaction |
| Mesure de la | Coût de développement du module |
| réponse | Temps de déploiement sur les systèmes concernés |
| Questions | 1. Quels paramètres devraient changer selon le type de transaction? |
|  | 2. Quels types de transaction sont susceptibles d’être ajoutés? |
|  | 3. Y a ­t-­il besoin de remplacer des composants physiques du système ATM afin de supporter des types de transactions additionnels? |
|  |

# Retirer de l’argent avec un téléphone intelligent par code QR

| Scénario | Retirer de l’argent avec un téléphone intelligent par code QR | |
| --- | --- | --- |
| objectifs d'affaires | Permettre aux utilisateurs de retirer de l’argent par ATM dans le cas où ils n’ont pas leur carte bancaire. | |
|  |
| Attribut de qualité | Modifiabilité | |
| Source | Développeur, technicien | |
| Stimulus | Ajout d’une fonctionnalité | |
| Artéfact | Interface utilisateur, système de transaction (logiciel), hardware | |
| Environnement | Design |  |
| Réponse | Faire les modifications et les tests | |
| Mesure de la | Coût des modifications | |
| réponse | Durée de développement des modifications | |
| Questions | 1. | Est­-ce que cela va affecter plusieurs artéfacts? |
|  | 2. | Est­-ce que le logiciel existant permet de faire le changement? |
|  | 3. | Est­-ce qu’il est possible d’ajouter un nouveau composant à l’ATM permettant d’effectuer le nouveau type d’authentification (ex. scanneur QR)? |
|  |  |
|  | 4. | Quelle est la probabilité qu’un utilisateur oublie sa carte bancaire? |

Tactique ­ Scénario I

**Description​**:Restrict dependencies

**Justification​**:Afin d’implémenter un système de transaction extensible, il est sécuritaire et efficace d’implémenter plusieurs couches logicielles réduisant les dépendances entre les modules du système. Ainsi, une couche Transactions située entre l’interface utilisateur et le module gérant les transactions concrètes peut faire le lien entre les deux et permettre l’ajout de nouveaux types de transaction sans avoir à modifier la couche plus sécurisée et stable du système bancaire.

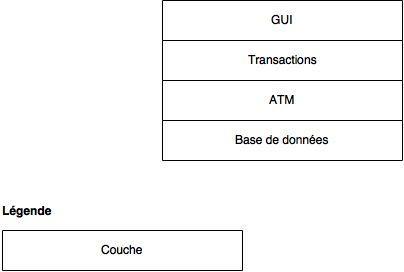
Tactique ­ Scénario II

**Description​**:Abstract common services

**Justification​**:Dans le cas traité, il y a deux modules d’authentification: le service standard et celui par codeQR,qui doivent offrir le service pour retirer l’argent de l’ATM. Le service va être placé dans le contrôleur de l’ATM. Quand c’est le temps de retirer, les deux modules vont demander au contrôleur de l’ATM pour qu’il appelle sa fonction qui communique au distributeur de billets afin de faire sortir le montant exact passé en paramètre.

Vue Architecturale

Diagramme ­ Scénario I:

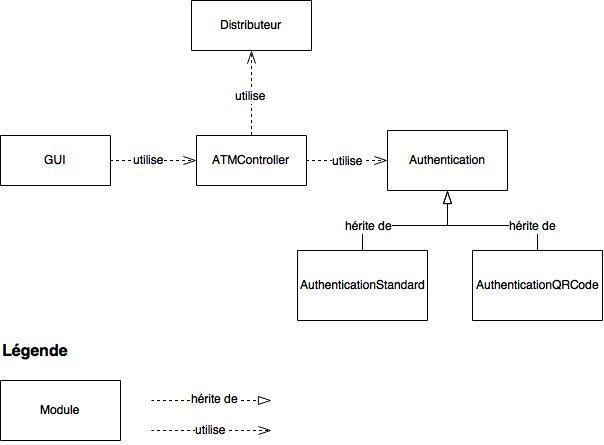


Le diagramme en couche représente la tactique pour le scénario 1.

| Éléments | Descriptions |
| --- | --- |
|  |  |
| GUI | La couche qui représente l’interface d’utilisateur. |
|  |  |
| Transactions | La couche contenant les types de transaction. |
|  |  |
| ATM | La couche qui représente le logique de la machine. |
|  |  |
| Base de données | La couche représentant la ou les base de données |
|  | de la banque. |
|  |  |

Pour faire le lien avec le scénario et le choix de la tactique, le couche Transactions a été ajoutée afin de permettre d’éviter de faire des modifications qui sont déjà stable et sécuritaire, donc cela veut dire qu’il n’y a pas de changements dans la couche ATM ou les couches inférieures. Si on veut ajouter un nouveau type de module de transaction,il faut simplement l’ajouter dans la couche Transactions.

Diagramme ­ Scénario II:



Le diagramme représente la tactique pour le scénario 2. Il montre les liens entre les modules qui sont affectés par le changement du système.

| Éléments | Descriptions |
| --- | --- |
|  |  |
| GUI | Le module qui représente l’interface d’utilisateur de la machine ATM. |
|  |
|  |  |
| ATMController | Le module qui représentelecontrôleurdel’ATM.Il  permet de rediriger les requêtes à la bonne service. |
|  |
|  |  |
| Authentication | Le module qui offre le service pour s’authentifier de deux manières : Standard ou Code QR. |
|  |
|  |  |
| Distributeur | Le module qui communique avec le distributeur de billets (composant physique). |
|  |
|  |  |

Pour faire le lien avec le scénario et le choix de la tactique, le module ATMController est le coeur du système et que peu importe la méthode d’authentification, il va toujours rediriger les requêtes à la bonne direction. Cela veut donc dire que si l’utilisateur veut retirer un montant, méthode standard ou code QR, le module ATMController va s’occuper de l’authentification.

LOG430 – Architecture logicielle

**Scénarios de qualité, tactiques et justifications des attributs produits par les équipes en classe**

**Modificabilité – Justification**

La modificabilité est un attribut de qualité important dans le cas d’un guichet automatique bancaire, car il faut que ce guichet puisse être capable de s’adapter aux nouvelles technologies (ex: empreinte digitale). De plus, il faut aussi que le guichet puisse s’adapter aux nouvelles règles d’affaires.

**Modificabilité – Scénario 1**

# Ajout d’une nouvelle fonctionnalité (Fingerprint)

| **Scénario** | Ajout d’une nouvelle fonctionnalité (Fingerprint) |
| --- | --- |
| **objectifs d'affaires** | Être capable d’ajouter une fonctionnalité sur le guichet sans procéder à une réingénierie du système. |
| **Attribut de qualité** | Modificabilité |
| **Source** | Développeur |
| **Stimulus** | À une demande d’ajouter une fonctionnalité (Fingerprint) sur les guichets. |
| **Artéfact** | Code |
| **Environnement** | Temps de développement |
| **Réponse** | * Le gestionnaire calcul le nombre d’employés et le temps pour entreprendre ce projet. * Un fonctionnaire autorise le gestionnaire de commencer le projet. * Les développeurs programme le nouveau plugin de cette fonctionnalité. * Le plugin est compilé et testé par des testeurs. * Lorsque la plupart des défauts ont été trouvé, le nouveau code est poussé sur les guichets par des techniciens. |
| **Mesure de la réponse** | * Le coût du projet. * Le temps dédié au projet. * Le nombres d’employés sur le projet. * Le nombre de nouveaux défauts introduits. |
| **Questions** | g - Est-il rentable de faire la modification?   * Quelles sont les composantes physiques d’un module de FingerPrint? * Quels sont les défauts que cette nouvelle fonctionnalité peut créer? |

# 

Modificabilité – Scénario 2

# Changement de règle d’affaire sur le serveur bancaire

| **Scénario** | Changement de règle d’affaire sur le serveur bancaire |
| --- | --- |
| **objectifs d'affaires** | Ajouter ou modifier les règles bancaires sur le serveur avec facilité et dans un temps minimum. |
| **Attribut de qualité** | Modificabilité |
| **Source** | Mainteneur |
| **Stimulus** | À une demande de modifier une règle d’affaire sur le serveur. |
| **Artéfact** | Code |
| **Environnement** | Temps de maintenance |
| **Réponse** | * Le gestionnaire calcul le temps et les ressources dédiés pour le projet. * Un ou des mainteneur(s) font les modifications du code relié aux règles d’affaire. * Les nouvelles modifications sont testées par des testeurs. * Un technicien pousse les nouvelles modifications sur les serveurs choisis. |
| **Mesure de la réponse** | * Le temps dédié à l’analyse du projet. * Le temps dédié aux modifications * Le temps dédié aux tests * Le coût relié aux modifications * Le nombre d’employés qui ont travaillé sur le projet. |
| **Questions** | 1. - Est-ce que les règles d’affaire changent souvent?  - Est-il rentable d’intégrer un système de plugins?  - Est-ce que les règles d’affaire changent souvent selon la région? |

# 

**Modificabilité – Tactique 1 (Split Module)**

**Description**: Cette tactique consiste a séparer les modules plus gros en leurs composantes les plus basique pour faciliter la réutilisation des sous-modules.

**Justification**: Décomposer les modules en composantes plus petites les rend plus facile à échanger, modifier ou les réutiliser plus tard sans causer de dérangements dans les autres parties du code.

**Modificabilité – Tactique 2 (Restrict Dependencies)**

**Description**: Réduire les dépendances entre les différents modules du système.

**Justification**: En rendant les différents modules plus indépendants les uns des autres, les modifications effectuées sur un module auront un moins grand impact sur les autres aux alentours, réduisant ainsi le temps et le coût de développement.