Scénarios et tactiques de testabilité

[1.Un nouveau programme d’ATM est lancé, nous voulons tester avec une base de donnée](#_gjdgxs)

[Tactique: Sandbox](#_30j0zll)

[2.On veut pouvoir repérer rapidement la source d’une défaillance dans un système lorsqu’elle n’est pas connue](#_1fob9te)

[Tactique: Executable assertion](#_3znysh7)

[3.Augmenter la couverture des tests sur l’application](#_tyjcwt)

[Tactique: Specialized Interfaces](#_3dy6vkm)

[Tactique: Executable Assertions](#_2s8eyo1)

[4.Tests unitaires du lecteur de carte d'accès du guichet.](#_17dp8vu)

[Tactique: Assertions exécutables](#_3rdcrjn)

[Tactique: Limiter les éléments non déterministes](#_26in1rg)

[5.Tests d’intégration des diverses composantes du guichet.](#_lnxbz9)

[Tactique: Enregistrer et rejouer](#_35nkun2)

[Tactique: Limiter la complexité structurelle](#_1ksv4uv)

[6.Améliorer la testabilité en vue d’un ajout de fonctionnalité de transaction au système actuel.](#_z337ya)

[Tactique: Limit structural complexity](#_3j2qqm3)

[7.Un test unitaire vérifie qu’il est impossible de retirer plus d’argent que la solde du compte.](#_1y810tw)

[Tactique: Executable assertion](#_4i7ojhp)

| **Scénario** | 1.Un nouveau programme d’ATM est lancé, nous voulons tester avec une base de données |
| --- | --- |
| **objectifs d'affaires** | Tout doit marcher lorsqu’on lance le programme en production |
| **Source** | Développeur |
| **Stimulus** | Les tests sur la base de donnée sécurité n’utilise pas celle de la production |
| **Artéfact** | Les bases de données et le code |
| **Environnement** | Development time, deployment time, integration time, compile time, design time |
| **Réponse** | Execute test suite |
| **Mesure de la réponse** | Effort to find a fault or class of faults |
| **Questions** | 1. Y-a t-il beaucoup de bug dans le système 2. Combien de temps en effort-personne cela va prendre pour arranger les bug |

### Tactique: Sandbox

**Description**: Il s’agit de tester la nouvelle partie du logiciel dans un “Sandbox”, ce qui permet tester facilement sans affecter les systèmes en production. Tout est en quelque sorte isolée.

**Justification**: Si des modifications a la nouvelle base de données sont effectués lors des tests, il est possible que des défaillances surviennent, Ces défaillances pourraient se répercutés vers d’autres bases de données et/ou vers d’autres système. En utilisant un sandbox, le système en production est protégé.

| **Scénario** | 2.On veut pouvoir repérer rapidement la source d’une défaillance dans un système lorsqu’elle n’est pas connue |
| --- | --- |
| **objectifs d'affaires** | 95% des défaillances doivent être trouvés en moins de 3 heures. |
| **Source** | Le développeur |
| **Stimulus** | Les asserts |
| **Artéfact** | Le code |
| **Environnement** | Developpement time |
| **Réponse** | Execute test suite |
| **Mesure de la réponse** | Effort to detect faults |
| **Questions** | 1. Y a-t-il assez d’asserts? 2. Quoi devrais t’on asserter? |

### Tactique: Executable assertion

**Description**: Les asserts servent à logger les problèmes qui se produise dans le logiciel.

**Justification**: Très important de logger les erreurs qui ne font pas crasher l’application mais qui ont un impact sur celui-ci.

| **Scénario** | 3.Augmenter la couverture des tests sur l’application |
| --- | --- |
| **objectifs d'affaires** | Les tests doivent couvrir 90% de l’application |
| **Source** | Testeurs unitaires |
| **Stimulus** | L’implémentation complète du système |
| **Artéfact** | Le système au complet, le code source |
| **Environnement** | Temps de développement |
| **Réponse** | Exécution de la suite de tests et capture des résultats  Control et monitoring du système |
| **Mesure de la réponse** | Réduction des risques  Probabilité de découvrir une erreur lors des prochains tests  Effort pour arriver à couvrir 90% de l’application  Temps d’exécution |
| **Questions** | 1. Combien de temps et d’argent est-ce qu’on a besoin pour développer les tests? 2. Quel est le pourcentage du programme qui a déjà été testé? 3. Combien de tests avons nous besoin pour couvrir la totalité du système? 4. Combien de fautes ont été découverts après l’exécution des tests? |

### Tactique: Specialized Interfaces

**Description**: Utiliser des interfaces spécialisées pour accéder aux données.

**Justification**: Mettre en place l'utilisation d'interface spécialisée va permettre de vérifier l'état des données durant l'exécution et de s'assurer de l'exactitude des informations. Cela va favoriser la détection des incohérences.

### Tactique: Executable Assertions

**Description**: Tests de vérification qui permettent de vérifier des assomptions

**Justification**: Permet de vérifier la validité des fonctions et assure que les résultats engendrer durant l’exécution reste identique. Assure que les modifications effectuées dans le système engendre les résultats voulus.

| **Scénario** | 4.Tests unitaires du lecteur de carte d'accès du guichet. |
| --- | --- |
| **objectifs d'affaires** | S'assurer que le lecteur de carte répond aux exigences fonctionnelles et aux attributs de qualités établie par l'architecte. |
| **Source** | Le testeur |
| **Stimulus** | exécute un jeu de tests unitaires pour s'assurer que le lecteur de carte fonctionne bien. |
| **Artéfact** | Le lecteur de carte d’accès du guichet. |
| **Environnement** | Durant la conception. |
| **Réponse** | Les tests sont exécutés. |
| **Mesure de la réponse** | Tous les tests ont réussi. |
| **Questions** | 1. Est-ce que la couverture des tests est suffisante ? 2. Est-ce que les tests actuels sont bien conçus (ont-ils été vérifiés par plus d’une personne) ? 3. Faut-il produire un rapport de tests ? |

### Tactique: Assertions exécutables

**Description :** Les tests unitaires comportent des assertions afin de valider que le lecteur retourne les données attendues en fonction des entrants qu'on lui fournit.

**Justification :** C'est une façon très efficace de concevoir des tests unitaires.

### Tactique: Limiter les éléments non déterministes

**Description :** Le lecteur de carte doit toujours produire les mêmes résultats pour des intrants identiques.

**Justification :** De cette façon, on peut assurer le bon fonctionnement du lecteur à l'aide de tests simples.

| **Scénario** | 5.Tests d’intégration des diverses composantes du guichet. |
| --- | --- |
| **objectifs d'affaires** | Les multiples composantes du guichet doivent être capable de communiquer avec le bus principal du guichet pour en assurer son bon fonctionnement. |
| **Source** | Les testeurs d’intégration |
| **Stimulus** | exécutent les tests d'intégration du système de communication principal entre les diverses composantes du guichet. |
| **Artéfact** | Le bus principal du guichet. |
| **Environnement** | La période d'intégration. |
| **Réponse** | Capturer les résultats des tests d’intégration. |
| **Mesure de la réponse** | Tous les tests doivent être exécutés à l'intérieur d'une période de trois heures. |
| **Questions** | 1. Est-ce que les composantes communiquent entre elles d'une autre façon que par le bus principal du guichet ? 2. Doit-on faire plus de tests ? 3. Faut-il produire un rapport ? |

### Tactique: Enregistrer et rejouer

**Description :** Simuler des messages envoyées aux composantes du système et analyser leurs réponses.

**Justification :** Après modification d'une composante, on va pouvoir re-tester et comparer les résultats pour savoir si la situation s'améliore ou se détériore.

### Tactique: Limiter la complexité structurelle

**Description :** Chaque composantes doit avoir un fort niveau de cohésion. Les composantes sont alors plus petites, plus simple à implémenter et à tester.

**Justification :** Augmenter la cohésion permet indirectement de réduire la complexité, d'augmenter la modularité et la testabilité.

| **Scénario** | 6.Améliorer la testabilité en vue d’un ajout de fonctionnalité de transaction au système actuel. |
| --- | --- |
| **Objectifs d'affaires** | Augmenter les profits occasionnés par les prochaines versions du système en diminuant les coûts de testabilité du système actuel en prévision des changements à venir. |
| **Source** | Unit testers, integration testers, automated testing |
| **Stimulus** | Complétion d’un nouvel incrément de code due à l’ajout de fonctionnalité d’un nouveau type de transaction |
| **Artéfact** | Système en entier (il faut tester les attributs de qualité ainsi que les exigences fonctionnelles du système post modification) |
| **Environnement** | Development time |
| **Réponse** | Les tests certifiant l’intégrité de nos transactions sont exécutés avec succès. |
| **Mesure de la réponse** | Temps pour écrire les tests  Temps d’exécution des tests en bas d’une minute pour 1000 tests  Pourcentage de couverture des fonctionnalités par les tests 90% code coverage |
| **Questions** | Quel est le niveau satisfaisant de code coverage?  Comment pouvons-nous reconnaître une mauvaise transaction?  Quels sont les valeurs à tester? Ex: min, max (Boundary testing)  Quelles divisions incorporés dans notre architecture pour réduire la complexité structurale et améliorer l’encapsulation?  Quelles sont les exigences fonctionnelles actuelles?  Quels sont les attributs de qualités actuels ? |

### Tactique: Limit structural complexity

**Description**: Technique pour résoudre les dépendances cycliques et simplification de l’héritage, dans le but de réduire la complexité de la structure et améliorer l’encapsulation

**Justification**: Dans notre cas, nous voulons réduire la complexité en vue d’un ajout de fonctionnalité dans un module avec des classes ayant des responsabilités similaires. Or, une structure plus simple est souvent lié à un nombre moins élevé de classes, ce qui facilite la création des tests.Dans cet ordre d’idée, nous pensons que cette tactique est adaptée à nos besoins et nous permettra de faire cette modification en minimisant les dégâts potentiels, et par le même fait, les coûts de tests de cette modification.

| **Scénario** | 7.Un test unitaire vérifie qu’il est impossible de retirer plus d’argent que la solde du compte. |
| --- | --- |
| **objectifs d'affaires** | Avoir un système couvert a 99% par les tests unitaires. |
| **Source** | Le développeur |
| **Stimulus** | Le test unitaire est lancé suite à la complétion de la fonctionnalitée |
| **Artéfact** | Le code de la transaction retrait |
| **Environnement** | développement |
| **Réponse** | Résultat du tests |
| **Mesure de la réponse** | Pourcentage de tests qui sont des succès.  Pourcentage de code couvert par les tests.  Durée des tests |
| **Questions** | 1. Quel pourcentage de code doit être couvert? 2. Quelles fonctionnalitées doivent être testé? 3. Quel est le budget alloué au tests? |

### Tactique: Executable assertion

**Description:** On ajoutes des *assert* dans le code

**Justification:** Cela nous donne un indicateurs instantané de quand les résultats attendu ne corresponde pas à la réalité.