Scénarios, tactiques et vues Architecturales

Usabilité

[**1 On veut qu’un utilisateur puisse annuler une transaction**](#_x28itky74d6h) **4**

[1.1 Tactique 1 : Cancel](#_a8sspkm7rpqm) 4

[1.2 Vue architecturale (à l’extérieur de la classe)](#_ni1zxdxzo6ps) 4

[1.2.1 Description du diagramme](#_1fob9te) 5

[1.2.2 Description des éléments](#_3znysh7) 5

[1.2.3 Relation entre éléments et tactiques](#_2et92p0) 5

[1.2.4 Tactique 1 : Cancel](#_tyjcwt) 5

[**2 Scénario : Transfert de compte en erreur**](#_snzul5rkamh7) **6**

[2.1 Tactique 1 : Undo](#_o4c88yvdixk6) 6

[2.2 Tactique 2 : Cancel](#_4bayzicd7cdn) 6

[2.3 Diagramme](#_3dy6vkm) 7

[2.3.1 Texte de description du diagramme](#_l85rmnx3uuvs) 8

[2.3.2 Table de description des éléments du diagramme](#_siv585hamdxh) 8

[2.3.3 Texte décrivant la relation entre les éléments et les tactiques](#_xz65n1vn98h7) 8

[**3 Scénario :**](#_bwtwd6h7lxtl) **9**

[3.1 Tactique 1 : Support System Initiative (Maintain Task Model)](#_cnii2ntw3wcm) 9

[3.2 Vue architecturale (à l’extérieur de la classe)](#_9dmsttrvwpi3) 10

[3.2.1 Diagramme](#_1msslngt53dd) 10

[3.2.2 Description du diagramme](#_xh3d8n3rwg4d) 10

[3.2.3 Table de description des éléments du diagramme](#_pvo448rsedk0) 11

[3.2.4 Relation entre les éléments et les tactiques](#_l3ib48uzxwuj) 11

[**4 Scénario : Le client fait une erreur lors lors de son interaction avec le guichet.**](#_v99j2t5fufcd) **12**

[4.1 Tactique 1 : Undo](#_ozthowna1hq) 12

[4.2 Tactique 2 : Cancel](#_6sgjfp7uje5b) 12

[4.3 Vue architecturale (à l’extérieur de la classe)](#_m1flvau4hiyl) 13

[4.3.1 Description:](#_gfoxt84g7tin) 13

[4.3.2 Relation avec la tactique:](#_3ek5sum4v4kn) 13

[4.3.3 Composantes :](#_23cgsr5viidc) 13

[4.4 Tactique 1 : Cancel](#_577nf5yl0mbp) 14

[4.5 Tactique 2 : Maintain Task Model](#_2lp3phcsrlpt) 14

[4.6 Diagramme 1.0 Modélisation de la sécurité du système d'ATM.](#_1t3h5sf) 15

[4.6.1 Description du diagramme:](#_5gtq6mnsc556) 15

[4.6.2 Deux composants du système, l’ATM et l’interface du système ont des relations d’usabilité en cas d’annuler une transaction en cours. Les deux systèmes respecteront la tactique de « Cancel » pour permettre aux utilisateurs d’annuler de transactions en cours. De plus, l’ATM respectera la tactique de « Maintain Task Model » pour que l’utilisateur soit informé par le système qu’il est possible d’annuler une transaction.](#_4d34og8) 15

[**5 Scénario : Présentation à l’usager de différents choix de montants d’argent lors d’un retrait ou d’un transfert**](#_cqs99dvj3zyz) **16**

[5.1 Tactique 1 : ”Maintain task model”](#_8i5d2qans0w1) 16

[5.2 Vue architecturale](#_9befwcy4rfde) 17

[5.2.1 Texte de description du diagramme](#_17dp8vu) 17

[5.2.2 Table de description des éléments du diagramme](#_ux9t0mn8cjtg) 18

[5.2.3 Texte décrivant la relation entre les éléments et les tactiques](#_btn0i5axt88h) 18

[**6 Scénario: Pour un utilisateur qui fait toujours les même transactions, il doit pouvoir faire ses transactions plus rapidement et facilement.**](#_3rdcrjn) **19**

[6.1 Tactique 1 : Aggregate](#_7o03jreaq9af) 19

[6.2 Tactique 2 : Maintain User Model](#_ga22yxusxvwe) 19

[6.3 Vue architecturale (à l’extérieur de la classe)](#_nfu41c5j2fsu) 21

[6.3.1 Diagramme](#_26in1rg) 21

[**7 Scénario : L’utilisateur désire de faire une transaction dépôt**](#_y7vy8ux9ai6c) **23**

[7.1 Tactique 1 : Undo](#_t0q5npma0fip) 23

[7.2 Tactique 2 : Cancel](#_1yogkhlnc68) 23

[7.3 Vue architecturale (à l’extérieur de la classe)](#_eq3tcek5xsbg) 24

[7.3.1 Description](#_wwluxgut1y9h) 25

[7.3.2 Relation avec la tactique](#_1335efnvmbvc) 25

[7.3.3 Composantes :](#_uc4n0lcwyvds) 25

[**8 Un utilisateur insère sa carte pour effectuer une transaction mais décide au milieu des étapes qu’il veut l’annuler et oublie sa carte.**](#_mzvu2omeawth) **26**

[8.1 Tactique 1 : Support User Initiative - Cancel](#_bc4llzn7p5kg) 26

[8.2 Tactique 2 : Support System Initiative - Maintain System Model](#_4xsg1eehrfox) 26

[8.3 Vue architecturale (à l’extérieur de la classe)](#_q78g4i7ujjga) 26

[8.3.1 Diagramme](#_3j4gnllk4c3s) 26

[8.3.2 Légende](#_jf8cacjh5k9d) 27

[8.3.3 Texte de description du diagramme](#_v1f2ddxpn5lj) 27

[8.3.4 Table de description des éléments du diagramme](#_g6tfqvec1nzf) 28

[8.3.5 Texte décrivant la relation entre les éléments et les tactiques](#_j0vwq4uhmv56) 28

[**9 Scénario : Permettre l’annulation d’une transaction dans un certain délai**](#_35nkun2) **28**

[9.1 Tactique 1 : Undo](#_yagocjwvucp3) 28

[9.2 Tactique 2 : Cancel](#_u0bsbayhma3g) 29

[9.3 Vue architecturale](#_di124lx9g9na) 29

[9.3.1 Diagramme](#_1ksv4uv) 29

[9.3.2 Légende](#_8k70qnlztfs) 29

[9.3.3 Texte de description du diagramme](#_42mcgg6epk7j) 29

[9.3.4 Table de description des éléments du diagramme](#_z63watd37f04) 30

[9.3.5 Texte décrivant la relation entre les éléments et les tactiques](#_jmdgchx4hsy9) 30

# 1 On veut qu’un utilisateur puisse annuler une transaction

|  |  |
| --- | --- |
| **objectifs d'affaires** | On veut qu’un utilisateur puisse utiliser plus facilement le GAB lors d’une transaction |
| **Source** | User |
| **Stimulus** | User utilise le système efficacement |
| **Artéfact** | Le GUI lors d’une transaction |
| **Environnement** | runtime |
| **Réponse** | Le système offre la fonctionnalité d'annuler une transaction |
| **Mesure de la réponse** | La satisfaction du user (en moyenne 4 sur une échelle de 1 à 5) |
| **Questions** |  |

## 

## 1.1 Tactique 1 : Cancel

**Description**: Cette tactique permet d'annuler une tâche du système.

**Justification**: Donné la possibilité à l’utilisateur d'annuler une transaction au lieu de faire deux transactions , soit une étant l'erreur et l'autre pour la défaire (exécuter l’opération inverse)

## 1.2 Vue architecturale (à l’extérieur de la classe)

### 

### **1.2.1 Description du diagramme**

Le système du GAB permet à l’utilisateur d’entrer une commande annuler. Celle-ci sera transmise au contrôleur qui décidera quoi faire avec. S’il est en train d’utiliser un transaction, il dira à celle-ci de s’annuler. Chaque type de transaction saura comment s’annuler elle-même.

### **1.2.2 Description des éléments**

**GAB** : Package contenant les fonctionnalité du GAB.

**BankingController** : Classe représentant le contrôleur du système.

**ITransaction** : Interface représentant le modèle d'un transaction.

**Depot** : Classe représentante la transaction de dépôt.

**Retrait** : Classe représentant la transaction de retrait.

**Input** : Classe représentant les entrés des usagers.

**GUI** : Package contenant les élément de l'interface utilisateur.

### **1.2.3 Relation entre éléments et tactiques**

### **1.2.4 Tactique 1 : Cancel**

ITransaction oblige l'implémentation de la fonction "cancel" à tous les transactions, ce qui permet à l'utilisateur de choisir l'option d'annulation pour appeler le contrôleur qui fera annuler la transaction.

# 2 Scénario : Transfert de compte en erreur

|  |  |
| --- | --- |
| **Objectifs d'affaires** | On voudrait que le nombre de transfert annulés diminue de 80% en un mois |
| **Source** | Client possédant un compte. |
| **Stimulus** | Minimiser les impacts d’erreurs de transferts bancaire. |
| **Artéfact** | Le guichet |
| **Environnement** | Runtime |
| **Réponse** | Affiche un message de confirmation avec une demande de validation et permettre d’annuler un transfert en dedans d’une heure. |
| **Mesure de la réponse** | Le nombre de transferts annulés par le service à la clientèle |
| **Questions** | 1. Combien de transfert sont-ils annulés par mois? 2. Quelle est la satisfaction générale des clients envers le guichet? |

## 2.1 Tactique 1 : Undo

**Description**: Garder en mémoire les informations des transferts permettant ainsi d’effectuer un retour à l’état précédant la demande de transfert.

**Justification**: Étant donné qu’il faut permettre le transfert de compte, il faut automatiquement implanter un moyen de permettre un retour en arrière d’un transfert. Par exemple , un problème technique ou un annulation par le client nécessite un retour en arrière.

## 2.2 Tactique 2 : Cancel

**Description**: Permettre à tout moment lors du transfert de compte d’annuler la transaction.

**Justification**: Un événement lors d’un transfert peut nécessité une annulation soudaine d’un transfert. L’ajout de la tactique d’annulation permet de libérer les ressources utilisés en plus de retourner à l’état avant l’évènement ayant été annulé.

Vue architecturale (à l’extérieur de la classe)

## 2.3 Diagramme



### 2.3.1 Texte de description du diagramme

Le diagramme ci dessus nous montre les différentes options afin de confirmer au client si les paramètres de la transaction sont bons. On peut annuler la transaction si les comptes ne sont pas bons et on peut modifier le montant de la transaction si jamais les comptes sont bons mais le montant n’est pas bon.

### 2.3.2 Table de description des éléments du diagramme

|  |  |
| --- | --- |
| Élément | Fonction |
| Bouton du clavier | Sert d’input au GUI |
| Écran | Sert à afficher les informations qui sont traitées par le guichet. |

### 2.3.3 Texte décrivant la relation entre les éléments et les tactiques

La tactique Undo est représenté par le fait que qu’il y a un bouton 0 pour revenir en arrière afin de modifier la transaction au cas ou une erreure a été commise.

La tactique Cancel est représenté par le fait qu’il y a un bouton cancel pour annuler le transfert et cela va donc permettre au système de retourner à l’écran d'accueil pour pouvoir permettre à l’utilisateur de rentrer une nouvelle transaction.

# 3 Scénario :

|  |  |
| --- | --- |
| **objectifs d'affaires** | La banque veut qu’un utilisateur non-régulier puisse comprendre en moins de 20 secondes comment l’ATM fonctionne pour faire ses actions. |
| **Source** | Un utilisateur |
| **Stimulus** | Un utilisateur non-régulier utilise l’ATM pour faire un retrait. |
| **Artéfact** | Interface de l’ATM que l’utilisateur utilise |
| **Environnement** | Runtime |
| **Réponse** | L’ATM anticipe les besoins en donnant des petites explications simple et précises tout au long du processus. |
| **Mesure de la réponse** | User satisfaction and amount of time. |
| **Questions** |  |

## 3.1 Tactique 1 : Support System Initiative (Maintain Task Model)

**Description**: Afficher des bulles d’informations tout au long du processus de retrait en appuyant sur un bouton qui permet d’en activer l’usage en utilisant une architecture MVC.

**Justification**: Permet aux utilisateurs non-réguliers d’avoir de l’aide rapidement et efficacement et pour les utilisateurs réguliers, avoir le choix de l’enlever l’aide rendra leur utilisation rapide, car ils n’auront pas à attendre que les infos-bulles s’affichent.

## 3.2 Vue architecturale (à l’extérieur de la classe)

### 3.2.1 Diagramme

Légende

|  |  |
| --- | --- |
|  | Un client qui utilise l’ATM. |
|  | ATM |

### 3.2.2 Description du diagramme

Lorsqu’un client clique sur un bouton dans l’interface pour activer ou désactiver les bulles d’aides, le système va changer de controller pour utiliser celui associé au bon comportement et changer les vues à afficher soit avec les bulles ou sans les bulles d’aide.

### 3.2.3 Table de description des éléments du diagramme

|  |  |
| --- | --- |
|  | Un utilisateur qui active ou désactive l’option pour avoir l’affichage des bulles d’aides. |
|  | Les controllers qui permettent soit d’afficher les vues avec infobulles avec des actions de contrôle spécifiques à celles-ci, soit de ne pas les afficher. |
|  | Les modèles de données qui ne changent pas dans cette option là, mais sont pour illustrer l’architecture MVC. |
|  | Ce sont les différentes vues qui possèdent les infos-bulles et celles qui n’en possèdent pas. |

### 3.2.4 Relation entre les éléments et les tactiques

L’architecture MVC va permettre de pouvoir faire des révisions de l’interface facilement et rapidement des vues avec et sans bulles, sans risquer d’augmenter le risque de problèmes, car on ne touche pas aux controlleurs.

# **4 Scénario : Le client fait une erreur lors lors de son interaction avec le guichet.**

|  |  |
| --- | --- |
| **Objectifs d’affaires** | Permettre au client de revenir en arriere apres une entrée ou une sélection pour augmenter sa satisfaction. |
| **Source** | Le client |
| **Stimulus** | ~~Le client~~ effectue une erreur lors d’une transaction et on souhaite en minimiser l’impact |
| **Artéfact** | Console client |
| **Environnement** | Runtime |
| **Réponse** | Le système offre l’option de corriger toute information entrée par le client. |
| **Mesure de la réponse** | * La satisfaction du client * Le ratio de succes des transactions. |
| **Questions** | * Le client est-il satisfait de son expérience * Le client se sent-il à l’aise de faire une erreur * Quel est le nombre de transaction contenant une erreur. |

## **4.1 Tactique 1 : Undo**

**Description**: Permet de revenir en arrière sur une action en conservant l’état précédent.

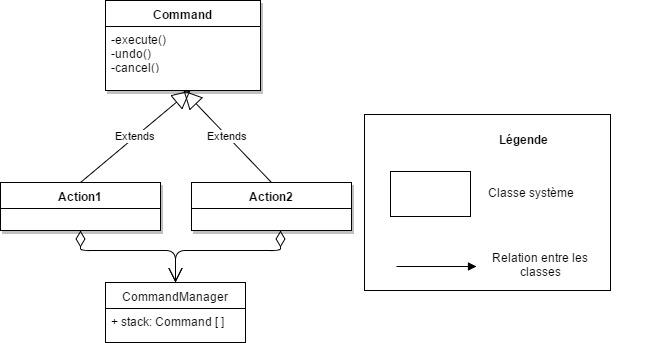
**Justification**: Quand il navigue dans les menus, l’usager doit pouvoir revenir en arrière en tout temps pour modifier une information qui serait erroné. Cette action prend place avant que la transaction soit envoyé au système bancaire.

## 4.2 Tactique 2 : Cancel

**Description**: Permet à l’usager de complètement annuler une action dans laquelle il identifie une erreur.

**Justification :** Quand une transaction est en cours de traitement, l’usager doit pouvoir annuler cette transaction si l’action est posé avant la complétion de la transaction avec le système bancaire.

## **4.3 Vue architecturale (à l’extérieur de la classe)**



### 4.3.1 Description:

Chaque action que l’utilisateur peut effectuer *extend* la class Command. Chaque action aura une fonction undo qui contiendra la logique permettant d’annuler l’action effectué. Toutes les actions seront agrégé dans la classe CommandManager.

### **4.3.2 Relation avec la tactique:**

Toutes les actions sont des Command

Elles sont créées et contenues dans CommandManager.

### 4.3.3 Composantes :

*Command*

Classe abstraite

*Action*

Classe qui représente une action que l’utilisateur peut effectuer. Ex: ChoixDeCompte

*CommandManager*

Sert de factory et contient les actions pour permettre leurs annulation en chaîne si nécessaire.

5 Scénario : L’utilisateur désire annuler une transaction en cours

|  |  |
| --- | --- |
| **objectifs d'affaires** | Permettre 99,99 % du temps d’annuler des transactions aux utilisateurs de l’ATM. |
| **Source** | Humain, utilisateur |
| **Stimulus** | Tentative d’annulation de transaction |
| **Artéfact** | module de transactions du système |
| **Environnement** | Mode d'opération normal |
| **Réponse** | 1. Journalisation de la transaction,  2. vérifier l’information de l’utilisateur  3. Annuler la transaction  Informer à l’utilisateur l’état de son compte  Revenir à l’interface initiale du système |
| **Mesure de la réponse** | Nombre de transactions annulées par les utilisateurs |
| **Questions** | 1. À quel moment l’utilisateur voudra annuler une transaction ? 2. Combien de fois l'utilisateur peut annuler des transactions? |

## 4.4 Tactique 1 : Cancel

**Description**: L’utilisateur veut annuler une transaction. Le système doit offrir cette option dans ce cas.

**Justification**: Le système doit offrir l’option annuler une transaction.

## 4.5 Tactique 2 : Maintain Task Model

**Description**: Lors du traitement de la transaction, le système doit savoir que l’utilisateur pourrait vouloir annuler une transaction et offrir cette option sur l’interface.

**Justification**: L’utilisateur doit savoir qu’il est possible d’annuler une transaction. Le fait que le système affiche cette option va montrer la possibilité qu’a l’utilisateur.

## 4.6 Diagramme 1.0 Modélisation de la sécurité du système d'ATM.

### 

### 4.6.1 Description du diagramme:

### 4.6.2 Deux composants du système, l’ATM et l’interface du système ont des relations d’usabilité en cas d’annuler une transaction en cours. Les deux systèmes respecteront la tactique de « Cancel » pour permettre aux utilisateurs d’annuler de transactions en cours. De plus, l’ATM respectera la tactique de « Maintain Task Model » pour que l’utilisateur soit informé par le système qu’il est possible d’annuler une transaction.

**Table de description des éléments du diagramme :**

|  |  |
| --- | --- |
| **Tactiques** | **Descriptions** |
| Cancel | Permets d’annuler une transaction en cours |
| Maintain Task Model | Informer à l’utilisateur qu’une transaction peut être annulée si elle est en cours. |

**Relations :**

Les lignes de l'ATM vers l’interface représentent la tactique de « Cancel ». De plus, la ligne qui sort et revient vers l’ATM représente la tactique de « Maintain Task Model ».

# 5 Scénario : Présentation à l’usager de différents choix de montants d’argent lors d’un retrait ou d’un transfert

|  |  |
| --- | --- |
| **objectifs d'affaires** | L’utilisateur doit pouvoir être présenté de divers choix de montant d’argent à retirer (ou à transférer) pour rendre la fonctionnalité plus rapide et éviter des erreurs |
| **Source** | Le client au guichet automatique bancaire |
| **Stimulus** | L’utilisateur apprend les propositions du GAB (qui sont toujours les mêmes) et cela augmente la rapidité d’utilisation et minimise le risque d’erreurs |
| **Artéfact** | L’interface (l’écran) qui propose des choix de montants |
| **Environnement** | Pendant que le GAB est en exécution et fonctionnel |
| **Réponse** | Le système anticipe un montant de retrait/transfert souhaité par l’utilisateur (et si ce montant n’est pas affiché, l’interface permet à l’utilisateur de choisir le sien dans un écran qui suit les choix). |
| **Mesure de la réponse** | L’utilisateur qui s’habitue au GAB sera de plus en plus rapide pour effectuer la tâche puisqu’il sait exactement où se situe les choix dans l’interface et cela réduit considérablement le risque d’erreurs d’entrer un montant invalide puisqu’il n’y a qu’un bouton à appuyer pour sélectionner un bon montant. |
| **Questions** | 1. Quel est le temps souhaité pour effectuer la tâche? 2. Combien de choix doit-il y avoir à l’écran? 3. Quels sont les propositions de choix de montants? 4. Quels fonctionnalités du GAB doivent être affectés par l’interface (retrait, dépôt, etc.)? 5. Pouvons-nous entrer un montant personnalisé pour retirer de l’argent? |

## 5.1 Tactique 1 : ”Maintain task model”

**Description**: Le système assiste l’usager dans ses opérations, dans notre situation en lui offrant des options pour les retraits ou les transferts.

**Justification**: Offrir des montants lors de ces opérations permet de sauver du temps à l’utilisateur, il n’a pas à saisir un montant, il n’a qu’à en sélectionner un ce qui lui sauve une certaine quantité de boutons à appuyer et réduit également le risque d’erreurs puisque l’utilisateur n’a pas à entrer un montant personnalisé à priori.

## 5.2 Vue architecturale

### 

### 5.2.1 Texte de description du diagramme

La vue architecturale “décomposition” ci-haut représente les divers éléments d’interface faisant partie du logiciel du GAB. On peut y retrouver un UI spécifiquement pour les choix de montants apparaissant à l’écran que l’utilisateur peut sélectionner. Tous ces interfaces font partie de l’interface global du GAB et il est possible de les modifier indépendamment. Ainsi, si l’utilisateur désire avoir plus de choix ou moins de choix, l’utilisabilité de l’application peut être améliorée en un clin d’oeil.

### 5.2.2 Table de description des éléments du diagramme

Le diagramme contient un élément qui est utilisé à plusieurs niveaux. Le “GAB\_UI” est un ensemble de plusieurs modules qui représente l’ensemble de la tactique qui est d’afficher des choix de montants. Tous les éléments contenus dans ce groupe de modules sont des modules qui représente des interfaces distinctes qui peuvent être affichés selon les actions de l’utilisateur.

### 5.2.3 Texte décrivant la relation entre les éléments et les tactiques

La relation entre les éléments en est une d’inclusion. En effet, on peut constater que chacun des modules d’interface est contenu dans le module de l’interface global du GAB.

# **6 Scénario: Pour un utilisateur qui fait toujours les même transactions, il doit pouvoir faire ses transactions plus rapidement et facilement.**

|  |  |
| --- | --- |
| **Objectifs d'affaires** | Minimizer le nombre d’étapes (clics) à 1 pour une transaction. |
| **Source** | Usager (le client) |
| **Stimulus** | Adapter l’efficacité du système pour l’utilisateur. |
| **Artéfact** | Le système.  Interface usagé |
| **Environnement** | Runtime |
| **Réponse** | Offrir les types de transactions les plus courantes du client  Offrir dans le menu les 4 transactions les plus courantes |
| **Mesure de la réponse** | Nombre d’étapes nécessaires pour faire une transaction  Nombre de tâches accomplies par l’utilisateur. |
| **Questions** | 1. Combien de “dernières” transactions sont offertes. 2. Quels types de transactions se retrouvent dans celles offertes à nouveaux (retrait, dépôt, transfert de fonds) 3. Quels transactions sont les plus courantes (types et montant) |

## **6.1 Tactique 1 : Aggregate**

**Description**: Plusieurs processus distincts d’une transaction, tels que le choix du type de transaction (Retrait, Dépôt, Transfert), le montant d’argent ainsi que les comptes ciblés, sont agrégés en une seule action initiée par l’utilisateur.

**Justification**: Grâce à cette tactique, l’utilisateur peut facilement effectuer une transaction complète en un clic/choix. L’utilisateur choisirait “Repeat last transaction (Deposit, 20$, Chèque Account)” et cette transaction s'effectuera au complet d’un coup.

## **6.2 Tactique 2 : Maintain User Model**

**Description:** Les choix de montants offerts directement (Ex: 20$, 40$, ...), lors d’une transaction, sont personnalisés selon les montants les plus utilisés par l’utilisateur. Cette tactique suggère automatiquement des choix de transaction basé sur l’utilisateur et son comportement.

**Justification:** L’utilisateur peut effectuer des transactions courantes rapidement et personnalisées. Si par exemple l’utilisateur choisit souvent (3 fois ou plus) “Autre” puis entre le montant “420$”, alors ce montant sera affiché à l’avenir parmi les choix par défaut, ce qui lui évitera de l’entrer manuellement. Si cependant une transaction personnalisée n’est pas réutilisée après 5 autres transactions, elle serait supprimée des choix disponibles. L’usager n’aura pas à passer à travers tous les écrans pour faire une opération répétitives.

## **6.3 Vue architecturale (à l’extérieur de la classe)**

### **6.3.1 Diagramme**

Ex du prof: composant/connecteur des tâches

Ex du prof: interface usagee

Usabilite.png

**Figure 1.0 :** Diagramme du scénario usabilité

**Table de description des éléments du diagramme**

|  |  |
| --- | --- |
| **Élément** | **Responsabilité** |
| UI | Interface utilisateur |
| Transaction | Module démarrant les transactions offertes |
| System | Fonctionnalités du système traitant les transactions |
| LastTransactions | Liste dynamique des 5 dernières transactions. |

**Table de description des relations des tactiques entre ces éléments**

|  |  |
| --- | --- |
| **Élément** | **Responsabilité** |
| Aggregate | Choix de transactions antérieures pour réduire le temps de décision et le nombre de clics. |
| Maintain User Model | Mise à jour de l’interface de l’utilisateur après chacune de ces transactions. |

**Description des relations entre les éléments**

Lorsque l’utilisateur veut exécuter une transaction, le guichet utilise un affichage comprenant des actions simplifiée en utilisant la tactique Aggregate, permettant de réduire le nombre de clics nécessaire pour chaque transaction courante,cette tactique est basée sur des actions répétées de l’utilisateur ainsi ceci va permettre à l’utilisateur de réduire le nombre de clic. La tactique Maintain User Model est aussi utilisé afin de garder l'interface utilisateur à jour selon ses dernière transactions, augmentant ainsi sa rapidité de prise de décision lors de transaction courante.

# **7 Scénario : L’utilisateur désire de faire une transaction dépôt**

|  |  |
| --- | --- |
| **Objectifs d’affaires** | Diminuer le taux d’erreur sans mettre de chiffre |
| **Source** | Utilisateur |
| **Stimulus** | * Minimiser l’impact d’erreur |
| **Artéfact** | * Système |
| **Environnement** | * Mode normale |
| **Réponse** | * Le système doit être en mesure d’y répondre, si l’utilisateur a commis une erreur. |
| **Mesure de la réponse** | * Le temps de mesures allouer pour que l’utilisateur réagir aux erreurs * Le nombre d’erreurs que l’utilisateur peut refaire * Le temps de mesures pour que l’utilisateur puisse refaire une transaction |
| **Questions** | * Quels sont les options sont fournies afin que l’utilisateur puisse corriger son erreur ? |

## **7.1 Tactique 1 : Undo**

**Description**: Si l’utilisateur a effectué une erreur lors de la transaction dépôt.

**Justification**: L’utilisateur peut effectuer une option « Undo » afin d’aller dans la page précédente pour refaire la transaction dépôt.

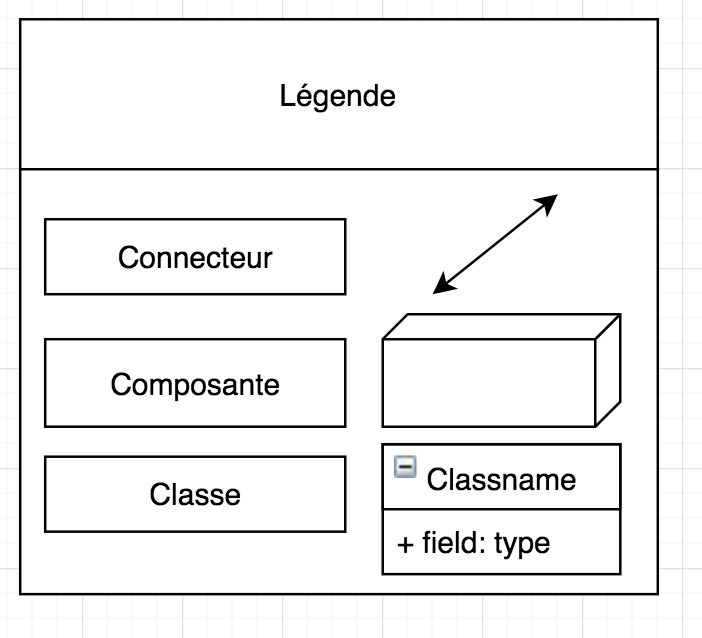
## 7.2 Tactique 2 : Cancel

**Description**: Si l’utilisateur décide qu’il veut d’annuler sa transaction de dépôt.

**Justification :** L’utilisateur peut effectuer une option « Cancel » afin d’annuler les transactions et il peut retirer sa carte.

## **7.3 Vue architecturale (à l’extérieur de la classe)**

VueArchitecturalAndy.png



### 7.3.1 Description

Lors de la transaction dépôt, l’utilisateur aura le choix de choisir l’option « Undo » et « Cancel » afin d’y retourner aux pages précédentes pour refaire la transaction ou annuler complètement la transaction pour retirer la carte.

### **7.3.2 Relation avec la tactique**

*Cancel :*

Il servira à annuler la transaction pour retirer la carte par la suite

*Undo:*

*Il servira à retourner aux pages précédentes pour que l’utilisateur puisse refaire sa transaction*

### 7.3.3 Composantes :

ATM : c’est un composant de la transaction dépôt

Cancel : Il contient les fonctionnalités de l’option cancel et retirer la carte.

Undo : il contient les fonctionnalités de l’option undo et retourner aux pages précédentes.

# 8 Un utilisateur insère sa carte pour effectuer une transaction mais décide au milieu des étapes qu’il veut l’annuler et oublie sa carte.

|  |  |
| --- | --- |
| **objectifs d'affaires** | En tant que développeur nous voulons augmenter l'usabilité en maximisant la rétroaction donnée aux utilisateurs |
| **Source** | L’utilisateur |
| **Stimulus** | ~~L’utilisateu~~r insère sa carte dans l’atm |
| **Artéfact** | L’interface utilisateur et la borne physique de l’Atm où on insère la carte |
| **Environnement** | Runtime |
| **Réponse** | * L’interface utilisateur offre un bouton cancel à tout moment de la transaction * L’Atm sone si la carte et laisser dans la borne de l’insertion de carte |
| **Mesure de la réponse** | * L’utilisateur réussit à récupérer sa carte à n'importe quel moment après l’insertion * L’utilisateur est rappelé de reprendre sa carte si jamais il l’oublie après avoir annulé sa transaction |
| **Questions** | 1. Combien de temps après avoir annulé une transaction est-ce ça prend pour qu’un utilisateur reprend sa carte? 2. Est-ce que l’utilisateur peut appuyer sur annuler sans avoir donné de NIP? |

## **8.1 Tactique 1 : Support User Initiative - Cancel**

**Description**: L’interface utilisateur offre un bouton annuler à n’importe quel moment une fois que l’utilisateur ait insérer la carte.

**Justification**: L’utilisateur est imprévisible est peut désirer d’annuler sa transaction à n'importe quel moment après avoir déjà inséré sa carte. Pour cette raison afin d’augmenter d'usabilité nous devons offrir une façon qu’il puisse récupérer sa carte sans effectuer une transaction non désirée. Ainsi, l’utilisateur doit pouvoir annuler le processus d’une transaction avec que cette dernière ne soit complétée.

## **8.2 Tactique 2 : Support System Initiative - Maintain System Model**

**Description**: L’Atm répond avec un signal sonore pour avertir l’utilisateur qu’il a oublié sa carte à l’entrée de l’Atm après avoir annulé sa transaction.

**Justification**: L’Atm possède un capteur qui lui permet de vérifier si la carte demeure dans la borne de l’entrée de la carte. Parfois, par insouciance, un utilisateur peut oublier de reprendre sa carte. Pour cette raison l’Atm envoie un signal sonore ou lumineux pendant quelques secondes pour signaler à l’utilisateur la présence d’une carte dans le détecteur.

## **8.3 Vue architecturale (à l’extérieur de la classe)**

### 8.3.1 Diagramme

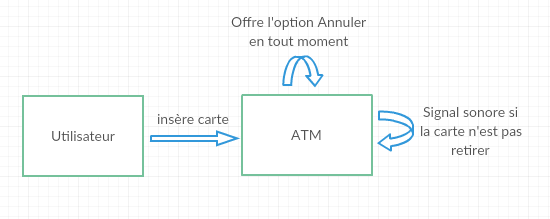


Fig 1.

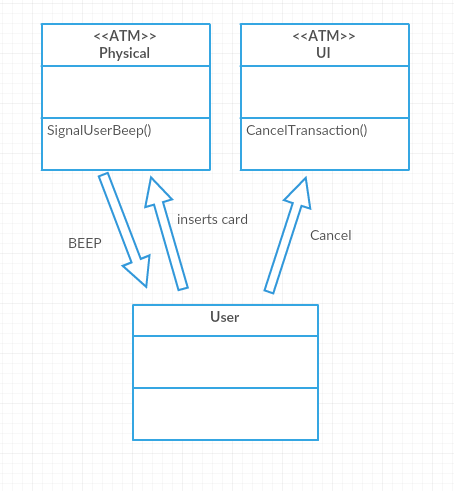


Fig 2.

### 8.3.2 Légende

Fig 1.

* La flèche directe démontre que l’utilisateur interagit avec l’Atm
* Les flèches cycliques démontrent qu'en tout temps l’Atm peut effectuer ces options si nécessaires

Fig 2.

* ATM physical représente les composantes physiques
* ATM UI représente l’interface utilisateur avec laquelle l'utilisateur interagit

### 8.3.3 Texte de description du diagramme

Le diagramme démontre que l’utilisateur insère une carte dans l’Atm et qu'en tout temps:

- l’utilisateur peut annuler sa transaction

- si la carte n’est pas retirée l’Atm envoie un signal sonore ou lumineux pour avertir l’utilisateur.

### 8.3.4 Table de description des éléments du diagramme

Utilisateur : la personne qui désire utiliser l’Atm

ATM Physical: la borne physique où nous insérons la carte et qui contient le capteur pour déclencher le système sonore si nécessaire

ATM UI: l’interface utilisateur qui permet à l’utilisateur d’annuler sa transaction à n'importe quel moment

### 8.3.5 Texte décrivant la relation entre les éléments et les tactiques

Les tactiques « Support user Initiative – Cancel » et « Support System Initiative - maintain System Model » servent à améliorer l’usabilité. Les deux tactiques sont offertes par l’Atm afin que l'utilisateur puisse quitter à n’importe quel moment et pour s’assurer qu’il n'oublie pas sa carte. Offrir ces deux actions au client, met en contrôle ce dernier et augmente le degré selon lequel le guichet peut être utilisé par le client.

N’oubliez pas que vous pouvez réutiliser des éléments des vues architecturales précédentes pour réaliser la vue courante. Toutes les tactiques peuvent être combinées dans une ou plusieurs vues selon les besoins.

# 

# 

# 9 Scénario : Permettre l’annulation d’une transaction dans un certain délai

|  |  |
| --- | --- |
| **objectifs d'affaires** | Permettre l’annulation d’une transaction dans un certain délai |
| **Source** | L’utilisateur du guichet |
| **Stimulus** | Corriger une erreur dans un montant à retirer |
| **Artéfact** | Système, interface usager |
| **Environnement** | Exécution normale |
| **Réponse** | Le système affiche les options pour corriger l’erreur faite par l’usager |
| **Mesure de la réponse** | Le nombre d’erreurs effectuées  le temps nécessaire à un utilisateur pour corriger une erreur |
| **Questions** | 1. Est-il possible de faire une certaine validation de la transaction avant de l'effectuer? 2. Quel est l’effet sur le système lors de l’annulation d’un retrait ? |

## 9.1 Tactique 1 : Undo

**Description**: Avec ce patron, il est possible de défaire une opération précédemment effectuée. Cela permet à un usager de corriger une erreur qu’il a faite ou de défaire une opération qu’il ne voulait pas faire.

**Justification**: En permettant de défaire une opération, le système permettra à l’utilisateur de corriger une transaction s’il a fait une erreur lors de celle-ci.Elle lui permet aussi de corriger un montant erroné avant de pouvoir confirmer la transaction. Cela rend le système plus agréable pour l’utilisateur et lui permet de corriger ses erreurs.

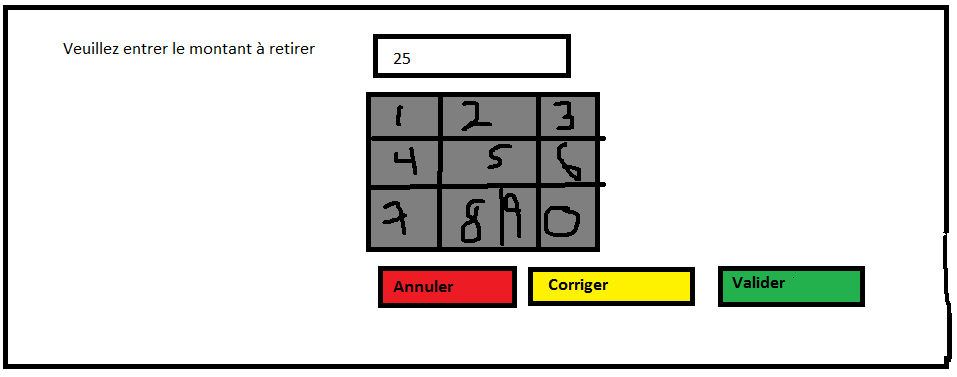
## 9.2 Tactique 2 : Cancel

**Description**: Il est possible d’annuler une transaction avant d’avoir retiré sa carte. Les transactions ne sont appliquées qu’au retrait de la carte, ce qui permet à l’utilisateur de les annuler ou de les modifier avant de se déconnecter de l’ATM.

**Justification**: En permettant d’annuler les transactions, le système supporte l’initiative de l’utilisateur et permet à l’utilisateur de réviser ses transactions avant de quitter l’ATM. Donc , s’il a fait une erreur, il pourra la corriger.Donc, cela rend le système plus agréable pour l’utilisateur et lui permet de corriger ses erreurs.

## 9.3 Vue architecturale

### 9.3.1 Diagramme



### 9.3.2 Légende

Le bouton rouge représente le patron Annuler et le bouton jaune représente le patron Undo.

### 9.3.3 Texte de description du diagramme

Dans ce diagramme , on représente la situation dans laquelle un utilisateur essaie de retirer 25 de son compte. On lui demande combien d’argent il veut retirer. Il y a 3 boutons sur le diagramme. Un bouton rouge qui représente le patron annuler. Il permet d’annuler la transaction. Il y a le bouton jaune qui représente le patron Undo. Il nous permet de corriger le montant que nous avons inscrit. Il y a aussi le bouton vert qui permet de confirmer la transaction.

### 9.3.4 Table de description des éléments du diagramme

|  |  |
| --- | --- |
| Éléments | Description |
| Bouton rouge “Annuler“ | Un bouton rouge qui représente le patron annuler. Il permet d’annuler la transaction. |
| Bouton jaune “Corriger” | Il y a le bouton jaune qui représente le patron Undo. Il nous permet de corriger le montant que nous avons inscrit. |
| Bouton vert “Confirmer” | Il y a aussi le bouton vert qui permet de confirmer la transaction. |
| Clavier de sélection | Le clavier nous permet d’entrer des montants dans le champ de texte. Avec ces montants, il est possible de faire les différentes transactions. |

### 9.3.5 Texte décrivant la relation entre les éléments et les tactiques

En utilisant le patron Cancel en combinaison avec le patron Undo , il est possible de donner le contrôle à l’utilisateur lorsqu’il fait ses différentes transactions. En effet, grâce au patron Cancel , il pourra décider d’annuler une transaction en cours et de revenir à l’écran principal du guichet. De plus, grâce au patron Undo, il est possible de donner la possibilité à l’usager de corriger le montant qu’il a inscrit dans la case de montant. Donc, l’utilisateur a donc un contrôle total sur son expérience lors de l’utilisation du guichet. Ces différents patrons sont représentés dans l’interface avec le bouton rouge et le bouton jaune.