La méthodologie UML

Benkhaldoun KAMEL
kamel@club-internet.fr

1 Introduction à la conception orientée objet et à UML

1.1 Le logiciel, état des lieux

- **Les temps de conception et de développement sont longs :** évaluation du besoin, phase de conception souvent peu tangible.
- **La communication et l'implications des utilisateurs pendant le développement passent pour rester de faible niveau :** Le langage de l'informaticien et celui de l'utilisateur final.
- **Le test du logiciel reste un des grand points noirs de l'informatique :** La testabilité doit faire partie de la conception.
- **La maintenance prend une part de plus en plus importante des ressources informatiques :** Structure complexe du logiciel et dépendances inextricables.
- **Les besoins s'élargissent et les exigences s'accroissent :** Ergonomie, performance.
- **L'évolution des techniques introduit davantage de complexité qu'elle n'en supprime :** Multimédia, interfaces graphiques

1.2 Les conséquences d'une complexité sans limite

- Plus un système est complexe, plus il est susceptible d'effondrement.
- Gaspillage des ressources humaines.
- Comment résoudre cette crise du logiciel ?
- Comment sont organisés les systèmes complexes dans d'autres disciplines ?
- Le programmeur amateur réalise des logiciels à durée de vie limitée.

Exemples d'application complexes

- Système gérant des entités du monde réel : Contrôle de trafic aérien.
- Système simulant l'intelligence humaine : Réseaux de neurones.

La complexité de ces systèmes dépasse les capacités intellectuelles de l'homme.
Il faut trouver des méthodes rigoureuses pour maîtriser la complexité.

1.3 Exemple de système complexe: la structure du PC

Les éléments d'un ordinateur personnel sont :

- Une unité centrale
- Un écran
- Un clavier
- Un disque dur

Une unité centrale englobe :

- Une mémoire centrale.
- Une unité arithmétique et logique.
- Un bus.

Une unité arithmétique et logique englobe:

- Des registres et logique de contrôle.

Nature hiérarchique d'un système complexe.
L'ordinateur ne peut fonctionner qu'à partir du moment où ses parties collaborent.
Chaque partie peut être indépendante.

1.4 Les cinq attributs d'un système complexe

1. La complexité prend souvent la forme d'une hiérarchie dans laquelle un système est composé de sous-systèmes reliés entre eux et ayant à leur tour leurs propres sous système, et ainsi de suite jusqu'à ce qu'on atteigne le niveau le plus bas des composants élémentaires.
2. Le choix des composants primaires d'un système est relativement arbitraire et dépend largement de l'observateur du système.
3. Les liaisons inter-composants sont généralement plus fortes que les liaisons inter - composants. Ceci a pour effet de séparer les dynamiques haute fréquence de composants - celles qui concernent la structure interne des composants. - des dynamiques basse fréquence - qui concernent l'interaction entre composants
4. Les systèmes hiérarchiques sont habituellement composés d'un petit nombre de genres de sous-systèmes qui forment des combinaisons et des arrangements variés.
5. Un système complexe qui fonctionne a toujours évolué à partir d'un système simple qui a fonctionne...Un système complexe conçu ex nihilo ne fonctionne jamais et ne peut être rapiécé pour qu'il fonctionne. Il faut tout recommencer, à partir d'un système simple qui fonctionne.

1.5 Les méthodes de conception

1.5.1 Principe de base: la décomposition

La taille et la complexité des logiciels imposent la décomposition du problème en sous problèmes, jusqu'à atteindre des unités plus facilement maîtrisables.

1.5.2 Approche structurée

Guidée par les traitements - module principal, décomposition progressive...,on obtient des modules fonctionnels (FORTRAN).

1.5.3 Approche par les données

Analyse en structure de données (Warnier, Cobol)

1.5.4 Approche objet

Conception d'unités autonomes qui en-capsulent à la fois les données et les traitement qui s'y rapportent.

Exemple de l'approche structurée

Pour construire une remise:

1. Mettre en place les fondations
2. Bâtir les murs
3. Mettre en place les planchers.
4. Placer le toit dessus.

1.6 L'approche objets selon Stroustrup

Le paradigme de programmation d'origine est le suivant:

- Définissez vos procédures.
- Utilisez le meilleur algorithme possible.

Avec l'encapsulation le paradigme devient:

Définissez vos modules.
Partitionnez le programme de sorte que les données soient cachées au sein des modules.

L'introduction des types abstraits de données le transforme en:

Définissez vos types.
Rattachez un ensemble d'opérations à chaque type

L'héritage permet d'aboutir à la programmation objet.

Définissez vos classes.
Rattachez un ensemble d'opérations à chaque à chaque classe.
Explicitez ce qui est partageable en ayant recours à l'héritage.

1.7 Conception par objets

La conception par objets n'est qu'un guide pour la réalisation d'une application. Quatre principes essentiels dans la conception par objets:

1. La notion de réification.
2. L'autonomie et la localité.
3. La composition et l'affinage.
4. La généricité.

1.7.1 La notion de réification

Il faut focaliser sur ce qui doit être manipulé.

Exemple.

Pour un système de régulation de processus industriel:

`Vanne, relais, conduite, régulateur, pompes.`

Définition.

La réification est l'opération essentielle du paradigme objet par lequel quelle que chose (chose physique, relation événement, situation, idée, loi, etc.) est représentée informatiquement sous la forme d'un objet

Principe.

Si l'on parle de quelque chose en lui attribuant des propriétés, ou si cette chose doit être manipulée, alors, il faut la réifier, c'est à dire la représenter sous forme d'objet.

1.7.2 Autonomie et localité

Il faut décomposer le logiciel en unités indépendantes.

Exemple: Cas procédural.

```
OuvrirPorte(laPorte)
      Si laPorte est de type "un seul battant"
            ouvrirPorteUnSeulBattant(laPorte)
      Sinon
            Si laPorte est de type "un double battant"
                  ouvrirPorteUndoubleBattant(laPorte)
Fin ouvrirPorte
```

Exemple: Cas objets.

```
OuvrirPorte(laPorte)
      laPorte ouvreToi
Fin ouvrirPorte

      Classe PorteUnSeulBattant
            sorte de porte
            méthode ouvre Porte
            Fin ouvreToi
      Fin PorteUnSeulBattant

      Classe PorteUnDoubleBattant
            sorte de porte
            méthode ouvreToi
            ............
            Fin ouvreToi
      Fin PorteUnDoubleBattant
```

1.7.3 Composition et affinage

Exemple

Un fenêtre est composée: D'un cadre, d'un titre, de barres de déplacement et de défilement, etc.

Définition.

Affiner, c'est décrire un logiciel en termes de logiciels déjà existants, et en ne définissant que ce qui est nouveau dans le logiciel en cours de création.

1.7.4 La généricité.

1.8 Les grandes étapes de la conception par objets

La démarche est moins descendante et plus itérative.

1. Identifier les entités du domaine.
2. Structurer le domaine en analysant les propriétés de ces entités et leurs relations.
3. Identifier les opérations que savent effectuer ces entités.
4. Décrire précisément ces opérations en les reliant à des messages.
5. Décrire le lancement du programme.

1.8.1 Identifier les entités du domaine

C'est le principe de réification

Exemple:

Pour un logiciel de gestion :
Les classes sont: Facture, Personne, Compte, Produit, Commande.
Pour un logiciel de simulation de trafic automobile :
Les classes sont: les voies, les carrefours, les signalisations fixes et mobiles (agents de police).

Il faut obtenir les bonnes classes, plusieurs solutions:

- Prendre un petit noyau de classes évidentes
- Donner une liste exhaustive des entités du domaine.
- Analyser les informations et procéder par classification.
- Faire parler le spécialiste (souvent noms = classes et verbes = méthodes)

1.8.2 Structurer le domaine par l'analyse des propriétés et des relations

Il faut organiser et relier les classes
Une propriété et une valeur simple : Nombre booléen, symboles, chaînes de caractères.
Une relation est un lien entre les objets plus complexes :

- Lien qui unit le chauffeur à son camion
- (C'est le chauffeur - de, a - comme - propriétaire).

Une propriété peut se ramener à une relation.

1.8.3 Identifier les opérations

Déterminer les messages auxquels doivent répondre les objets

Exemple:

Pour une interface graphique : Se visualiser, s'effacer, se déplacer.

Déterminer ces comportements de la manière la plus abstraite possible : Définir un protocole général des objets graphiques caractérisés par un comportement par défaut.

1.8.4 Décrire les opérations et lancer l'exécution

Phase de codage
Lancer l'exécution
Déterminer le processus en créant quelques instances initiales
De message en message. le programme se déroule.

Conseils

1. Il faut savoir revenir aux étapes précédentes.
2. Il faut prototyper rapidement avant de passer à une spécification complète.
3. Il faut penser abstrait.
4. Il faut penser local.

Les opérations peuvent-elles être des objets ?

Si les méthodes de résolution sont caractérisées par des propriétés, ou si elles doivent être manipulées, alors, il faut en faire des objets.
Autrement, si leur seule rôle est seulement de s'appliquer, il suffit de les garder comme méthodes.

Les erreurs communes

1. Faire des tests répétés.
2. Confondre héritage et composition.
3. Manipuler les objets au loin.
4. Ne pas penser en termes de fonctions, penser en objets.

1.8.5 Décrire le lancement du programme.

2 La méthodologie UML

Définition d'une modélisation est une conception orientée objets.

- Modèles organisés autour des concepts du monde réel
- Comprendre les problèmes
- Préparer la documentation
- Concevoir les programmes.

Les phases d'UML

1. L'analyse.
2. La conception.
3. La programmation.

2.1 La notion d'objet

Quatre aspects fondamentaux de l'objet

2.1.1 L'identité

Objets distinguables.

2.1.2 La classification

Même structure de données et même comportement mis dans une classe.

2.1.3 Le polymorphisme

Une opération se comportant différemment selon les classes.

2.1.4 L'héritage

Partage des attributs et des opérations.

2.2 Les trois modèles d'UML

- **Le modèle objet :** Structure statique des objets et leurs relations.
- **Le modèle dynamique** : Interactions entre les objets dans le système.
- **Le modèle fonctionnel** : Transformation des valeurs des données dans le système.

2.3 Thèmes orientés objet

- **L'abstraction** **:** Prendre les aspects essentiels d'une entité.
- **L'encapsulation** **:** Masquage de l'information.
- **Le polymorphisme** **:** Opération se comportant différemment sur différentes classes.
- **L'héritage** **:** Partage de code.

2.4 Utilité d'une approche orienté objets

- Pas forcément de réduction en ce qui concerne le temps de développement
- La maintenance et l'évolution du logiciel est facilité.
- Révisions plus localisées.
- Plus de problème découverts pendant le développement.
- Applications.
- Compilateurs, applications graphiques, bases de données.

3 Le modèle objet

3.1 Les objets

- Un **objet** est un concept, une abstraction ou chose ayant un contour net et une signification pour le problème posé: instance de classe : Fenêtre à droite, processus 28, Félix le chat, etc.
- L'**identité** est une caractéristique distinctive d'un objet qui dénote l'existante séparée de l'objet, même si cet objet peut avoir des valeurs de données identiques à celles d'un autre objet.
- L'**instance** est un objet décrit par une classe.

3.2 Les classes

Une classe est la description d'un groupe d'objets ayant des propriétés similaires, un comportement commun, des relations et une sémantique commune : Personne, Société, Processus, Fenêtre.
Une classe ou un objet est souvent un nom dans les descriptions du problème.
On abstrait un problème en groupant les objets en classe.
L'abstraction est la capacité mentale permettant aux humains de représenter les problèmes du monde réel avec un degré variable de détail e fonction du contexte courant du problème.

3.3 Les diagrammes d'objets

Un diagramme d'objets propose une notation graphique formelle qui permet de modéliser les objets, les classes et les relations qu'ils ou elles entretiennent.

- Diagramme de classes.
- Diagramme d'instances.

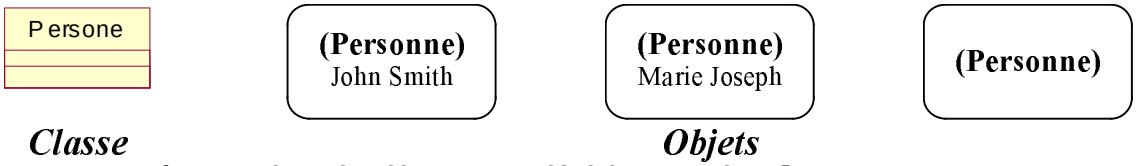


Figure 1: Représentation du diagramme d'objts et de classes

3.4 Les attributs

Un attribut est une valeur de donnée détenue par les objets d'une classe : Non, Âge, poids d'une personne.

Un attribut devra être une valeur pure et non un objet.

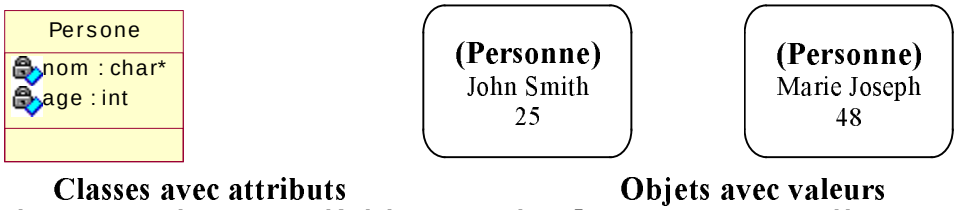


Figure 2: Diagramme d'objets et de classes avec attributs

3.5 Les opérations est les méthodes

- **Une opération** est une fonction qui s'applique aux objets d'une classe.
- Une opération est **polymorphe**.
- **Une méthode** est l'implémentation d'une opération pour une classe.
- **Une requête** ne modifie pas l'objet.
- **Une attribut dérivé** est un attribut calculé à partir d'autres attributs.

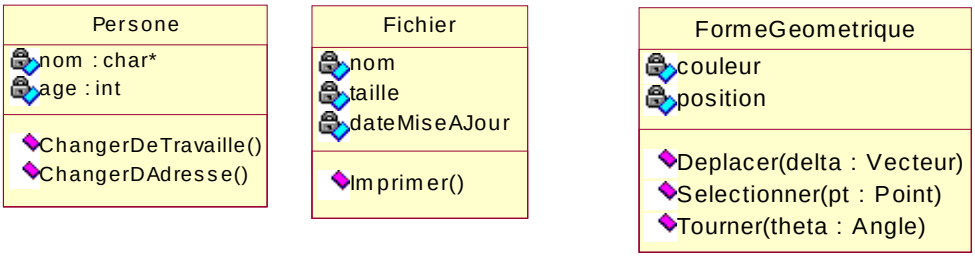


Figure 3: Diagramme de classes avec attributs et méthodes

3.6 Notation des classes d'objets

- Une classe est représentée par une boîte de trois zones
  - Le nom de la classe.
  - La liste des attributs. Suivie du type de la valeur par défaut (facultatif)
  - La liste des opérations. Suivie de la liste des arguments et le type des résultats (facultatif)

Nom de classe
Nom d'attribut 1 : type de donnée 1= valeur par défaut 1
Nom d'attribut 2 : type de donnée 2= valeur par défaut 2
Nom d'attribut 3 : type de donnée 3= valeur par défaut 3

3.7 Liens et associations

Une **association** est une relation entre instances de deux ou plusieurs classes décrivant un groupe de liens avec une structure et une sémantique commune.

Un **lien** est l'instance d'une association: connexion conceptuelle ou physique entre objets.

Diagramme de classes

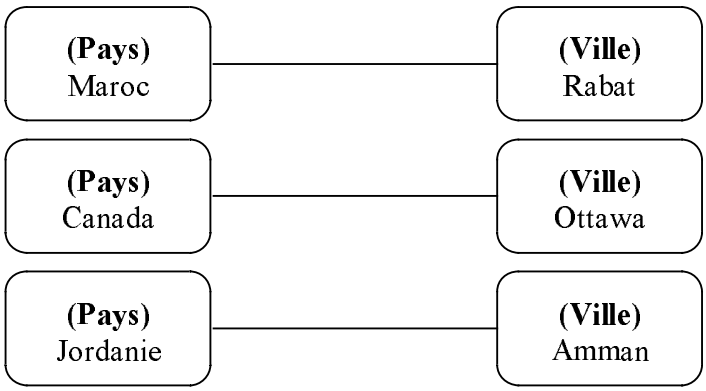


Diagramme d'instance

Figure 4: Diagramme de classes et d'instance avec liens

3.8 La multiplicité

La multiplicité spécifie combien d'instances d'une classe peuvent se connecter à chaque instance d'une autre classe.

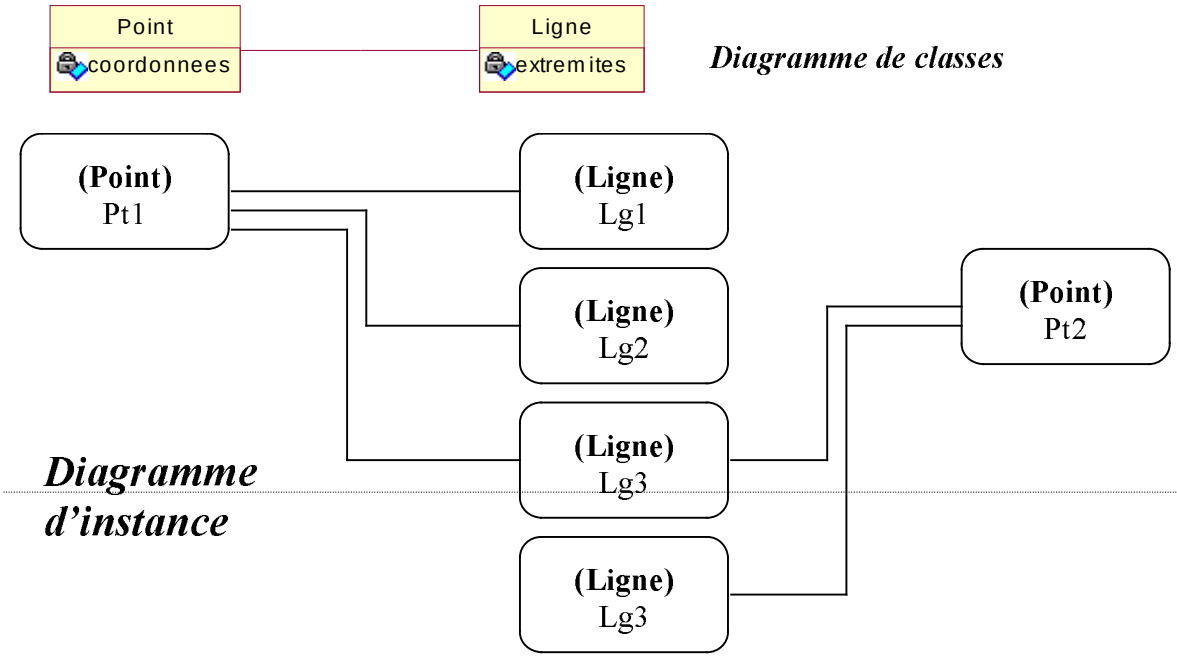
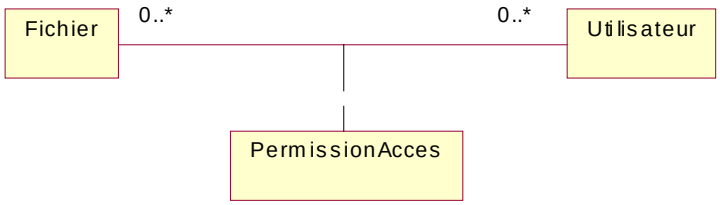


Figure 5: Diagramme de classes et d'instance avec multplicité

3.9 Les attributs de lien

Un attribut de lien est une valeur de donnée prise par chaque lien dans une association.



/usr/doc/login	/usr/doc/login	John Nasser
bin/include	bin/include	Zoubïr Ali
bin/include	bin/include	Ammar Yassir

Figure 6: Attributs de liens pour une association de plusieurs à plusieurs

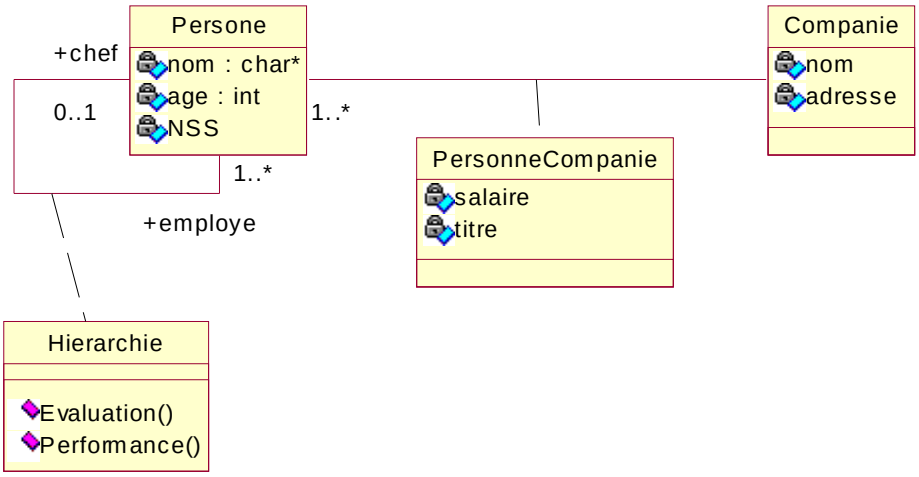


Figure 7:Attribut de lien pour une association "un à plusieurs"

3.10 Les noms de rôle

Un nom de rôle est un non identifiant de manière unique une des extrémités de l'association. Un attribut dérivé est un attribut calculé à partir d'autres attributs.

Figure 8: Noms de rôle pour une association

3.11 L'ordre

L'ordre est un type particulier de contrainte de l'association : Fenêtres explicitement ordonnées

Un ensemble ordonné d'objets du côté "plusieurs" de l'association est indiqué par {ordonné} près du point de multiplicité du rôle.

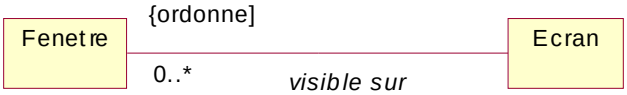


Figure 9: Ensembles ordonnés dans une association

3.12 La qualification

Une association qualifiée est une association reliant deux classes et un qualificatif. Il s'agit d'une association binaire dont la première partie est un composé comprenant la classe et le qualificatif, et la seconde partie une classe.

La qualification améliore la précision sémantique et augmente la visibilité des chemins de navigation.

Figure 10: Une association qualifiée

3.13 L'agrégation

L'agrégation est une forme particulière d'association entre un tout et ses parties, dans laquelle le tout est composé des parties.

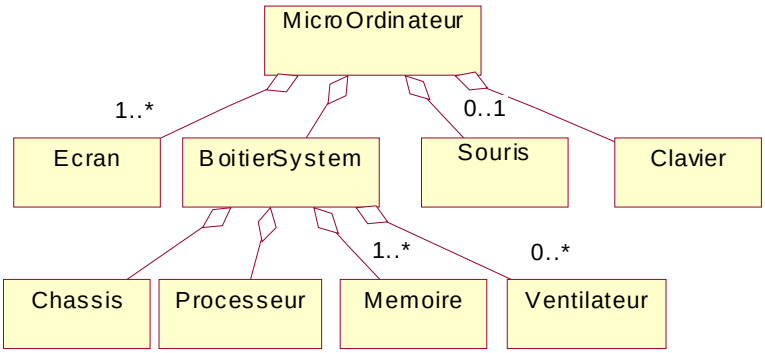


Figure 11: Agrégation à niveaux multiples

3.14 L'agrégation par rapport à l'association et la généralisation

La décision d'utiliser l'agrégation est affaire de jugement et fréquemment arbitraire

- Peut-on utiliser l'expression "Partie de" ?
- Les valeurs des attributs sont elles propagées du composé vers tous ou quelques composants ?

L'agrégation met en relation des instances
La généralisation met en relation les classes et constitue une façon de structurer la description d'un objet unique.

3.15 Généralisation et héritage

La **généralisation** est une relation entre une classe et une ou plusieurs versions raffinées ce celles-ci.

La **spécialisation** est la création de sous classes depuis une super classe en raffinant la super classe

Un **discriminant** est un attribut d'un type énuméré qui indique quelle propriété d'une classe se voit abstraite par une généralisation particulière.

Un **héritage** est un mécanisme orienté objet qui permet aux classes de partager des attributs et des opérations en se basant sur une relation, habituellement une généralisation.

3.16 Exemple 1

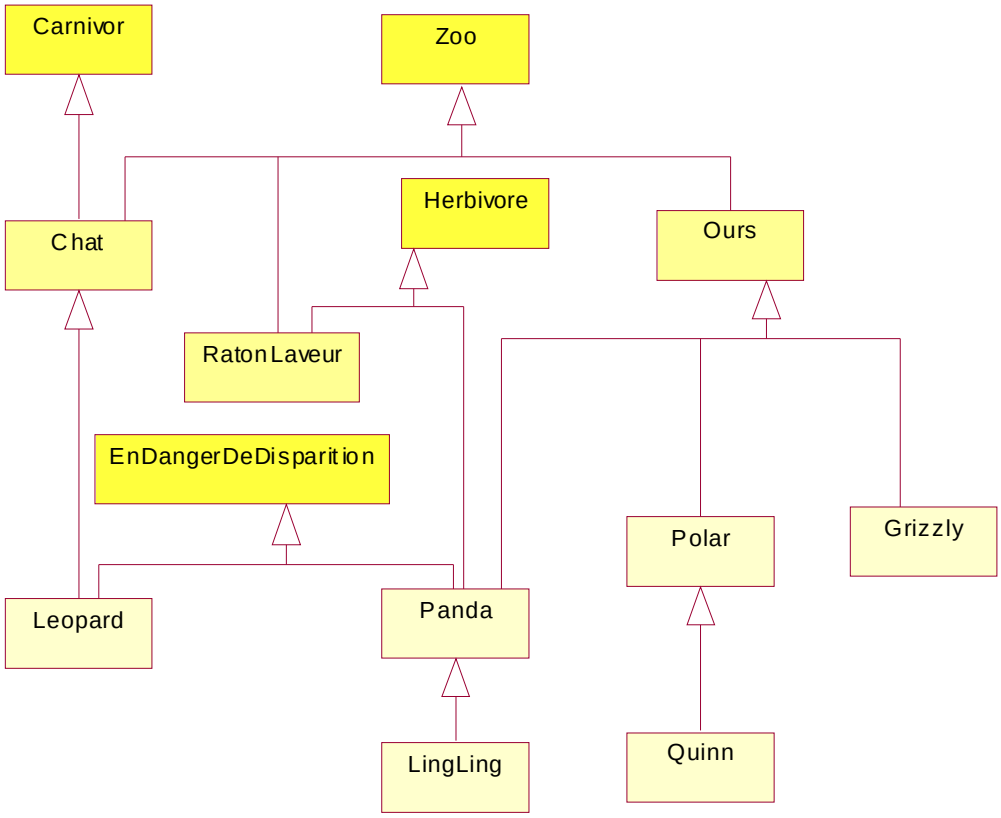


Figure 12: Héritage à niveaux multiples, avec instances

Remarquons que **Panda** hérite de **Ours** et **Raton Laveur** qui à leurs tours héritent de **Zoo**, il s'agit donc d'un héritage virtuel, symbolisé par le triangle noir.

3.17 Exemple 2

Une **redéfinition** est la définition d'une méthode pour une opération remplaçant la méthode héritée pour la même opération.

Une **signature** est pour un attribut le type et pour une opération le nombre et le type de ses arguments et type de sa valeur de retour.

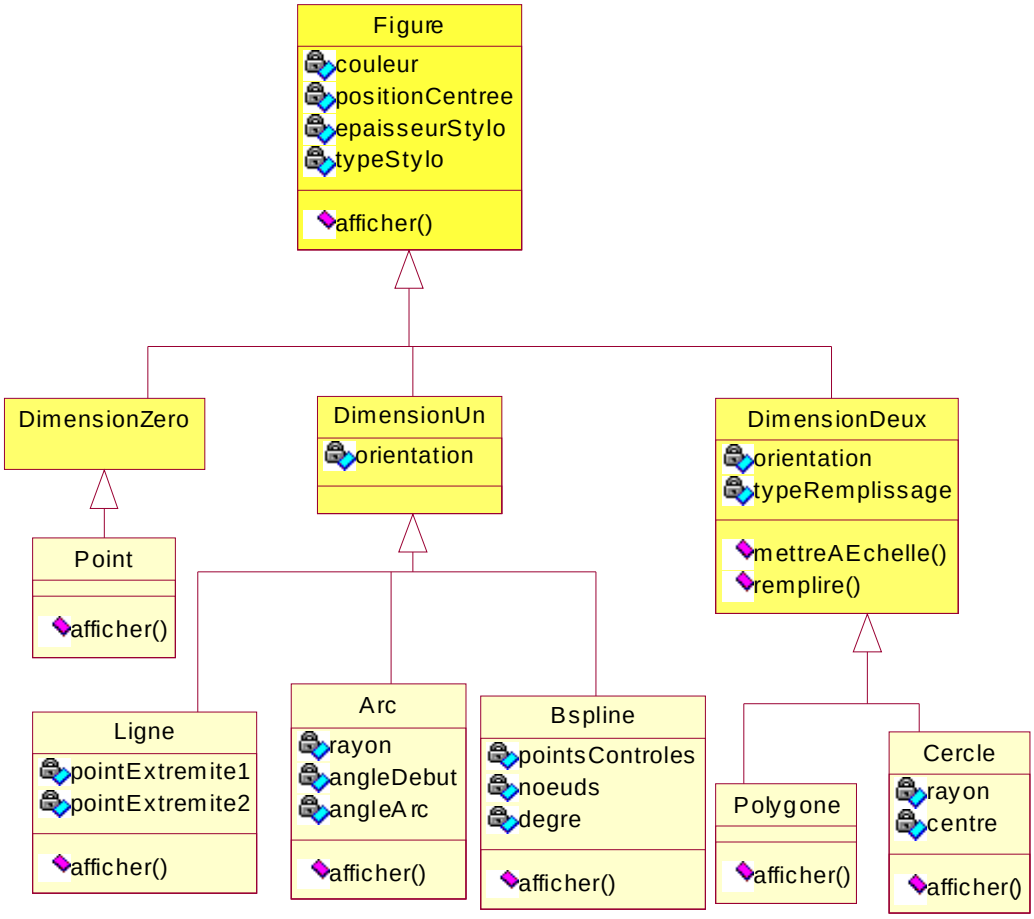


Figure 13: Héritage pour des gigures graphiques

3.18 Les modules

Un **module** est un sous ensemble cohérent d'un système contenant un groupe de classes intimement liées et leurs relations.

Le **feuillet** est un mécanisme pour découper un grand modèle objet et une série de pages. Chaque module consiste en un ou plusieurs feuillets.
Un feuillet est une conversion de notation, pas une construction logique.

3.19 Le modèle objet d'un système de fenêtrage

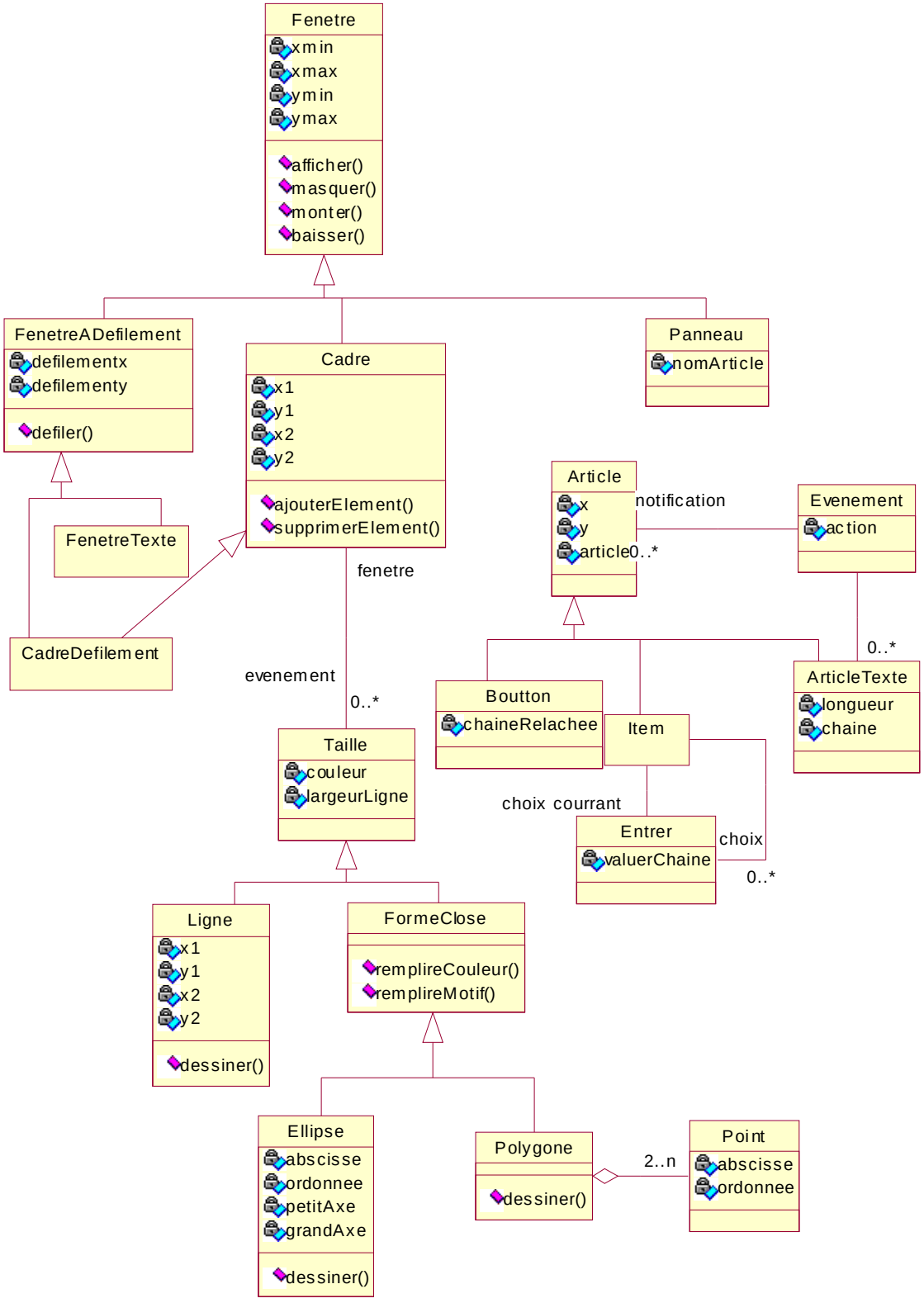


Figure 14: Modèle objet d'un système de fenêtrage

3.20 Conseils de Rumbaugh sur la construction de diagrammes d'objets

1. Ne commencez pas à construire un modèle objet en jetant pêle-mêle classes, association et héritage. Vous devez avant tous, **comprendre le problème** à résoudre. Le contenu d'un modèle objet est conditionné par la pertinence de la solution.
2. Efforcez-vous à garder votre **modèle simple**. Éviter les complication inutiles
3. **Choisissez le noms** avec soins. Ils sont importants et portent des connotations puissantes. Ils doivent être descriptifs, précis, sans ambiguïtés et ne doivent pas subir l'influence exclusive d'un seul aspect de l'objet. Le choix de bons noms est l'une des facettes les plus délicates de la modélisation des objets.
4. N'enfouissez pas les **pointeurs ou les références** d'objets dans les objets en tant qu'attributs. Modélisez les plus tôt comme **associations**. C'est plus clair et cela saisi la véritable intention plus tôt qu'une approche particulière d'implémentation.
5. **Évitez** si possible **les associations ternaires** et n-aires. La plus part de celles-ci peuvent être décomposées en associations binaires, éventuellement avec les qualificatifs et des attributs de lien.
6. N'essayez par d'exprimer parfaitement les **multiplicités trop tôt** dans le développement.
7. **Ne dissolvez** pas les **attributs de lien** dans une classe.
8. Utilisez les **associations qualifiées** quand c'est possible.
9. Essayez d'**éviter les généralisations profondément imbriquées**.
10. Reconsidérez les associations un à un. Souvent l'objet à chaque extrémité est optionnel et une multiplicité zéro à un peut être plus appropriées. À d'autres moments, une multiplicité "plusieurs" sera nécessaire.
11. Ne soyez pas surpris que votre **modèle** ait besoin d'une **révision**.
12. Soumettez vos modèles à des **avis extérieurs**.
13. **Documentez** systématiquement vos modèles objets.
14. Ne vous sentez **pas obligé d'éprouver** toute les constructions de modélisations.

3.21 Les classes de jointure

Une classe de jointure est une classe ayant une super classe.

L'héritage multiple est le type d'héritage permettant à une classe d'avoir plus d'une super classe et d'hériter des caractéristiques de tous ses ancêtres.

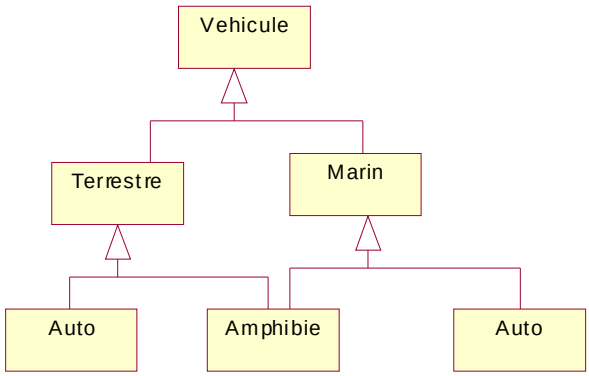


Figure 15: Héritage multiple de classes avec recouvrement

3.22 Les méta données

Une méta-donnée est une donnée qui en décrit une autre.

- Les classes.
- Les systèmes de gestion de bases de données relationnelles.
- Les dictionnaires.
- Un plan (décrit une maison).

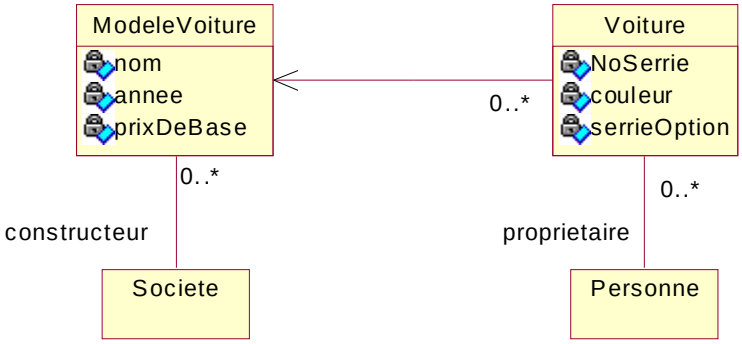


Figure 16: Modèles et individus

3.23 Les descripteurs de classe

Un descripteur de classe est un objet décrivant la classe elle même, contenant la liste des attributs et les méthodes autant que les valeurs de tout attribut de classe. Un descripteur de classe est une instance d'une méta-classe.
Un attribut de classe est un attribut dont les valeurs sont communes à toutes les instances plutôt que différentes pour chaque instance.

Une opération de classe est une opération sur une classe, plutôt que sur des instances de la classe.

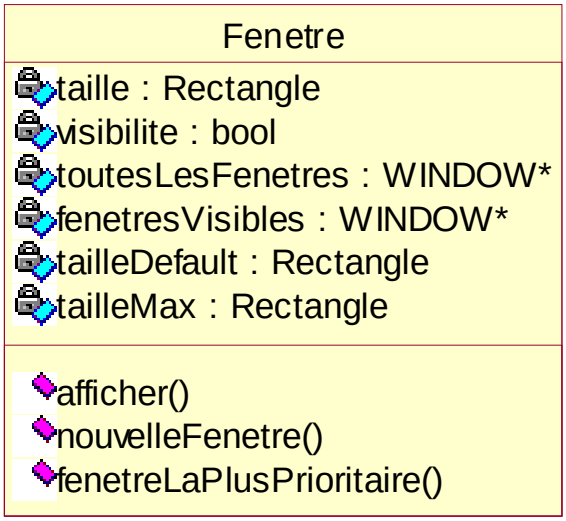
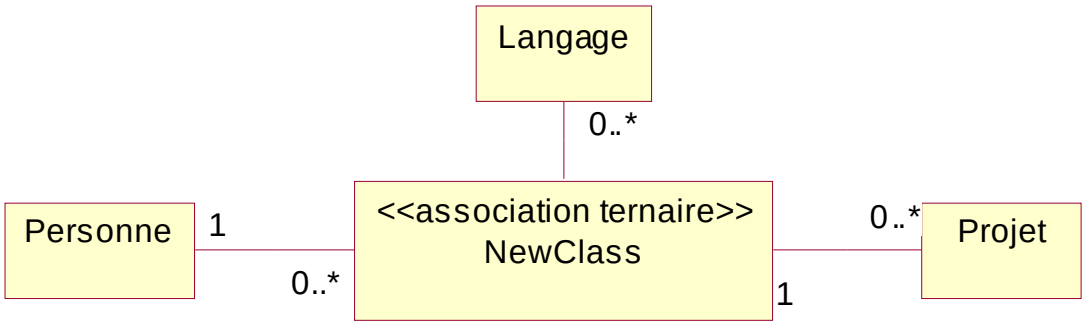


Figure 17: Classe avec priorité de classe

3.24 Les clés candidates

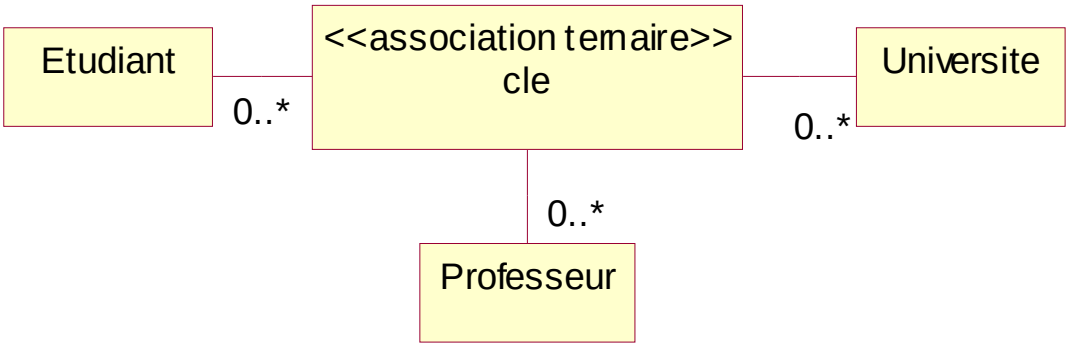
Une clé candidate est un ensemble minimal d'attributs identifiant une instance ou un lien.



Clé candidate: (projet, personne, langage)

Projet	Personne	Langage
Logiciel de CFAO	Stéphane	C
Programme de test	Marie	Ada
R & D	KAMAL	C++
Logiciel de CFAO	John	Assembleur
Compilateur C++	Hassane	C++
Logiciel de CFAO	Luis	C

Figure 18: Association ternaire



Clé candidate: (étudiant, Université)

Etudiant	Professeur	Univesité
Mamadou	Prof Farin	McGill
Mike	Prof Barnhil	Concordia
KAMAL	Prof Brezis	Jusieux
Stéphane	Prof Weisler	Orsay
Compilateur C++	Hassane	C++
Logiciel de CFAO	Luis	C

Figure 19: Association ternaire

3.25 Les contraintes

Une contrainte et une relation fonctionnelle entre objets, classes, attributs, liens et associations: instruction sur une certaine condition ou relation qui doit être maintenue vraie.

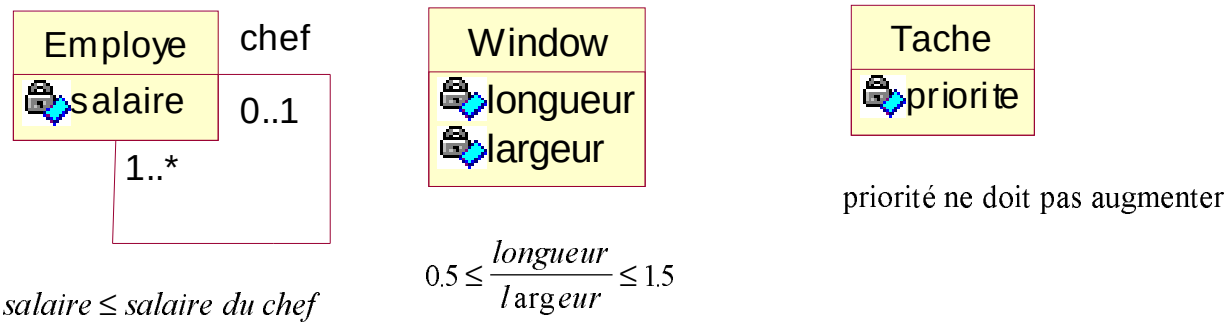


Figure 20: Contrintes sur des objets

4 L'analyse de la méthode UML

4.1 Le but de l'analyse

L'analyse est l'étape du cycle de développement dans laquelle le problème du monde réel est examiné afin d'en comprendre les spécifications, sans entreprendre d'implémentation.

La méthode d'analyse produit trois modèles dont l'importance diffère suivant le type d'application à réaliser

- Les modèle objet.
- Le modèle dynamique.
- Le modèle fonctionnel.

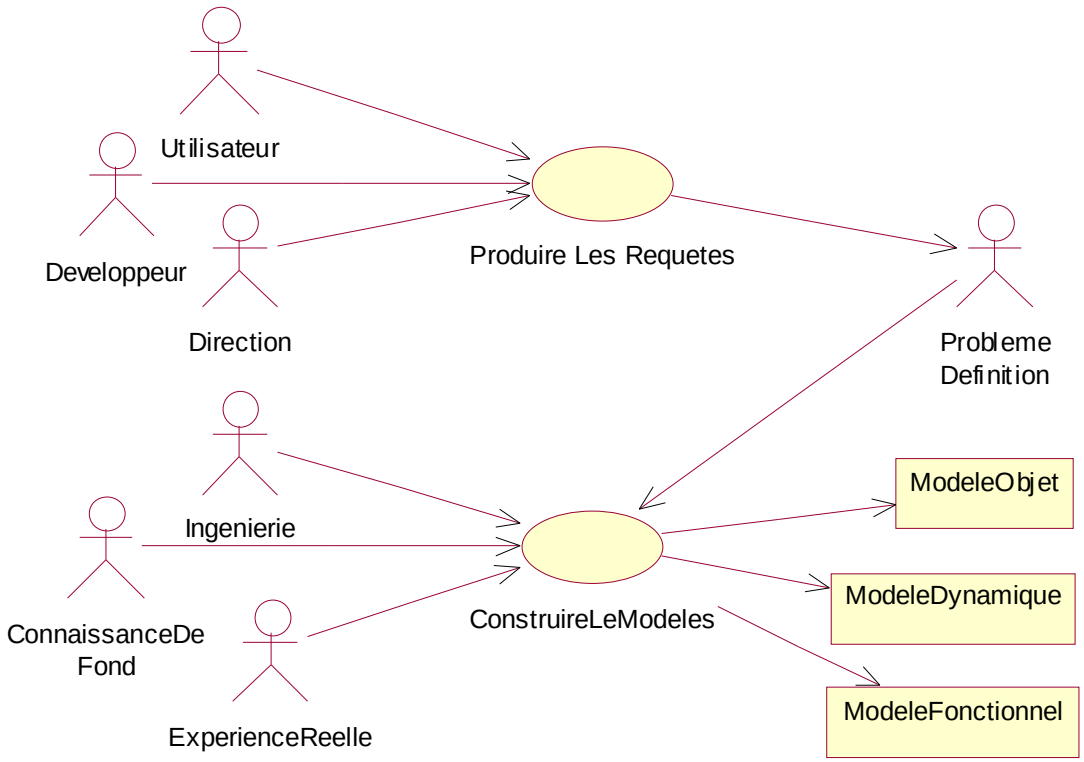


Figure 21: Vue d'ensemble du processus d'analyse

4.2 Étude de cas: les guichets automatiques bancaires

Sujet: Analyser la partie d'un logiciel bancaire permettant aux clients d'effectuer des opérations de retrait d'argent à partir d'un distributeur. Le distributeur lit la carte bancaire insérée par l'utilisateur. puis établit un dialogue avec celui-ci pour connaître les données de la transaction. Ensuite, il communique avec l'ordinateur central qui gère les comptes, pour réaliser l'opération. Enfin, il délivre si accord de la banque, la somme demandée et imprime un état.

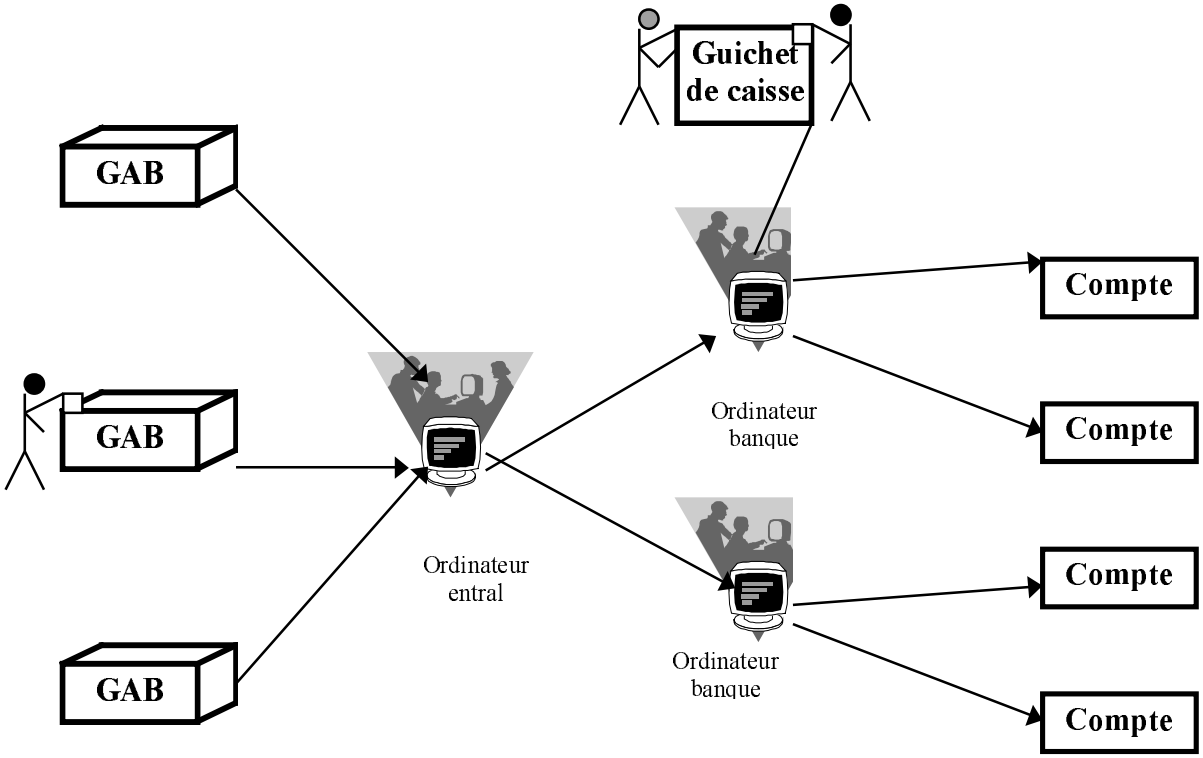


Figure 22: Réseau de guichets automatiques bancaires (GAB).

4.2.1 La construction du modèle objet

1. Identifier les classes d'objets
2. Préparer un dictionnaire des données
3. Identifier les associations entre les objets.
4. Identifier les attributs des objets
5. Organiser et simplifier le classes en utilisant l'héritage.
6. Vérifier les chemins d'accès.
7. Itérer en raffinant le modèle.
8. Grouper les classes dans des modules.

4.2.1.1 Identifier les classes d'objets

Les noms utilisés dans l'énoncé du problème sont souvent de bons candidats à l'élaboration des classes

Les objets représentent les entités physiques mais aussi des concepts (trajectoire, etc.).

Il ne faut pas au départ se soucier de l'héritage et des classes plus générales.

Exemple: un gestionnaire de bibliothèques

- Les livres
- Les journaux
- etc.

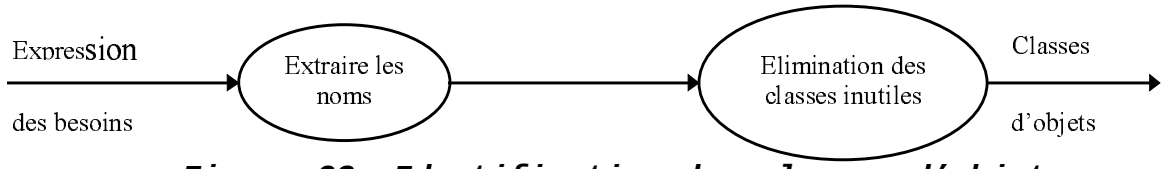


Figure 23: Identification des classes d'objets

Cas du GAB

Selon l'énoncé:

LOGICIEL **BANQUE CLIENT DISTRIBUTEUR CARTE** UTILISATEUR **COMPTE** SOMME ETAT

D'après les connaissances du problème:

LIGNE COMMUNICATION

4.2.1.1.1 Les critères pour éliminer les classes incorrectes

- Classes redondantes : conserver la plus descriptive.
- Classes jugées sans rapport avec le problème traité.
- Classes imprécises ou d'une portée trop large.
- Noms transformés en classes mais qui ne sont en fait que des attributs d'autres classes.
- Noms décrivant des opérations sur des objets et n'ayant pas de caractéristiques propres.
- Noms de classes représentant un rôle joué par la classe et non sa valeur intrinsèque (Propriétaire -> personne).
- Structure de conception (Arbre, liste, etc.)

Cas du GAB

Mauvaise classes:

UTILISATEUR: redondant avec **CLIENT**.

SOMME ETAT DONNEES: attributs de **OPERATION**.

LOGICIEL LIGNE COMMUNICATION: implèmentation.

Bonnes classes:

BANQUE CLIENT OPERATION DISTRIBUTEUR CARTE ORDINATEUR COMPTE

4.2.1.1.2 Préparer un dictionnaire des données

Le rôle, la portée, les restrictions d'utilisation, toutes les informations qui évitent de mauvaises interprétations doivent être précisés pour chaque objet.
Méthode:

- Écrire un paragraphe décrivant chaque classe.
- Décrire le rôle de la classe dans le problème.
- Décrire les associations, les attributs et les opérations.

Cas du GAB

COMPTE: Compte financier d'un particulier géré par une banque. Les comptes peuvent être de plusieurs types (compte chèque, compte épargne, etc.). Un client peut posséder plus d'un compte dans une même banque et/ou plusieurs comptes dans des banques différentes.

DISTRIBUTEUR: Machine permettant à un client possédant une carte, de réaliser des opérations sur ses comptes. Le distributeur demande les informations nécessaires pour la transaction, les envoi à l'ordinateur central pour validation et traitement puis délivre l'argent au client. On considère qu'il ne peut fonctionner indépendamment de l'ordinateur central.
BANQUE: Organisme financier qui gère les comptes des clients et délivre les cartes autorisant l'accès aux distributeurs.
ORDINATEUR: Ordinateur central avec lequel la banque gère les comptes de ses clients.
CARTE: Carte donnée à un client de la banque, lui permettant d'utiliser les distributeurs. A chaque carte sont associés un code d'accès et un numéro de la banque qui la délivre. Le code d'accès détermine les comptes accessibles par la carte, car certains peuvent ne pas l'être.
CLIENT: Personne physique ou morale possédant un ou plusieurs comptes dans une banque. La distribution entre personnes physiques et morales, nous apporte peu dans notre cas. Un client ayant des comptes dans des banques différentes sera considérée comme un client différent.
OPERATION: Demande réalisée par un client sur un des comptes. Nous-nous intéressons seulement au retrait d'argent.

4.2.1.2 Identifier les associations entre les objets.

Toutes les dépendances entre deux classes ou plus sont des associations.
Elles correspondent souvent aux verbes utilisés dans l'énoncé du problème.

Exemple:

- Lieu physique. (se trouver près de)
- Action directes (Conduite).
- Communication (Parler à).

Cas du GAB

Selon les verbes de l'énoncé:

1- Les clients effectuent des opérations.
2- Les opérations sont réalisées à partir d'un distributeur.
3- Le distributeur lit une carte.
4- Le distributeur établie un dialogue avec le client.
5- Le distributeur communique avec l'ordinateur central.
6- L'ordinateur central gère les comptes.
7- Le distributeur délivre la somme demandée.
8- Le distributeur imprime un état.

1. **Implicites**:

9- Les opérations concernent les comptes.
10- La banque maintient les comptes.
11- Les clients possèdent des cartes.
12- Les clients possèdent des comptes.
13- La banque possède un ordinateur.

2. **D'après les connaissances du domaine:**

14- La banque délivre les cartes.
15- La carte permet d'accéder aux comptes.
16- 16- La carte autorise les opérations.

4.2.1.2.1 Les critères pour supprimer les mauvaises associations

- Association dont l'une des classes a été supprimée.
- Associations jugées sans rapport avec le problème ou représentant une structure de conception.
- Une association doit décrire une propriété durable pas un événement éphémère. Certains besoins exprimant une action sous-entendent souvent une association.
- Les associations ternaires (ou plus) peuvent fréquemment être décomposées en associations binaires.
- Associations dérivées, redondante ou définies par une condition sur une attribut d'objet (plus jeune que)

Cas du GAB

En tenant compte des 16 associations dé[illegible]phe précedant, on peut faire les remarques suivantes:

La 1 peut être remplacée par: " **CLIENT** possède **CARTE** autorise **OPERATION**"
La 3 et la 4 ne sont pas des relations permanentes, mais des états passagers.
La 6 peut être remplacée par " **BANQUE** possède **ORDINATEUR** et gère **COMPTE**".
La 7 à supprimer, car SOMME n'a pas désignée comme classe.
La 8 à supprimer, car ETAT n'a pas désignée comme classe.

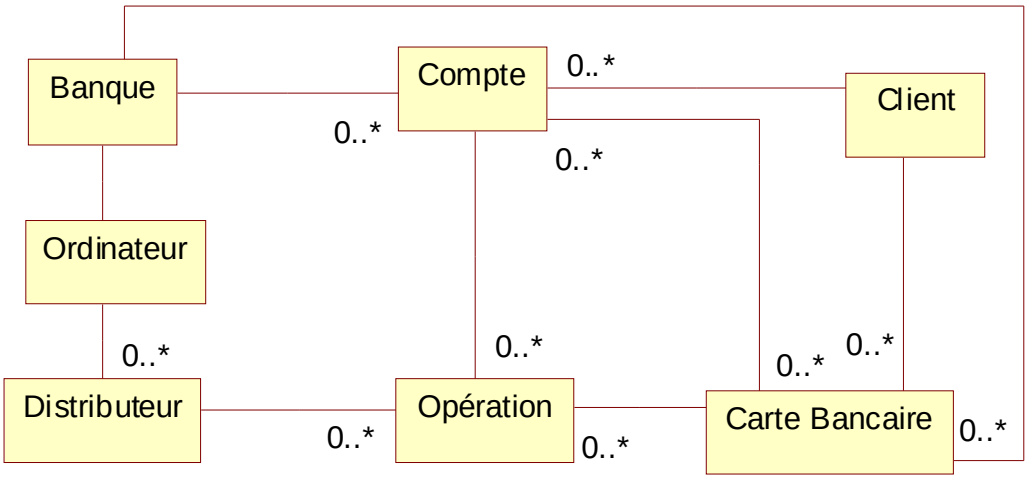


Figure 24:Diagramme Objet Initial.

4.2.1.3 Identifier les attributs des objets

Les attributs définissent des propriétés intrinsèques des objets : Nom, poids, couleur.
Les adjectifs utilisés dans l'énoncé du problème représentent certains attributs mais la plupart ne sont pas clairement exprimés.
Les attributs ne doivent pas être des objets (utiliser une association pour montrer la relation entre deux objets)
Il ne faut pas considérer que les attributs étant directement en relation avec l'application.

Cas du GAB

L'énoncé du problème ne nous donne be ations. Il faut rechercher à partir de ses propres connaissances.
Voici le diagramme des objets avec attributs.

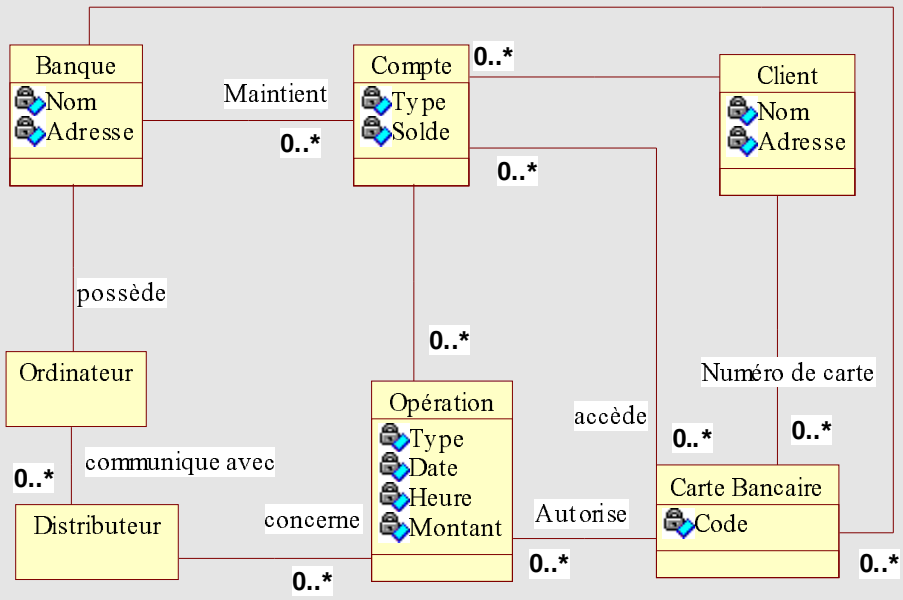


Figure 25: Diagramme Objet avec attributs.

Remarque:

L'association qualifiée permet de transformer des relations de type 1-n en des relations de type 1-1. L'analyse réel consiste à rechercher plusieurs attributs pour chaque classe.

4.2.1.3.1 Les critères pour éliminer les mauvais attributs

- Un attribut représente uniquement une valeur d'une entité. Si son existence propre est plus importante que sa valeur, alors il s'agit d'un objet et non d'un attribut.
- Seuls les attributs importants sont à considérer, les autres peuvent être omis.
- Un attribut paraissant sans relation ou incohérent avec les autres peut indiquer une classe à diviser. Une classe doit être simple et cohérente.
- Les attributs décrivant un état interne ne sont pas à prendre en compte dans l'analyse.

4.2.1.4 Raffiner en utilisant l'héritage

La notion d'héritage des classes, vers le haut - généralisation - ou vers le bas - spécialisation - doit être introduite pour clarifier le modèle.

Le but est de partager des structures communes entre différentes classes.

La recherche d'associations, d'opération ou d'attributs communs entre plusieurs classes permettent de donner lieu à ces regroupements.

L'héritage multiple doit être utilisé qu'en cas de nécessité car il complique les problèmes de conception. parfois, la création d'une super classe permet de L'éviter.

Cas du GAB

La simplicité du sujet ne nous permet pas d'introduire la notion d'héritage. Mais si les c pérations étaient réalisables par un employé depuis un guichet, nous aurions pu introduire l' héritage de la façon suivante:

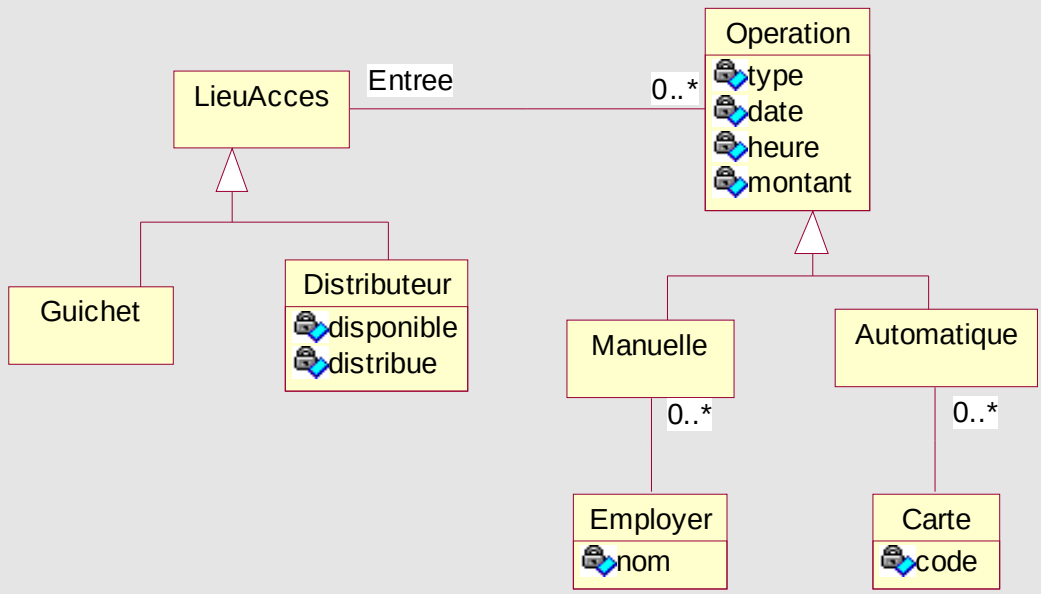


Figure 26:Extrait du Diagramme Objet avec héritage;

4.2.1.5 Tester les chemins d'accès

- La trace des chemins à travers le modèle doit fournir des résultats cohérents.
- Là où une valeur unique est attendue, un seul chemin doit y conduire.
- Le modèle peut-il répondre à toutes les questions que vous avez envie de poser ?
- Quelque chose de simple dans la réalité modélisée de façon complexe indique une incomplétude du modèle.

Cas du GAB

Selon le diagramme Objet de la figure 26, le client peut posséder une ou plusieurs cartes, qui chacune permet l'accès à un ou plusieurs comptes. De même un compte est accessible par une ou plusieurs cartes. Seul le code est transmis à l'ordinateur. Le sujet est extensible dans la mesure où on peut introduire des opérations de transfert de compte à compte.

4.2.1.6 Itérer en raffinant le modèle

Le modèle est rarement correct après un seul passage.

- Signes d'objets manquants
  - Attributs et opérations disparates dans une classe
  - Généralisation simple et claire difficile à obtenir
- Signe de classes non nécessaires.
  - Manque d'attributs, d'opérations, d'associations pour une classe.
- Signe d'associations manquantes.
  - Manque de chemin d'accès pour certaines opérations.
- Signe d'attributs mal placés.
  - Besoins d'accéder à un objet par un de ses attributs. Une association qualifiée est préalable.

Cas du GAB

Classe manquante: La carte peut être considérée comme support pour le code lu par l'utilisateur et même temps comme une autorisation d'accès aux comptes. Seul le code est transmis à l'ordinateur central par le support en plastique.
Classe non nécessaire: Le fait que le distributeur communique avec un ordinateur n'ajoute vraiment pas d'informations importantes au modèle, l'étude du logiciel de communication entre les deux machines aurait d'autres conséquences, mais dans notre cas, nous pouvons considérer l'ordinateur comme un moyen d'implémentation et non comme une classe.

4.2.1.7 Grouper les classes dans des modules

Pour faciliter la manipulation, des feuilles de même taille servent à la représentation des modules ou parties de module qui composent le modèle

- Les modules pour un système d'exploitation:
- La gestion des processus
- La gestion de la mémoire
- La gestion des périphériques.

Une feuille peut servir à l'organisation générale des classes les plus générales.

Cas du GAB

Intuitivement, on découpe le modèle en trois modules:

- Module **BANQUE**: Banque.
- Module **DISTRIBUTREUR**: Distributeur.
- Module **COMPTE**: Compte, Client, Opération, Carte support, Carte autorisation.

4.2.2 Diagramme objet initial

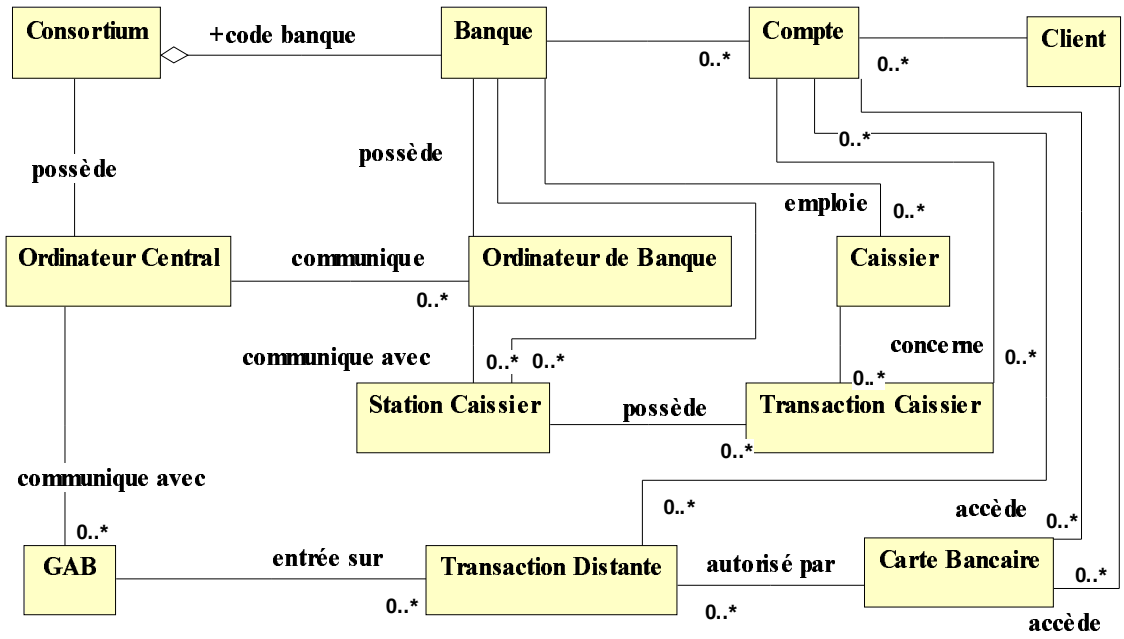


Figure 27:Diagramme d'objets initial pour le système GAB

5 Le modèle dynamique

5.1 Les caractéristiques du modèle dynamique

Le modèle dynamique montre le comportement du système en réponse aux externes.
Il indique le contexte et l'enchaînement des opérations responsables de l'évolution
Il ne prend pas en compte la façon dont les opérations sont réalisées, ni sur ce quoi elles agissent.
L'importance de ce modèle dépend du type de l'application étudiée. Elle sera importante dans des systèmes interactifs et minime dans des systèmes statistiques.
Un ensemble de diagrammes d'état représente graphiquement ce modèle

5.2 Les événements

Un événement est quelque chose qui se produit à un moment donné dans le temps.
Deux événements sans liens de causalité sont dit **concurrents**.
Les événements sont groupés dans des **classes d'événements** indiquant une structure et un comportement commun.

Exemple:

- **Vol part:** compagnie, numéro de vol, ville.
- **Bouton de souris appuyé:** bouton, position.
- **Chaîne de saisie composée:** texte.
- **Combiné soulevé**.
- **Chiffre composé:** chiffre.
- **Vitesse moteur entre en zone danger**.

5.3 Les scénarios

Un scénario est une séquence d'événements qui se produisent pendant une certaine exécution du système.

Exemple:

- L'appelant soulève le combiné
- La tonalité est déclenchée
- L'appelant tape un chiffre (3)
- La tonalité s'arrête.
- L'appelant tape un chiffre (3)
- L'appelant tape un chiffre (2)
- L'appelant tape un chiffre (5)
- L'appelant tape un chiffre (6)
- L'appelant tape un chiffre (3)
- L'appelant tape un chiffre (4)
- Le téléphone appelé commence à sonner.
- La tonalité de sonnerie commence dans le téléphone appelant.

- L'appelé répond.
- Le téléphone de l'appelé cesse de sonner.
- La tonalité de sonnerie cesse dans le téléphone appelant.
- Les téléphones sont connectés.
- L'appelé raccroche.
- Les téléphones sont déconnectés.
- L'appelant raccroche.

Cas du GAB

Voici un exemple de scénarios:

Le distributeur demande l'insertion d'une carte, le client insert la carte
Le distributaire accepte le carte et demande le code d'accès
Le client entre 4 5 8 7
Le distributeur demande validation à la banque, La banque valide
Le distributeur demande le type de l'opération, le client répond retrait
Le distributeur demande le montant désiré, le client entre 300 $
Le distributeur vérifie si le montant désiré est dans les limites autorisées et transmet l'opération à la banque qui la réalise et retourne le nouveau solde du compte
Le distributeur délivre la somme et demande au client de la prendre, le client la prend
Le distributeur demande au client s'il désire réaliser une autre opération, le client indique non.
Le distributeur imprime l'état, éjecte la carte et demande au client de les retirer, le client les retire
Le distributeur demande d'insérer une carte.

5.3.1 Un diagramme de suivi d'événements

Un diagramme d'événements représente chaque objet par une ligne verticale et chaque événement par une flèche horizontale reliant l'objet émetteur à l'objet récepteur.

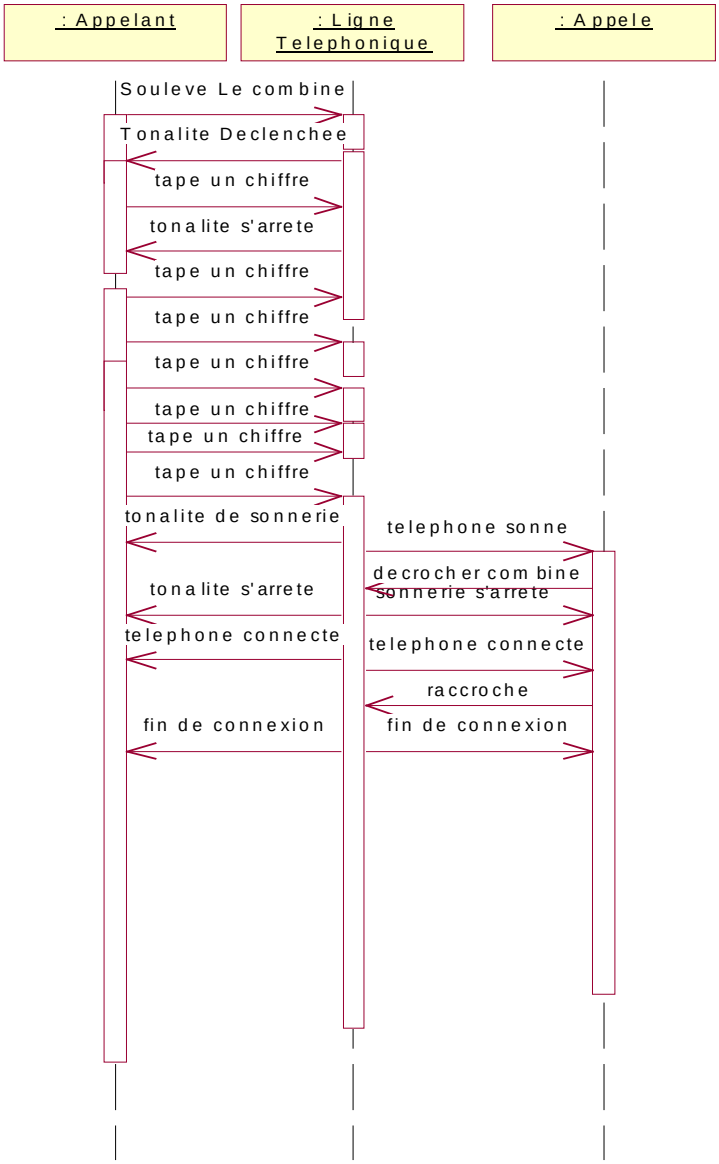


Figure 28: Suivie d'événements pour une appel téléphonique

5.4 Les états

Un état est une abstraction des valeurs des attributs et des liens d'un objet. Il correspond à l'intervalle entre deux événements reçus par une objet.

Exemple:

- État d'une banque solvable ou non.
- État liquide de l'eau.

État:	Sonnerie alarme.
Description:	L'alarme d'une montre sonne pour indiquer une heure cible
Séquence d'événement	- Régler une alarme (heure cible). - Une séquence sauf initiation alarme.

provoquant un état:	- Heure courante = heure cible.
Condition caractérisant l'état:	Alarme = { en marche ; heure cible ≤ heure courante ≤ heure cible + 20 sc ; aucun bouton n'a é té pous é depuisheure cible
Événements acceptés dans l'état:	**événement** / **action** / **prochain état** heure courante = heure cible + 20 / réinitialiser alarme / normal bouton poussé (n'importe lequel) / réinitialiser alarme / normal

Figure 29: Différentes caractéristiques d'un état.

5.4.1 Les diagrammes d'états

Un diagramme d'états est un graphe dirigé dans lequel les noeuds représentent les états du système et les transitions entre les états.

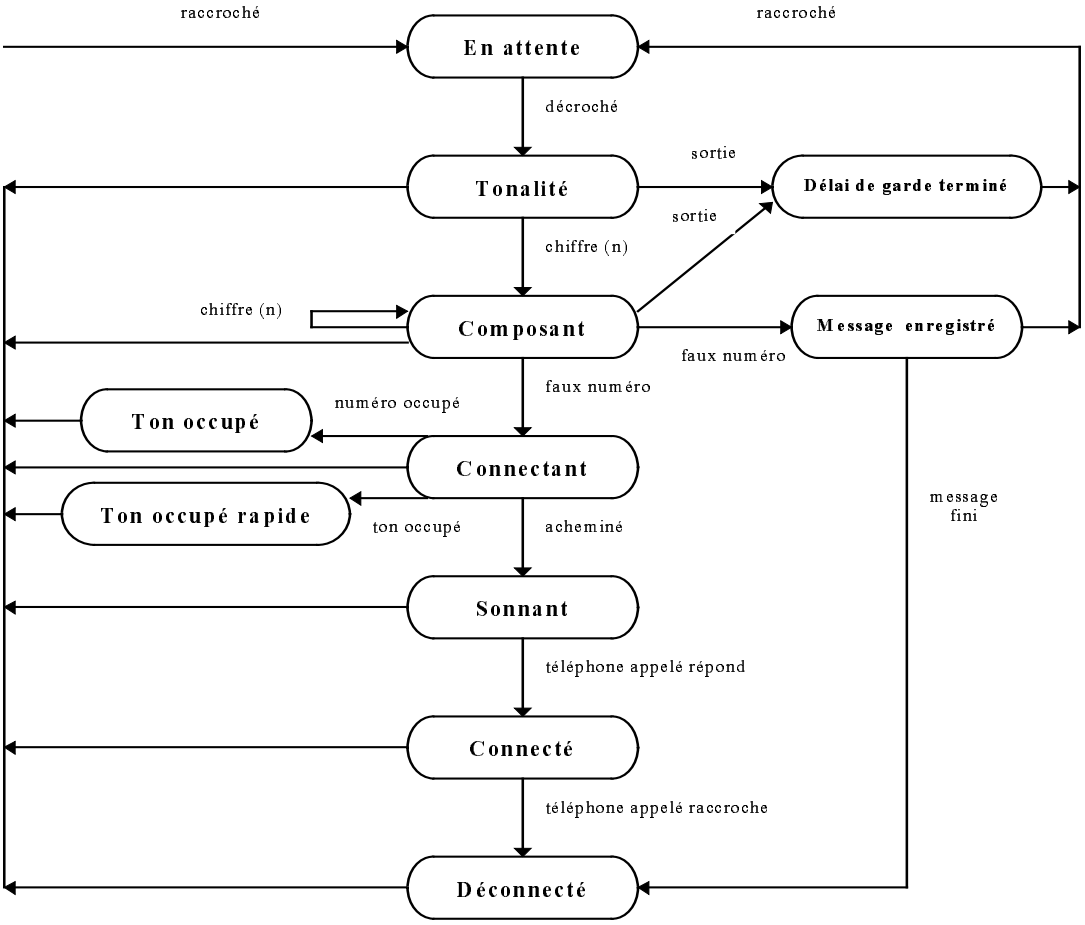


Figure 30: Diagramme d'états pour la ligne téléphonique

5.5 Les conditions et les opérations

Une condition est une fonction booléenne de valeurs d'objets valide sur un intervalle de temps.
Une activité est une opération prenant du temps à se réaliser. Les activités sont associées à des états et représentent les réalisations du monde réel.
Une action est une opération instantanée.
Les actions sont associées aux événements et sont généralement de nature formelle.

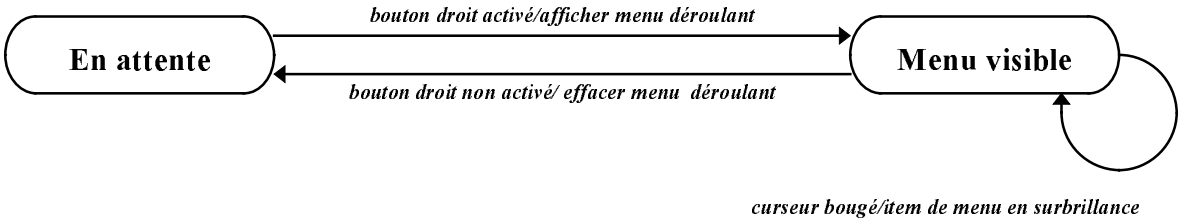


Figure 31 : Actions pour le menu déroulant

Exemple de diagramme d'état

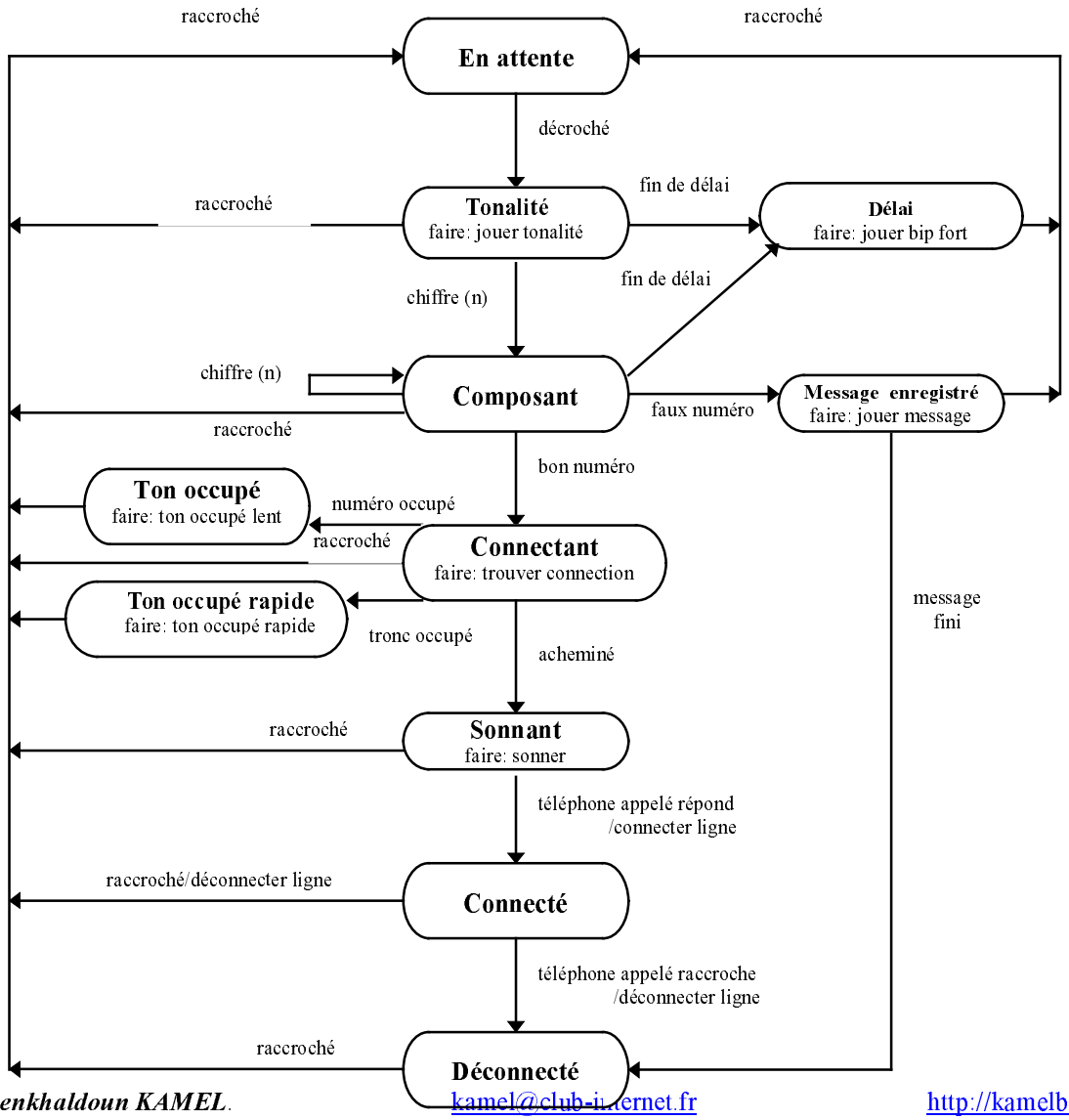


Figure 32: Diagramme d'états pour une ligne téléphonique

5.6 Les diagrammes d'états imbriqués

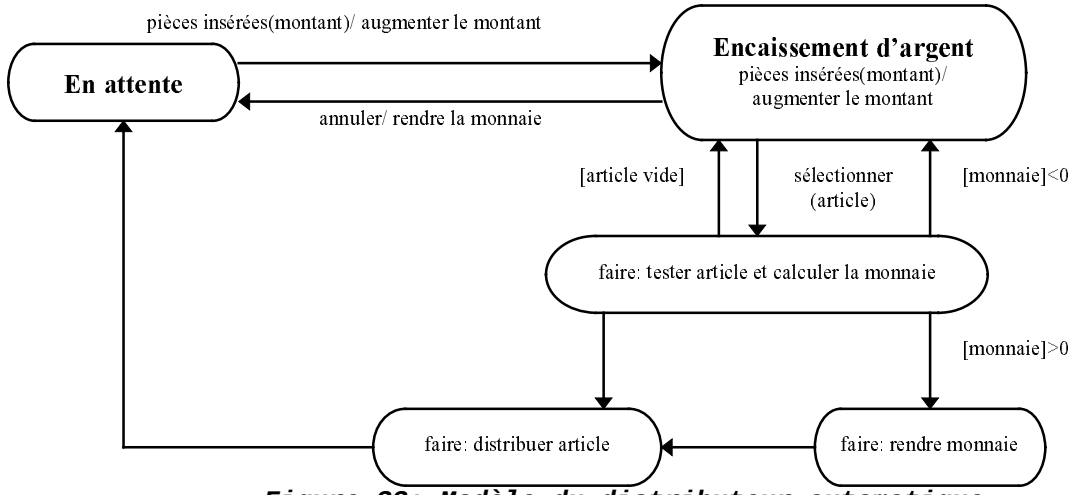


Figure 33: Modèle du distributeur automatique

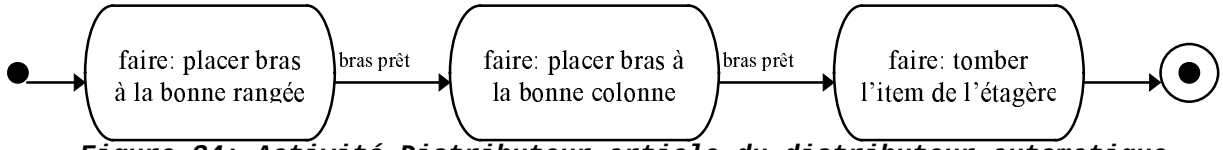


Figure 34: Activité Distributeur-article du distributeur automatique

5.7 La généralisation d'événements

Les événements peuvent être organisés en une hiérarchie de généralisation avec héritage des attributs d'événements
Fournir une hiérarchie d'événements permet d'utiliser différents niveaux d'abstraction à différents endroits du modèle.

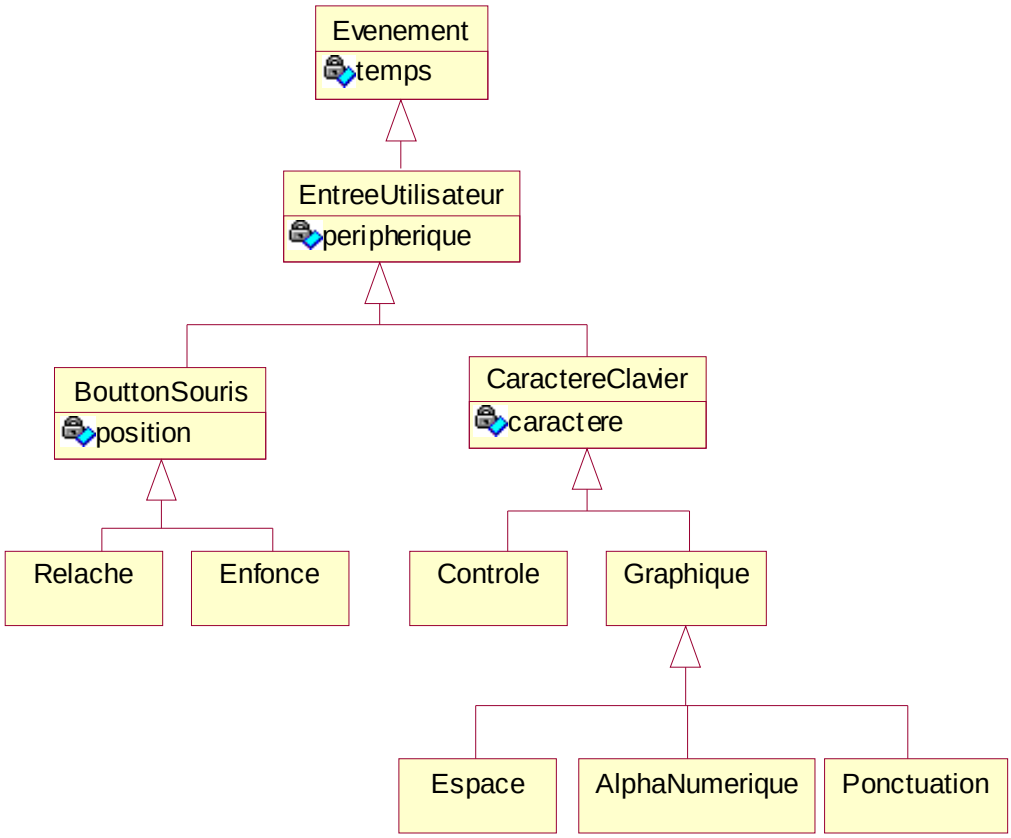
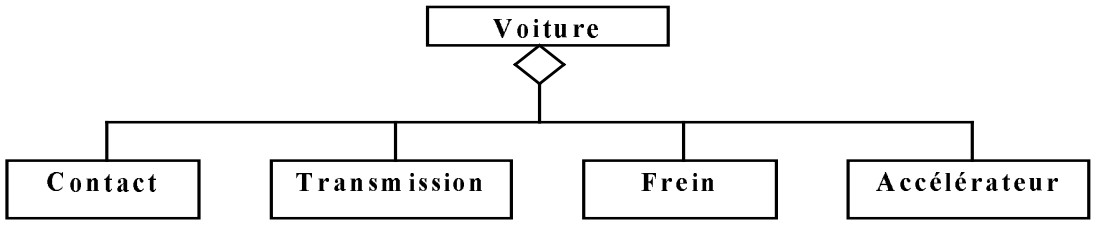
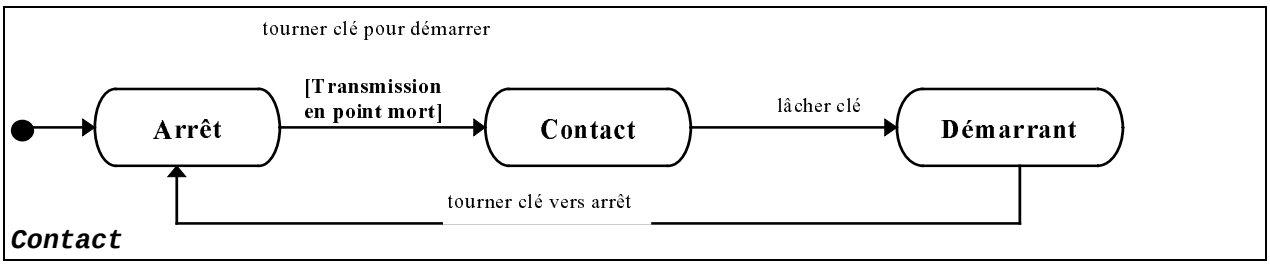


Figure 35: Représentation partielle de la hiérarchie pour les événements homme-machine

5.8 La concurrence d'agrégation





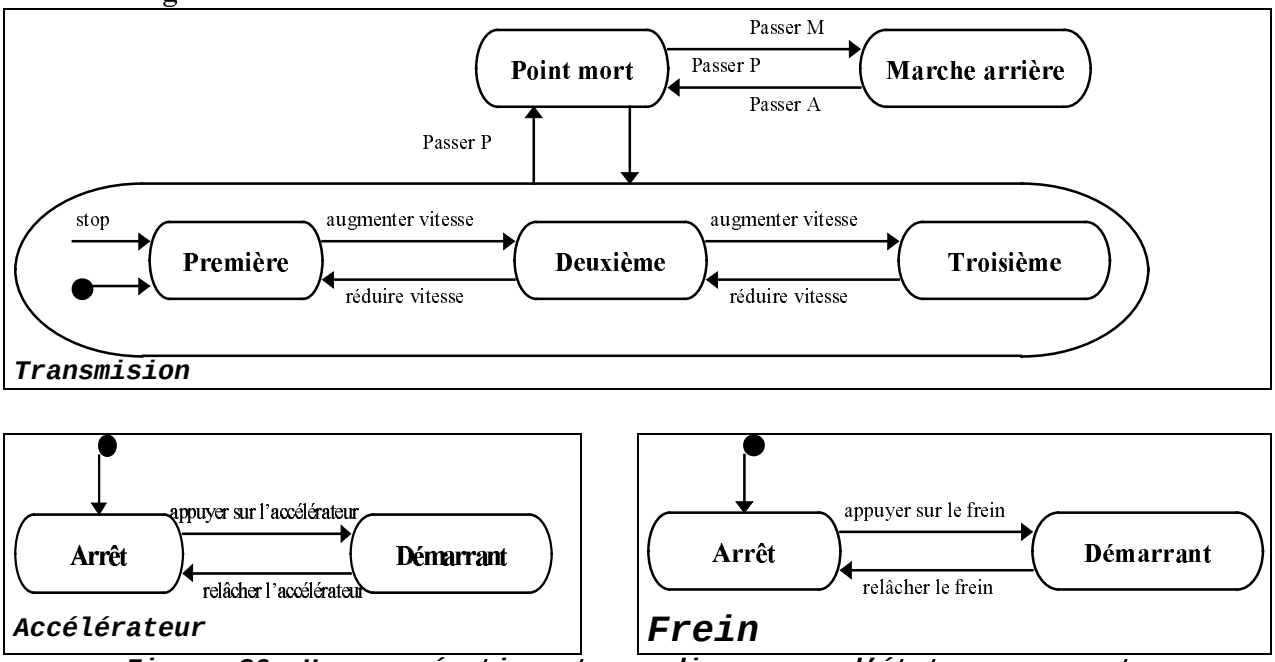


Figure 36: Une aggrégation et ses diagrammes d'états concurrents

5.9 Les étapes pour créer le modèle dynamique

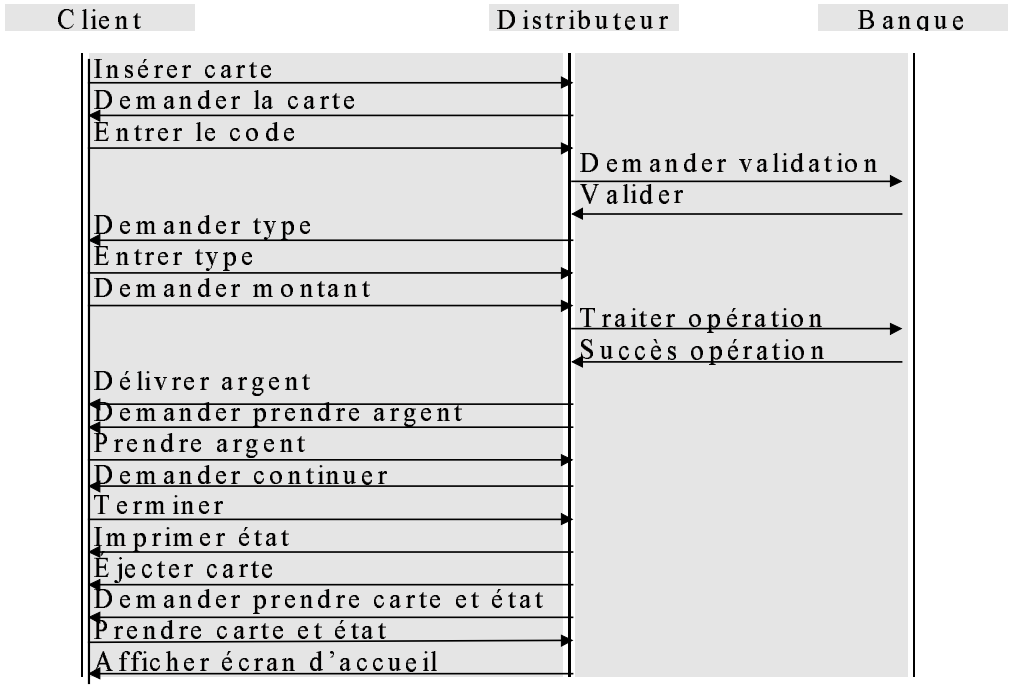
1. Préparer un scénario.
2. Identifier les événements.
3. Construire un diagramme d'état.
4. Vérifier la correspondance des événements entre les objets.

5.9.1 Préparer un scénario.

Un scénario est un dialogue entre le système et un objet extérieur (utilisateur, capteur, etc.). Il représente une suite d'événements.
Les premiers scénarios à réaliser, correspondent à des échanges de messages dans des cas normaux de fonctionnement.
Les suivants font référence aux cas particuliers et aux erreurs.

- Valeurs incorrectes.
- Oublis.

Cas du GAB


Client
Distributeur
Banque
Insérer carte
Demander la carte
Entrer le code
Demander validation
Valider
Demander type
Entrer type
Demander montant
Traiter opération
Succès opération
Délivrer argent
Demander prendre argent
Prendre argent
Demander continuer
Terminer
Imprimer état
Éjecter carte
Demander prendre carte et état
Prendre carte et état
Afficher écran d'accueil

5.9.2 Identifier les événements

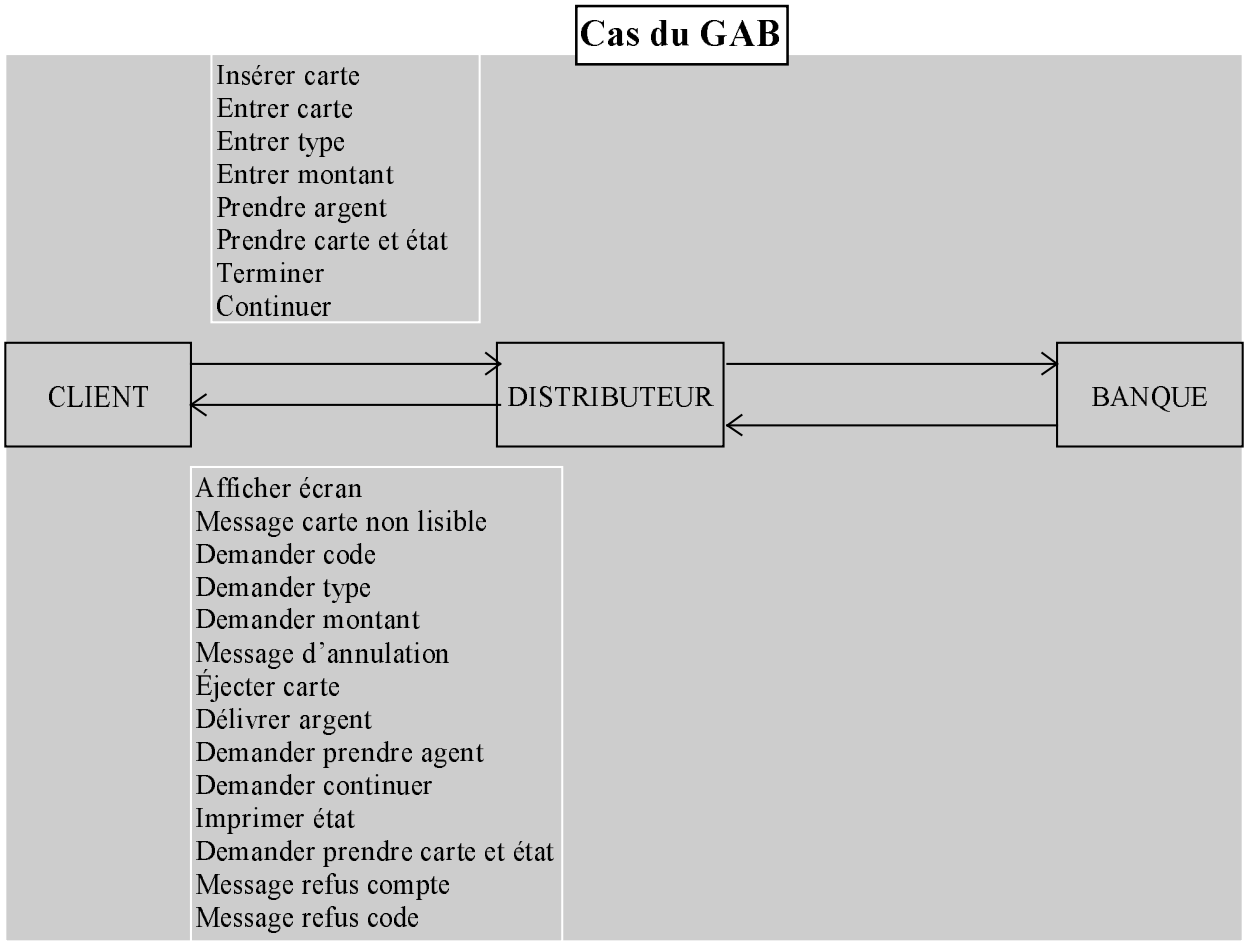
Toute forme de communication entre le système et le monde extérieur est une événement.
Les acteurs sont à associer aux objet, tendis que les informations échangées appartiennent aux attributs de l'événement.
Les événements ayant des effets identiques sur le flot de contrôle sont regrouper en classe.
Des diagramme de trace sont réalisés à partir de chaque scénario. Une liste ordonnée d'événements
- flèches - entre différents objets - colonnes - représente ces diagrammes.
Les événements entre classes ou groupe de classe (module) figure dans un diagramme de flots d'événements.

Cas du GAB

Exemple:

- Insérer la carte
- Demander le code d'accès
- Entrer le code d'accès
- etc.

5.9.3 Diagramme de trace et de flot d'événements



5.9.4 Construire un diagramme d'état

Considérer tous les événements qui affectent la classe dans un des diagrammes de trace d'événements.
Donner si besoin un nom aux différents états représentés par les intervalles entre les événements.
Le diagramme d'état initial traduit la séquence des événements et des états relevé dans le diagramme de trace.
Intégrer les autres scénarios au diagramme
Le diagramme est terminé lorsqu'il couvre tous les scénarios et lorsqu'il prend en compte tous les événements pouvant survenir.

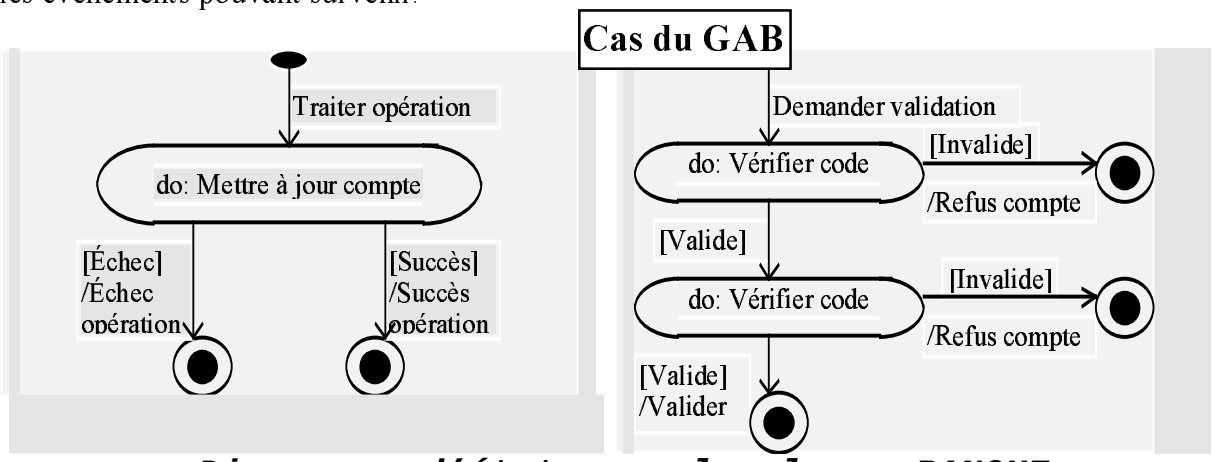


Diagramme d'état pour la classe BANQUE

5.9.5 Correspondance des événements entre les objets

Les états sans prédécesseurs ou sans successeurs représentent-il bien des points de départ ou d'arrivée ?
Une correspondance est nécessaire entre la propagation des événements d'objets à objets dans les diagrammes d'états et dans le scénarios.
Tout événement doit avoir un émetteur et un récepteur, occasionnellement le même objet.

6 Le modèle fonctionnel

6.1 Rôle du modèle fonctionnel

Le modèle fonctionnel décrit des aspects du système qui transforment les valeurs en utilisant des fonctions, des représentations, des contraintes et des dépendances fonctionnelles

Exemples:

L'ensemble des feuillets d'impôts et des instructions
- **Valeurs entrantes**
  - revenus, charges, déductions, etc.
- **Valeurs sortantes**
  - calcul d'impôts

Un compilateur
- **Valeurs entrantes**
  - texte d'un programme
- **Valeurs sortantes**
  - fichier objet dans un autre langage

6.2 Les diagrammes à flot de données

Un diagramme à flots de données DFD est une représentation graphique d'un modèle fonctionnel, montrant les dépendances entre les valeurs et le calcul des valeurs de sortie sans considération du moment ou de l'exécution même de la fonction

Un flot de données contient:
- **des traitements**
  - transformation des données
- **des flots de données**
  - transport des données.
- **des objets acteurs.**
  - production et consommation des données
- **des objets réservoirs**
  - stockage des données

6.3 Les traitements

Un traitement dans un DFD est une opération qui transforme les valeurs des données.

Somme de deux nombres.

Facturation d'un ensemble de transactions de cartes de crédit.

Les traitements sont implémentés comme des méthodes d'opérations sur des classes d'objets.

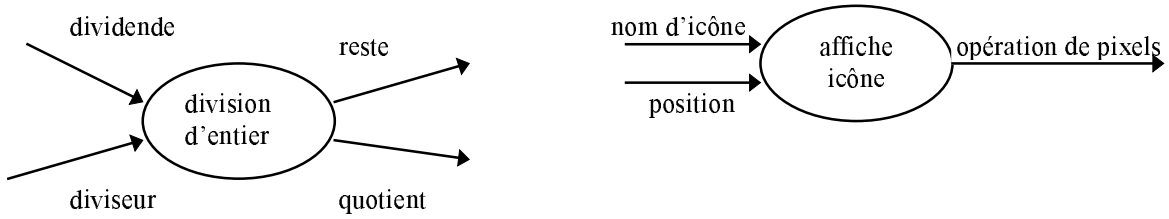


Figure 37: Traitements

6.4 Les flots de données

Un flot de données dans un DFD relie la sortie d'un objet ou d'un traitement à l'entrée d'un autre objet ou d'un autre traitement. Il représente une valeur de donnée intermédiaire dans un calcul.

Un flot de données est représenté par une flèche entre le producteur et le consommateur de la valeur de donnée.

Fufure : Flots de données pour coopier une valeur et décomposer une valeur d'aggrégat

6.5 Les acteurs

Un acteur est un objet qui dirige le graphe de flots de données en produisant ou en consommant des valeurs

L'utilisateur d'un programme.

Un thermostat.

Un moteur piloté par un ordinateur.

Les acteurs sot attachés aux entrées et aux sorties d'un diagramme à flots de données

Un acteur est représenté par un rectangle.

6.6 Les réservoirs de données

Un réservoir de données est un objet passif à l'intérieur d'un diagramme à flots de données qui stocke des données pour un accès ultérieur.

Les listes
Les tables
Un compte en banque.

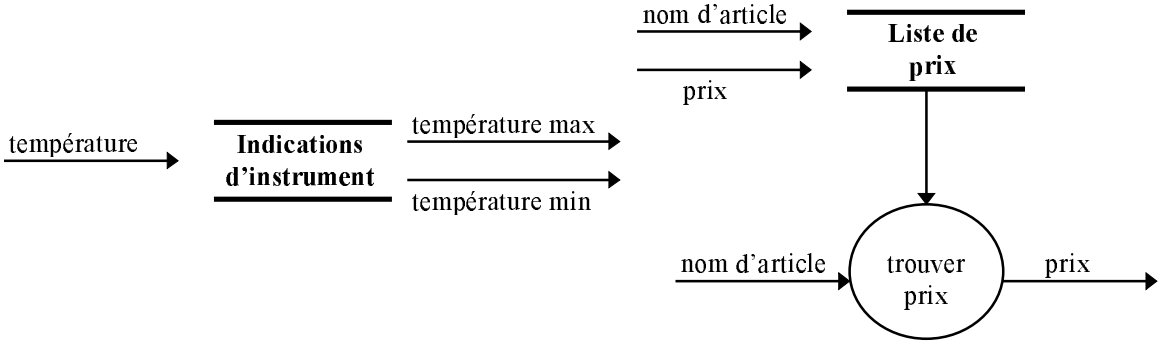


Figure 38: Exemple de réservoirs de données

6.7 Les flots de contrôle

Un flot de contrôle dans un DFD est une valeur booléenne qui décide de l'exécution d'un processus.

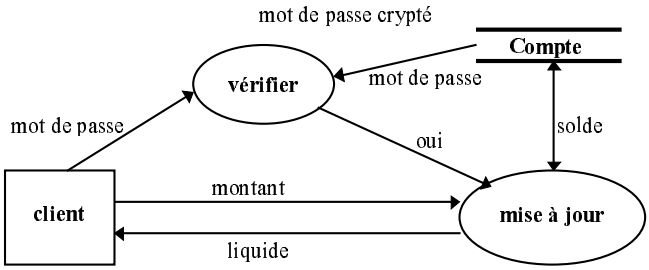


Figure 39: Flot de contrôle

7 Les différentes étapes pour créer le modèle fonctionnel

1. Identifier les valeur d'entrée et de sortie : Ces valeurs sont les paramètres des événements.
2. Construire les DFD : Un DFD indique comment une valeur de sortie est obtenue à partir des valeurs en entrée.
3. Décrire les fonctions. : Sous forme déclarative ou procédurale.
4. Identifier les contraintes entre les objets : Dépendance fonctionnelle entre les objets.
5. Spécifier les critères d'optimisation : Examiner les opérations similaires
6. Itération de l'analyse : Raffiner le modèle, reformuler les besoins.