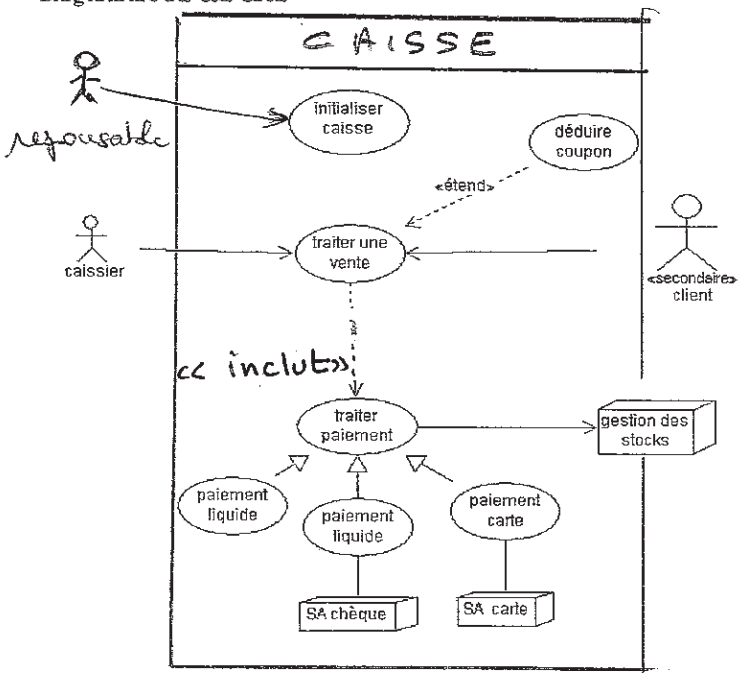
Diagramme des Uses Cases

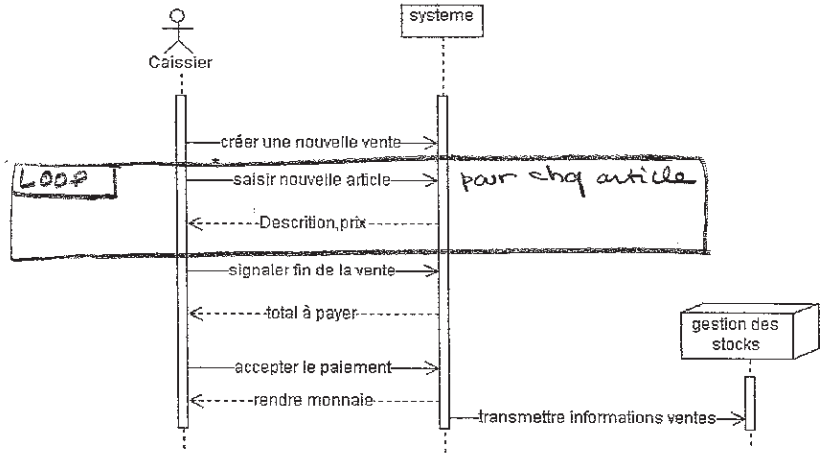


Chaque cas doit être décrit (scénario nominal, alternatif et d'exception) accompagné d'un glossaire.

A titre d'exemple, les scénarii de traiter une vente avec paiement en liquide sont donnés.

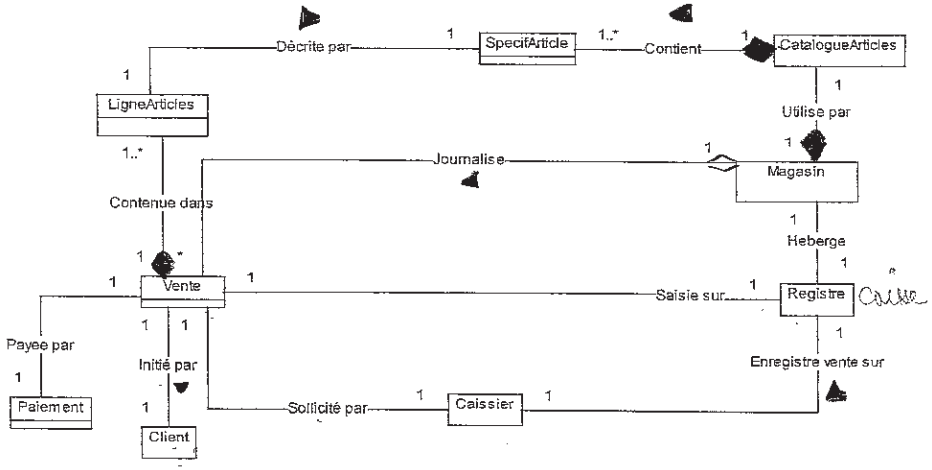
ne pas confondre interaction et UC :
1 UC est 1 ensemble de SCENARIOS,
eux mêmes composés d'interactions.

Diagramme de séquence système de traiter une vente : approche fonctionnelle !



Ce diagramme montre que l'acteur aura à sa disposition 4 opérations dans l'interface de son application.

Diagramme de classe du domaine :



Sommaire d'identification

Titre : Traiter une vente

Type : essentiel détaillé

Résumé : un client arrive à une caisse avec des articles qu'il souhaite acheter. Le caissier enregistre les achats et récupère le paiement. À la fin de l'opération, le client part avec les articles.

Acteurs : Caissier (principal), *Client (secondaire).*

Date de création : 17/05/00

Date de mise à jour : 10/11/00

Description des enchaînements

Préconditions :

- La caisse est ouverte ; un caissier y est connecté.

Scénario nominal :

1. Ce cas d'utilisation commence quand un client arrive à la caisse avec des articles qu'il souhaite acheter.	
2. Le caissier enregistre chaque article. S'il y a plus d'un exemplaire par article, le caissier indique également la quantité.	3. La caisse détermine le prix de l'article et ajoute les informations sur l'article à la vente en cours. La caisse affiche la description et le prix de l'article en question. + montant
4. Après avoir enregistré tous les articles, le caissier indique que la vente est terminée.	5. La caisse calcule et affiche le montant total de la vente.
6. Le caissier annonce le montant total au client.	
7. Le client choisit le type de paiement : a. En cas de paiement cash, exécuter le cas d'utilisation « Traiter le paiement en liquide » ; b. En cas de paiement par carte de crédit, exécuter le cas d'utilisation « Traiter le paiement par carte de crédit » ; c. En cas de paiement par chèque, exécuter le cas d'utilisation « Traiter le paiement par chèque».	
	8. La caisse enregistre la vente effectuée et imprime un ticket.
9. Le caissier donne le ticket de caisse au client.	
10. Le client s'en va avec les articles qu'il a achetés.	

Enchaînements « alternatifs » :

A1 : numéro d'identification inconnu

L'enchaînement A1 démarre au point 2 du scénario nominal.

3. La caisse indique au caissier que le numéro d'identification de l'article est inconnu. L'article ne peut alors être pris en compte dans la vente en cours.

Le scénario nominal reprend au point 2.

Enchaînements d'exception :

E1 : client ne pouvant payer

L'enchaînement E1 démarre au point 6 du scénario nominal.

7. Le client ne peut payer le total avec aucun moyen de paiement autorisé.
8. Le caissier annule l'ensemble de la vente et le cas d'utilisation est terminé.

Sommaire d'identification

Titre : Traiter le paiement en liquide

Résumé : un client paie en liquide le total affiché par la caisse enregistreuse.

Acteurs : Caissier (principal), *Client (secondaire)*.

Date de création : 17/05/00 **Date de mise à jour** : 06/12/00

Version : 1.1

Description des enchaînements

Préconditions :

- La vente est clôturée.

Scénario nominal :

1. Ce cas d'utilisation commence quand un client choisit de payer en espèces, après avoir été informé du montant total de la vente.	
2. Le client donne en paiement une somme en espèces ; elle est éventuellement plus élevée que le montant total de la vente.	
3. Le caissier enregistre la somme donnée par le client.	4. La caisse affiche la somme qui doit être rendue au client.
5. Le caissier encaisse l'argent reçu et sort la monnaie qu'il doit rendre.	
6. Le caissier rend la monnaie au client.	

Enchaînements « alternatifs » ou d'exception :

E1 : client ne pouvant payer

L'enchaînement E1 démarre au point 1 du scénario nominal.

2. Le client n'a pas assez de liquide pour payer.
3. Le caissier annule l'ensemble de la vente et le cas d'utilisation est terminé, ou le client paie avec un autre moyen de paiement (Cf. « Traiter le paiement par chèque », ou « Traiter le paiement par carte de crédit »).

E2 : caissier ne pouvant rendre la monnaie

L'enchaînement E1 démarre au point 4 du scénario nominal.

5. Le tiroir de la caisse ne contient pas assez d'espèces pour qu'il soit possible de rendre la monnaie.
6. Le caissier demande des espèces supplémentaires à son supérieur ou propose une autre méthode de paiement au client (Cf. « Traiter le paiement par chèque », ou « Traiter le paiement par carte de crédit »).

Diagramme de séquences d'analyse de traiter une vente :

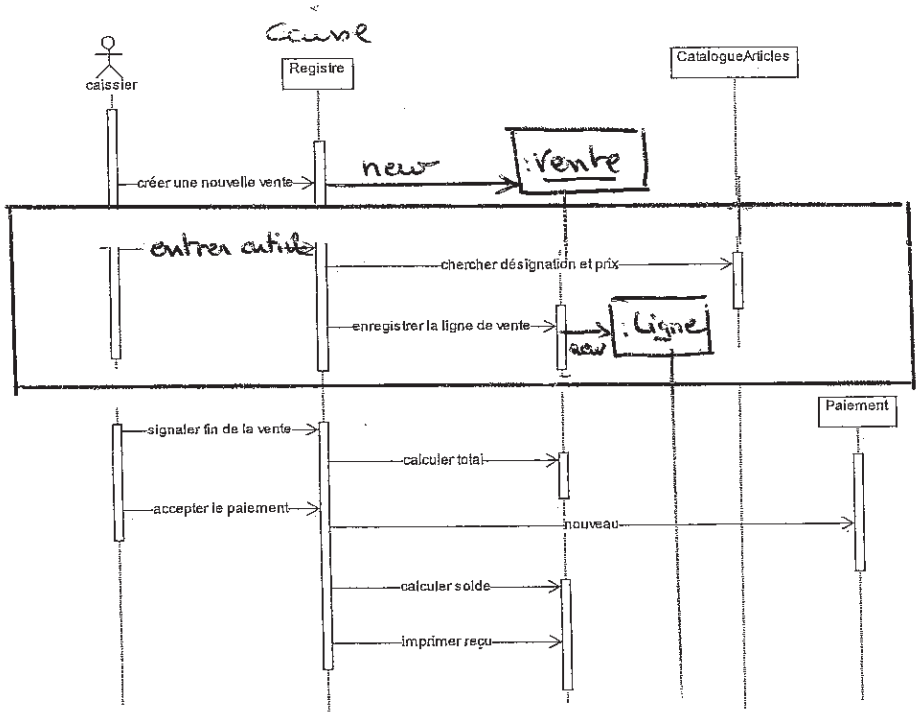
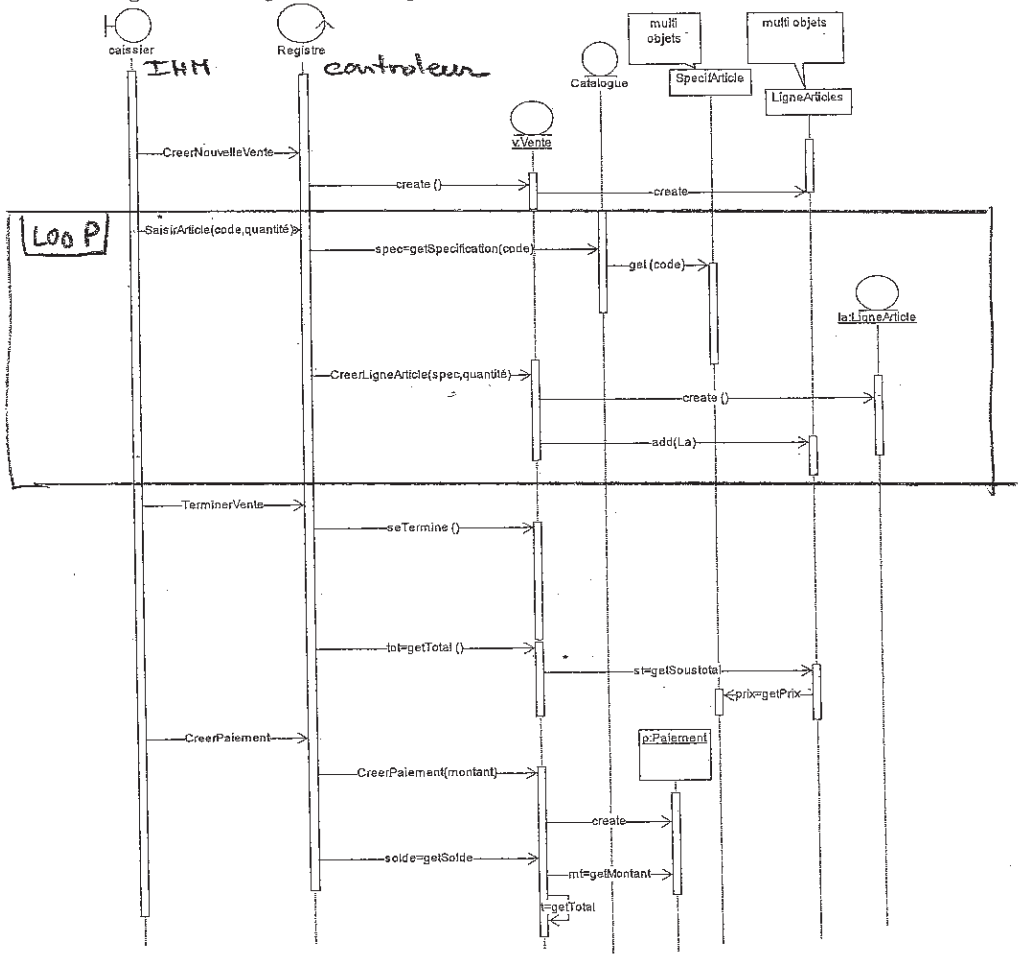
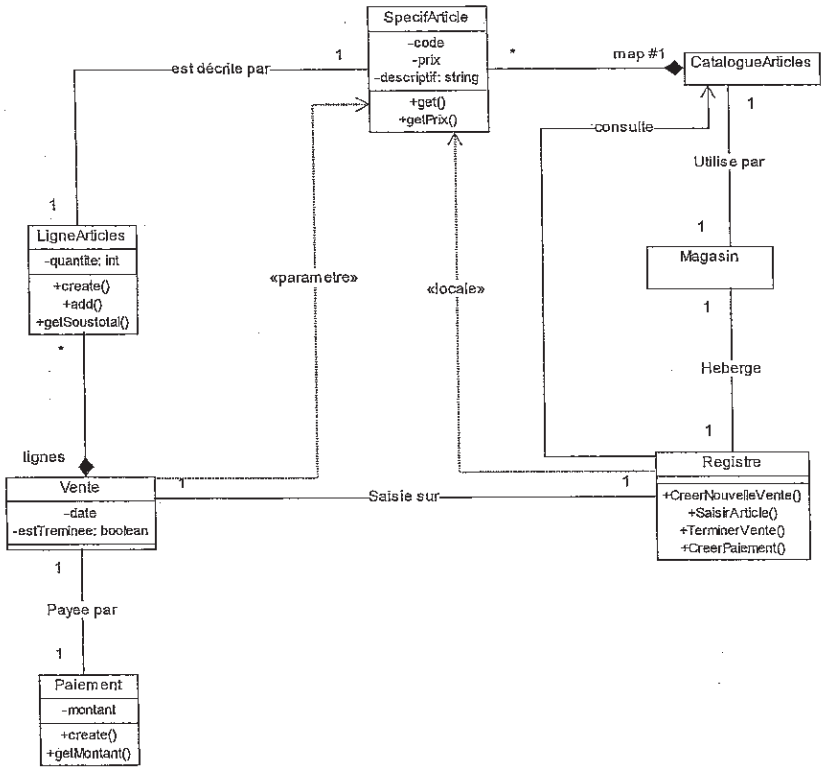


Diagramme de séquence de conception de traiter une vente :



Les opérations create (sont des constructeurs).
A partir de ce diagramme, il est possible de créer un premier diagramme de classe de conception qui montre les classes intervenant dans le cas d'utilisation traiter une vente avec paiement en liquide. Ce diagramme est indépendant de tout langage d'implantation.

Les dépendance en pointillés montrent que :
Un vente reçoit en paramètre une spécification article, le registre doit stocker en local la spécification donnée par le catalogue.


SpecifArticle
-code
-prix
-descriptif: string
+get()
+getPrix()
est décrite par
1
*
map #1
CatalogueArticles
1
consulte
Utilise par
1
Magasin
1
Heberge
1
Registre
+CreerNouvelleVente()
+SaisirArticle()
+TerminerVente()
+CreerPaiement()
1
LigneArticles
-quantite: int
+create()
+add()
+getSoustotal()
*
«parametre»
«locale»
lignes
Vente
-date
-estTreminee: boolean
1
Saisie sur
1
1
Payee par
1
Paiement
-montant
+create()
+getMontant()

La classe Paiement

```
public class Paiement
{
   private Monnaie montant ;
   public Paiement(Monnaie montantPresente) { montant =
                                              montantpresente ; }
   public Monnaie getMontant { return montant ; }
}
```

La classe CatalogueProduits

```
public class CatalogueProduits
{
   private Map specificationsProduits = new HashMap() ;
   public CatalogueProduits()
   {
      // échantillon de données
      CodeArticle id1 = new CodeArticle(100) ;
      CodeArticle id2 = new CodeArticle(200) ;
      Monnaie prix = new Monnaie (3) ;

      SpecificationProduit sp ;
      sp = new SpecificationProduits (code1, prix, "produit 1") ;
      specificationsProduits.put (code1, sp) ;
      sp = new SpecificationProduits (code2, prix, "produit 2") ;
      specificationsProduits.put (code2, sp) ;
   }
   public SpecificationProduit getSpecification (codeArticle, code)
   {
      return (SpecificationProduit)specificationsProduits.get(code) ;
   }
}
```

La classe LigneArticles

```
public class LigneArticles
{
   private int quantite ;
   private SpecificationProduit specProduit ;
   public LigneArticles(SpecificationProduit spec, int quantite)
   {
      this.specProduit = spec ;
      this.quantite = quantite ;
   }
   public Monnaie getSousTotal()
   {
      return specProduit.getPrix().times(quantite) ;
   }
}
```

La classe Magasin

```
public class Magasin
{
   private CatalogueProduits catalogue = new CatalogueProduits() ;
   private Registre registre = new Registre(catalogue) ;
   public Registre getRegistre() { return registre ; }
}
```

La classe Registre

```
public class Registre
{
   private CatalogueProduits catalogue ;
   private Vente vente ;
   public Registre (CatalogueProduits catalogue)
   {
      this.catalogue = catalogue ;
   }
   public void terminerVente()
   {
   vente.seTerminer() ;
   }
   public void saisirArticle(CodeArticle code, int quantite)
   {
      SpecificationProduit spec = catalogue.getSpecification(code) ;
      vente.creerLigneArticles(spec, quantite) ;
   }
   public void creerNouvelleVente()
   {
      vente = new Vente() ;
   }
   public void creerPaiement (Monnaie montantPresente)
   {
      vente.creerPaiement(montantPresente) ;
   }
}
```

La classe SpecificationProduit

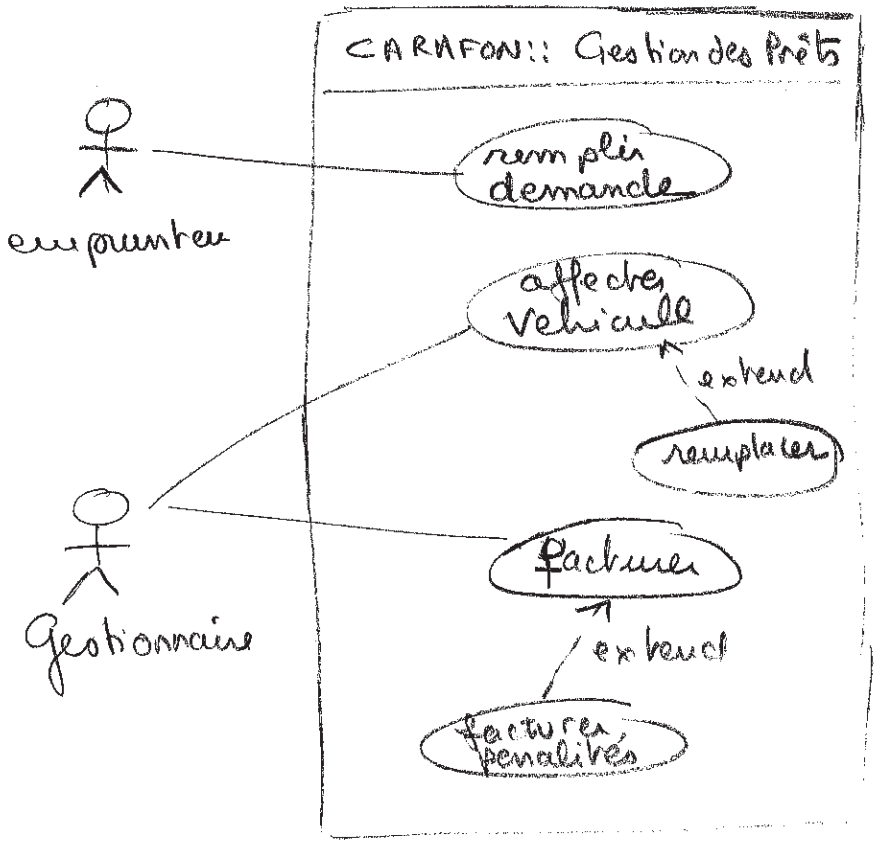
```
public class SpecificationProduit
{
   private CodeArticle code ;
   private Monnaie prix ;
   private String descriptif ;
   public SpecificationProduit
      (CodeArticle code, Monnaie prix, String description)
   {
      this.code = code  ;
      this.prix = prix ;
      this.descriptif = descriptif ;
   }

   public CodeArticle getCodeArticle() { return code ;}
   public Monnaie getPrix() { return prix ; }
   public String getDescriptif() { return descriptif ; }
}
```

La classe Vente

```
public class Vente
{
   private List lignes = new ArrayList() ;
   private Date date = new Date() ;
   private boolean estTerminee = false ;
   private Paiement paiement ;
   public Monnaie getSolde() ;
   {
   return paiement.getMontant().minus(getTotal()) ;
   }
   public void seTerminer() { estTerminee = true ;}
   public boolean estTerminee() { return estTerminee ;}
   public void creerLigneArticles
      (SpecificationProduit spec, int quantite)
   {
   lignes.add(new LigneArticles (spec, quantite)) ;
   }
   public Monnaie getTotal()
   {
   Monnaie total = new Monnaie() ;
   Iterator i = lignes.iterator() ;
   while(i.hasNext())
   {
   LigneArticles la = (LigneArticles) i.next() ;
   total.add(la.getSousTotal()) ;
   }
   return total ;
   }
   public void creerPaiement(Monnaie montantPresente)
   {
      paiement = new Paiement (montantPresente) ;
   }
}
```

Carafon UC

Acteurs emprunteur, gestionnaire, chef Atelier
On peut faire 2 catégories (ou 2 paquets)



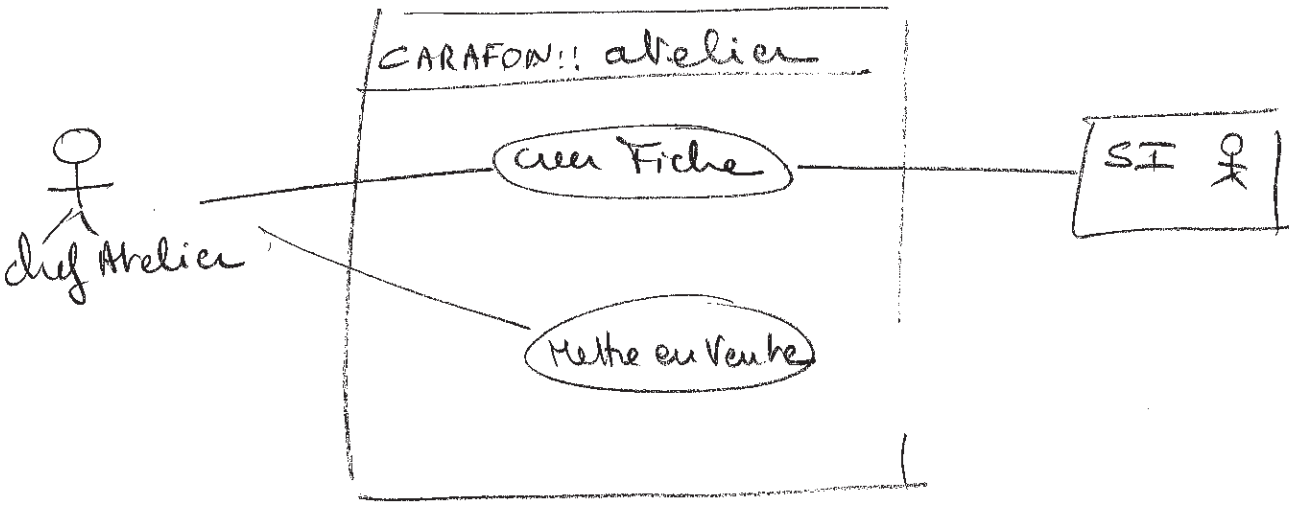


Diagramme de séquences Système