الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Centre Universitaire
Abdelhafid Boussouf Mila

المركز الجامعي
عبد الحفيظ بوالصوف ميلة

Institut des sciences et de la technologie
Département des Mathématiques et de
l'Informatique

معهد العلوم والتكنولوجيا
قسم الرياضيات والاعلام الالي

www.centre-univ-mila.dz

Cours de
Modélisation des applications web

Conforme au programme de la 1ere année Master STIC
(Science et Technologie de l'Information et de la Communication)

Présenté par :
Dr. BOUZAHZAH Mounira

Année Universitaire : 2019/2020

Objectif du cours

Cette matière aborde l’aspect architectural d’une application Web, puis présente un guide méthodologique à travers un processus de développement (UP) d'application web, et une extension d’UML pour le web.

Contenu de la matière

Chapitre1 : Introduction aux applications web

1. Introduction
2. Vue d'ensemble des technologies Web, Définition d'une application Web
   2.1. Définitions
   2.2. Les avantages des applications web
   2.3. Les types des applications web
   2.4. Applications web Vs applications native (classique)
   2.5. Applications web Vs sites web
   2.6. Les principes de base des applications web
3. Définition de l'architecture
   3.1. Architecture à 3 tiers ou 3 niveaux
   3.2. Les avantages de l'architecture à 3 tiers
   3.3. Les modèles d'architectures web
4. Conclusion

Chapitre2 : Les modèles d'architecture web et les technologies web utilisées

1. Introduction
2. Modèles d'architecture d'une application web
3. Le modèle client léger
4. Le modèle client dynamique (Objets JavaScript, Applets Java, Beans, ActiveX/COM)
5. Le modèle des objets distribués ((RMI, CORBA, DCOM et XML)
6. Conclusion

Chapitre3 : Construction des applications web

1. Introduction
2. Construction des applications web
   2.1. La gestion de projet
   2.2. Le rassemblement des exigences
   2.3. L'analyse
   2.4. La conception
   2.5. L'implémentation
   2.6. Le test
   2.7. Le déploiement
   2.8. La configuration et la gestion de la modification
   2.9. Les risques
3. Définition de l'architecture d'une application web
4. Conclusion

Chapitre4 : Les phases du processus

Partie 1 : Spécifications et cas d'utilisation

1. Introduction
2. Les exigences
3. La collecte des exigences
4. Les cas d'utilisations
5. Le paquetage
6. Le diagramme de séquence
7. L'analyse des cas d'utilisation
8. Conclusion

Partie 2 : L'analyse

1. Introduction
2. L'analyse du système
3. L'itération
4. Le paquetage
5. Définir le modèle de haut niveau
6. Le diagramme de classes dans le modèle d'analyse
7. Le diagramme de séquence
8. Le diagramme de collaboration
9. Le diagramme d'activité
10. Conclusion

Partie 3 : La conception

1. Introduction
2. L'étape de conception
3. Les diagrammes de la conception
4. Les activités de la conception
5. L'extension de l'UML pour les applications web
6. La conception des applications web
7. Les diagrammes de séquence de la conception
8. Conclusion

Chapitre 01

Introduction aux applications web

1. Introduction

De par leur complexité croissante, les applications Web sont des candidates idéales à la modélisation graphique et à une méthodologie de développement [1]

2. Vue d'ensemble des technologies Web, Définition d'une application Web

Cette partie est consacrée à la représentation du domaine des applications web

2.1. Définitions

Quelques définitions nécessaires

A. Serveur [2]

Un ordinateur détenant des ressources particulières et qu'il met à la disposition d'autres ordinateurs par l'intermédiaire d'un réseau. On parle de l'architecture client-serveur

Il existe plusieurs types de serveurs : serveur d'applications, serveur web ; serveur de fichiers, serveur de bases de données...

B. Serveur web [2]

C'est un programme situé sur une machine connectée au réseau interne. Dédié au protocole HTTP, il attend des requêtes client (navigateur web). Il traite les requêtes et retourne une page HTML en réponse.

C. Application web [2]

Une application web est une extension dynamique d'un serveur web. Une application Web est une application délivrée par un serveur web sur un réseau. Elle repose sur une architecture client/serveur à plusieurs niveaux.

Une application est formée d'un ensemble de composants web, de ressources statiques (images, sons,…) et de bibliothèques et de classes utilitaires. Les composants web fournissent cette capacité d'extension (exemples : les servlets, les pages JSP). Les composants web sont supportés par un container web qui fournit des services tels que la sécurité et la concurrence

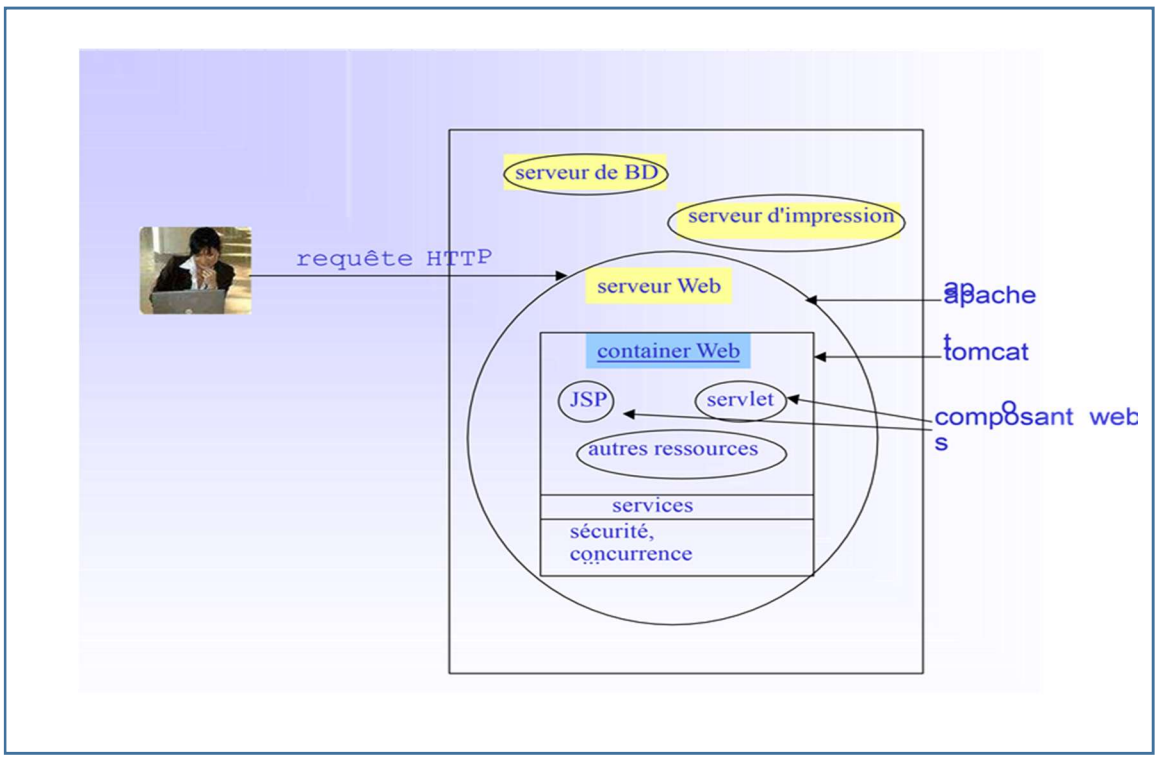

Figure 1.1 : Les composants d'une application web

- L'application est placée sur un serveur et les utilisateurs y accèdent par un simple navigateur
- Il n'est plus nécessaire d'installer un logiciel sur chaque poste, ce qui diminue en grande partie les frais de maintenance et élimine certaines incompatibilités ;
- Une application bien conçue peut être utilisée par différents types de terminaux (ordinateurs, laptops, tablettes, smartphones)

2.2. Les avantages des applications web

Les applications web permettent plusieurs avantages, parmi lesquels on cite :

- Il n'existe plus de contrainte géographique et on peut envisager un accès pour les utilisateurs nomades en toute sécurité grâce au cryptage des données
- Des centaines de personnes réparties dans des lieux différents travaillent simultanément sur ce type d'applications car elles sont conçues pour supporter une grande charge qui est souvent répartie sur plusieurs serveurs ;
- Certaines parties de ces applications sont ouvertes aux clients et fournisseurs et facilitent ainsi les échanges d'informations entre les partenaires (B2B)
- Ces applications sont capables de gérer des données multimédia (textes, photos, vidéos) et les utilisateurs sont à l'aise car habitués à utiliser un navigateur dans le cadre privé ;
- Pour mettre à jour l'application, il suffit de modifier l'application sur le serveur et tous les postes accèdent instantanément à la nouvelle version.

2.3. Types d'applications web

La classification des applications web peut se faire en fonction de la façon dont le contenu de l'application est présenté. On se basant sur ce critère 6 types d'applications web sont distingués :

2.3.1. Les applications web statiques

Ce type d'application contient peu d'informations et, en général, son contenu n'évolue pas ou très peu. Le développement de ce type d'applications web se fait habituellement en HTML et CSS. Il peut, néanmoins y avoir des objets animés tels que bannières, GIF, vidéos, etc. Ces applications peuvent être développées avec jQuery et Ajax.

La modification du contenu des applications statiques n'est pas facile. Pour ce faire, vous devez télécharger le code HTML, l'éditer, puis l'uploader de nouveau sur le serveur.

Deux exemples typiques des applications web statiques sont le portfolio professionnel et le curriculum vitae numérique.

2.3.2. Les applications web dynamiques

Les applications web dynamiques sont plus complexes sur le plan technique. Elles utilisent des bases de données pour charger des informations. Le contenu de ces applications mis à jour à chaque fois que l'utilisateur se connecte à l'application.

Il existe de nombreux langages de programmation pour le développement d'applications web dynamiques. PHP et ASP sont les plus répandus, car ils facilitent l'organisation du contenu.

L'actualisation d'une application dynamique est très simple, et il n'est même pas nécessaire d'entrer dans le serveur pour faire des modifications.

2.3.3. Les applications web de type e-commerce

Le développement d'une application web de type e-shop (commerce numérique) est plus complexe, car elle doit permettre les paiements électroniques par carte de crédit, PayPal ou autre mode de paiement.

Le développeur doit également créer un panel de gestion pour l'administrateur afin que ce dernier puisse mettre en vente des produits, faire des mises à jour et gérer les commandes.

2.3.4. Les applications web de type portail

Il s'agit d'une application dont la page d'accueil permet d'accéder aux différentes sections ou catégories. Son contenu peut être très varié : forums, chats, e-mail, moteurs de recherche, formulaire d'enregistrement, contenu le plus récent, etc.

2.3.5. Les applications web animées

La technologie FLASH est indispensable pour le développement d'applications web animées. Elle sert à créer le contenu avec des effets d'animation.

Permettant un design plus créatif et moderne, FLASH est l'une des technologies les plus utilisées par les designers.

L'inconvénient des applications web animées est le risque d'un référencement faible, car la technologie utilisée empêche les moteurs de recherche de lire correctement les informations.

2.3.6. Les applications de type « contenant manager»

Les applications web dont le contenu doit être souvent mis à jour nécessitent l'installation d'un système de gestion de contenu (Content Management System, CMS) à travers lequel l'administrateur aura la possibilité d'apporter des modifications.

Ces systèmes de gestion sont intuitifs et très faciles à gérer. Les CMS les plus connus sont : Wordpress, Joomla, Drupal

Considérations sur le développement d'applications web

Chaque type d'application web présente des avantages et des inconvénients, mais il faut garder à l'esprit qu'il s'agit d'applications web et non d'applications natives.

Il faut respecter les règles de gestion de cookies et s'assurer de la sécurité pour empêcher les attaques des pirates, comme pour un site web.

2.4. Les applications web Vs les applications natives

Les différences entre une application Web et une application native classique relèvent principalement de l'ordre technique, à savoir la programmation.

2.4.1. Les applications natives

Les applications natives sont programmées pour une plateforme spécifique. Cette plateforme peut être un système d'exploitation comme Android ou iOS ou encore un système d'exploitation comme Windows pour appareils de bureau.

Si des développeurs souhaitent couvrir plusieurs systèmes d'exploitation, ils doivent programmer une application pour chacune de ces plateformes. Cela nécessite donc un investissement bien plus important qu'avec une application Web.

Une application native utilise la mémoire de l'appareil utilisé.

Concernant la sécurité : avec une application native, les failles existantes peuvent seulement être comblées en mettant à jour l'application ou en téléchargeant sa dernière version.

2.4.2. Les applications web

Les applications Web sont moins adaptées aux supports sur lesquels elles se trouvent, mais elles fonctionnent correctement sur tous les matériels et appareils possibles.

Une seule application suffit pour recouvrir toutes les plateformes.

Concernant la sécurité, un navigateur Web comporte des mises à jour automatiques directes pour ses applications, afin que ses utilisateurs n'accèdent qu'à la version la plus sûre. Les applications Web sont donc plus avantageuses et plus rapides à produire.

2.5. Application Web Vs Site Web

2.5.1. Un site web [2]

Un site web est ce que l'on retrouve lorsqu'on entre une adresse et qu'on atterrit sur une page. Le site correspond à cette page et toute page reliée qui est gérée par la même entité.

Le rôle principal d'un site web est de fournir et présenter de l'information aux visiteurs

Un site de nouvelles ou un site d'information sur un produit ou une compagnie sont de bons exemples de sites web.

2.5.2. Une application web [2]

Une application web est tout site web qui permet à ses utilisateurs d'accomplir des tâches spécifiques.

Une application Web s'appuie sur ou étend un système Web pour ajouter des fonctionnalités métier.

Une application gère donc généralement des utilisateurs et toutes sortes de données selon les requis spécifiques au projet.

2.5.3. Application Web Vs Site Web

- Un site web n'est rien d'autre qu'une interface graphique.
- Une application Web est un programme dont l'interface graphique est un site web au travers d'un navigateur. Exemple : Facebook est une application web. Son interface graphique est accessible via le site web https://www.facebook.com. Donc, une application Web utilise un site Web comme frontal pour une application plus typique.

2.6. Principes de base de l'application Web

Les applications Web ont évolué à partir de sites Web ou de systèmes Web. Les premiers sites Web ont constitué un système hypermédia distribué permettant aux chercheurs d'accéder directement depuis leur ordinateur à des documents et à des informations publiés par d'autres chercheurs.

Les documents ont été consultés et visualisés avec un logiciel appelé un navigateur, une application logicielle qui s'exécute sur un ordinateur client. Le navigateur permet à l'utilisateur de demander des documents Web à partir d'autres ordinateurs sur le réseau et rendre les documents sur l'affichage de l'utilisateur.

Pour afficher un document, l'utilisateur doit démarrer le navigateur et entrer le nom du document et le nom de l'ordinateur hôte. La demande est gérée par une application logicielle appelée un serveur Web.

Le serveur Web reçoit la demande, localise le document sur son système de fichiers local et renvoie le document au navigateur

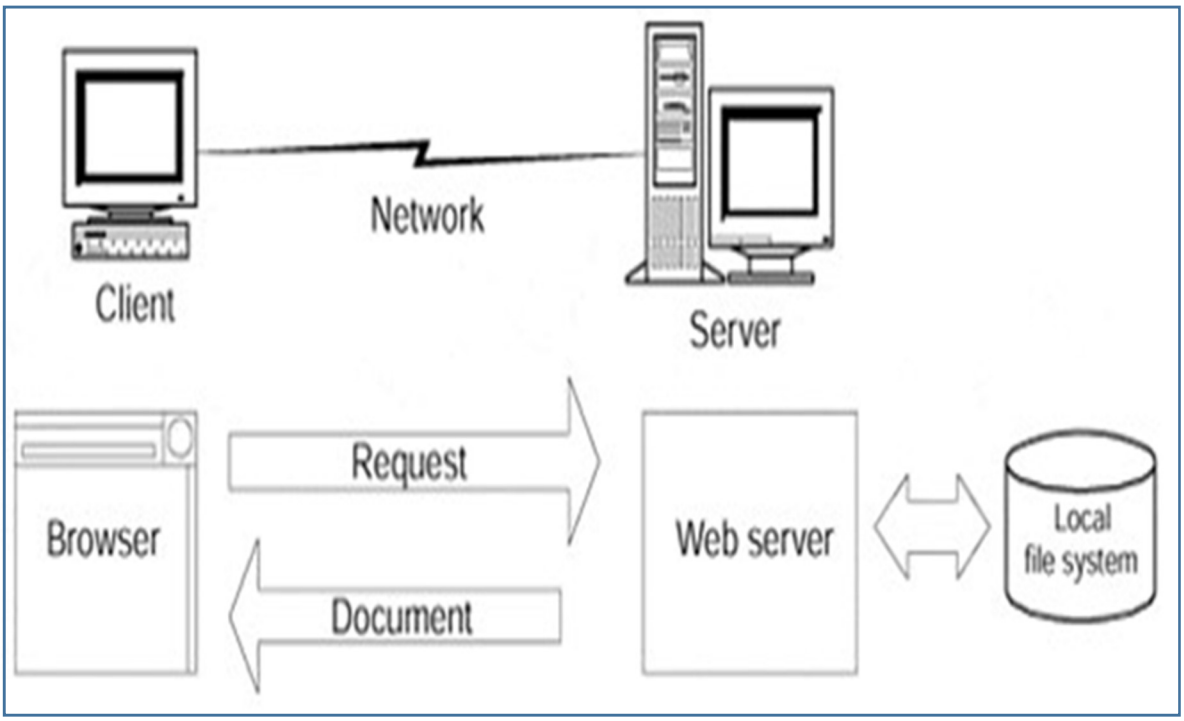

Figure 1.2 : Principes de base d'une application web

3. Définition de l'architecture des applications web

3.1. Architecture à 3 tiers ou 3 niveaux :

Elle se compose de trois couches :

A. **Couche de présentation** : représente la partie de l'application responsable des données et de l'interaction avec les utilisateurs (application HTML exploitée avec un navigateur web)

B. **Couche métier** : reçoit les requêtes utilisateur. Le serveur d'application fournit les traitements métiers, c'est là qu'est implémentée la logique du système et ses règles de gestion. Cette couche protège les données de l'accès direct des clients

C. **Couche d'accès aux données** : est responsable de la gestion des données. Elle permet de rendre l'accès aux données transparent quelle que soit La méthode utilisée pour les stockées.

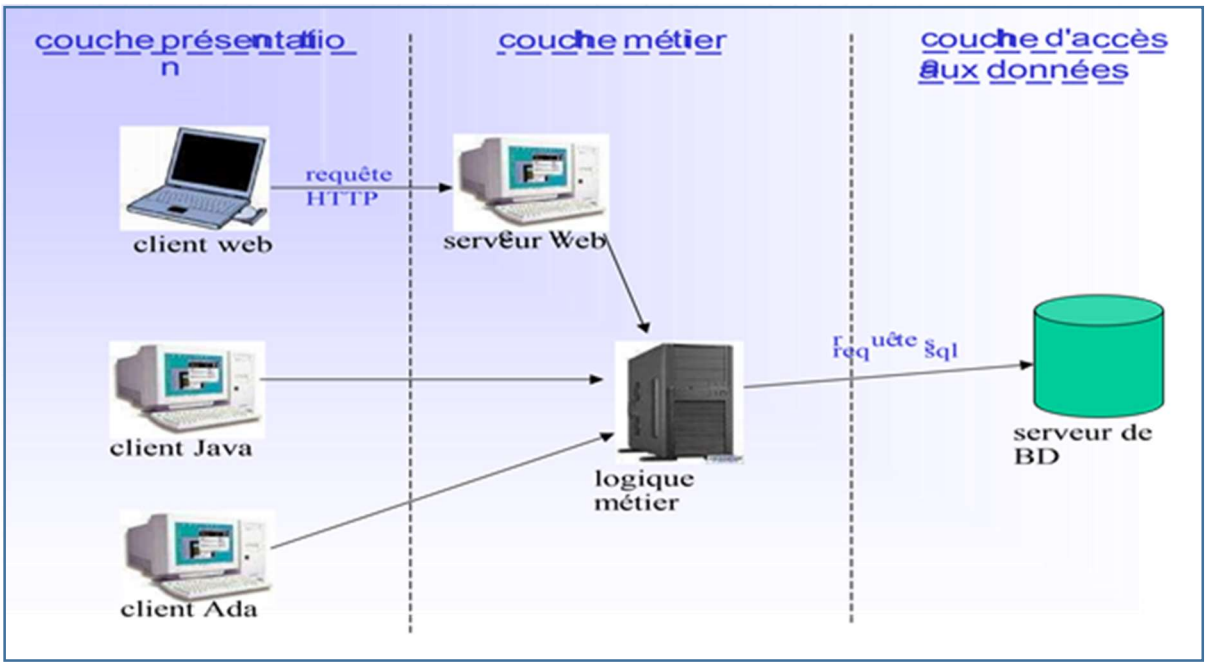

Figure 1.3 : L'architecture à trois tiers

3.2. Les avantages de l'architecture à 3 niveaux :

On cite les avantages suivants :

- La séparation entre les trois couches permet un développement rapide de l'application web par réutilisation des composants métiers prédéfinis
- La protection et la sécurité des données est plus facile à obtenir
- Le changement d'implémentation des composants est possible sans la réécriture de l'ensemble de l'application

3.3. Les modèles d'architectures web

Il existe trois modèles :

1. Le modèle client web léger (thin web client)
2. Le modèle client dynamique (thick web client)
3. Le modèle web services ou objets distribués (Web delivery)

4. Conclusion

- Ce chapitre donne une vue générale concernant le domaine des applications web (terminologie, principes et avantages)
- Le chapitre prochain est permet de faire une étude détaillée des modèles d'architecture d'une application web et des technologies utilisées par ces modèles.

Chapitre 02

Les modèles d’architecture web et les technologies web utilisées

1. Introduction

Ce chapitre représente une étude détaillée des différents modèles d’architecture d’une application web

2. Modèles d'architecture d'une application Web

Fondamentalement, une application Web étend un site Web en permettant à son utilisateur d'invoquer une logique métier et de modifier ensuite l'état de l'entreprise sur le serveur. Cette définition d'une application Web implique au minimum trois composants architecturaux significatifs pour une application Web :

- le navigateur client
- Le serveur Web
- Le serveur d'applications.
- Un serveur de base de données.

Généralement, on considère une application Web comme un logiciel client / serveur qui possède, au minimum, les composants architecturaux suivants :

1) un navigateur HTML / XML sur un ou plusieurs clients communiquant avec un serveur Web via le protocole http
2) un serveur d'application qui gère la logique métier.

Les trois modèles d'architecture les plus courants sont les suivants :

2.1 Le modèle client web léger (Thin web client)

2.1.1. Applicabilité

Ce modèle est le plus approprié pour les applications Web basées sur Internet ou pour les environnements dans lesquels le client a une puissance de calcul minimale ou n'a aucun contrôle sur sa configuration.

2.1.2. Utilisations connues

La plupart des applications Internet de commerce électronique utilisent ce modèle pour les visiteurs, car il n'est pas judicieux d'éliminer un secteur de clients simplement parce qu'il ne dispose pas de suffisamment de capacités client.

2.1.3. Structure

Les principaux composants de ce modèle d'architecture existent sur le serveur. Cette architecture représente l'architecture minimale de l'application Web.

Les principaux composants sont :

A. Navigateur client [1]

L'utilisateur de l'application utilise le navigateur pour demander des pages Web, HTML ou serveur.

La page renvoyée contient une interface utilisateur entièrement formatée (contrôles de texte et de saisie) qui est rendue par le navigateur sur l'affichage du client.

Toutes les interactions de l'utilisateur avec le système s'effectuent via le navigateur.

B. Serveur Web

Les navigateurs clients de l'architecture client web léger accèdent au système uniquement via le serveur Web, qui accepte les demandes de pages Web, qu'il s'agisse de pages HTML statiques ou de pages de serveur.

Selon la requête, le serveur Web peut lancer un traitement côté serveur.

C. Connexion http

Chaque fois que le client ou le serveur envoie des informations à l'autre, une nouvelle connexion distincte est établie entre les deux.

Une variante de la connexion HTTP est une connexion HTTP sécurisée via SSL (Secure Sockets Layer).

Ce type de connexion chiffre les informations transmises entre le client et le serveur à l'aide de la technologie de clé de chiffrement publique / privée.

D. Page HTML [1]

C'est une page Web avec une interface utilisateur et des informations de contenu qui ne sont pas traitées par le serveur.

Lorsqu'un serveur Web reçoit une demande de page HTML, le serveur récupère simplement le fichier et l'envoie sans filtrage vers le client demandeur.

E. Page de serveur [2]

Ce sont des pages Web qui subissent une forme de traitement côté serveur.

Ces pages sont implémentées sur le serveur en tant que pages scriptées (pages Active Server, pages Java Server, pages Cold Fusion).

Ces pages ont potentiellement accès à toutes les ressources côté serveur, y compris les composants de logique métier, les bases de données…

F. Base de données

Les applications Web utilisent une base de données pour rendre les données métier persistantes.

La base de données peut également être utilisée pour stocker les pages elles-mêmes.

Cette utilisation d'une base de données, représente un modèle architectural différent.

Ce modèle architectural étend le modèle client web léger à l'aide de scripts et d'objets personnalisés côté client, tels que les contrôles ActiveX et les applets Java.

2.2. Le modèle client web dynamique (Thick Web Client)

Le modèle client web dynamique (Thick Web Client) tire son nom du fait que le client peut exécuter une partie de la logique métier du système.

2.2.1. Applicabilité

Le modèle client web dynamique est utilisé pour améliorer l'interface utilisateur et exécuter la logique métier.

Dans certaines situations, la logique métier peut être exécutée sur le client seul. Dans ces situations, toutes les données requises pour effectuer le processus doivent être disponibles sur le client. La logique peut être aussi simple que la validation des données entrées.

2.2.2. Utilisations connues

L'utilisation la plus évidente des scripts côté client, des applets, des contrôles et des plug-ins est sur Internet sous la forme d'interfaces utilisateur améliorées.

Exemple : hors Internet, une société de logiciels de soins de santé a développé une application intranet basée sur le Web pour gérer les dossiers et la facturation des patients. L'interface utilisateur basée sur le Web fait un usage intensif des scripts côté client pour effectuer des validations de données

2.2.3. Structure

Les communications entre le client et le serveur sont effectuées avec HTTP.

Les scripts côté client, les contrôles ActiveX et les applets Java sont limités à l'interaction uniquement avec les objets sur le client.

Le modèle client web dynamique est une extension du modèle client web léger, la plupart des éléments significatifs sur le plan architectural sont identiques.

Ce modèle introduit quelques éléments supplémentaires, comme suit :

A. **Script client**
   JavaScript ou VB Script incorporé dans des pages au format HTML.
   Le navigateur interprète le script.
   Le W3C a défini l'interface HTML et Document Object Model que le navigateur propose aux scripts clients
B. **Document XML [2]**
   Un document formaté avec le langage XML (Extensible Markup Language).
   Les documents XML représentent le contenu (données) sans mise en forme de l'interface utilisateur.
C. **Contrôle ActiveX [1]**
   Un objet COM pouvant être référencé dans un script client et "téléchargé" sur le client, si nécessaire.
   Il dispose d'un accès complet aux ressources client.
   Le principal mécanisme de sécurité pour protéger les machines clientes est l'authentification
D. **Applet Java [1]**
   Un composant autonome et compilé qui s'exécute dans le contexte d'un navigateur.
   Pour des raisons de sécurité, l'applet a un accès limité aux ressources côté client.
   Les applets Java sont utilisés à la fois comme éléments d'interface utilisateur sophistiqués et pour analyser des documents XML

E. **JavaBean [1]**

Un petit composant Java à usage unique qui implémente un certain ensemble d'interfaces, lui permettant d'être facilement intégré dans des systèmes plus grands et plus complexes.

ActiveX est l'analogue du JavaBean dans les architectures centrées sur Microsoft.

2.3. Le modèle des objets distribués (services web)

Ce modèle d'architecture est ainsi nommé car le Web est principalement utilisé comme mécanisme de distribution pour un système client / serveur d'objets distribué traditionnel.

Ce type d'application est une application client / serveur distribuée qui inclut simplement un serveur Web et un navigateur client en tant qu'éléments architecturaux importants.

2.3.1. Application

Ce modèle architectural est le plus approprié lorsqu'il existe un contrôle important sur les configurations client et réseau.

La plus grande force de cette architecture est sa capacité à tirer parti des objets métier existants dans le contexte d'une application Web.

Le client peut être exploité pour exécuter une logique métier significative dans une plus grande mesure.

2.3.2. Utilisation connue

L'un des sites de nouvelles les plus fréquentés sur le Net est le site Web CNN Interactive. La plupart de son accès public est fait avec les navigateurs conventionnels et HTML droit 3.2 ; Cependant, derrière le site Web se trouve un réseau sophistiqué basé sur CORBA de navigateurs, de serveurs et d'objets distribués.

2.3.3. Structure

La différence la plus significative entre le modèle diffusion web et les autres modèles d'architecture d'application Web est la méthode de communication entre le client et le serveur.

Les éléments significatifs sur le plan architectural dans le modèle de diffusion Web incluent tous ceux spécifiés dans le modèle client web léger, ainsi que les suivants :

A. **DCOM [1]**

Distributed COM est le protocole d'objet distribué de Microsoft. Il permet aux objets sur une machine d'interagir avec et d'appeler des méthodes sur des objets sur une autre machine.

B. **IIOP [1]**

Le protocole Internet Inter-ORB est le protocole CORBA d'OMG pour interagir avec des objets distribués sur Internet ou sur tout réseau basé sur TCP / IP.

C. RMI (JRMP) [1]

L'invocation de méthode distante est la manière Java d'interagir avec des objets sur d'autres machines. JRMP (Java Remote Method Protocol) est le protocole natif pour RMI mais pas nécessairement le seul protocole qui peut être utilisé. Le RMI peut être implémenté avec l'IIOP de CORBA.

3. Conclusion

Durant ce chapitre on a décrit les différents modèles d'architecture d'une application web et les outils utilisés par chaque modèle.

Le choix d'un modèle d'architecture pour la construction d'une application web est très important et il est lié à plusieurs critères.

Chapitre 03

Construction des applications web

1. Introduction

Ce cours permet de décrire la procédure du développement d'une application web et de définir l'architecture de cette application.

2. Construction des applications web

Les étapes suivantes permettent le développement d'une application web :

2.1. La gestion du projet

La gestion globale de l'application et inclut les responsabilités typiques de la gestion des personnes et des budgets, ainsi que le rôle de porte-parole externe du projet.

Le flux de travail de gestion de projet contribue également aux artefacts du projet à travers

- Planification du projet
- Planification d'itération
- Gestion des risques
- Suivi des progrès

Le principal responsable du flux de travail de gestion de projet est le chef de projet, qui est responsable de l'exécution correcte du processus.

Le chef de projet est directement responsable de plusieurs artefacts clés du projet :

2.1.1. Plan de gestion de la configuration et du changement

Ce plan décrit comment les demandes de modification sont effectuées et gérées et définit les outils et les processus de gestion de la configuration, notamment le contrôle des versions d'artefact, les rapports et la collecte de mesures.

2.1.2. Glossaire de projet

Les métriques de projet, telles que les heures facturables, l'utilisation des ressources et les découvertes techniques, doivent être évaluées par rapport au plan d'itération.

2.1.3. Vision et analyse de rentabilisation

Le document de vision, lu et utilisé par presque tous les membres de l'équipe, est la base de la production de l'analyse de rentabilisation, une évaluation économique de la réalisation de la vision.

2.1.4. Plan de projet

Le plan de projet contient les étapes majeures et mineures : les dates de début et de livraison.

Le plan contient une première estimation des itérations et de leurs dates et objectifs. C'est une vue large et radicale de tout le cycle de vie du projet.

2.1.5. Plans d'itération

La planification des itérations comprend la dotation en personnel et les objectifs de chaque équipe, ainsi que les artefacts qui devraient être terminés et quand.

À tout moment, il y a toujours «deux» plans d'itération : le plan actif et le plan de l'itération suivante.

Le plan actif, l'itération en cours sur laquelle travaille l'équipe, est référencée par les membres de l'équipe et mesuré par le chef de projet.

Le plan d'itération suivant est modifié et amélioré avec les informations obtenues à partir de l'évaluation des résultats du présent et des itérations précédentes.

2.1.6. Évaluations d'itération

Le plan d'itération suivant est modifié et amélioré avec les informations obtenues à partir de l'évaluation des résultats du présent et des itérations précédentes.

L'objectif est d'affiner et de recentrer continuellement la direction du progrès vers la satisfaction ultime des exigences.

2.2. Le rassemblement des exigences

Une exigence est une déclaration de ce que le système devrait faire. La collecte de toutes les exigences du système est la spécification des exigences.

L'objectif général des exigences est d'exprimer sans ambiguïté ce que le système proposé devrait faire.

Un élément clé de l'expression des exigences de manière à ce que toutes les parties concernées puissent les comprendre est les cas d'utilisation.

Les cas d'utilisation expriment des scénarios d'utilisation du système dans la langue du domaine.

2.3. L'analyse

L'analyse est le processus d'examen des exigences et de construction d'un modèle conceptuel du système à construire.

L'analyse est souvent mentionnée dans le même souffle que la conception, car les mêmes personnes contribuent à ces flux de travail, mais les activités et les motivations sont nettement différentes.

Les artefacts d'analyse incluent des classes et des collaborations détaillées, des diagrammes de séquence, des diagrammes d'états et des diagrammes d'activités.

Ce sont les mêmes artefacts utilisés et élaborés pendant la conception, la principale différence est que l'architecture n'est pas encore appliquée. Les artefacts d'analyse représentent le système extrait de l'architecture.

2.4. La conception

L'objectif principal de la conception est de rendre le modèle d'analyse du système réalisable dans le logiciel.

La conception prend les artefacts produits pendant la conception et leur applique l'architecture.

Dans certaines situations, l'application d'une architecture affecte tellement le modèle que deux modèles conceptuellement séparés du système sont maintenus : le modèle d'analyse et le modèle de conception. Ces deux modèles ne sont que des vues différentes du même système.

2.5. L'implémentation

L'implémentation et la mise en œuvre d'un système logiciel va au-delà de l'écriture et de la compilation du code, même si cela représente une grande partie du flux de travail.

La mise en œuvre prend les artefacts du design et leur applique des outils de développement logiciel.

Les applications Web impliquent souvent un certain nombre de technologies qui doivent être gérées. Les langages de programmation et les compétences pour le développement côté client sont principalement HTML, JavaScript, Java, ActiveX et, éventuellement, certaines technologies d'objets distribués.
Les langages et les technologies sur le serveur ont une plus grande portée et impliquent des langages typiques de programmation d'objets et de troisième génération (C / C ++, Java, Smalltalk, Ada, Eiffel), ainsi que des technologies de composants telles que JavaBeans et COM.
Le côté serveur traite également les technologies traditionnelles de base de données et de surveillance de traitement des transactions (TPM).
Sur le serveur, une application Web est à peu près comme n'importe quel autre système client / serveur.

2.6. Le test

Le test se concentre sur l'évaluation des artefacts exécutables du système. Le test est distinct de l'assurance qualité (AQ), car le contrôle qualité affecte toutes les parties du système, tandis que le test est effectué uniquement sur les parties exécutables (ou presque exécutables) du système.
De nombreux tests différents du système sont réalisés. Chaque test tente de déterminer une qualité du système.
Un test de performance accède à la capacité du système à fonctionner rapidement et sous de lourdes charges.
Le test de charge met spécifiquement l'accent sur le système et établit le point de rupture du système ou seulement les courbes de performance sous charge.
Les tests fonctionnels déterminent si des fonctions spécifiques, telles que définies dans la spécification des exigences, ont été implémentées correctement. Les tests fonctionnels sont souvent dérivés directement des spécifications de cas d'utilisation.
En plus de tester les qualités spécifiques du système, il existe des tests pour certaines étapes du système :

- **Test unitaire :** test effectué par les exécutants sur les petites unités de système qu'ils ont développées. Souvent, ces tests sont effectués sur un composant singulier ou une petite collaboration de composants.
- **Test d'intégration :** test qui valide les interfaces individuelles des composants et leur capacité à travailler les uns avec les autres. De tels tests sont effectués lorsque certaines parties du système sont connectées ensemble.
- **Test du système :** le test pour vérifier que toutes les exigences ont été satisfaites. Ce test est effectué lorsque tous les composants du système sont assemblés, pour valider le système dans son ensemble.
- **Test d'acceptation** : test formel effectué par la communauté d'utilisateurs sur le système. Si le système réussit ce test, la communauté d'utilisateurs accepte le système et est prête à le déployer.
- **Les tests de régression** : sont très importants dans un processus de développement itératif. Le test de régression est le test d'un système qui a potentiellement changé.

Le test des applications Web est pratiquement effectué de la même manière que pour les autres systèmes. En fonction de la nature de l'application Web, des tests peuvent être nécessaires sur un certain nombre de plates-formes client et de configurations de navigateur.

Les applications Internet doivent prendre en compte tous les différents navigateurs pouvant être utilisés, même les plates-formes sur lesquelles ils s'exécutent.

2.7. Le déploiement

Déployer une application Web peut être très simple ou très compliqué. L'application intranet simple qui s'exécute sur un serveur et exploite un réseau existant peut être très facile à déployer. Seul le serveur doit être configuré.

Si l'application est conçue pour utiliser uniquement les fonctionnalités client les plus élémentaires, il n'y a plus rien à faire. Mais si l'application doit gérer les problèmes de sécurité et les charges lourdes sur Internet, une planification importante du déploiement est nécessaire.

Le traitement des problèmes de basculement et d'équilibrage de la charge tels que définis par l'architecture implique souvent l'utilisation de plusieurs composants tiers et prêts à l'emploi qui doivent être intégrés.

La plupart des grandes applications Internet ont des connexions Internet redondantes et des systèmes de sauvegarde hors site. Le déploiement dans ces types d'applications nécessite une planification et une gestion soignées

2.8. Configuration et gestion des modifications

La configuration et la gestion des modifications sont un flux de travail en soi, car elles jouent un rôle essentiel dans un processus itératif.

Le flux de travail gère les modifications afin qu'elles puissent être introduites et surveillées de manière contrôlée.

Les zones qui connaissent des changements fréquents et inattendus sont des indications de zones à haut risque et d'absence probable d'analyse initiale.

2.9. Les risques

L'un des objectifs importants de ce processus est de s'attaquer rapidement aux risques. Au lieu de laisser le risque apparaître incontrôlé, le processus cherche activement les zones à risque du système et les implémente en premier.

Le mécanisme pour cela est de laisser les cas d'utilisation conduire le processus. L'approche basée sur les cas d'utilisation permet de gérer et d'attaquer les risques en mettant l'accent sur le développement.

L'équipe de test utilise le plan d'itération et l'état actuel des cas d'utilisation et des exigences pour préparer des plans de test et des scripts afin d'évaluer chaque livraison d'itération

Le chef de projet évalue les résultats et les utilise pour apporter des ajustements aux calendriers de développement ou pour réévaluer l'ensemble minimal d'exigences nécessaires à la réussite du projet.

3. Définition de l'architecture de l'application web

À mesure que l'équipe des exigences rassemble les exigences et construit le modèle de cas d'utilisation, l'architecte ou l'équipe d'architecture examine le modèle de cas d'utilisation et explore les architectures système possibles.

3.1. Définition

L'architecture représente les vues de plus haut niveau des composants architecturaux significatifs dans le système. Un composant dans ce sens est une entité autonome avec une interface publique.

Les composants d'importance architecturale sont ceux qui apparaissent dans les vues les plus élevées du système.

3.2. Les activités de l'équipe d'architecture

Le modèle de cas d'utilisation, en tant que vue du comportement dynamique du système souhaité, est hiérarchisé en fonction du risque perçu.

Le processus consiste à traiter les cas d'utilisation présentant le plus de risques précoces et à éviter ainsi les «pièges» indésirables plus tard.

Un cas d'utilisation risqué peut impliquer l'utilisation d'une technologie ou nécessiter un haut niveau de performance.

Les trois modèles d'architecture les plus courants sont les suivants :

1) Le modèle client web léger
2) Le modèle client web dynamique
3) Le modèle des objets distribués (services web)

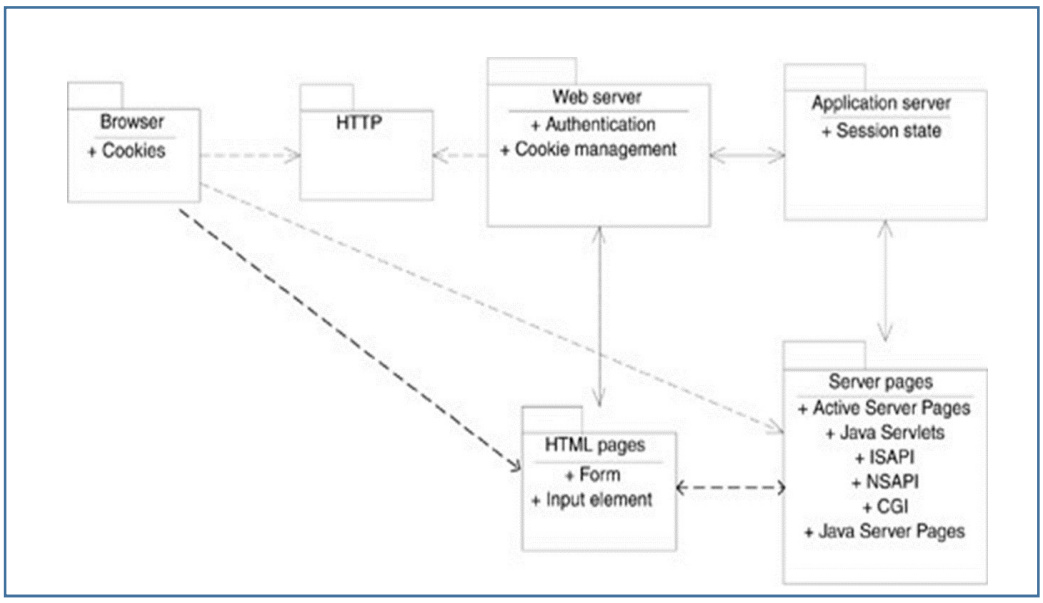

Figure 3.1 : Diagramme de la vue logique pour l'architecture client web léger

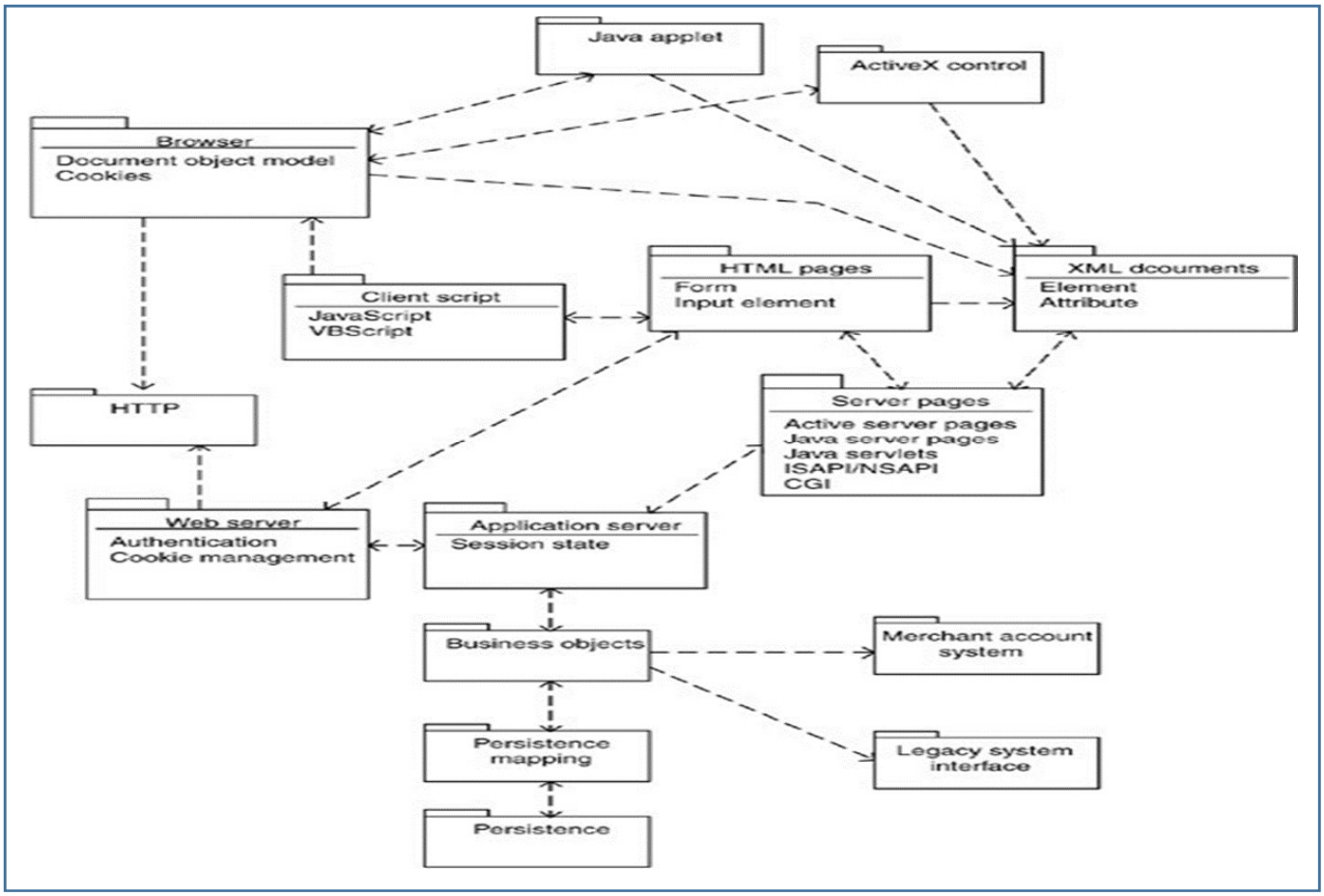

Figure 3.2 : Diagramme de la vue logique pour l'architecture client web dynamique

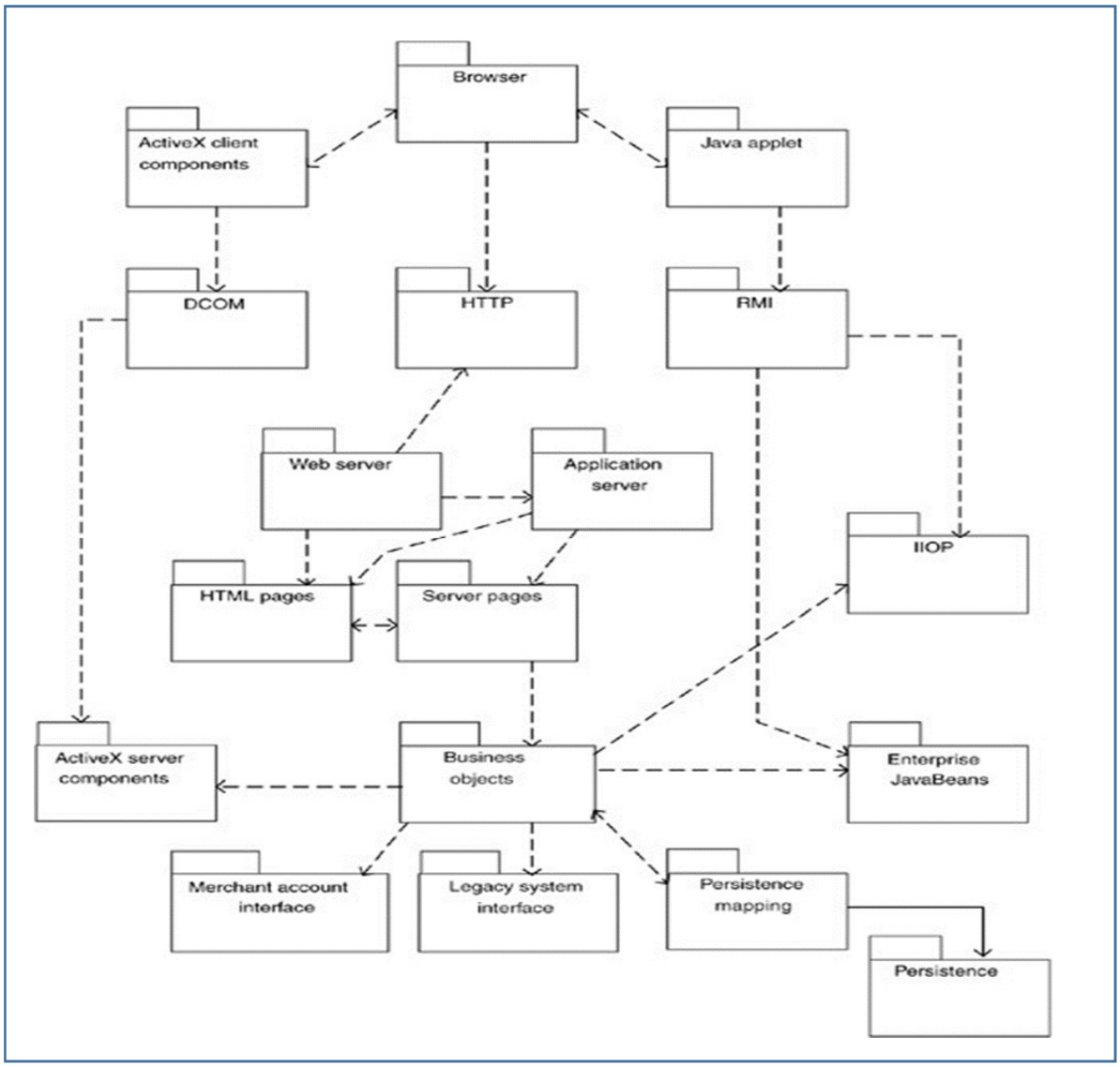

Figure 3.3 : Diagramme de la vue logique pour l'architecture objets distribués

4. Conclusion

Ce chapitre décrit les étapes de constructions d'une application Web

Le chapitre prochain est consacré à la description des différents phases de la modélisation de l'application web

Chapitre 04

Les phases du processus

Spécification des exigences et cas d'utilisation

1. Introduction

La spécification des exigences, la collection de documents et de modèles, tente à décrire sans ambiguïté le système logiciel à construire. La spécification des exigences doit être mise à la disposition de presque toutes les personnes liées au projet.

Une exigence ou une contrainte que le système doit observer est généralement exprimée sous la forme d'une instruction commençant par une phrase du type "Le système doit ..."

2. Les exigences

Le but d'une déclaration d'exigence est d'exprimer un comportement ou une propriété que le système devrait avoir. Une qualité très importante d'un bon énoncé des exigences est qu'il peut être vérifié par l'équipe de test lorsqu'elle est livrée.
Les exigences peuvent être catégorisées comme fonctionnelles ou non fonctionnelles.

2.1. Les exigences fonctionnelles

Les exigences fonctionnelles expriment une action que le système doit effectuer et définissent à la fois le stimulus et la réponse.
Les exigences fonctionnelles, le type d'exigence le plus commun, identifient les choses que le système peut faire, généralement en réponse à une entrée externe.

2.2. Les exigences non fonctionnelles

Les exigences non fonctionnelles sont regroupées en catégories pour les rendre plus faciles à comprendre et à suivre.

A. **Les exigences d'utilisabilité** se rapportent aux aspects généraux de l'interface entre l'utilisateur et le système. Dans le cas d'une application web, les exigences d'utilisabilité peuvent inclure la configuration de navigateur minimale à utiliser ou les éléments HTML à utiliser.
B. **Les exigences de performances** décrivent les performances d'exécution du système et sont généralement liées au temps. Une exigence courante pour les applications Web est de spécifier un temps de chargement maximal pour une page.
C. **Les exigences de robustesse** et fiabilité doivent indiquer clairement le degré de disponibilité de l'application. Ces exigences sont également concernées par les problèmes de sauvegarde et de stockage.
D. **Les exigences de sécurité** ont tendance à spécifier les niveaux d'accès au système et correspondent souvent aux rôles humains de l'entreprise. Les exigences de sécurité devraient également inclure l'accès au système par d'autres systèmes «externes», s'ils sont utilisés.
E. **Les exigences matérielles** indiquent souvent le matériel minimal requis pour implémenter le système.
F. **Les exigences de déploiement** décrivent comment l'application est livrée aux utilisateurs finaux. Il fournit des contraintes sur la façon dont le système doit être installé, maintenu et accessible par le personnel de maintenance.

3. La collecte des exigences

La collecte des exigences est généralement effectuée par des groupes. Une équipe d'exigences est composée d'un représentant de la communauté des utilisateurs ou des parties prenantes et d'un membre technique du personnel de développement.
Des compétences supplémentaires peuvent compléter l'équipe.
Chaque exigence spécifique du système doit avoir un identifiant unique utilisé pour la traçabilité.
Au cours de la conception, les éléments classes et les packages du modèle de conception sont vérifiés pour s'assurer qu'ils pointent au moins une exigence et qu'ils capturent vraiment l'esprit des exigences qu'ils pointent.

3.1. Directives

- Chaque exigence doit être claire et concise. Évitez les descriptions verbeuses qui peuvent être interprétées de différentes façons.
- Une déclaration d'exigence doit se concentrer sur un point. La granularité plus fine permet une meilleure traçabilité à travers le modèle.
- Chaque exigence doit être vérifiable.
- "L'interface utilisateur doit être intuitive" ne peut pas être testée objectivement et n'est donc pas une déclaration d'exigence vérifiable.

4. Les cas d'utilisation [3]

Les exigences fonctionnelles décrivent comment le système se comporte en réponse à l'entrée de l'utilisateur et du système externe. Ils ont tendance à être plus dynamiques et nécessitent souvent plus de détails afin de les comprendre clairement.

Les cas d'utilisation sont une technique puissante pour capturer et exprimer le comportement détaillé du système. Ils sont un moyen formel de capturer et d'exprimer l'interaction et le dialogue entre les utilisateurs du système (appelés acteurs) et le système lui-même.

Le terme acteur est utilisé pour représenter un rôle générique d'utilisateur.

Un cas d'utilisation peut contenir plusieurs scénarios. Cependant, il y a toujours un scénario principal. Les autres scénarios sont appelés scénarios alternatifs ou chemins alternatifs. Un cas d'utilisation décrit le comportement du système vu de l'extérieur. Tout le comportement est sous la forme de résultats observables.

Chaque cas d'utilisation est nommé et son nom reflète son objectif ou son but.

Les cas d'utilisation capturent d'autres informations clés :

- **Identifiant unique :** idéalement, un numéro automatisé ou un système de numérotation qui permettra à un nom de cas d'utilisation d'évoluer tout en maintenant la traçabilité tout au long du processus.
- **Énoncé de l'objectif :** Une déclaration simple qui résume l'objectif de l'ensemble du cas d'utilisation.
- **Auteurs :** noms des membres de l'équipe qui ont directement contribué au texte dans le cas d'utilisation.
- **Priorité :** priorité globale du cas d'utilisation. Cette valeur aidera les chefs de projet à déterminer combien d'efforts sont justifiables pour le cas d'utilisation.
- **Risque:** Une évaluation des risques qui identifie la probabilité relative que quelque chose puisse mal se passer pendant sa mise en œuvre, ou le niveau d'inexpérience de l'équipe de développement .
- **Hypothèses :** une description textuelle des éléments supposés, c'est-à-dire l'état du système.
- **Conditions préalables :** Une description textuelle des conditions qui doivent être remplies avant que ce cas d'utilisation puisse être effectué.
- **Post-conditions :** Une description textuelle des conditions qui doivent être remplies avant que ce cas d'utilisation puisse être complété.
- **Problèmes en suspens :** Une collection d'éléments qui doivent être résolus avant que le cas d'utilisation puisse être élaboré dans l'analyse et la conception.

- **Exigences satisfaites :** une liste d'identificateurs d'exigences que ce cas d'utilisation particulier est supposé satisfaire.

4.1. Le modèle de cas d'utilisation

Les relations entre les cas d'utilisation sont documentées dans un diagramme de cas d'utilisation. La collection complète de cas d'utilisation, d'acteurs et de diagrammes forme un modèle de cas d'utilisation.

Dans un diagramme, un cas d'utilisation est rendu par un ovale avec le nom de cas d'utilisation imprimé dans ou juste en dessous.

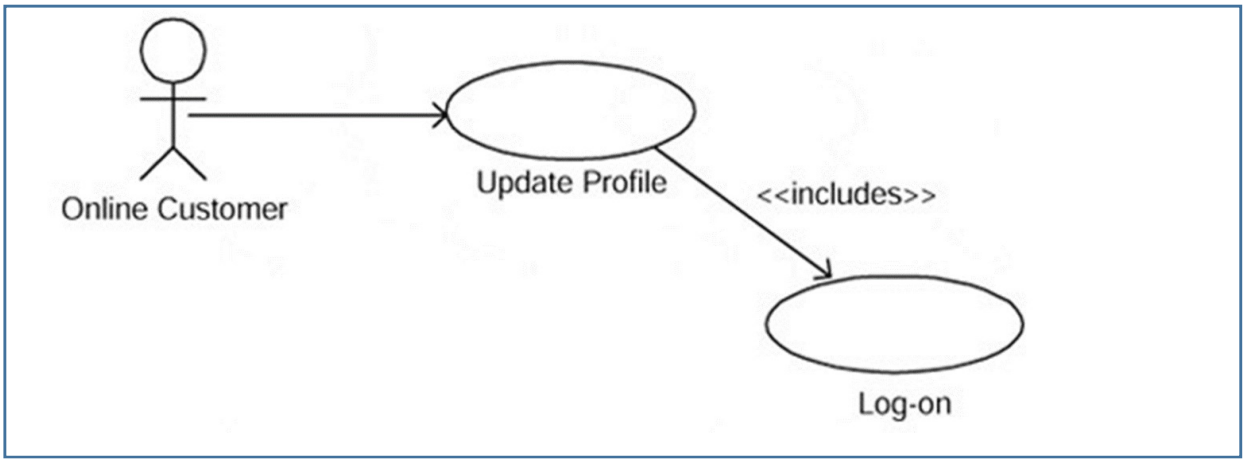

Figure 4.1 : diagramme du cas d'utilisation

5. Le paquetage (package) [3]

Un modèle de cas d'utilisation est souvent divisé en paquets. Chaque paquetage (souvent appelé package) possède un ensemble de cas d'utilisation ou même d'autres paquets.

Le package est un mécanisme UML permettant de décomposer un modèle en éléments plus maniables.

Dans un diagramme, un package est rendu sous la forme d'un dossier à onglets.

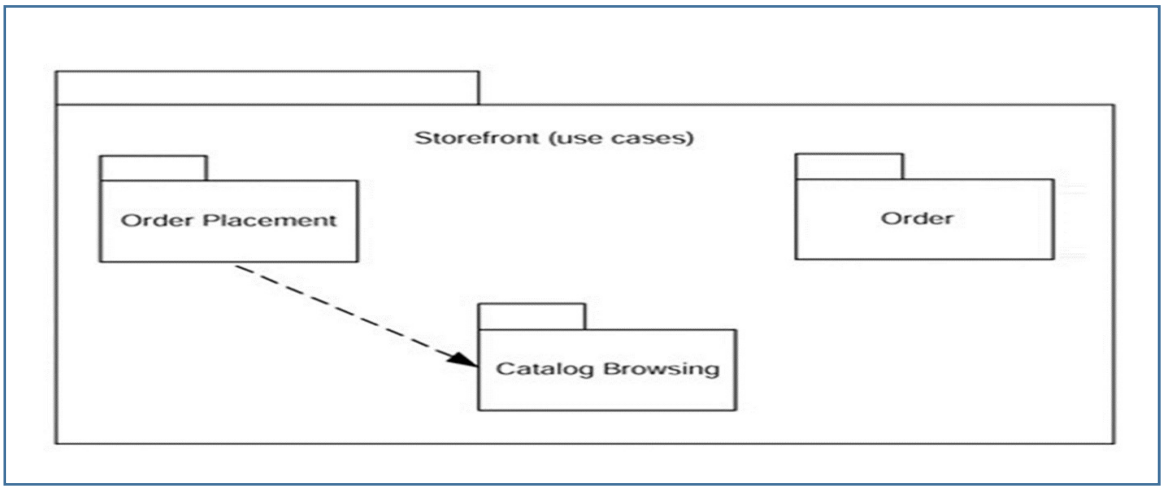

Figure 4.2 : Un package

6. Le diagramme de séquence

Un diagramme de séquence, un type spécifique de diagramme d'interaction, exprime l'interaction entre un acteur et le système, avec un accent particulier sur la ligne de temps.

Chaque scénario dans un cas d'utilisation doit être schématisé. Les diagrammes de séquence commencent par placer le texte du scénario dans le diagramme. Chaque étape du scénario doit être placée séparément dans le diagramme, en commençant par la première étape en haut et en continuant sur le côté gauche du diagramme.

L'axe vertical du diagramme représente la dimension temporelle ; le haut est le début du scénario, et le bas est la fin. Les deux principaux acteurs du scénario l'acteur et le système - sont placés dans le diagramme en haut.

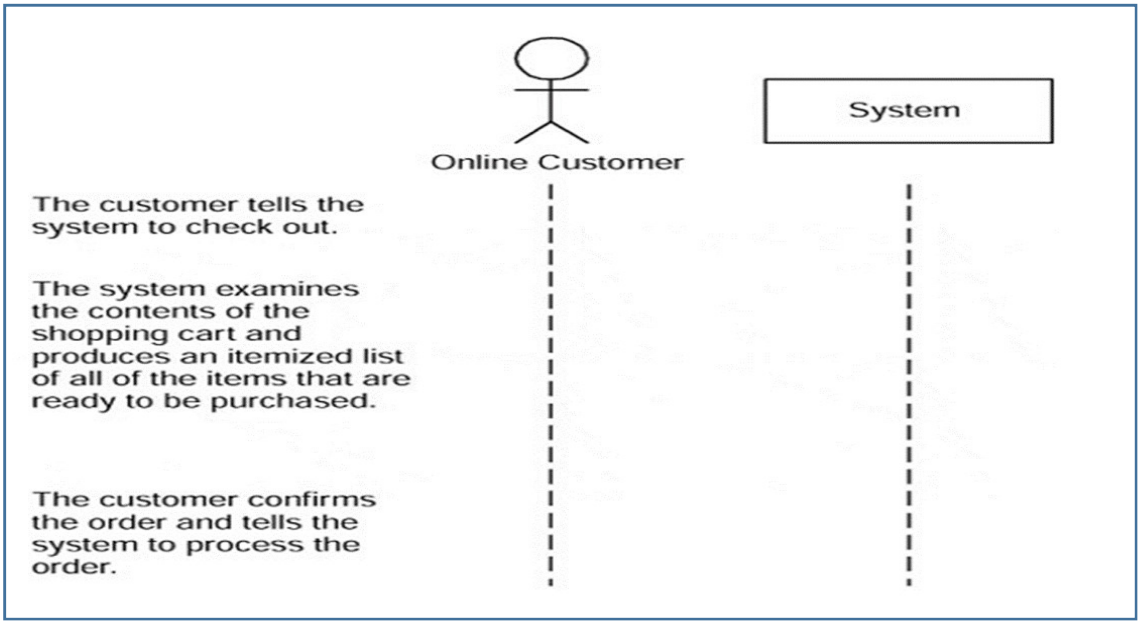

Figure 4.3 : diagramme de séquence

L'analyste complète le diagramme de séquence en ajoutant sur le diagramme des interactions entre l'acteur et le système. L'entrée du système, fournie par l'acteur, est indiquée en tirant des flèches entre la ligne de l'acteur et celle du système. Les réponses du système, telles que vues par l'acteur, sont indiquées par des flèches remontant à l'acteur.

Ces interactions sont essentiellement des messages envoyés d'un objet à un autre. Les messages sont classés séquentiellement de haut en bas.

En plus des messages envoyés d'un objet à un autre, un objet peut lui envoyer un message. Ce type d'action peut être considéré comme l'appel d'un sous-programme, ou la rupture d'une réaction complexe en plus petits.

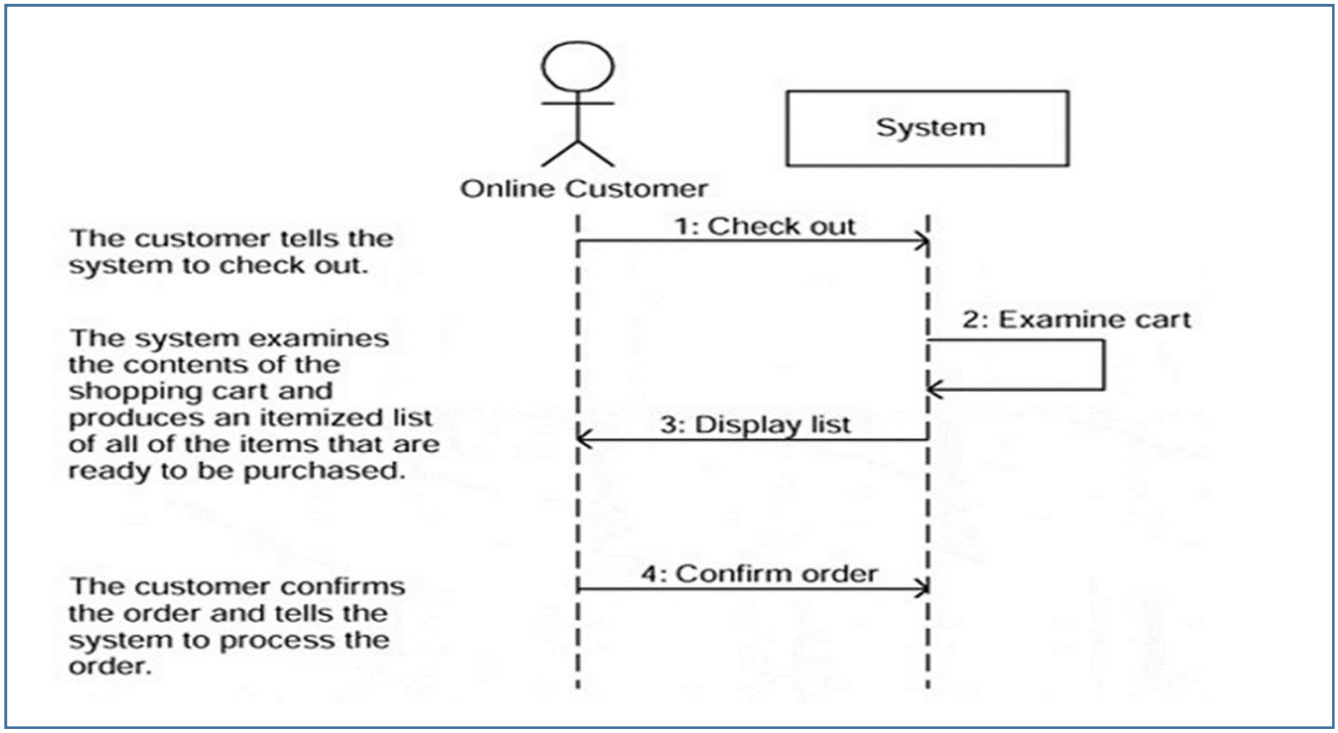

Figure 4.4 : diagramme de séquence

7. L'analyse des cas d'utilisation

L'analyse de cas d'utilisation est une autre activité qui est effectuée lorsque les cas d'utilisation sont presque terminés.

Il s'agit d'une analyse de robustesse et fait partie intégrante de ce processus. Les objectifs de l'analyse de cas d'utilisation sont les suivants :

- Identifier les classes et les objets qui vont exécuter le flux d'événements d'un cas d'utilisation
- Identifier les responsabilités, les attributs et les associations des classes.
- Notez l'utilisation des mécanismes architecturaux
- Une réalisation de cas d'utilisation est un cas d'utilisation stéréotypé qui réalise un cas d'utilisation normal.
- La relation stéréotypée << réalise >> lie un cas d'utilisation à la réalisation de son cas d'utilisation

La réalisation d'un cas d'utilisation distincte permet l'implémentation du cas d'utilisation de base par plusieurs systèmes.

Une réalisation de cas d'utilisation distincte fournit une couche d'indépendance par rapport aux exigences du système et de sa mise en œuvre tout en maintenant un lien dans la chaîne de traçabilité. L'analyse de cas d'utilisation commence par l'élaboration des diagrammes de séquence du cas d'utilisation de base avec des objets de niveau analyse.

- Les objets d'analyse sont des instances de classes d'analyse qui représentent les «choses» majeures du système qui ont une responsabilité et un comportement.
- Les classes d'analyse sont les premiers prototypes des classes de niveau conception.
- Les classes d'analyse peuvent être stéréotypées en trois types : Dialogue (boundary), entité (entity) et contrôle (control).

7.1. Les Dialogues

Les Dialogues (objets limités) représentent l'interface entre l'acteur et le système. Les instances de ces objets sont généralement des écrans d'entrée ou des commandes d'interface utilisateur spéciales. Dans les applications Web, elles peuvent représenter des pages Web entières.

7.2. Les objets d'entité

Les objets d'entité sont ceux qui sont décrits dans le cas d'utilisation mais qui dureront plus longtemps. Les commandes, les clients, les produits et la paie sont des objets d'entité dont les instances peuvent apparaître dans de nombreuses invocations de cas d'utilisation.

7.3. Les objets de contrôle

Les objets de contrôle représentent des processus. Ces objets représentent les activités du système qui peuvent souvent être nommées. Calculer la masse salariale, traiter et facturer, ré-cataloguer l'inventaire sont tous des processus suffisamment importants pour être nommés. Les objets de contrôle dirigent les activités de l'entité et des objets d'interface.

Une façon de commencer l'analyse de cas d'utilisation consiste à examiner le texte de cas d'utilisation des noms clés et des verbes. Les noms sont des candidats pour les objets d'entité ; les verbes, pour les contrôleurs.

8. Conclusion

Ce chapitre détaille les différentes étapes de la spécification des exigences et donne une vue générale sur l'analyse des cas d'utilisation. La phase d'analyse est bien détaillée dans la partie suivante de ce chapitre

Chapitre 04

Les phases du processus

L’analyse

1. Introduction

Les activités d'analyse et de conception aident à transformer les exigences du système en une conception qui peut être réalisée dans un logiciel. L'analyse et la conception peuvent être effectuées séparément ou combinées dans le cadre du même ensemble d'activités.

2. L’analyse du système

L'analyse comprend les activités qui prennent les cas d'utilisation et les exigences fonctionnelles pour produire un modèle d'analyse du système.

Le modèle d'analyse est composé de classes et de collaborations de classes qui présentent les comportements dynamiques détaillés dans les cas d'utilisation et les exigences.

Le modèle physique représente la structure du système proposé à un niveau d'abstraction au-delà de la mise en œuvre physique du système.

Les classes représentent généralement des objets, tels que le panier, la commande, l'élément de campagne ou le produit, dans le domaine métier (espace de problème).

Le niveau d'abstraction est tel que ces mêmes classes pourraient être appliquées indifféremment à des architectures autres que les applications web.

L'analyse se concentre sur les exigences fonctionnelles du système, ignorant pour l'instant les contraintes architecturales du système.

L'accent est mis sur le fait de s'assurer que toutes les exigences fonctionnelles, telles qu'exprimées par les cas d'utilisation et autres documents, sont réalisées quelque part dans le système.

A la fin de cette étape, chaque élément d'exigence et chaque cas d'utilisation est lié aux classes et aux packages qui les réalisent.

3. L’itération

Dans un processus de développement incrémentiel, le modèle de cas d'utilisation complet ou l'ensemble d'artefacts d'exigences n'ont pas besoin d'être complétés avant que des activités dans les phases ultérieures du processus de développement puissent avoir lieu.

Un autre aspect important du développement itératif est la possibilité d'aborder le risque dès le début.

Le risque est habituellement identifié par l'expérience - ou le manque d'expérience - des membres seniors de l'équipe. Souvent, les inconnues sont liées à des exigences non fonctionnelles, telles que les performances, la sécurité ou les interfaces système externes.

4. Le paquetage (package) [3]

L'une des premières activités de l'équipe d'analyse consiste à créer la hiérarchie des paquetages du modèle d'analyse. «Diviser pour mieux régner» est le principe qui s’applique très efficacement à la résolution d’un problème complexe.

Un paquetage n'est rien de plus qu'un "morceau" du modèle : assez petit pour que l'on puisse comprendre son but et sa signification dans le modèle. Les paquetages contiennent des éléments du modèle ; classes, diagrammes, composants, interfaces, etc.

Chaque élément du modèle appartient à un seul paquetage souvent dit package. Les éléments du modèle peuvent apparaître dans les diagrammes des autres packages ou

participer à des relations d'éléments dans d'autres packages grâce à La notion de classe publique ou privée dans d'un package.

- Une classe publique dans un package est visible et peut être utilisée par des éléments extérieures du package. D'une certaine manière, ces classes représentent l'interface publique du paquet et doivent être choisies avec soin.
- Les packages eux-mêmes peuvent être subdivisés en plusieurs packages ; par conséquent, il est possible qu'un modèle soit représenté par une hiérarchie de packages.
- Un package est rendu graphiquement sous la forme d'un dossier à onglets.
- Un paquetage à un nom unique dans tout le modèle. Chaque paquetage forme un espace de nommage, ce qui signifie que deux éléments peuvent avoir le même nom tant qu'ils appartiennent à deux packages différents.

Les packages peuvent avoir des relations les uns avec les autres. Les deux types de relations sont la dépendance et la généralisation.

- **Une relation de dépendance** signifie qu'un paquetage dépend de la structure des éléments d'un autre package. Cette relation est représentée avec une ligne pointillée et une flèche pointant vers le package dont l'autre dépend.
- **Une relation de généralisation** est comme la généralisation dans les classes ; les sous-packages représentent des spécialisations d'un paquet. Par exemple, un package d'interface utilisateur peut comporter deux sous-packages : l'interface utilisateur activée par ActiveX et l'interface utilisateur Java. Les deux contiennent des éléments qui prennent en charge l'objectif de fournir une interface utilisateur, mais chaque sous-package le fait avec une architecture différente.

5. Définir le modèle de haut niveau

Au cours des activités de définition de cas d'utilisation, le modèle de cas d'utilisation a été divisé en packages.

Pendant l'analyse, la même hiérarchie de packages pourrait être utilisée pour modéliser la vue structurelle du système.

La hiérarchie de la vue dynamique du système (cas d'utilisation) peut fournir un début initial, mais elle est insuffisante pour définir la vue structurelle du système (classes).

La raison est que certains objets participent probablement à un grand nombre de cas d'utilisation et de packages et ne peuvent logiquement pas être affectés à un seul package de cas d'utilisation.

Au niveau le plus élevé, les packages des deux modèles sont souvent les mêmes, mais, Aux niveaux inférieurs de la hiérarchie, la division des packages peut être fonctionnellement plus explicite.

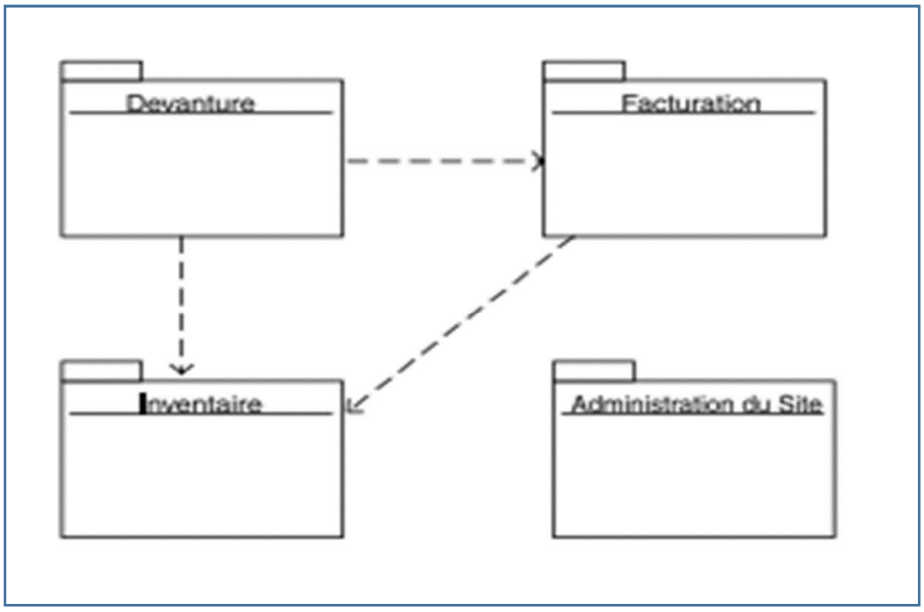

Figure 4.5 : Vue de haut niveau d'une application de e-commerce

Ce même diagramme pourrait également être utilisé pour le modèle d'analyse de niveau supérieur.

Aux niveaux inférieurs - par exemple, dans le package Devanture il peut y avoir des packages supplémentaires pour séparer les principales fonctions du système telles qu'elles sont disponibles pour l'utilisateur en ligne.

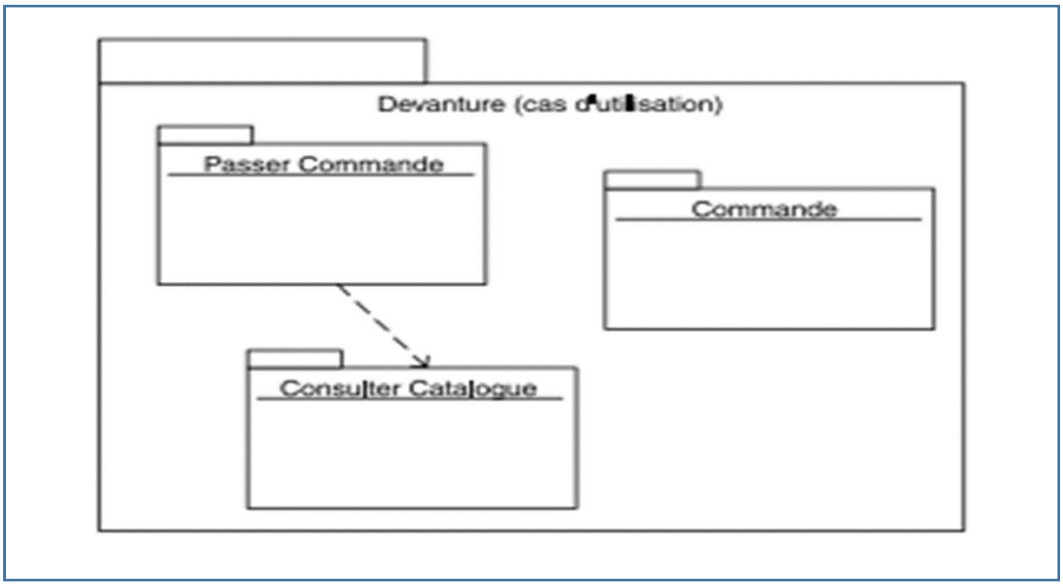

Figure 4.6 : Paquetage du cas d'utilisation Devanture

Le package Devanture pour le modèle d'analyse peut être très différent, avec les packages suivants : Catalogue, Panier, Profil client et Personnalisations de produit (décorations, couleur, la taille).

Dans le modèle d'analyse, les packages ont tendance à représenter les choses plutôt que les actions.

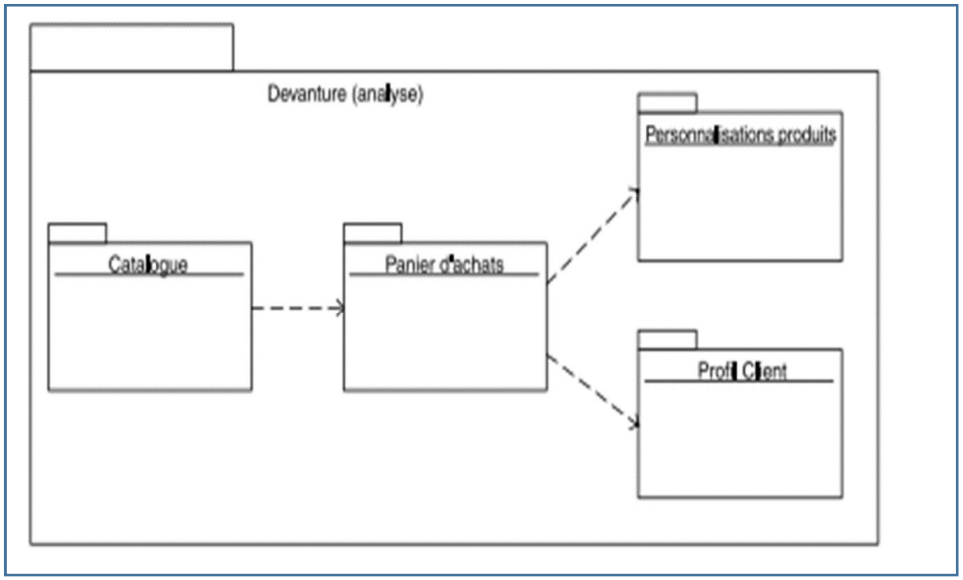

Figure 4.7 : Vue structurelle de paquetage Devanture

Une bonne façon de démarrer le modèle d'analyse consiste à commencer par les packages de diagrammes de cas d'utilisation de haut niveau.

À partir de là, il est préférable d'examiner les cas d'utilisation et les exigences fonctionnelles d'un nouveau point de vue qui consiste à diviser le modèle en fonction de choses similaires (classes d'objets).

6. Le diagramme de classe dans le modèle d'analyse

L'analyse des substantifs est une autre technique pour identifier les classes et les objets. Les textes de cas d'utilisation et d'exigences sont analysés pour déterminer les noms importants. Ces noms indiquent des classes d'objets possibles. D'autre part, les verbes indiquent les opérations et les processus possibles.

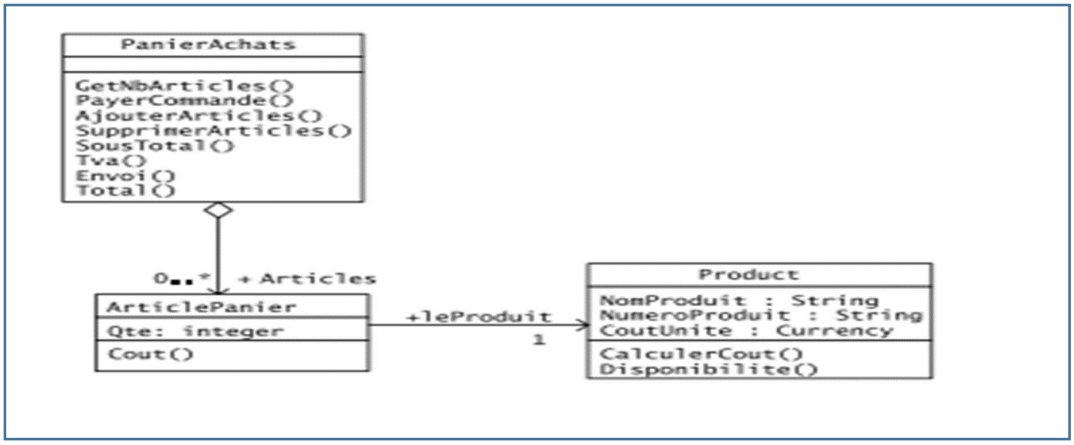

Figure 4.8 : Le diagramme de classe dans le modèle d'analyse

7. Le diagramme de séquence

Exprimer la collaboration entre les classes fait autant partie de l'analyse que la définition de la classe.

Le mécanisme UML pour exprimer la dynamique de la collaboration de classe est le diagramme d'interaction, qui est le générique de plusieurs types de diagrammes : collaboration, séquence et activité. Ces diagrammes expriment le comportement dynamique du système, en utilisant la classe structurelle et les éléments de relation du modèle.

Les diagrammes de séquence et de collaboration, en particulier, fournissent un lien critique de traçabilité entre les scénarios de cas d'utilisation et les structures des classes.

Ces diagrammes peuvent exprimer le flux dans un scénario de cas d'utilisation en termes de classes qui les implémenteront éventuellement.

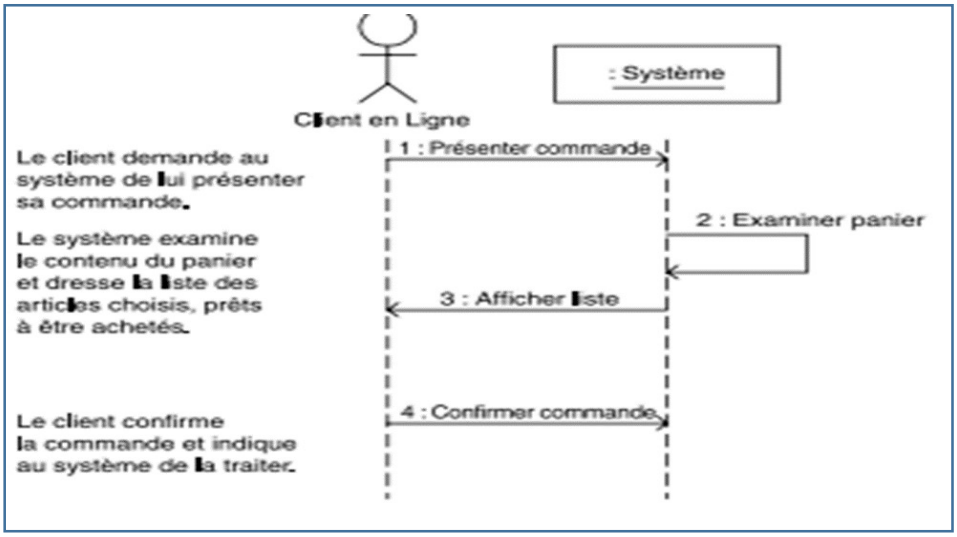

Figure 22 : Diagramme de séquence simple à partir d'un scénario de cas d'utilisation

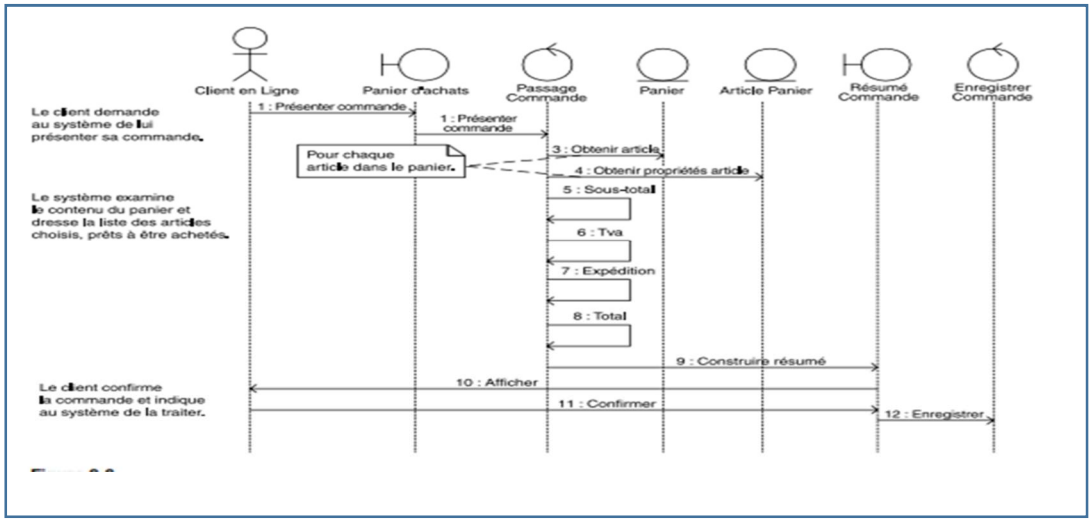

Figure 4.9 : Diagramme de séquence élaboré pendant l'analyse

8. Le diagramme de collaboration

Les diagrammes de collaboration sont essentiellement les mêmes que les diagrammes de séquence.

Même si sémantiquement, ils disent la même chose, chaque type de diagramme exprime l'information avec une vue différente. Dans les diagrammes de collaboration, l'accent est mis sur les instances d'objets. Les objets d'un diagramme de collaboration peuvent être placés n'importe où dans le diagramme, avec une seule ligne représentant tous les messages d'un objet à un autre. Chaque message est numéroté (pour préserver la dimension temporelle) et regroupé en une seule association entre chaque objet.

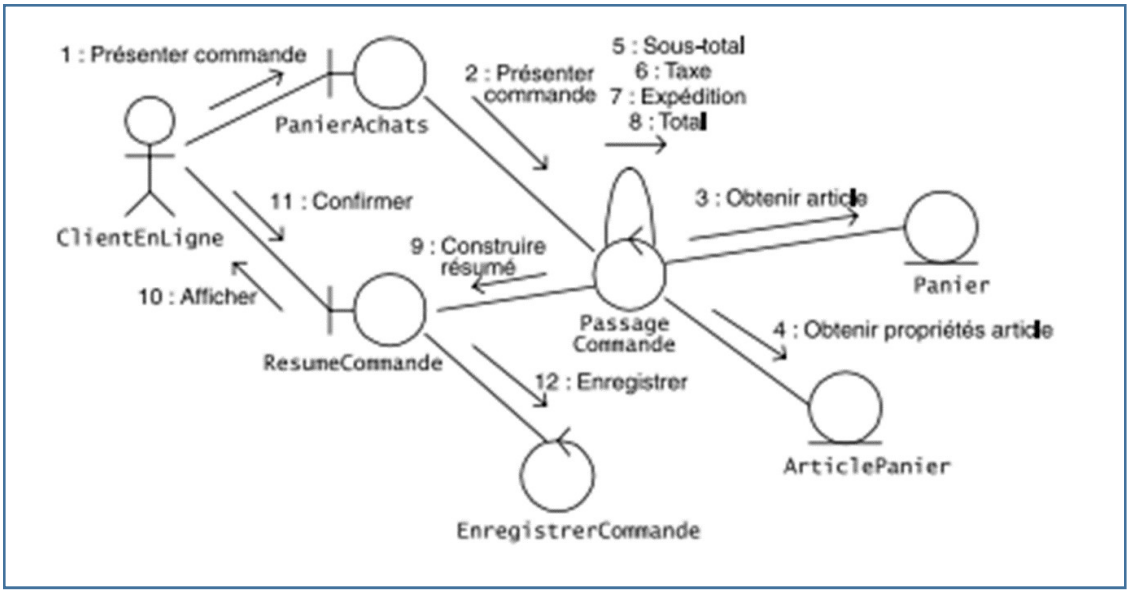

Figure 4.10 : Le diagramme de collaboration

9. Le diagramme d'activité

Les diagrammes d'activités sont utiles pour exprimer le flux de travail.

Par définition, ils montrent le flux des activités, qui à son tour aboutissent à des actions.

Les diagrammes d'activité peuvent être utilisés pour modéliser les activités d'une opération spécifique. Lorsqu'ils sont utilisés de cette manière, ils sont similaires aux organigrammes.

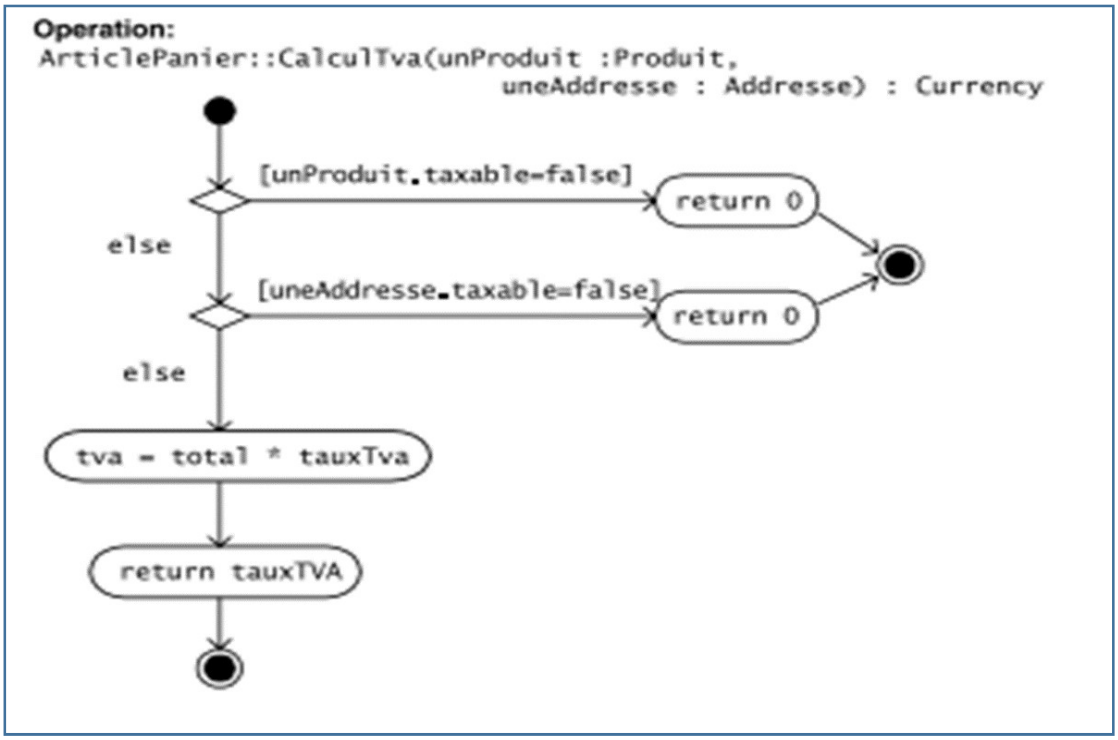

Figure 24 : Le diagramme d'activité de l'opération calcul TVA

10.Conclusion

Ce chapitre résume les différentes étapes à réaliser durant la phase d'analyse
Le chapitre prochain décrit la phase de conception.

Chapitre 04

Les phases du processus

La conception

1. Introduction

La conception commence par le modèle d'analyse et l'architecture en tant qu'apports principaux.

L'activité principale de la conception est d'affiner le modèle d'analyse de sorte qu'il puisse être implémenté avec les composants de l'architecture.

Même si cela semble simple, cela peut être la phase la plus complexe d'un projet de développement, en particulier lorsque des progrès significatifs dans la technologie logicielle se produisent si fréquemment.

2. L’étape de conception

Les activités de conception tournent autour des diagrammes de classe et d'interaction.

Les classes deviennent plus définies, avec des propriétés pleinement qualifiées (nom et type) et des opérations (signatures complètes).

Des classes supplémentaires, principalement des classes d'assistance et d'implémentation, sont souvent ajoutées lors de la conception.

À la fin, le modèle de conception résultant est quelque chose qui peut être mappé directement dans le code. C'est le lien entre les abstractions de l'entreprise et les réalités du logiciel.

3. Les diagrammes de la conception

Au cours de l'analyse, nous nous sommes contentés de travailler uniquement avec des diagrammes de classes et des diagrammes d'interaction.

Lors de la conception, un nouveau diagramme, en fait, une nouvelle vue du modèle : la vue des composants est introduite.

3.1. Le diagramme des composants

Le diagramme de composant exprime le physique - si quelque chose dans le logiciel peut être considéré comme un «physique» - modules exécutables qui seront distribués en tant que système.

Un composant est généralement mappé vers des fichiers exécutables, des fichiers de classe Java, des bibliothèques statiques ou des bibliothèques de liens dynamiques (DLL). Un composant est quelque chose qui réalise un ensemble d'interfaces. Une interface définit le nom de la fonction, ses paramètres et leurs types de données, qu'ils soient facultatifs, qu'ils soient en entrée ou en sortie, et le type de valeur de retour de la fonction. Un composant peut réaliser plusieurs interfaces.

La réalisation d'un composant se fait avec les classes et collaborations exprimées dans la vue logique. Chaque classe de la vue logique est implémentée par au moins un composant. Des classes abstraites définissant des interfaces peuvent être implémentées par de nombreux composants.

Les composants sont rendus dans un diagramme avec un ensemble de rectangles. Les interfaces sont rendues avec une "sucette" ou un cercle sur un bâton. Les dépendances sont indiquées avec des lignes pointillées et des pointes de flèches.

Les diagrammes de composants visualisent les composants, les interfaces et leurs interrelations.

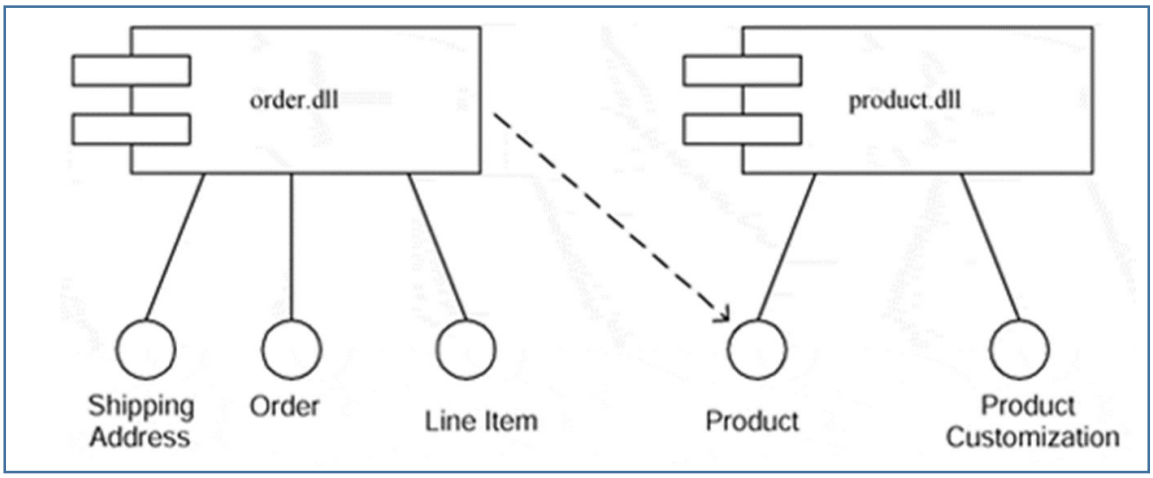

Figure 4.11 : Le diagramme de composant

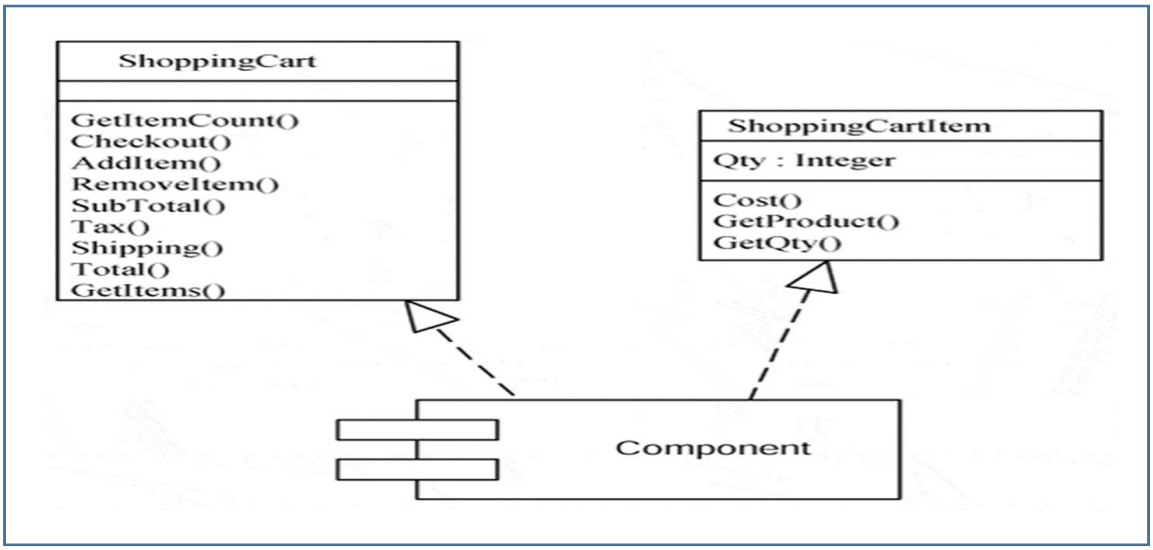

Figure 4.12 : Panier d'achat réalise les classes ShoppingCart et ShoppingCartItem.

4. Les activités de conception

Lors de la conception, l'interface doit être élaborée en un ensemble d'interfaces spécifiques capables de gérer la communication entre les acteurs et le système.
Les activités de conception comprennent aussi :

1) Partitionner des objets en niveaux, tels que client, serveur, etc.
2) Séparer et définir des interfaces utilisateur ou des pages Web (problème de modélisation des pages web scriptées avec UML)

5. L'extension de l'UML pour les applications web

Une extension à UML est exprimée en termes de stéréotypes, de valeurs marquées et de contraintes. Combinés, ces mécanismes nous permettent de créer de nouveaux types de blocs de construction que nous pouvons utiliser dans le modèle.

5.1. Les stéréotypes [1]

Un stéréotype représente une extension du vocabulaire de la langue. Il nous permet d'associer une nouvelle signification sémantique à un élément du modèle.

Les stéréotypes sont représentés sous la forme d'une chaîne entre deux guillemets «» Cependant, ils peuvent également être rendus par une nouvelle icône.

5.2. Les valeurs étiquetées [3]

La valeur étiquetée est une extension à une propriété d'un élément de modèle.

Une valeur étiquetée est la définition d'une nouvelle propriété pouvant être associée à un élément de modèle.

Une valeur étiquetée est rendue sur un diagramme sous la forme d'une chaîne entre crochets [].

5.3. Les contraintes [4]

Une contrainte est une extension de la sémantique de la langue. C'est une règle qui définit comment le modèle peut être assemblé.

Une contrainte spécifie les conditions dans lesquelles le modèle peut être considéré comme «bien formé».

Les contraintes sont affichées en tant que chaînes entre deux accolades {}.

6. La conception d'une application web

Le partitionnement correct des objets métier dans une application Web est essentiel et dépend beaucoup de l'architecture. Les objets peuvent résider exclusivement sur le serveur, le client ou les deux.

Certains objets, tels que Facture, peuvent avoir des vies dans les deux. Un objet de facture client peut exister sur le serveur. Cet objet pourrait être envoyé sous la forme d'un document XML au client. Le document XML pourrait être utilisé comme état de la facture.

Lorsque le travail est à portée de main, le partitionnement des objets est simple.

Les applications basées sur l'architecture client web léger placent tous les objets derrière le serveur.

Les applications client web dynamique permettent à certains objets de s'exécuter sur le client.

Les applications objets distribués ont le plus de liberté dans le placement des objets, car ce sont essentiellement des systèmes d'objets distribués qui utilisent simplement un navigateur.

6.1. Partitionnement d'objets pour les applications web avec l'architecture client dynamique

Lors du premier passage, les objets persistants, les objets conteneur, les objets partagés et les objets complexes appartiennent tous au serveur.

Les objets associés à des ressources de serveur telles que des bases de données et des systèmes hérités appartiennent également au niveau serveur.

Les objets qui gèrent des associations statiques ou des dépendances avec l'un de ces objets doivent également exister sur le serveur.

Il est plus facile d'identifier les objets pouvant exister sur le client.

Si un objet n'a pas d'associations ou de dépendances avec des objets sur le serveur et possède des associations et des dépendances uniquement avec d'autres ressources client, telles que les navigateurs et les applets Java, il peut exister sur le client.

Les objets candidats pour le partitionnement sur le client sont des objets de validation de champ, des contrôles d'interface utilisateur et des contrôles d'assistance à la navigation.

6.2. Partitionnement des objets pour les applications web avec l'architecture des objets distribués

Le modèle d'architecture d'objets distribués (services web) est essentiellement un système d'objets distribués basé sur un site Web.

Ce type d'application utilise des protocoles de communication client et serveur autres que HTTP.

Les objets réels peuvent s'exécuter dans le contexte du client ou du navigateur et ont donc accès à ses ressources.

La partition d'objets pour ce type d'architecture dépend principalement de la nature des objets individuels.

L'une des raisons principales de la distribution d'objets au client est de retirer une partie de la charge du serveur.

Il est également naturel de placer les objets là où ils seront le plus efficaces dans le système.

7. Les diagrammes de séquence de la conception

La définition des pages web implique la découverte de pages Web et de leurs relations entre elles et avec les objets du système. Cette étape dépend aussi fortement du modèle architectural de l'application.

Pendant cette activité, les objets Système des diagrammes de séquence créés lors de l'analyse évoluent en objets et en pages Web, l'interface utilisateur principale de l'application Web.

7 .1. La conception avec le modèle client web léger

Dans les applications client web léger, les acteurs n'interagissent qu'avec les pages client et les pages du serveur n'interagissent qu'avec les ressources du serveur.

Par conséquent, il faut mettre une page client et serveur dans le diagramme de séquence.

La solution la plus simple consiste à transformer directement les objets frontières du modèle d'analyse en pages client et à transformer les objets du contrôleur en pages serveur.

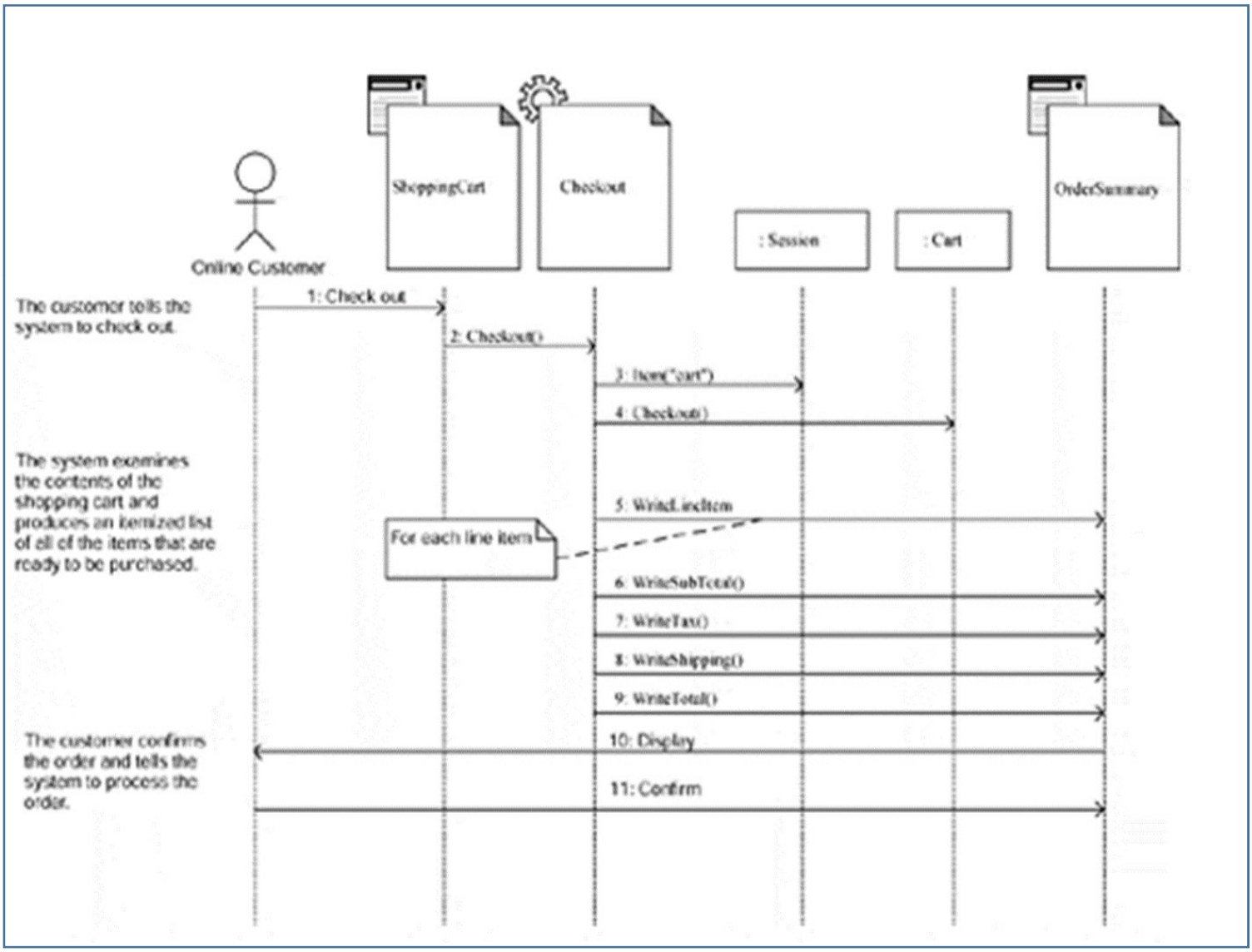

Figure 4.13 : Le diagramme de séquence élaboré pour l'opération de paiement

7.1.1. **Le diagramme de séquence**

Une grande partie de l'activité détaillée du processus de vérification n'est pas représentée dans le diagramme mais peut être exprimée dans des diagrammes supplémentaires. Ce type d'activité peut inclure la création de transactions, ou d'objets intermédiaires, nécessaires pour vérifier un panier.

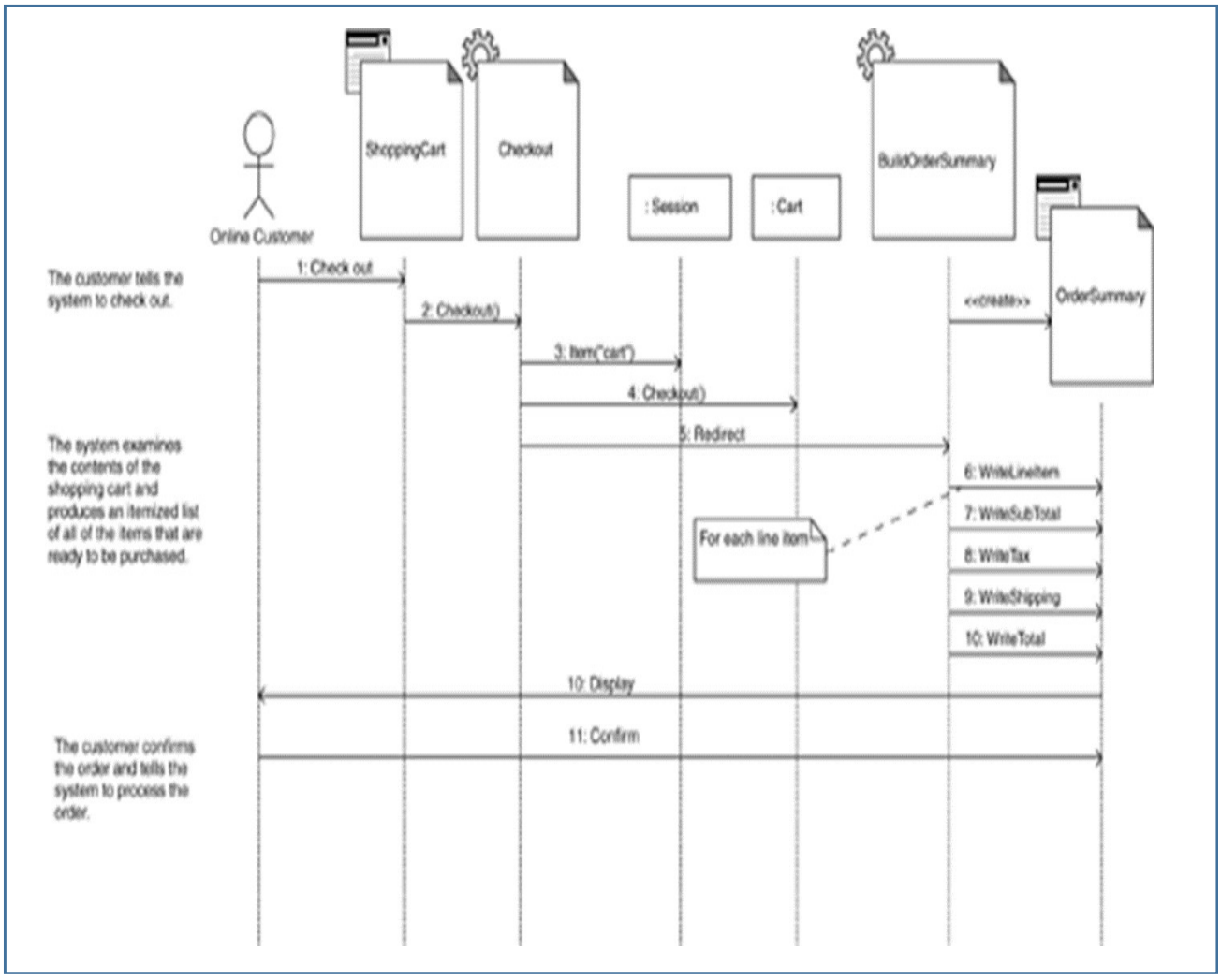

Figure 4.14 : Séparer les responsabilités de la page du serveur

Explication

Les parties restantes de ce scénario simple commencent par la nouvelle page client, OrderSummary, car elle a remplacé la page Commander dans le navigateur du client.

Dans le scénario, l'acteur confirme le processus de commande en envoyant un message à la page OrderSummary, puisque c'est la page maintenant disponible pour l'acteur. Ce message est aussi probablement un lien hypertexte vers une autre page du serveur.

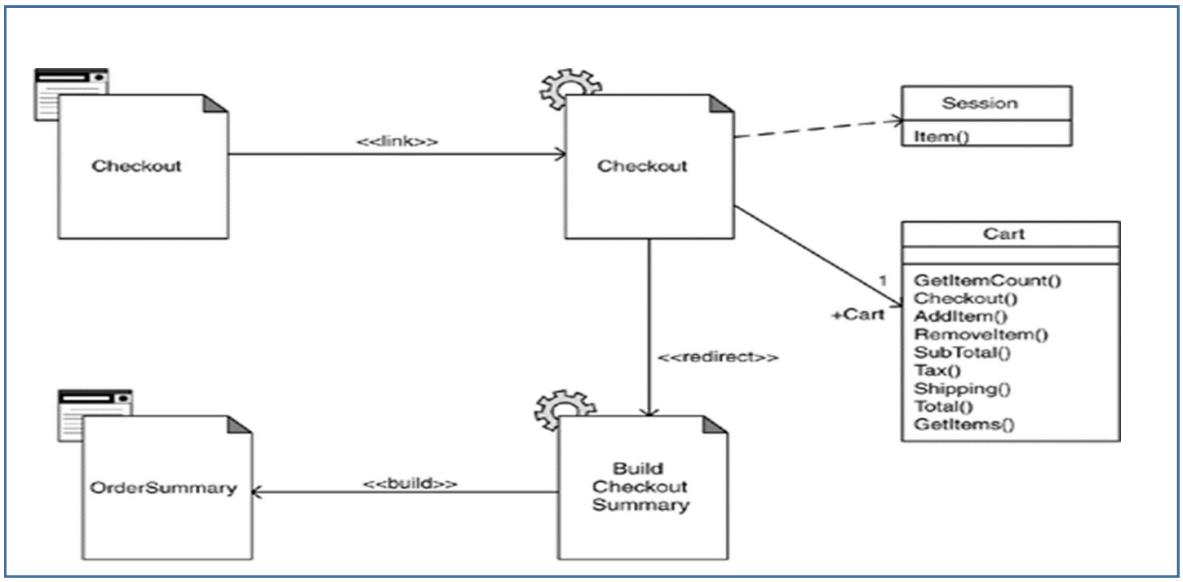

Figure 4.15 : La vue logique des classes liées à la vérification d'un panier

71.2. Le diagramme de composants

En fin de compte, ces pages Web conceptuelles (serveur et client) doivent se retrouver dans un composant du système.

Le diagramme qui suit montre trois composants, chacun représentant une page Web (ShoppingCart.asp, Checkout.asp et BuildOrderSummary.asp) qui sera fournie avec le système final. Ce diagramme est une réalisation de composants des pages de paiement

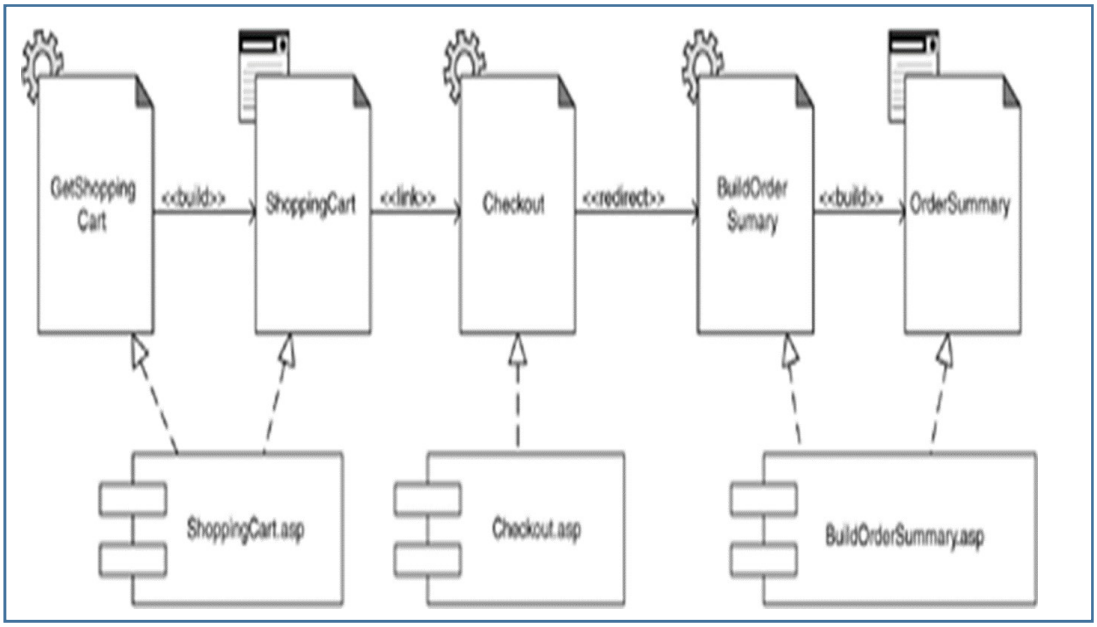

Figure 4.16 : Réalisations de composants des pages de paiement

7.2. **La conception avec le modèle client dynamique**

La conception d'applications Web comportant des pages client dynamiques nécessite une attention particulière au partitionnement des objets.

Dans ce type d'applications, il n'y avait aucune tentation de dessiner des associations entre les objets client et les objets serveur, car il n'y a pratiquement aucun objet client définissable par l'utilisateur. Ce type d'applications peut cependant avoir toutes sortes d'objets et d'activités sur le client.

7.2.1. Les scriptes cotés client

La conception des systèmes client web dynamique commence par les diagrammes de séquence produits avec les cas d'utilisation. Une page client peut exécuter des opérations sur elle-même. Les opérations sont des fonctions JavaScript dans la page

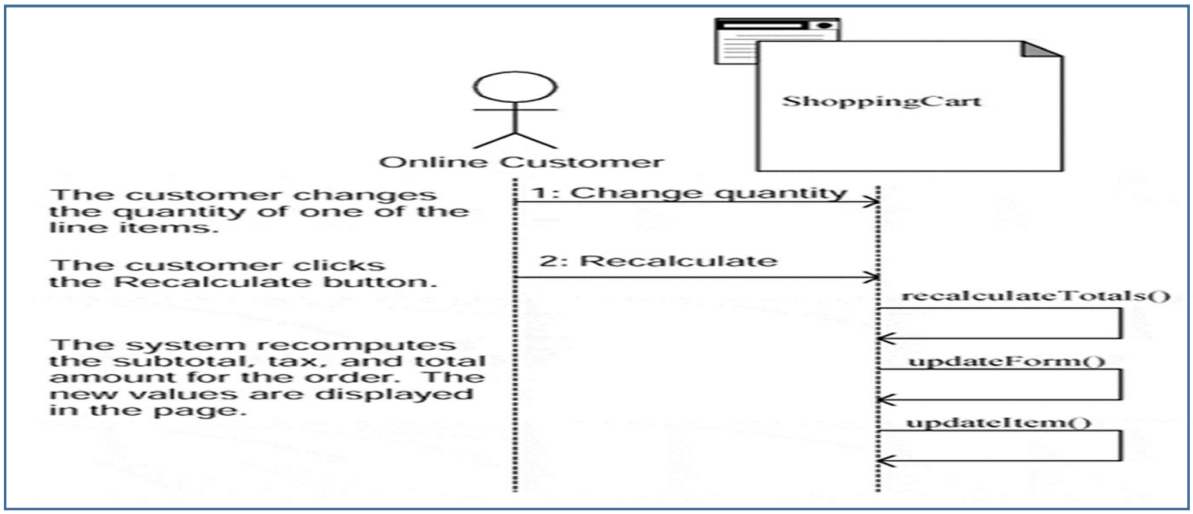

Figure 4.17 : Exprimer des scripts dans une page client

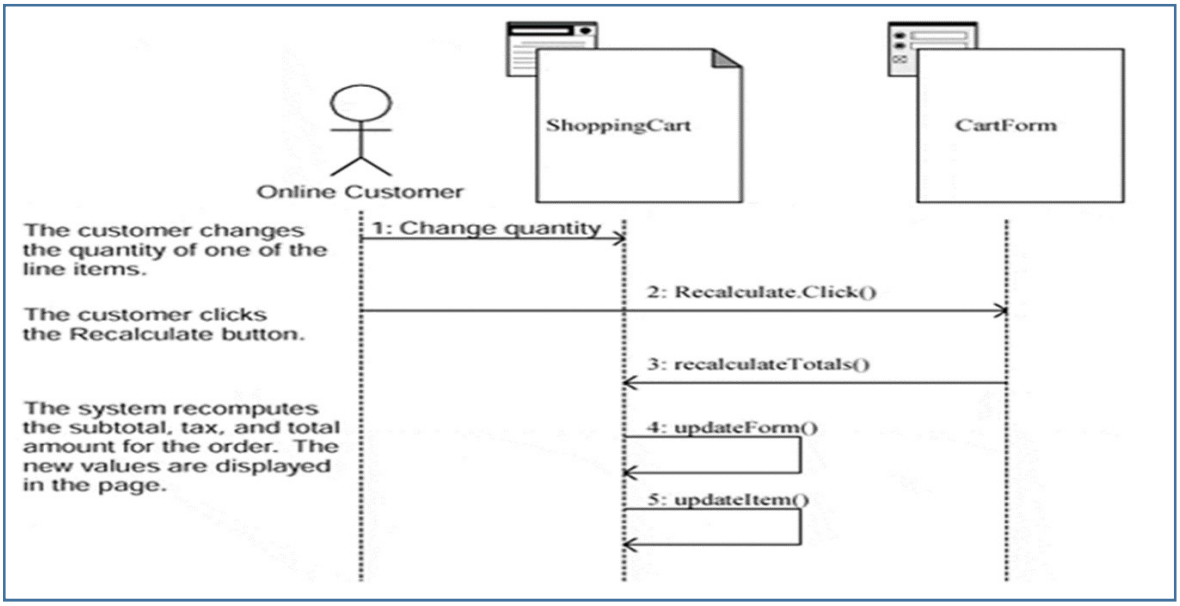

Figure 4.18 : Manière alternative pour exprimer un script dans une page client

7.2.2. Objets client

La fonctionnalité peut être fournie non seulement par les scripts, mais également avec des objets tels que les contrôles ActiveX, les applets Java et les JavaBeans. Ces types de composants, souvent utilisés lorsque des fonctionnalités vraiment sophistiquées sur le client sont requises.

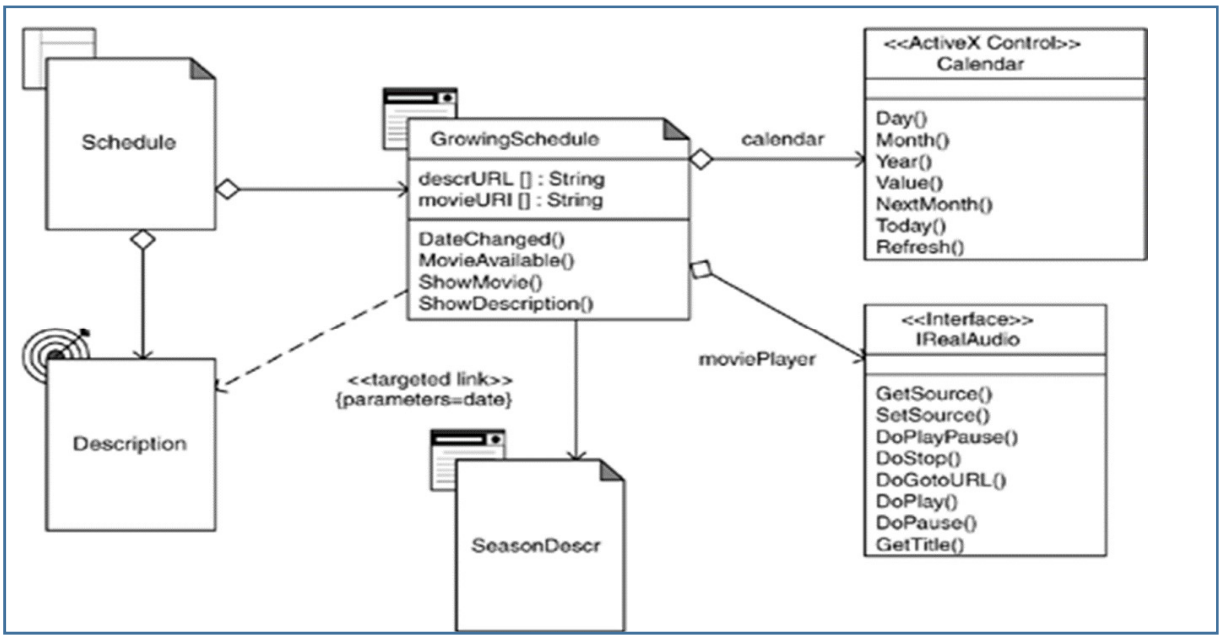

Figure 4.19 : Diagramme de classes avec une page client utilisant des composants ActiveX

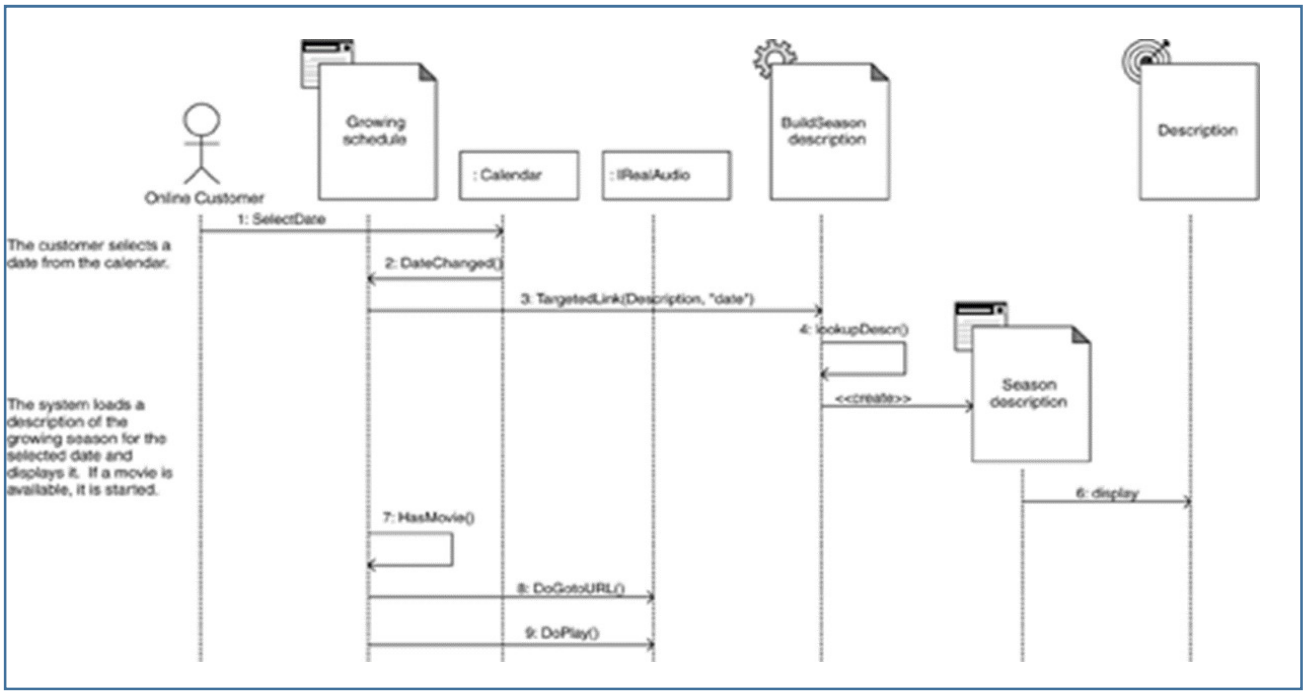

Figure 4.20 : Diagramme de séquence impliquant des composants côté client

7.3. La conception et le modèle des objets distribués (services web)

Pour une flexibilité maximale, les applications Web peuvent utiliser des objets réels distribués. La décision de savoir quand utiliser les applets Java et les beans, RMI / IIOP, ActiveX ou DCOM dépend des besoins de l'application et du niveau d'expérience de l'équipe de développement avec ces technologies. La modélisation des applications Web compatibles DCOM est simple.

Les objets sur le client peuvent interagir et envoyer des messages à d'autres objets sur le client (et de même pour les objets serveur), mais la communication entre les objets client et serveur représente quelque chose d'important sur le plan architectural.

En raison de cette importance, les associations entre les objets s'exécutant sur le client et les objets s'exécutant sur le serveur sont stéréotypées << DCOM >>, en supposant que c'est le mécanisme de communication.

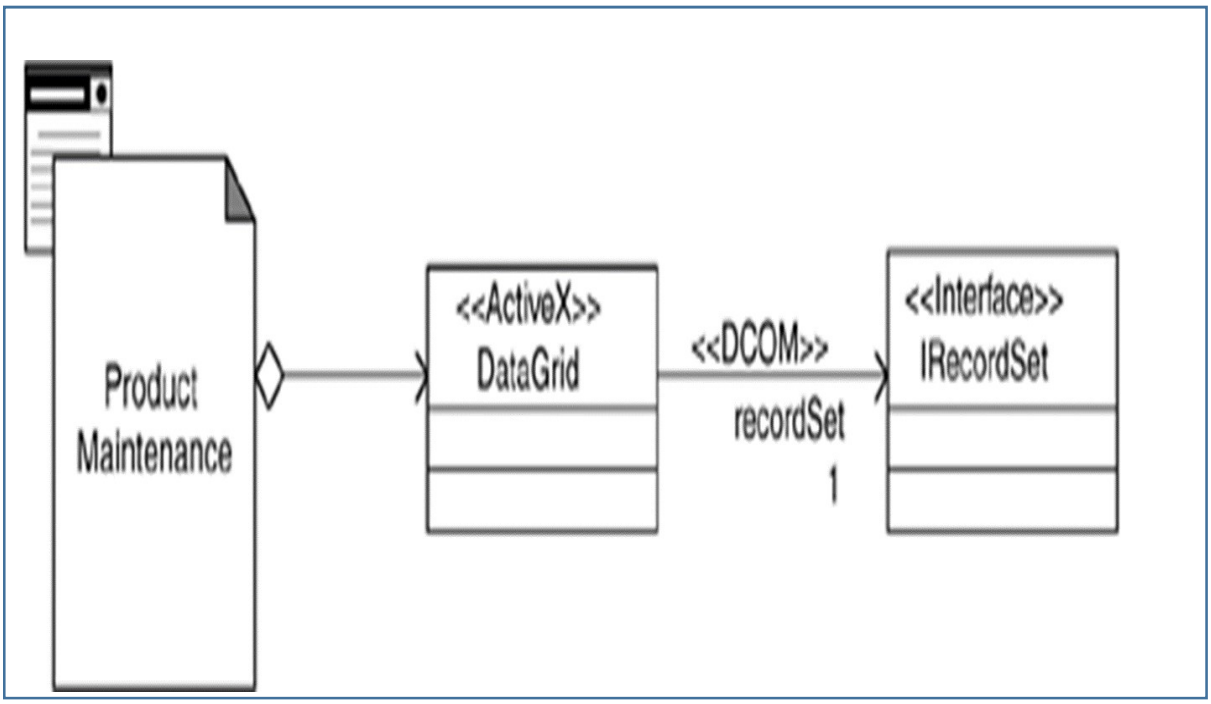

Figure 4.21 : Diagramme de classes avec des objets distribués

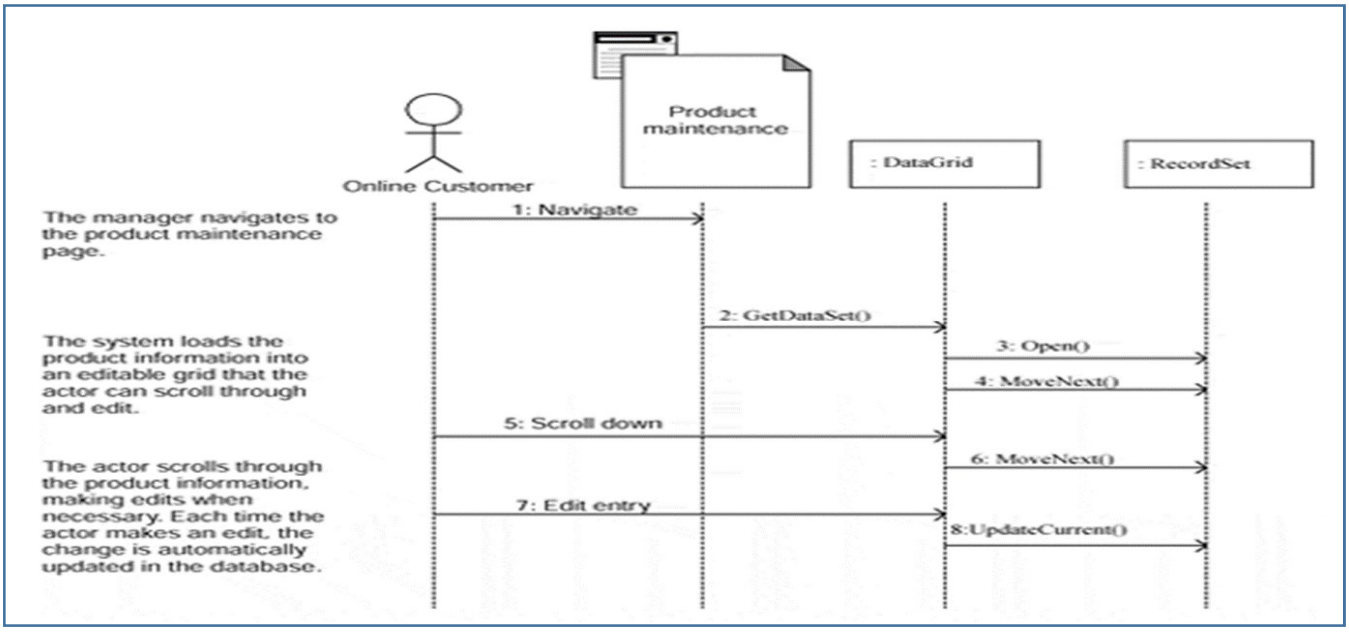

Figure 4.22 : Diagramme de séquence du cas d'utilisation Entrée de produit

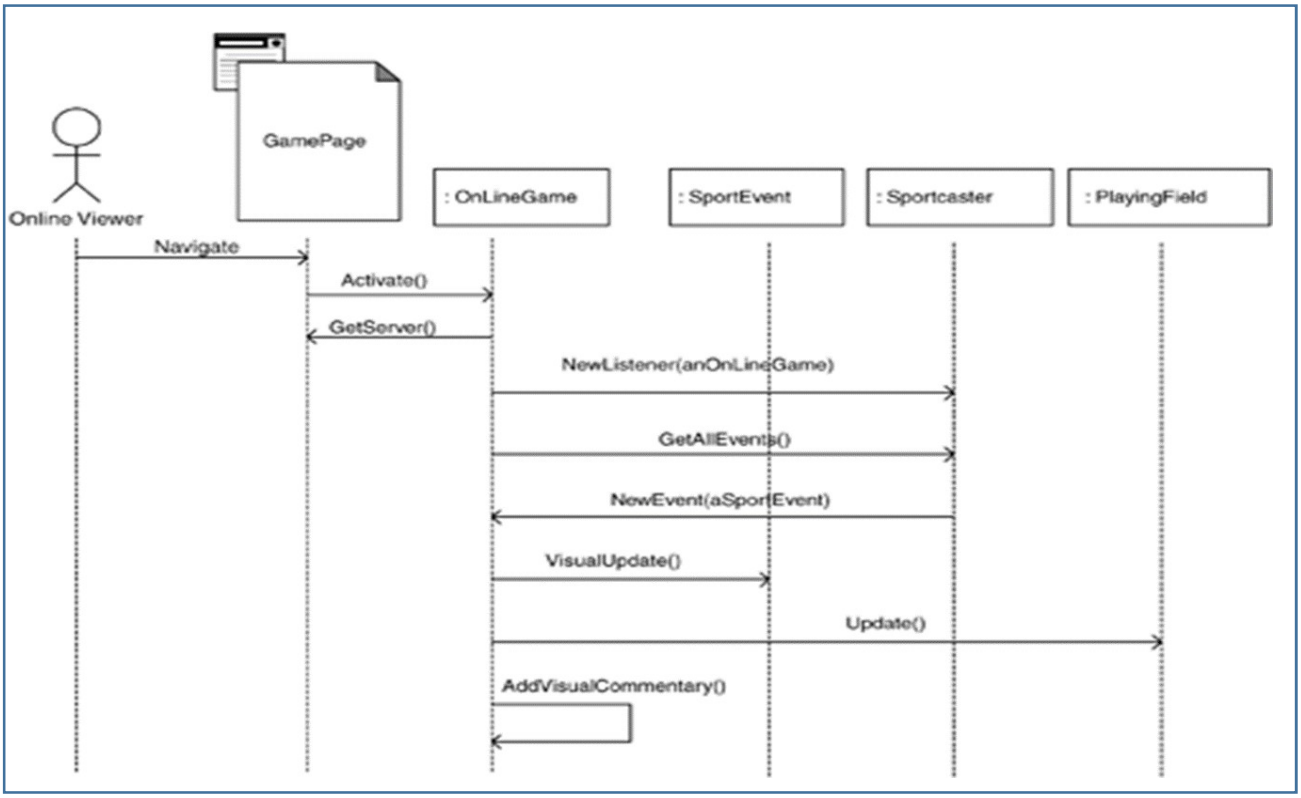

Figure 4.23 : Un système de surveillance d'événements sportifs en ligne

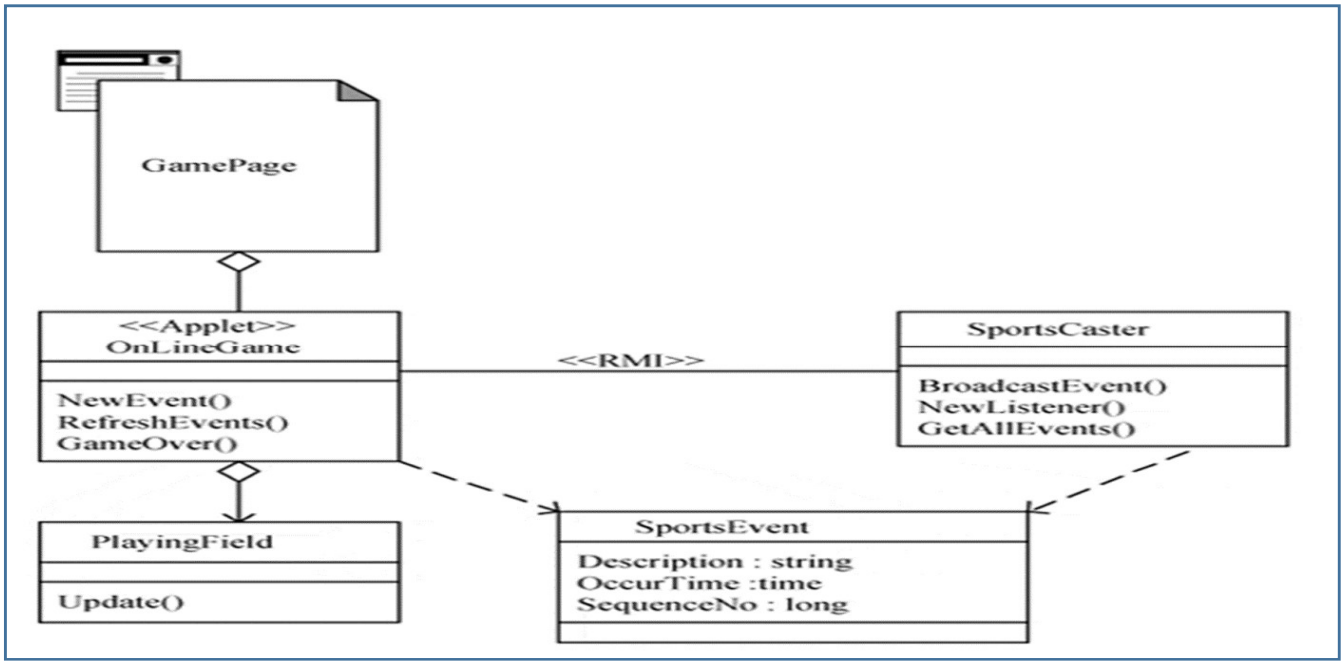

Figure 4.24 : Vue logique de l'application de surveillance d'événements sportifs

8. Conclusion

L'étape de conception permet de faire la liaison entre l'analyse et l'architecture choisie pour l'application modélisée.

Les diagrammes réalisés durant cette étape peuvent être mappés directement vert le code

Annexe

Cette partie contient un cours additive qui concerne le langage de modélisation UML et la démarche à suivre pour la modélisation d'une application web durant les séances des travaux dirigés.

L'objectif de ce cours est de donner une vue générale sur le langage UML et les extensions apportées à ce langage pour supporter la modélisation des applications web.

Chapitre supplémentaire : UML et la modélisation d'une application web

Plan du cours

1. **Introduction**
2. **Concevoir une application**
3. **Pourquoi créer un modèle ?**
4. **Modélisation d'une application web**
5. **Le langage de modélisation unifié**
6. **Les but de l'UML**
7. **Les principes de l'UML**
8. **Convention et terminologie**
9. **Les bases de l'UML**
10. **Le processus simplifié pour les applications web**
11. **Les principes de la méthode Agile**
12. **La démarche de modélisation d'une application web**
13. **Conclusion**

1. Introduction

La modélisation est un processus visuel utilisé pour construire et documenter la conception et la structure d'une application.

C'est une bonne idée de faire au moins quelques grandes lignes d'une application, montrant les interdépendances et les relations entre les composants et les sous-systèmes, au cours du développement.

Les outils de modélisation facilitent ce processus ; lorsqu'un changement est apporté au modèle, l'effet d'entraînement de ce changement est indiqué.

L'utilisation d'outils de modélisation donne aux développeurs une vue de haut niveau de ce qui pourrait représenter des milliers de lignes de code.

2. Concevoir une application

Un obstacle majeur à une ingénierie réussie est l'incapacité d'analyser et de communiquer les nombreuses activités d'interaction qui composent le processus métier.

Les langues conversationnelles s'avèrent trop ambiguës pour être efficaces

Ce qui est nécessaire est plutôt une technique qui structure le langage conversationnel pour éliminer l'ambiguïté, facilitant la communication et la compréhension efficaces

3. Pourquoi créer un modèle ?

La modélisation est une technique efficace pour la compréhension et la communication qui a été utilisée pendant des siècles.

Dans un modèle de processus, les détails superflus sont éliminés, réduisant ainsi la complexité apparente du système étudié

Le détail restant est structuré pour éliminer toute ambiguïté tout en mettant en évidence des informations importantes.

Les graphiques (images, lignes, flèches, normes graphiques) sont utilisés pour fournir une grande partie de la structure, de sorte que la plupart des gens considèrent les modèles de processus comme des représentations picturales.

La modélisation du processus métier cible est une première étape nécessaire dans le développement d'une application

Le modèle devient une carte routière qui établira la destination finale.

4. La modélisation d'une application web

Les applications Web deviennent de plus en plus complexes et essentielles à la mission. La modélisation peut aider à gérer cette complexité croissante. La norme actuelle pour

la modélisation de systèmes à forte composante logicielle est le langage UML (Unified Modeling Language).

Lors de la modélisation d'une application Web avec UML, il devient évident que certains composants ne correspondent pas automatiquement aux éléments de modélisation UML standard.

Afin d'utiliser une seule notation de modélisation pour un système complet comprenant des composants Web et des composants de niveau intermédiaire traditionnels, UML doit être étendu.

5. Le langage de modélisation unifié

La force d'UML est enracinée dans sa sémantique ainsi que dans le méta-modèle qui constitue la base des futures méthodes. Le méta-modèle est une couche d'une architecture de méta-modelage à quatre couches.

Le langage de modélisation unifié est un langage de modélisation orienté objet permettant de spécifier, de visualiser et de documenter les artefacts d'un système orienté objet pendant le développement. UML est approprié pour les systèmes en temps réel, fournissant un support pour les classes de modélisation, les objets et les relations entre eux. Ces relations incluent l'association, l'agrégation, l'héritage, la dépendance et l'instanciation.

Les cas d'utilisation sont directement pris en charge avec des scénarios pour des descriptions détaillées du comportement du système requis.

Les scénarios de modèle sont représentés graphiquement par des diagrammes d'interaction qui peuvent inclure à la fois des annotations de synchronisation du message.

Les fonctionnalités en temps réel sont prises en charge grâce à la modélisation améliorée des machines à états finis, y compris la simultanéité, la propagation d'événements et les états imbriqués.

Les créateurs de l'UML ont réalisé qu'il n'est pas toujours suffisant de capturer la sémantique pertinente d'un domaine ou d'une architecture particulière. Un mécanisme d'extension formel a été défini afin que les praticiens étendent la sémantique de l'UML. Ce mécanisme permet de définir des stéréotypes, des valeurs étiquetées et des contraintes qui peuvent être appliquées aux éléments du modèle.

- **Un stéréotype** : permet de définir une nouvelle signification sémantique pour un élément de modélisation.
- **Les valeurs marquées :** sont des paires de valeurs clés pouvant être associées à un élément de modélisation.
- **Les contraintes** : sont des règles définissant la meilleure façon d'exprimer un modèle.

6. Les buts d'UML [3]

L'UML a sept objectifs :

1) Fournir aux utilisateurs un langage de modélisation visuel expressif prêt à l'emploi pour le développement et l'échange de modèles significatifs.

2) Fournir des mécanismes d'extensibilité et de spécialisation afin d'étendre les concepts centraux.
3) Être indépendant des langages de programmation et des processus de développement spécifiques.
4) Fournir une base formelle pour comprendre le langage de modélisation.
5) Encourager le développement du marché des outils OO.
6) Pour soutenir les concepts de développement de plus haut niveau, y compris les collaborations, les cadres, les modèles et les composants.
7) Intégrer les meilleures pratiques.

7. Les principes de l'UML [4]

Un méta-modèle a été créé afin d'offrir une déclaration commune et claire du modèle sémantique de la méthode. Une fois ce méta-modèle développé, des décisions sur la syntaxe graphique de surface pour les éléments visuels de ce modèle ont été prises.

Un certain nombre de principes pour guider ces efforts ont été établis **:**

7.1. La simplicité

Une méthode ne devrait nécessiter que quelques concepts et symboles.

7.2. L'expressivité

Une méthode devrait être applicable à une grande variété de systèmes de qualité de production.

7.3. L'utilité

Une méthode doit uniquement se concentrer sur des éléments significatifs pour l'ingénierie système et logicielle.

7.4. L'auto-consistance

Le même concept et le même symbole doivent être utilisés de la même manière tout au long de la méthode.

7.5. Orthogonalité

Les concepts non liés doivent être modélisés indépendamment.

7.6. Concepts avancés

Les couches de concepts avancés doivent être traitées comme des ajouts à la méthode pour des concepts plus basiques.

7.7. Stabilité

Les concepts et les symboles déjà compris et utilisés doivent être adoptés.

7.8. Imprimabilité

L'approche de la visualisation doit être basée sur un format qui peut être étendu. Les utilisateurs devraient être en mesure d'esquisser la méthode manuellement ou de l'afficher en tant qu'image imprimée.

7.9. Adaptabilité

Les utilisateurs extensibles et les constructeurs d'outils devraient avoir la possibilité d'étendre et d'adapter la méthode.

8. Conventions e terminologie

En langage UML, l'explication de chaque élément suit un modèle simple. La sémantique du concept est décrite, suivie de sa syntaxe graphique.

- **Sémantique :** Cette sous-section fournit un bref résumé de la sémantique
- **Notation :** Cette sous-section explique la représentation notationnelle du concept sémantique
- **Options de présentation** : cette sous-section décrit différentes options de présentation des informations de modèle, telles que la possibilité de supprimer ou de filtrer des informations, d'autres manières de montrer des choses et d'autres manières de présenter des informations dans un outil.

9. Les bases de l'UML2

UML 2 s'articule autour de treize types de diagrammes, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système logiciel.

9.1. Les diagrammes de l'UML

Les types de diagrammes sont répartis en deux grands groupes :

9.1.1. Les diagrammes structurels

Il existe six diagrammes structurels :

A. **Diagramme de classe :** Il montre les briques de base statiques : classes, associations, interfaces, attributs, opérations, généralisations, etc.

B. **Diagramme d'objets** : **[3]** Il montre les instances des éléments structurels et leurs liens à l'exécution.

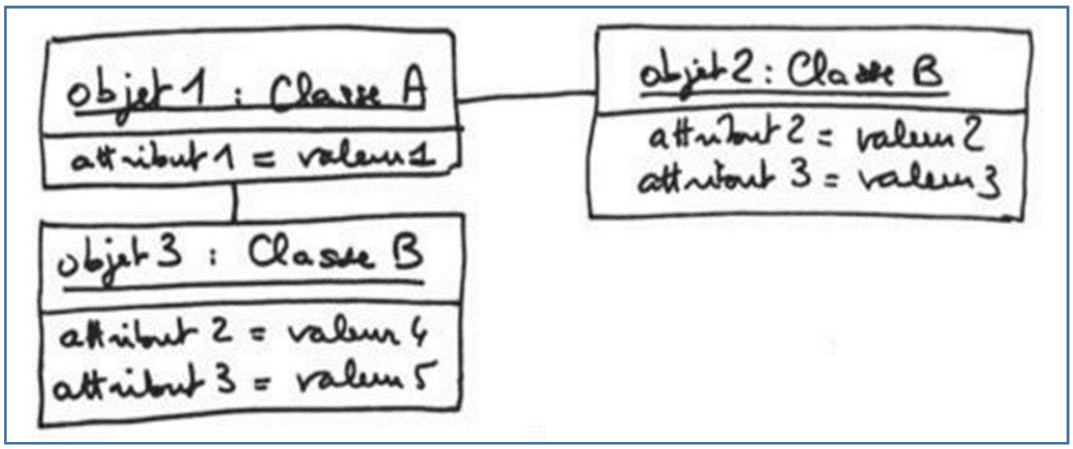

Figure 1 : Diagramme d'objets

Le diagramme d'objets est un instantané, une photo d'un sous-ensemble des objets d'un système à un certain moment du temps.

C. **Diagramme de packages** : **[3]** Il montre l'organisation logique du modèle et les relations entre packages.

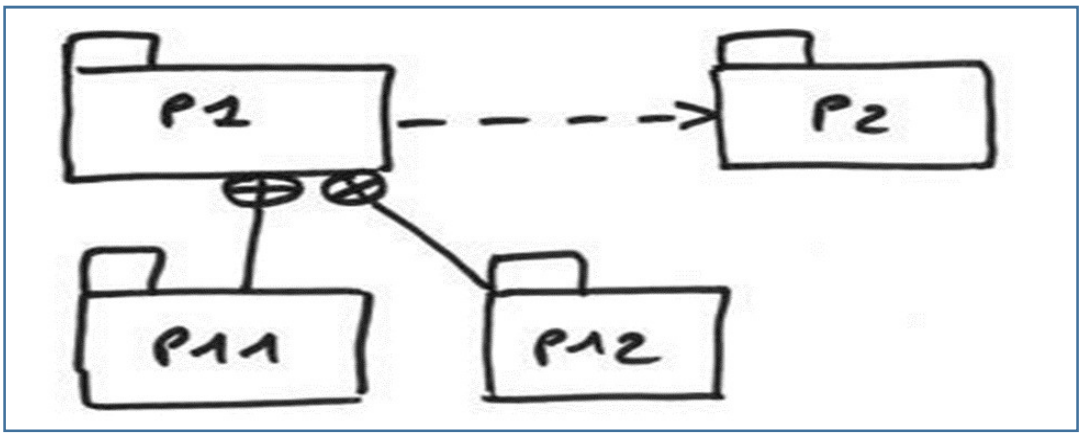

Figure 2 : Diagramme de packages

Le diagramme de packages qui montre l'organisation logique du modèle et les relations entre packages. Il permet de structurer les classes d'analyse et de conception et aussi les cas d'utilisation.

D. **Diagramme de structure composite :** Il montre l'organisation interne d'un élément statique complexe sous la forme d'un assemblage de parties, de connecteurs et de ports.

E. **Diagramme de composants** : **[3]** Il montre des structures complexes, avec leurs interfaces fournies et requises.

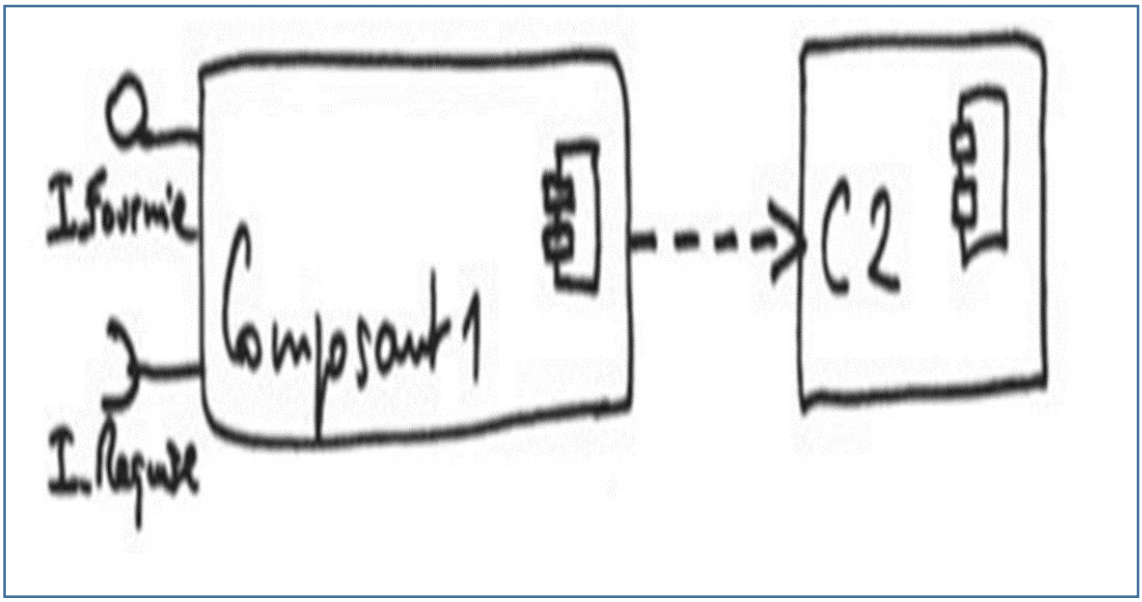

Figure 3 : diagramme de composants

Le diagramme de composants montre les unités logicielles à partir desquelles on a construit le système informatique, ainsi que leurs dépendances.

F. **Diagramme de déploiement** : Il montre le déploiement physique des «artefacts» sur les ressources matérielles.

9.1.2. Les diagrammes comportementaux :

Il existe sept diagrammes comportementaux

A. **Diagramme de cas d'utilisation** : Il montre les interactions fonctionnelles entre les acteurs et le système à l'étude.
B. **Diagramme de vue d'ensemble des interactions** : Il fusionne les diagrammes d'activité et de séquence pour combiner des fragments d'interaction avec des décisions et des flots.
C. **Diagramme de séquence :** Il montre la séquence verticale des messages passés entre objets au sein d'une interaction.
D. **Diagramme de communication** : **[3]** Il montre la communication entre objets dans le plan au sein d'une interaction.

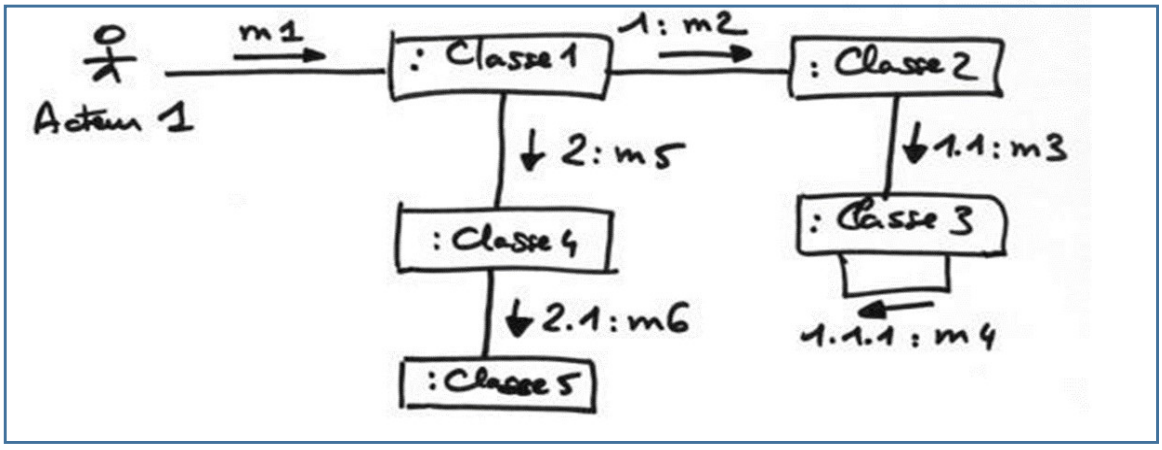

Figure 4 : Diagramme de communication

Les diagrammes de communication représentent des échanges de messages entre éléments, dans le cadre d'un fonctionnement particulier du système.

E. **Diagramme de temps :** Il fusionne les diagrammes d'états et de séquence pour montrer l'évolution de l'état d'un objet au cours du temps et les messages qui modifient cet état.
F. **Diagramme d'activité** : Il montre l'enchaînement des actions et décisions au sein d'une activité.

G. **Diagramme d'états : [3]** Il montre les différents états et transitions possibles des objets d'une classe.

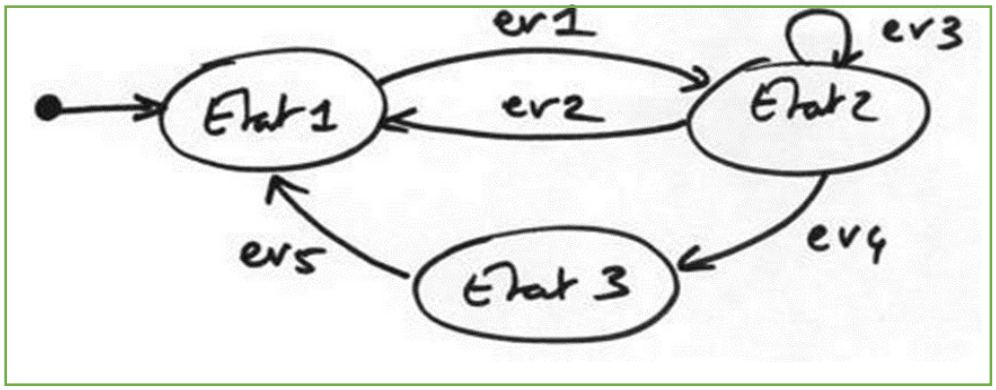

Figure 5 : Diagramme d'états

Le diagramme d'états représente le cycle de vie commun aux objets d'une même classe. Ce diagramme complète la connaissance des classes en analyse et en conception en montrant les différents états et transitions possibles des objets d'une classe à l'exécution.

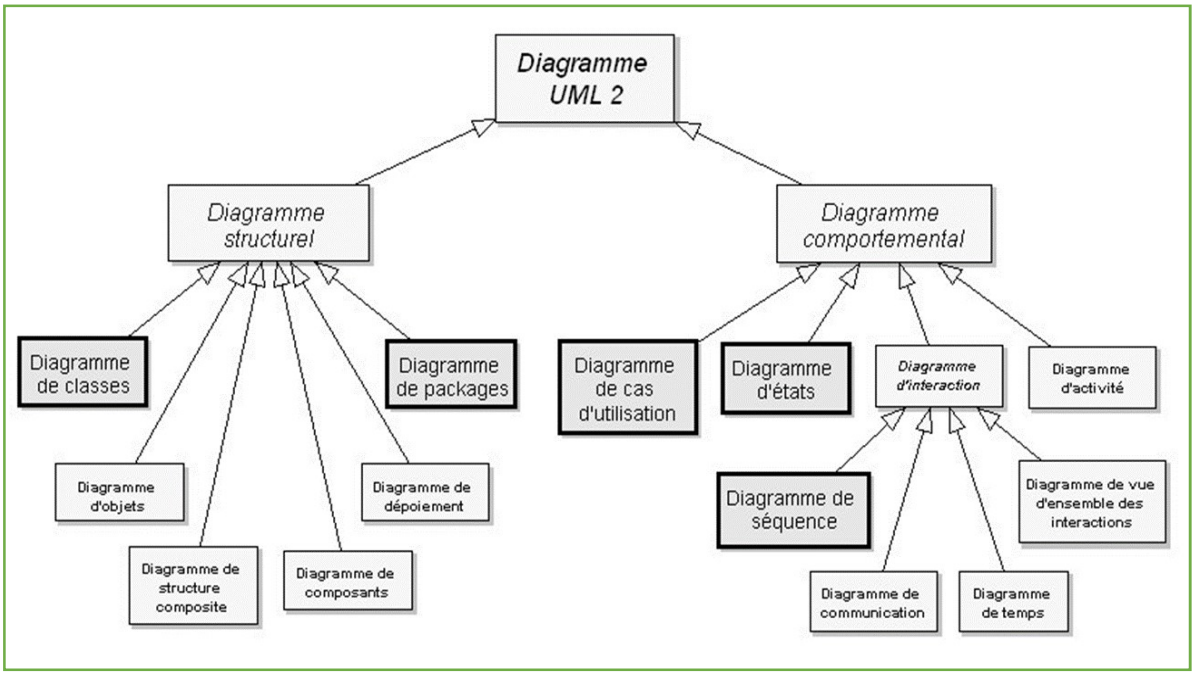

Figure 6 : Les diagrammes UML utilisés pour la modélisation des applications

10. Le processus simplifié pour les applications web

Un processus définit une séquence d'étapes, partiellement ordonnées, qui concourent à l'obtention d'un système logiciel ou à l'évolution d'un système existant.

L'objet d'un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles.

Un processus doit permettre de répondre à la question fondamentale : « Qui fait quoi et quand ? ».

Le processus à suivre pour le développement d'applications web se situe à mi-chemin entre Le processus UP (Unified Process) et la méthode agile.

10.1. Le processus unifié (UP)

C'est un processus de développement logiciel « itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation et piloté par les risques »

- **Itératif et incrémental** : le projet est découpé en itérations de courte durée qui aident à mieux suivre l'avancement global. À la fin de chaque itération, une partie exécutable du système final est produite, de façon incrémentale.
- **Centré sur l'architecture** : tout système complexe doit être décomposé en parties modulaires afin de garantir une maintenance et une évolution faciles. Cette architecture (fonctionnelle, logique, matérielle, etc.) doit être modélisée en UML
- **Piloté par les risques** : les risques majeurs du projet doivent être identifiés au plutôt, mais sur tout levés le plus rapidement possible. Les mesures à prendre dans ce cadre déterminent l'ordre des itérations.
- **Conduit par les cas d'utilisation** : le projet est mené en tenant compte des besoins et des exigences des utilisateurs. Les cas d'utilisation du futur système sont identifiés, décrits avec précision et priorisés.

10.2. Les phases du processus unifié

Le processus est organisé suivant les quatre phases suivantes : initialisation, élaboration, construction et transition.

10.2.1. La phase d'initialisation

Cette phase conduit à définir la « vision » du projet, sa portée, sa faisabilité, son business case, afin de pouvoir décider au mieux de sa poursuite ou de son arrêt.

10.2.2. La phase d'élaboration

Cette phase poursuit trois objectifs principaux en parallèle :

a) Identifier et décrire la majeure partie des besoins des utilisateurs.
b) Construire l'architecture de base du système.
c) lever les risques majeurs du projet.

10.2.3. La phase de construction

Consiste surtout à concevoir et implémenter l'ensemble des éléments opérationnels

10.2.4. La phase de transition

Permet de faire passer le système informatique des mains des développeurs à celles des utilisateurs finaux. Les mots-clés sont : conversion des données, formation des utilisateurs, déploiement, bêta-tests.

Le résultat de chaque phase est un système testé, intégré et exécutable.

11.Les principes de la méthode Agile

- **« Personnes et interactions plutôt que processus et outils » :**

Dans l'optique agile, l'équipe est bien plus importante que les moyens matériels ou les procédures.

- **« Logiciel fonctionnel plutôt que documentation complète » :**
  Il est vital que l'application fonctionne. Le reste, et notamment la documentation technique, est secondaire, même si une documentation succincte et précise est utile comme moyen de communication.
- **« Collaboration avec le client plutôt que négociation du contrat »** :
  Le client doit être impliqué dans le développement. On ne peut se contenter de négocier un contrat au début du projet, puis de négliger les demandes du client.
- **« Réagir au changement plutôt que suivre un plan » :**
  La planification initiale et la structure du logiciel doivent être flexibles afin de permettre l'évolution de la demande du client

12.La démarche de modélisation d'une application web

a) Les besoins donnent lieu aux cas d'utilisation et à une maquette
b) Une maquette est un produit jetable donnant aux utilisateurs une vue concrète mais non définitive de la future interface de l'application. Cela peut consister en un ensemble de dessins réalisés avec des outils spécialisés tels que Dreamweaver, Adobe Illustrator ou plus simplement avec Powerpoint ou même Word.
c) La maquette intégrera des fonctionnalités de navigation pour que l'utilisateur puisse tester l'enchaînement des écrans, même si les fonctionnalités restent fictives.
d) Les diagrammes de classes de conception représentent bien la structure statique du code, par le biais des attributs et des relations entre classes.
e) Les diagrammes d'interaction nous aident à attribuer les responsabilités aux classes.
f) Chaque cas d'utilisation est décrit textuellement de façon détaillée, mais donne également lieu à un diagramme de séquence simple représentant graphiquement la chronologie des interactions entre les acteurs et le système.
g) Les diagrammes de classes participantes décrivent, cas d'utilisation par cas d'utilisation, les trois principales classes d'analyse et leurs relations : les dialogues, les contrôles et les entités.
h) Les classes qui permettent les interactions entre le site web et ses utilisateurs sont qualifiées de « dialogues ». C'est typiquement les écrans proposés à l'utilisateur : les formulaires de saisie, les résultats de recherche, etc. Elles proviennent directement de l'analyse de la maquette.

i) Les classes qui contiennent la cinématique de l'application seront appelées « contrôles ». Elles font la transition entre les dialogues et les classes métier, en permettant aux écrans de manipuler des informations détenues par un ou plusieurs objet(s) métier.
j) Les classes qui représentent les règles métier sont qualifiées d'«entités ». Elles proviennent directement du modèle du domaine, mais sont confirmées et complétées cas d'utilisation par cas d'utilisation.

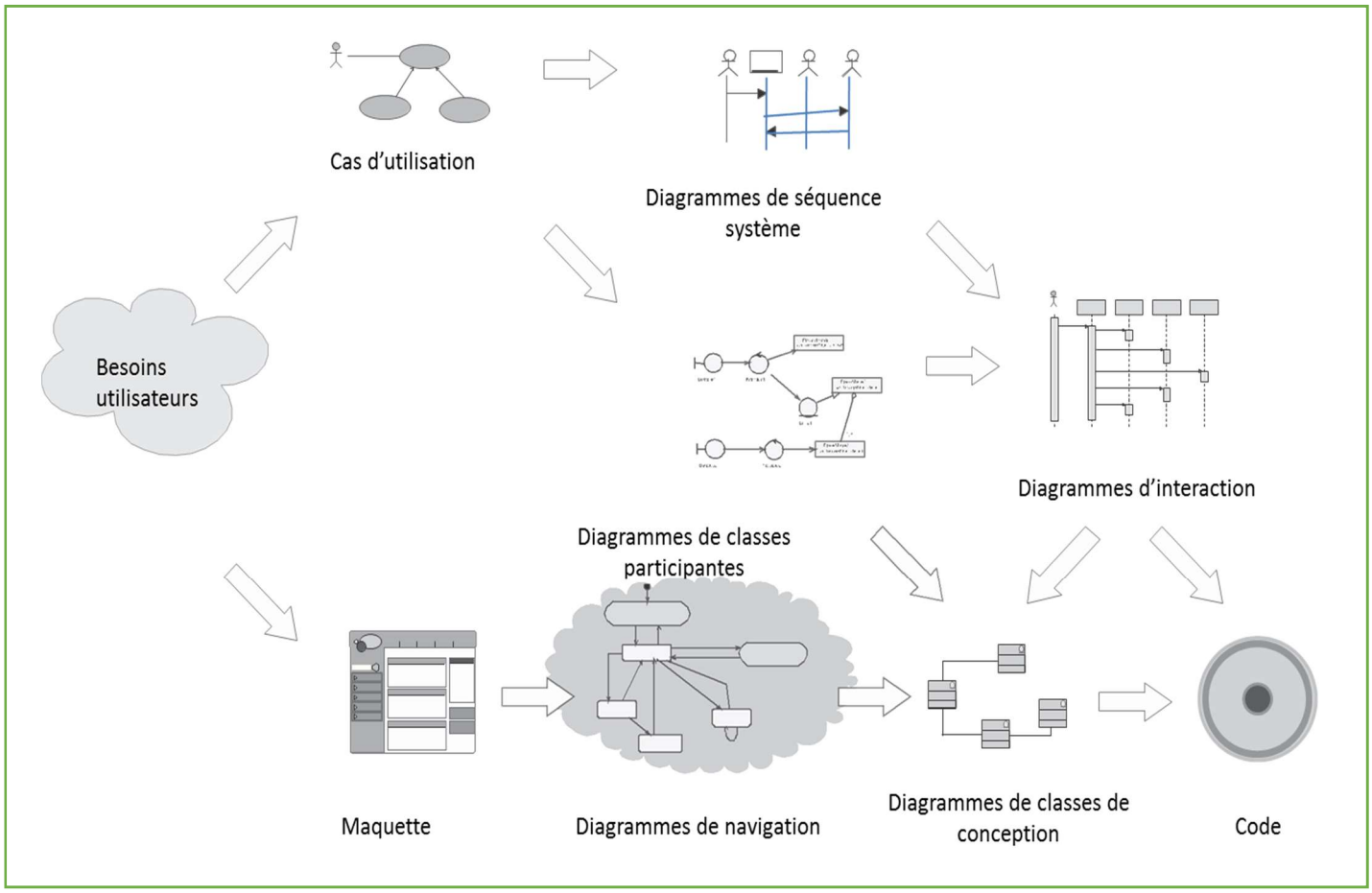

Figure 7 : La démarche de modélisation d'une application web

13. Conclusion

Ce chapitre cite les différentes étapes à suivre pour la modélisation d'une application web. C'est étapes seront détaillées durant les séances des travaux dirigés

Références

[1] Conallen, J. (2000). Building web application with UML. Eyrolles.

[2] Cours 8 dveloppement d'applications Web.. (2017). Récupéré sur Institute of management and career courses : https://www.coursehero.com/file/36408667/8-applis-webpdf/

[3] Roques, P. (2002). UML2 Modéliser une application web. Eyrolles.

[4] What is Unified Modeling Language (UML)? (2004). Récupéré sur Visual Paradigm : https://www.visual-paradigm.com/guide/uml-u