Préparation d’un Projet Web

Introduction

La préparation d’un projet web est une étape cruciale qui consiste à définir les objectifs, recueillir les besoins, analyser et modéliser le système. Cela garantit un déroulement efficace du projet, respectant les attentes, les délais et le budget tout en optimisant les ressources disponibles.

Concepts Fondamentaux de la Programmation

1. Encapsulation

Regroupe les données et méthodes dans une classe, protégeant les attributs avec des modificateurs d'accès (*private*, *protected*, *public*).

2. Héritage

Permet à une classe d'étendre les fonctionnalités d'une autre, favorisant la réutilisation du code et l'organisation hiérarchique

3. Polymorphisme

Permet à un objet d'exécuter différentes méthodes via une référence parent, avec surcharge (statique) ou redéfinition (dynamique)

Cycle de Vie d’un Projet :

Un projet suit plusieurs étapes depuis sa conception jusqu'à sa livraison finale. Les cycles de vie peuvent varier selon les méthodologies adoptées :

1. Cycle en Cascade

Modèle linéaire où chaque phase (besoins, conception, développement, tests, déploiement, maintenance) doit être validée avant de passer à la suivante.

2. Cycle en V

Chaque phase de développement a un test associé, renforçant la validation à chaque étape.

3. Cycle en W

Similaire au cycle en V, il ajoute des validations intermédiaires pour mieux gérer les risques.

4. Cycle en Y

Sépare le développement logiciel et matériel, qui convergent lors de l'intégration.

Méthodes de Gestion de Projet

1. Extreme Programming (XP)

Méthode agile avec des cycles courts, des tests automatisés et une grande adaptabilité aux changements.

2. Unified Process (UP)

Processus itératif divisé en phases : Inception, Élaboration, Construction, Transition.

3. Rational Unified Process (RUP)

Version structurée de UP, mettant l’accent sur la gestion des risques et les tests continus.

4. Two Tracks Unified Process (2TUP)

Distingue le développement technique de la modélisation fonctionnelle.

La Modélisation

La modélisation est un processus qui représente visuellement un système ou projet pour en simplifier la structure, le fonctionnement et les interactions. Dans le développement logiciel, elle clarifie les besoins et prépare le projet pour sa réalisation. L'UML (Unified Modeling Language) est un outil largement utilisé pour modéliser les systèmes complexes.

Diagramme de Contexte

Le diagramme de contexte représente globalement le système dans son environnement. Il montre le système comme une entité centrale et les acteurs externes avec lesquels il interagit.

- **Système** : représenté par une boîte au centre du diagramme.
- **Acteurs** : entités externes qui interagissent avec le système (personnes, systèmes ou organisations).
- **Interactions** : liaisons entre le système et les acteurs, représentant les échanges de données ou de services.

Diagramme de Cas d'Utilisation

Ce diagramme montre en détail comment les acteurs interagissent avec les fonctions spécifiques du système, appelées cas d'utilisation. Chaque cas d'utilisation représente une action distincte que le système permet à l'acteur de réaliser.

- **Relations entre acteurs et cas d'utilisation** : Un acteur peut interagir avec un ou plusieurs cas d'utilisation. Par exemple, un utilisateur peut accéder à la fonction "acheter un produit" ou "consulter un catalogue".
- **Relations entre acteurs** : Certains acteurs peuvent avoir des interactions entre eux, comme un client et un vendeur échangent des informations pour compléter une transaction.
- **Relations entre cas d'utilisation** :
    - **Inclusion ("include")** : Un cas d'utilisation inclut un autre cas, signifiant qu'une action spécifique doit toujours être exécutée dans un processus plus large. Exemple : "payer" est inclus dans "acheter un produit".
    - **Extension ("extend")** : Un cas d'utilisation est optionnel et ne s'exécute que sous certaines conditions, comme "appliquer un coupon" lors d'un achat.
    - **Généralisation ("héritage")** : Un cas d'utilisation général peut être subdivisé en sous-cas plus spécifiques, héritant des caractéristiques générales mais ayant des comportements distincts.

Description Textuelle :

La description textuelle des cas d'utilisation complète le diagramme de cas d'utilisation en fournissant un texte explicatif des étapes et interactions possibles. Elle se divise en trois parties principales : identification, scénarios et exigences non fonctionnelles.

- **Identification :**
    - **Nom :** Un nom clair, en utilisant un verbe à l'infinitif pour refléter l'action principale (ex. : "Consulter le solde de compte").

- ❖ **Objectif :** Décrit en quelques phrases l'objectif principal du cas d'utilisation.

- ❖ **Acteurs :** Liste les acteurs principaux (ceux qui interagissent directement) et les acteurs secondaires (ceux qui fournissent des informations ou soutiennent le processus sans interagir directement).

- ❖ **Gestion :** Informations administratives comme la date de création, les mises à jour, le responsable du cas d'utilisation, et le numéro de version.

- **Scénarios :**
  - ❖ **Pré-conditions :** Conditions nécessaires avant le début du cas d'utilisation. Elles posent le contexte de démarrage (ex. : l'utilisateur doit être connecté).

  - ❖ **Scénario Nominal :** Ce scénario principal décrit les étapes standard de l'interaction, sans erreur ou déviation.

  - ❖ **Scénarios Alternatifs :** Définissent des variations du parcours nominal pour répondre à des cas particuliers ou des choix alternatifs.

  - ❖ **Scénarios d'Exception :** Précisent les erreurs ou échecs qui peuvent survenir durant l'utilisation, et les actions de récupération ou de gestion de l'erreur (ex. : une tentative de connexion échoue en raison d'un mot de passe incorrect).

- **Exigences Non Fonctionnelles :** Elles incluent les spécifications additionnelles pour le cas d'utilisation, telles que des exigences de performance, sécurité ou ergonomie, spécifiques à ce cas d'utilisation.

Diagramme de classe :

Le diagramme de classes est la représentation statique du système. Il est composé de classes qui contiennent des attributs (propriétés) et des méthodes (comportements), ainsi que de relations entre les classes.

1. **Structure d'une Classe :**

- **Nom de la classe :** En général, commence par une majuscule (ex. : "Compte", "Utilisateur").

- **Attributs :** Propriétés de la classe, qui stockent les données spécifiques à chaque instance. Les attributs incluent des informations comme la visibilité (public, privé, protégé) et peuvent être calculés ou dérivés dans certains cas.

- **Méthodes :** Les actions ou fonctions associées à la classe, qui permettent de manipuler les attributs ou de réaliser des opérations spécifiques (ex. : calculerSolde(), validerConnexion()).

- **Visibilité et Encapsulation :**
  - ❖ **Encapsulation :** Protége les données d'une classe en limitant leur accès aux méthodes publiques (interface de la classe), en cachant les attributs (données privées ou protégées).

- **Visibilités :**
  Public (+) : Accessible depuis n'importe où.
  Protected (#) : Accessible uniquement aux sous-classes.
  Private (-) : Accessible uniquement dans la classe elle-même.
  Package (~) : Visible uniquement par les classes du même package.

2. **Relations entre Classes :**

- **Association :** Une relation entre deux classes montrant qu'elles interagissent (ex. : une classe "Compte" associée à "Utilisateur").

- **Agrégation :** Relation "partie-tout" non exclusive, où une classe contient une autre classe sans en dépendre entièrement (ex. : "Bibliothèque" et "Livre").

- **Composition :** Une relation "partie-tout" forte, où une classe ne peut exister sans la classe qui la contient (ex. : "Voiture" et "Moteur").

- **Héritage :** Relation de généralisation où une classe enfant hérite des attributs et méthodes de la classe parent.

Diagramme de séquence :

Un diagramme de séquence illustre comment les objets interagissent dans un scénario spécifique au fil du temps. Il montre l'ordre des messages échangés entre les objets pour réaliser une fonction ou un processus.

1. **Caractéristiques**

- **Ordre temporel :** Les messages sont disposés de haut en bas, ce qui indique l'ordre dans lequel les interactions se produisent.

- **Conditions et Boucles :** Les diagrammes de séquence peuvent inclure des conditions (sous-branches) et des boucles (répétition de messages) pour représenter des logiques complexes.

2. **Utilisation**

- **Description des scénarios :** Utilisé pour décrire des scénarios d'utilisation spécifiques, illustrant les interactions nécessaires pour atteindre un objectif.

- **Analyse des interactions :** Aide à comprendre comment les objets collaborent et quelles méthodes sont appelées dans un ordre spécifique.

- **Documentation :** Sert de documentation visuelle pour les développeurs et les parties prenantes, facilitant la communication sur la conception et le comportement du système.

3. **Exemple**

Dans un scénario de commande en ligne, un diagramme de séquence peut montrer les interactions entre un utilisateur, un système de paiement, et un système de gestion des commandes, illustrant les étapes depuis la création de la commande jusqu'à la confirmation du paiement.

Diagramme d'Activités :

Le diagramme d'activités est une vue comportementale qui montre le flux des processus ou des actions dans le système. Il détaille l'enchaînement des étapes et la logique de contrôle entre les actions.

1. **Éléments du Diagramme d'Activités :**
   - **Activités :** Chaque activité représente une action ou une étape spécifique dans le processus (ex. : "Vérifier Identité", "Calculer Total").
   - **Transitions :** Des flèches entre les activités indiquant le flux de contrôle, montrant quelle action suit une autre.
   - **Nœuds de Décision :** Ils permettent des conditions logiques qui divisent le flux en plusieurs chemins (ex. : condition "Solde Suffisant ?" avec deux sorties, "Oui" et "Non").
   - **Fork et Join :** Gèrent les processus parallèles. Le Fork divise le flux en plusieurs branches parallèles, et le Join les rassemble.
   - **Début et Fin :** Représentés respectivement par des cercles noirs pour le point de départ et un cercle encerclé pour le point final.
2. **Scénarios Alternatifs et Boucles :**

Les diagrammes d'activités peuvent inclure des boucles pour des processus répétitifs et des scénarios alternatifs pour traiter des choix différents dans le parcours d'activité.

- **Scénarios Alternatifs**
  - **Définition :** Représentent des choix différents dans un processus.
  - **Illustration :** Utilisent des losanges pour les points de décision. Chaque branche correspond à une option, comme "Oui" ou "Non".
  - **Exemple :** Dans un achat en ligne, le flux peut diverger selon que le paiement est accepté ou rejeté.
- **Boucles**
  - **Définition :** Représentent des actions répétitives jusqu'à ce qu'une condition soit remplie.
  - **Illustration :** Montées par une flèche retournant à une activité antérieure ou par des symboles de répétition.
  - **Exemple :** Un processus de validation où l'utilisateur corrige des erreurs jusqu'à ce que les données soient valides.