UNIFIED MODELING LANGUAGE (UML)

Approche de la décomposition fonctionnelle: Methode Merise

La méthode Merise est une méthode d'analyse et de conception de bases de données. Elle utilise une approche de la décomposition fonctionnelle pour diviser un système en sous-systèmes fonctionnels, et ainsi faciliter sa modélisation.

Demarche

La méthode Merise se compose de plusieurs étapes, notamment :

1. **L'analyse des besoins** : Il s'agit de l'étape de collecte des besoins des utilisateurs, des objectifs du système et des contraintes techniques et fonctionnelles.
2. **La spécification des exigences** : Il s'agit de l'étape de formalisation des besoins en termes de fonctionnalités attendues, d'entités impliquées et de relations entre ces entités.
3. **La conception préliminaire** : Il s'agit de l'étape de définition des sous-systèmes fonctionnels et de leur interdépendance.
4. **La conception détaillée** : Il s'agit de l'étape de conception détaillée de chaque sous-système, en utilisant des diagrammes de flux de données, des diagrammes entité-association, etc.
5. La mise en œuvre : Il s'agit de l'étape de réalisation effective du système, en utilisant un langage de programmation et un système de gestion de base de données.

Avantages

Les avantages de la méthode Merise sont nombreux :

1. Elle permet une **modélisation** claire et précise du système, en utilisant des diagrammes et des représentations graphiques.
2. Elle facilite la **communication** entre les parties prenantes, en utilisant un langage commun pour décrire les besoins et les fonctionnalités attendues.
3. Elle permet une **conception modulaire** et **évolutive** du système, en divisant le système en sous-systèmes fonctionnels.
4. Elle permet une **validation** précoce des choix de conception, en identifiant les erreurs et les incohérences dès les premières étapes.

Inconvenients

Cependant, la méthode Merise présente également certains inconvénients :

1. Elle peut être **coûteuse** et **chronophage**, en raison du grand nombre d'étapes et de documents requis pour la conception complète d'un système.
2. Elle peut être **rigide** et **difficile à adapter à des changements de besoins**, car elle repose sur une analyse préalable exhaustive.
3. Elle peut nécessiter une expertise **technique** et **fonctionnelle** importante pour être mise en œuvre efficacement.

Approche orientée objet avec UML :

L'approche orientée objet est un paradigme de programmation qui modélise les systèmes informatiques en utilisant des objets qui ont des caractéristiques (attributs) et des comportements (méthodes). UML (Unified Modeling Language) est un langage de modélisation graphique standardisé utilisé pour concevoir et documenter des systèmes orientés objet.

Démarche :

La démarche de l'approche orientée objet avec UML comprend les étapes suivantes :

1. Analyse des besoins : Collecte des exigences du système et identification des objets clés impliqués.

2. Modélisation des classes : Définition des classes qui représentent les objets et leur structure interne, y compris les attributs et les méthodes.

3. Modélisation des relations : Définition des relations entre les classes, telles que l'héritage, l'agrégation, l'association, etc.

4. Modélisation du comportement : Définition des interactions et du flux de contrôle entre les objets en utilisant des diagrammes de séquence, des diagrammes d'activité, etc.

5. Modélisation de l'architecture : Définition de l'architecture logicielle globale du système en utilisant des diagrammes de composants, des diagrammes de déploiement, etc.

Avantages:

Avantages de l'approche orientée objet avec UML :

1. Abstraction et modularité : Permet de modéliser des systèmes complexes en utilisant des objets abstraits et en les regroupant en modules réutilisables.

2. Réutilisabilité : Les classes et les objets peuvent être réutilisés dans différents projets, ce qui accélère le développement logiciel.

3. Encapsulation : Les objets encapsulent à la fois les données et les méthodes, ce qui permet de maintenir l'intégrité des données et de cacher les détails internes.

4. Flexibilité et extensibilité : Permet des modifications et des ajouts faciles grâce à l'héritage, la polymorphisme et la composition.

5. Communication efficace : UML fournit des diagrammes graphiques qui facilitent la communication entre les parties prenantes techniques et non techniques.

Inconvénients:

Inconvénients de l'approche orientée objet avec UML :

1. Complexité : L'approche orientée objet et UML peuvent être complexes à comprendre et à maîtriser, surtout pour les débutants.

2. Surdimensionnement : Pour les petits projets, l'utilisation complète de l'approche orientée objet et UML peut être excessive et entraîner un surdimensionnement.

3. Temps et coût : La modélisation complète d'un système avec UML peut nécessiter du temps et des ressources considérables.

4. Surcharge de documentation : La méthode UML encourage la création de nombreux documents et diagrammes, ce qui peut entraîner une surcharge de documentation.

Etude comparative entre la méthode Merise et UML

Approche 4 vues

UML adopte une approche en 4 vues pour la modélisation des systèmes. Ces vues sont :

1. Vue logique :
Cette vue se concentre sur la modélisation des classes, des objets, des relations et des comportements. Les diagrammes de classes, de séquence, d'activité, d'états et de communications font partie de cette vue.

2. Vue des processus :
Cette vue se concentre sur la modélisation des processus métier. Les diagrammes de flux de données, de flux de contrôle et de flux de traitements font partie de cette vue.

3. Vue de composants :
Cette vue se concentre sur la modélisation des composants logiciels et de leurs relations. Les diagrammes de composants, de déploiement et de packages font partie de cette vue.

4. Vue de déploiement :
Cette vue se concentre sur la modélisation de la configuration matérielle et logicielle sur laquelle le système est déployé. Les diagrammes de déploiement font partie de cette vue.

5. Vue des besoins :
Cette vue se concentre sur la modélisation des besoins des utilisateurs et des stakeholders. Les cas d'utilisation, les diagrammes de séquences d'utilisation, les diagrammes d'activité d'utilisation et les diagrammes de classes d'analyse font partie de cette vue.

Chacune de ces vues fournit une perspective unique sur le système, et la combinaison de toutes les vues permet une modélisation complète du système.

Les 13 diagrammes UML

1. Diagramme de cas d'utilisation : Modélise les cas d'utilisation du système.

2. Diagramme de classes : Modélise les classes, les attributs et les relations entre les classes.

3. Diagramme d'objets : Modélise les instances d'objets et leurs relations.

4. Diagramme de séquence : Modélise les interactions entre les objets dans une séquence chronologique.

5. Diagramme de collaboration : Modélise les interactions entre les objets dans un contexte particulier.

6. Diagramme d'états : Modélise les états et les transitions des objets.

7. Diagramme d'activité : Modélise les activités et les actions du système.

8. Diagramme de déploiement : Modélise la configuration matérielle et logicielle sur laquelle le système est déployé.

9. Diagramme de composants : Modélise les composants logiciels et leurs relations.

10. Diagramme de packages : Modélise les packages et leur structure.

11. Diagramme de profil : Modélise les profils UML personnalisés.

12. Diagramme de temps : Modélise les aspects temporels du système.

13. Diagramme d'interaction générale : Modélise les interactions entre les vues.

Classification des diagrammes par aspect et par vue

Les diagrammes UML peuvent être classés en fonction de leur aspect (fonctionnel ou architecture) et de leur vue.

En termes d'aspect, les diagrammes fonctionnels sont ceux qui modélisent les aspects du système qui sont liés aux processus métier, aux besoins des utilisateurs et aux interactions entre les acteurs. Les diagrammes d'architecture modélisent les aspects techniques du système, tels que la structure des classes, les composants logiciels, la configuration matérielle et logicielle, etc.

En termes de vue, les diagrammes UML peuvent être classés en fonction de la perspective qu'ils offrent sur le système. Voici une classification basée sur les vues principales :

1. Vue logique
• Diagramme de cas d'utilisation • Diagramme de classes • Diagramme d'objets • Diagramme de séquence • Diagramme de collaboration

2. Vue des processus
• Diagramme de cas d'utilisation • Diagramme de séquence • Diagramme de collaboration • Diagramme d'états • Diagramme d'activité • Diagramme de flux de données • Diagramme de flux de contrôle • Diagramme de flux de traitements

3. Vue de composants
• Diagramme de composants • Diagramme de déploiement • Diagramme de packages

4. Vue de déploiement
• Diagramme de déploiement

5. Vue des besoins
• Diagramme de cas d'utilisation • Diagramme de séquences d'utilisation • Diagramme d'activité d'utilisation • Diagramme de classes d'analyse

Il convient de noter que certains diagrammes peuvent être utilisés dans plusieurs vues pour représenter différents aspects du système. Par exemple, le diagramme de cas d'utilisation peut être utilisé dans toutes les vues pour représenter les fonctionnalités du système du point de vue des utilisateurs. De même, le diagramme de séquence peut être utilisé dans les vues logique et des processus pour modéliser les interactions entre les objets. La sélection des diagrammes à utiliser dépendra des besoins spécifiques du projet et de la vue nécessaire pour représenter les aspects pertinents du système.