Centre de Réadaptation de Mulhouse

CDA

CONCEPTEUR DEVELOPPEUR APPLICATION

Méthode de Développement Evénementiel

Dossier d'informations

Module 1

Séquence 3

1. Objectif de la démarche

Avoir une démarche d'analyse pour le développement d'un composant logiciel à réaliser dans un environnement de développement événementiel.

Obtenir une description détaillée de ce composant logiciel avant le codage.

Démarche :

- ✓ Réaliser le dialogue écran,
- ✓ Construire les menus si nécessaire,
- ✓ Construire les écrans (fenêtres) avec les objets du langage,
- ✓ Définir l'accès aux données,
- ✓ Etablir le tableau des objets/propriétés,
- ✓ Etablir le tableau des procédures événements,
- ✓ Présenter l'architecture des composants
- ✓ Spécifier les procédures événements (pseudo-code),

Les différentes étapes de la démarche sont formalisées dans un dossier de spécifications détaillées, à chaque étape un document est produit.

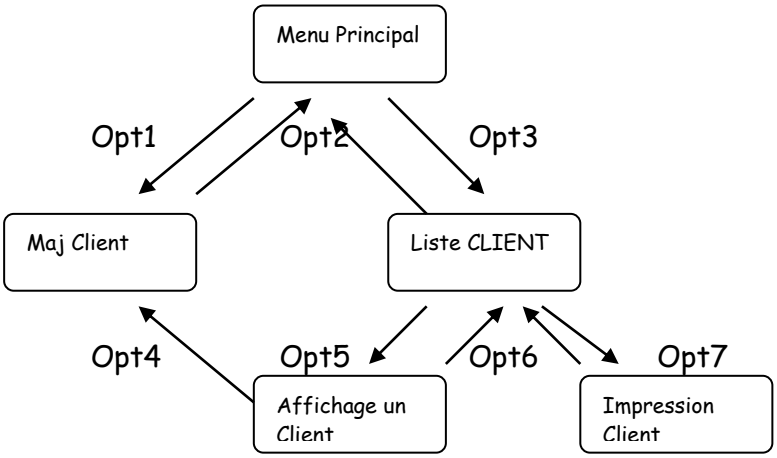
Suivre avec le canevas du dossier de spécifications détaillées les étapes de la démarche développées dans ce support.

2. Réaliser le dialogue Ecran

Avant d'implémenter les différentes fenêtres de l'application, il est nécessaire d'avoir une vue d'ensemble de l'enchaînement de celle-ci.

On utilise pour formaliser l'enchaînement un schéma sous forme de graphe que l'on appelle : Graphe du Dialogue Ecran.

Exemple de Graphe de Dialogue Ecran



On passe d'une fenêtre à une autre par une commande ou un contrôle.

Exemples de commande ou contrôle

- Un bouton de commande
- Une Option de menu

En analyse événementielle, sur chaque objet on déclenche différents événements. La liste de ces conditions/actions est utilisée dans la suite de la démarche avec le tableau des procédures événements.

3. Construire les menus

Certaines applications ont comme principe d'enchaînement l'utilisation d'option dans des menus.

Cette étape consiste à hiérarchiser ces options et à indiquer sous chaque option quelle partie de l'application est déclenchée.

Le graphe du dialogue écran fait apparaître les différents chemins ou parcours possibles pour une application.

Dans la suite de la démarche, on analyse chaque partie de l'application (une option de menu) indépendamment l'une de l'autre.

4. Construire les écrans (fenêtre)

A ce niveau de la démarche informatique les écrans ont été conçus dans le dossier d'analyse. Si le niveau de détail n'est pas suffisant, compléter ces maquettes avec tous les objets et contrôles à utiliser pour le bon fonctionnement de l'application. Les objets ou contrôles doivent être présents dans le langage événementiel utilisé en développement.

- ✓ Dessiner la version définitive de l'écran, en utilisant un support papier ou mieux encore en utilisant l'outil de développement.

  Voir le guide d'ergonomie Windows par exemple pour la conception générale des écrans ; couleurs, taille des boutons, type d'option, format des messages….

  On cherchera toujours à utiliser une ergonomie connue des utilisateurs.

- ✓ Définir le style des boites dialogue :

  Informations,
  Avertissements,
  Erreurs.

5. Définir l'accès aux données

Les langages de programmation événementielle proposent différentes méthodes d'accès aux données. Ces méthodes ont leurs propres caractéristiques et s'utilisent en fonction du type d'application à développer.

Lorsque l'application est un simple affichage de données, les langages fournissent généralement des assistants ou méthodes d'accès simplifiées ou pré-programmées.

Dans le cas d'application complexe on utilise plutôt les méthodes qui permettent une programmation sur mesure afin d'optimiser les accès.

Dans tous les cas de figure, une réflexion est à mener à chaque fois pour déterminer quelle est la meilleure méthode d'accès aux données dans une application voire dans une feuille ou une autre.

La documentation des langages donne une synthèse des accès aux données en fonction des besoins et des types d'application.

Dans le dossier de spécifications détaillées on précise par feuille ou fenêtre les données utilisées (base de données, table(s)) avec la méthode d'accès utilisée.

Dans le cas d'utilisation d'objets spécialisés pour l'accès aux données, on les décrit dans le tableau des objets/propriétés.

6. Etablir le tableau des Objets/Propriétés

Les langages de la programmation visuelle permettent d'implémenter des valeurs sur les propriétés des objets utilisés dans les fenêtres. Ces valeurs personnalisent l'objet en fonction des besoins ergonomiques.

Etablir le tableau des objets-propriétés consiste à reprendre chaque objet sur chaque fenêtre et de définir :

- ✓ Un nom à l'objet : suivre la codification définie au niveau application.

- ✓ Le type de l'objet :

  Exemples :
  Zone-texte, Bouton de Commande, Bouton-option, Boite-à-cocher, liste-déroulante, liste-simple, ascenseur,…

- ✓ Les valeurs de propriétés initiales :

  Couleur, visible/non visible, taille, …

7. Etablir le tableau des Procédures/Evénements

Un des problèmes de la programmation événementielle est le suivant :

Où placer le code programme qui effectuera les traitements ?

Les langages de programmation événementielle ont comme caractéristique de proposer pour chaque objet graphique utilisé dans une fenêtre des procédures associées aux événements possibles sur cet objet.

L'objectif de ce tableau est de déterminer quels sont les événements qui sont déclenchés sur les objets.

Dans le tableau on précise le nom de l'objet et le ou les événement(s) associé(s).

Exemples :

Sur un objet Bouton de commande : utilisation de la procédure CLICK.
Sur un objet Zone-texte : utilisation de la procédure CHANGE
Pour un objet fenêtre, utilisation de la procédure LOAD pour initialiser une valeur à des variables.

C'est dans ces procédures que les traitements sont définis.

8. Présenter l'architecture des composants

Par un schéma hiérarchique, l'objectif est de représenter, pour chaque feuille, les composants rattachés. Les composants peuvent être des procédures événementielles ou non. Les échanges de paramètres seront précisés.

9. Spécifier les procédures ou fonctions non événementielles

Une application événementielle est essentiellement composée de procédures événementielles (recensées dans le tableau précédent) et spécifiées plus loin
Comme pour un développement traditionnel, certains traitements peuvent être communs à plusieurs procédures événementielles, dans ce cas on peut mettre ces traitements en procédure ou fonction dans une partie du code de l'application accessible par l'ensemble des procédures événements

Documenter un composant de ce type consiste à :

- Indiquer son nom et le module qui le contient
- Donner une description générale
- Spécifier chaque composant (pseudo-code)

10. Spécifier les procédures événementielles

Généralités

Chaque procédure événement est un traitement associé à un objet. Dans cette étape il s'agit d'écrire le traitement sous forme de pseudo-code. Chaque procédure-événement sera un composant logiciel à part entière, avec les règles de la programmation structurée.

Dans une procédure-événement on utilise les structures classiques de la programmation structurée : répétitive (tant que , jusqu'à), alternative (si alors sinon), déclaration utilisation des variables

Spécificités programmation événementielle :

Les objets :

Une caractéristique de la programmation événementielle basée sur l'utilisation d'objet est de faire référence à ces objets dans la programmation soit :

- ✓ faire référence à une propriété d'un objet, en pseudo-code cela donne :

  Objet.propriété = …..

  Pour une valeur de propriété

- ✓ faire référence à une méthode (ou procédure) de cet objet, en pseudo-code

  …. Objet.méthode ….

  Une méthode peut être de deux types soit :

  1) Une fonction avec la possibilité de tester le retour

  Syntaxe :

  Variable = Objet.méthode(Param)

Exemple :

Booléen : Etat
Etat = Table.Ouvrir("LECTURE")

2)-Une procédure

Syntaxe :

Objet.méthode(param1,param2,……,paramX)

Exemple :

Fenêtre2.Afficher(P1,P2)

Les Variables

Portée des variables, principes :

Il y a deux types de variables : les globales et les locales

Une **variable globale** est définie au niveau de l'application et est connue ou visible pour l'ensemble des objets de l'application. En Visual Basic, on définit ces variables au niveau d'un module commun.

Une **variable locale** est définie au niveau d'une procédure et n'est connue ou visible que dans cette procédure.

Portée des objets :

On accède à un objet, propriétés et méthodes, de trois manières différentes :

- ✓ Directement à l'objet lui-même
  (En programmation objet on dit Self ou This pour nommer l'objet)

  Syntaxe :

  Self.propriété ou propreté = …..
  Self.méthode() ou méthode()

  Contexte de l'objet

- ✓ Accès aux objets d'un "conteneur"

  Syntaxe :

  Objet.Propriété = …..
  Objet.Méthode()

  Contexte du Conteneur

- ✓ Accès aux objets externes d'un "conteneur" ou objet parent

  Syntaxe :

  ObjetParent.Objet.Propriété = …..
  ObjetParent.Objet.Méthode()

  Ou

  Parent.Objet.Propriété = …..
  Parent.Objet.Méthode()

  Parent est un <mot réservé pour indiquer un objet parent

Les différents objets:

Faire la liste exhaustive de tous les objets possibles en programmation événementielle est impossible, chaque langage en a une multitude. On peut citer quelques objets principaux :

- Objets globaux
  - Application ou projet
  - Imprimante
  - Ecran
  - Module
- Objet d'interface ou de présentation
  - Contrôles
    - Champs
      - Libelle ou texte
      - Liste
      - ComboBox
    - Case
      - Option
      - A cocher
    - Bouton
      - Texte
      - Graphique
    - Image
    - Ascenseur
  - Fenêtre
    - SDI
    - MDI
  - Boite de dialogue standard
  - Menu
    - OptionMenu

Objet d'accès aux données

Table
Curseur
Requête
Transaction
Base de données

Les méthodes :

Elles sont nombreuses et une typologie est difficile.

En pseudo-code on prend le mot français le plus proche de l'action à effectuer par la méthode.

Exemple :

Pour une fenêtre

Ouvrir(), Fermer(), Cacher(), Montrer()

Il existe des noms identiques de méthodes pour des objets différents.

Exemple :

Objet Liste ou objet ComboBox

AjouterLigne(), SupprimerLigne

Objet Table, BasedeDonnées, Transaction

Ouvrir(), Fermer()

C'est le type de l'objet qui permet de les différencier.