Support de cours en systèmes d'informations, méthodes avancées: Conception

Dans le processus 2TUP, la conception du système est divisée en trois étapes: conception générique qui est indépendante du métier, la conception préliminaire qui adapte la conception générique aux besoins métier et la conception détaillée qui détaille la conception préliminaire. Ce document regroupe ces trois types de conception dans l'ordre de leur utilisation dans le processus 2TUP.

Conception générique

1. Introduction

La conception générique consiste à développer la spécification technique proposée durant la capture des besoins techniques tout en restant indépendant des besoins fonctionnels. Dans ce qui suit, nous décrivons les éléments de base nécessaires à la construction des modèles de la conception générique.

2. Classes, frameworks techniques et interfaces

La notion de classe technique décrit une classe UML dont la responsabilité est technique, par opposition aux classes fonctionnelles. Ces classes peuvent être réutilisées quel que soit le domaine ou l'étape du processus métier modélisé.

Exemples de classes techniques

- la classe Tuplet qui représente un enregistrement dans une table relationnelle
- la classe Fenêtre qui est liée à la présentation

La notion de framework découle du fait que généralement une classe technique n'est pas utilisée seule, mais dans un ensemble de classes. Ce regroupement de classes techniques est appelé framework. Associé à la notion de framework est la notion d'interface. Une interface définit un ensemble d'opérations abstraites qui définissent les services offerts par une classe ou par un composant.

Une interface est réalisée pour donner lieu à des classes concrètes. Un ensemble d'interfaces interdépendantes définit un framework abstrait. Par contre, un ensemble d'interfaces concrètes définit un framework concret.

3. Elaboration du modèle logique de conception

Le modèle logique de conception est construit à partir des classes techniques. Les éléments d'un diagramme de classe vus dans la première partie de ce cours reviennent ici (classes,

associations, agrégations, etc.) avec la différence qu'ici, l'intérêt porte sur les classes techniques.

Les classes techniques et frameworks sont définis selon le modèle de couches logicielles adopté durant l'étape de capture des besoins techniques. Si on reprend le modèle à 5 couches, on retrouve les couches suivantes avec les différents objets qui peuvent y figurer :

3.1. Couche présentation

Cette couche contient les objets (et donc les classes) nécessaires à la présentation. On y trouve les concepts fenêtre, panneau, bouton, …

3.2. Couche application

Cette couche contient les objets nécessaires au contrôle et pilotage de l'application. On y trouve les concepts de modèle, listes d'objets…

3.3. Couche métier

Dans cette couche, on retrouve les objets métier avec les règles de gestion associées. On y retrouve les concepts d'objet métier, objet composite …

3.4. Couche d'accès aux données

A ce niveau, on prévoit des objets qui gèrent l'accès aux données. On peut à ce niveau prévoir une classe technique qui récupère les données pour chaque classe métier comme l'employé, les diplômes, …et une classe Tuplet qui porte les valeurs d'une ligne d'une table physique.

3.5. Couche de stockage de données

La couche de stockage concerne le stockage physique de données. On y retrouve les classes Table et Clé étrangère …

Exemple : prenons l'exemple de gestion des droits de lecture/écriture des utilisateurs dans des tables du système. Chaque utilisateur peut avoir trois droits : lecture, écriture et lecture/écriture. On peut définir ce besoin à différents niveaux de spécification

Au niveau de la couche présentation : un panneau renseigne des caractéristiques de l'utilisateur et un autre recense les tables sous la forme d'une liste, avec pour chacune les droit associés. On aura donc le schéma ci-dessous.

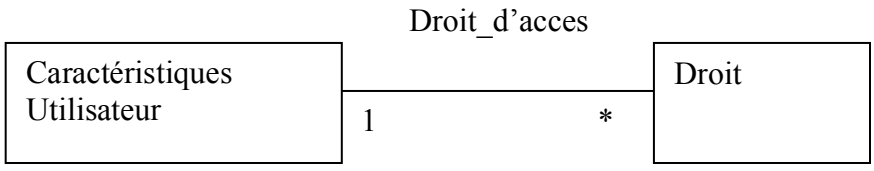

Figure. Exemple de classes techniques de gestion de droits d'utilisateurs

Par contre, au niveau de la couche de stockage, on trouvera trois tables relationnelles (si le choix physique porte sur le modèle relationnel): utilisateur, table_app et utilisateur_droit qui recense les droits de chaque utilisateur sur chaque table.

4. Conception dynamique d'un Framework

La notion de framework implique de décrire l'aspect statique composé des classes techniques. Cependant, il peut être nécessaire de représenter aussi la dynamique d'un framework. Cette dynamique peut être décrite en UML à l'aide des diagrammes de séquences ou de collaboration si le framework nécessite une synchronisation entre les classes, ou à l'aide de diagramme d'état s'il s'agit d'une seule classe concernée par les changements d'états.

5. Organisation du modèle logique de conception technique

Tout comme le modèle qui structure les classes métier en catégories, le modèle logique de conception générique structure les classes techniques en packages qui correspondent aux frameworks. Par ailleurs, les besoins de séparer les différentes responsabilités du logiciel en couches nécessitent d'avoir des frameworks dont les responsabilités de chacun sont liées à une seule couche logicielle. Toutefois, certains framework ont une transversalité par rapport aux couches logicielles, dans ce cas, ils seront liés aux autres frameworks des couches logicielles. **Exemple :** si l'on adopte toujours l'architecture à base de 5 couches logicielles, il en découle 5 frameworks techniques (présentation, application, métier, accès aux données, stockage de données) auxquels ont peut ajouter d'autres frameworks comme l'authentification, et la sauvegarde périodique de données. Le modèle d'organisation des frameworks est illustré comme suit.

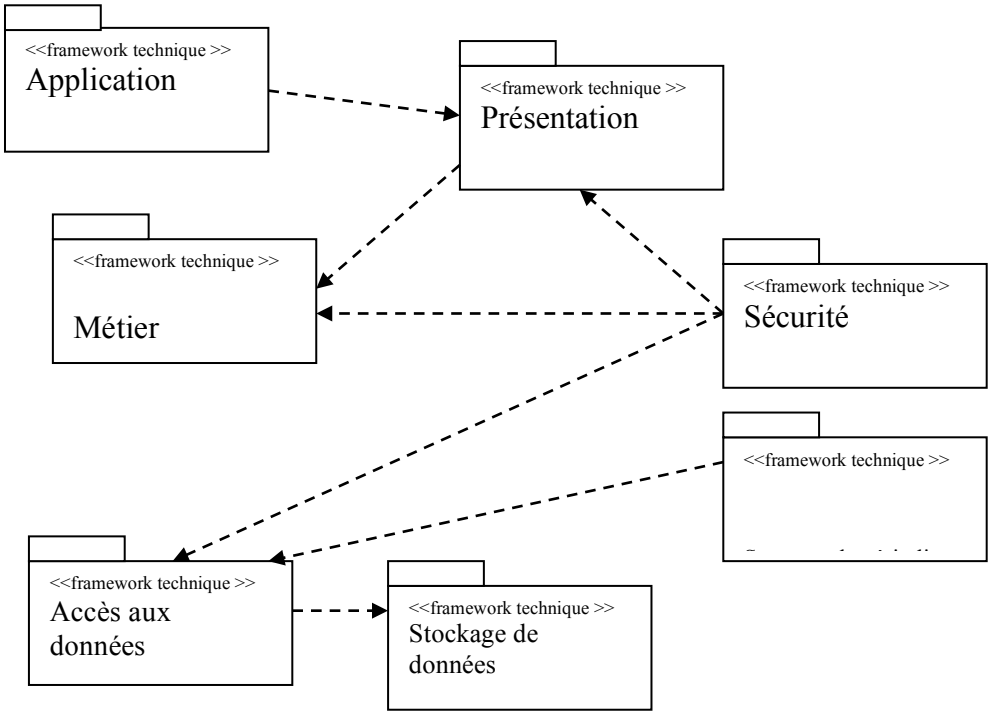

Figure. Exemple de modèle d'organisation de frameworks techniques

Les flèches en pointillé expriment des liens de dépendances. On remarque que l'utilisation du stéréotype <<framework technique>> et aussi les framework qui dépendent de plusieurs couches logicielles (frameworks « sécurité » et « sauvegarde »).

6. Elaboration du modèle d'exploitation de la conception technique

Le modèle d'exploitation reprend la notion de composant d'exploitation vue dans la spécification des besoins techniques, mais les détaille avec les composants nécessaires à configuration logicielle.

Pour reprendre l'exemple du système SGP, il s'agit d'inclure les composants nécessaires à la sécurisation de l'application et aux opérations de sauvegarde. Le cas de la sécurité nécessite de reposer sur un composant de base de données du même type que celui qui gère les données métier, tout en partageant le même composant qui représente le moteur de base de données.

Pour la sauvegarde, s'il l'on opte pour une sauvegarde sur un fichier externe, le composant d'exploitation « fichier » apparaitra. Nous aurons ce schéma illustratif des composants d'exploitation.

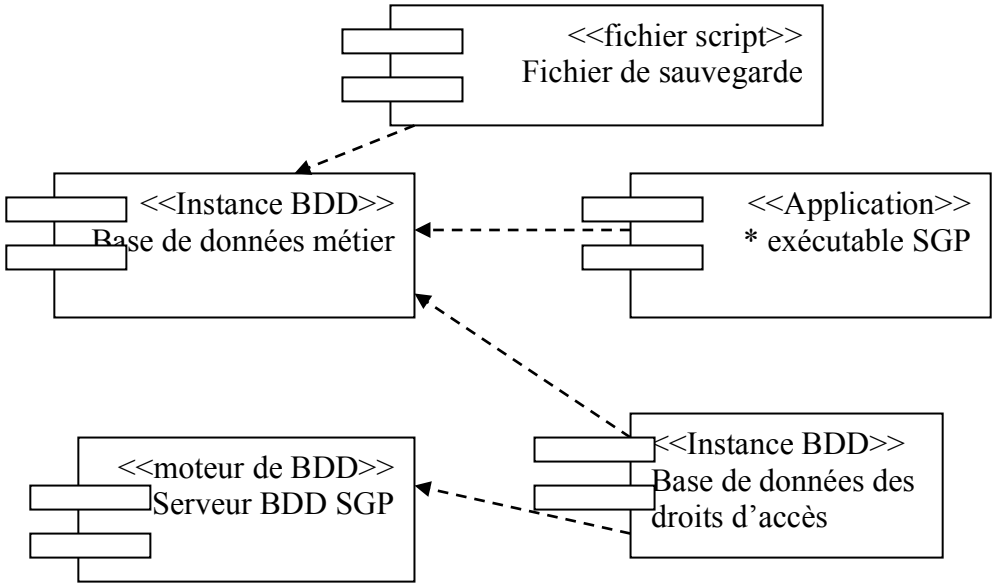

Figure. Exemple de composants d'exploitations

7. Conclusion

La conception générique est une étape importante dans laquelle les classes techniques du système sont développées, et ce, malgré que, de nos jours, les environnements de développement *clé en main* implémentent directement ces classes. La conception générique permet de détailler le modèle de spécification logicielle en détaillant les classes techniques qui serviront pour la conception des classes d'analyse. Ces classes sont alors organisées en paquetages appelés frameworks techniques correspondant à chaque couche du modèle logiciel choisi. En outre, des frameworks supplémentaires peuvent s'ajouter au modèle logiciel pour prendre en charge des aspects spécifiques du système tels que la sécurité, l'archivage, etc.

Conception préliminaire

1. Introduction

La conception préliminaire consiste à tenir compte des besoins exprimés dans la phase de capture des besoins fonctionnels dans la conception du logiciel. Il s'agit donc d'adapter la conception générique en termes de modèle de composants et de configuration logicielles aux métier et domaine du système à développer. La conception préliminaire est une première étape de la fusion qui sera suivie par le détail de conception plus tard (chapitre suivant). La conception préliminaire passe par les étapes décrites ci-après.

2. Concevoir le déploiement

Le modèle de déploiement décrit durant la spécification des besoins techniques ne définissait pas des nœuds spécifiques au domaine. En effet, à cette étape là, on introduit la notion de poste de travail. Un poste de travail représente un ou plusieurs acteurs localisés sur une machine de même type. Un poste de travail n'est pas réduit à une seule machine physique mais peut être matérialisé par plusieurs machines physiques à condition de répondre à la même configuration de déploiement. D'autre part, le modèle de déploiement peut contenir des nœuds qui ne correspondent pas nécessairement à des postes de travail au sens métier mais peuvent être considérés comme des postes de travail au sens technique. Des exemples de cela sont les nœuds représentant les serveurs et les navigateurs.

Exemple : dans le système SGP, dans un déploiement client serveur, on peut prévoir deux postes de travail qui correspondent au responsable de personnel et à l'agent administratif. Aussi, le nœud « serveur » permet de stocker et gérer les données.

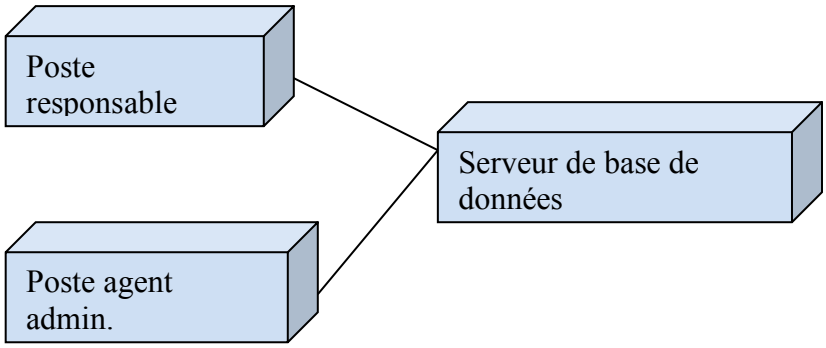

Figure. Exemple de diagramme de déploiement

3. Concevoir le modèle d'exploitation

Les modèles d'exploitation et de composants décrits jusque là permettent de spécifier, en général, les composants et leurs répartition sur les postes de travail. Parmi ces composants,

figurent les applications et les instances de bases de données. Hormis les composants techniques, les autres composants métier (application, instance de base de données métiers) n'ont pas été spécifiés. C'est à cette étape d'analyse préliminaire qu'on va le faire.

3.1. Découpage du système en applications

Le point de départ pour l'identification des applications est le regroupement des cas d'utilisation effectué durant l'étape de capture des besoins. Dans certains cas, un groupe de cas d'utilisation peut être affecté à un poste de travail. Ce poste de travail se verra assigner alors l'application qui réalise l'ensemble de ces cas d'utilisation. Par contre, on peut aussi trouver la situation où un cas d'utilisation est partagé entre plusieurs applications car exécuté par plusieurs acteurs sur des postes de travail différents.

Exemple : voici le diagramme des cas d'utilisation de gestion du personnel:

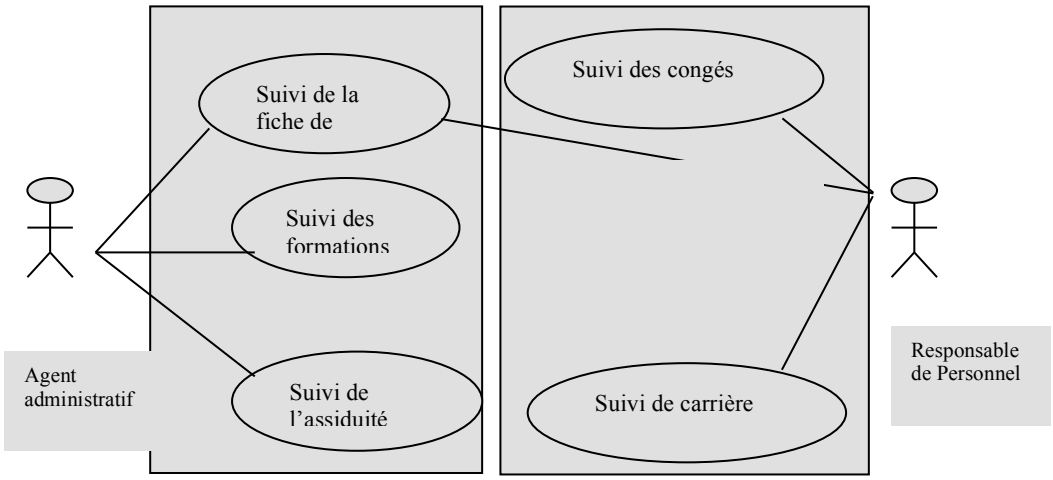

Figure 41. Exemple de découpage du système en applications

Remarquons qu'il est possible de regrouper les cas d'utilisation en deux groupes, répartis sur les postes de travail *responsable de personnel* et *agent administratif*. On peut dès à présent tracer le schéma de répartition des applications sur les postes de travail. On appelle la première application Gest_CC (carrière et congés) la deuxième Gest_AP (Affectations et positions). Ce schéma illustre la répartition des applications sur les postes de travail.

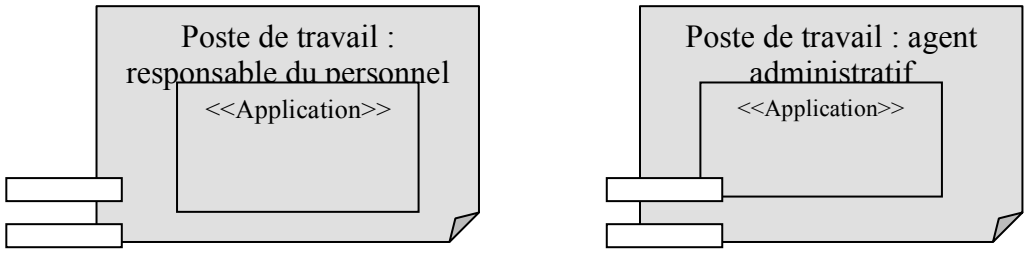

Figure. Exemple de postes de travail

3.2. Identification des composants distribués

Définition des composants : à ce niveau de conception, on s'intéresse aux composants métiers, issus du découpage en catégories. Puisque qu'une application reflète un regroupement de cas d'utilisation interdépendants, il en découle des classes regroupées en catégories. Par ailleurs, il est possible que certaines catégories soient partagées entre plusieurs applications. Par exemple, si on considère la catégorie Employé, on remarque qu'elle est

partagée entre les deux applications. Dans ce cas, les catégories qui sont partagées entre plusieurs applications donneront lieu à des composants partagés et distribués. En outre, à cette étape là on affine la définition du composant d'instance de base de données. Le détail de ce composant dépend de l'architecture.

Exemple : dans le cas du système SGP, on peut distinguer la catégorie Employé qui peut être utilisée qui est partagée entre les deux applications voire avec plusieurs applications qui seront centrées sur l'employé plus tard. De ce fait, on peut ressortir le composant Employé comme composant métier à part. Les autres catégories donneront lieu chacune à un composant métier. En ce qui concerne la base de données, l'architecture du système étant en 2 tiers, une seule instance de base de données suffira. Le schéma suivant illustre le modèle d'exploitation comportant des composants métiers. Ici, le composant Employé est partagé et est constitué d'une seule catégorie d'analyse. Par contre, les composants Gest_CC et Gest_AP ne sont pas partagés.

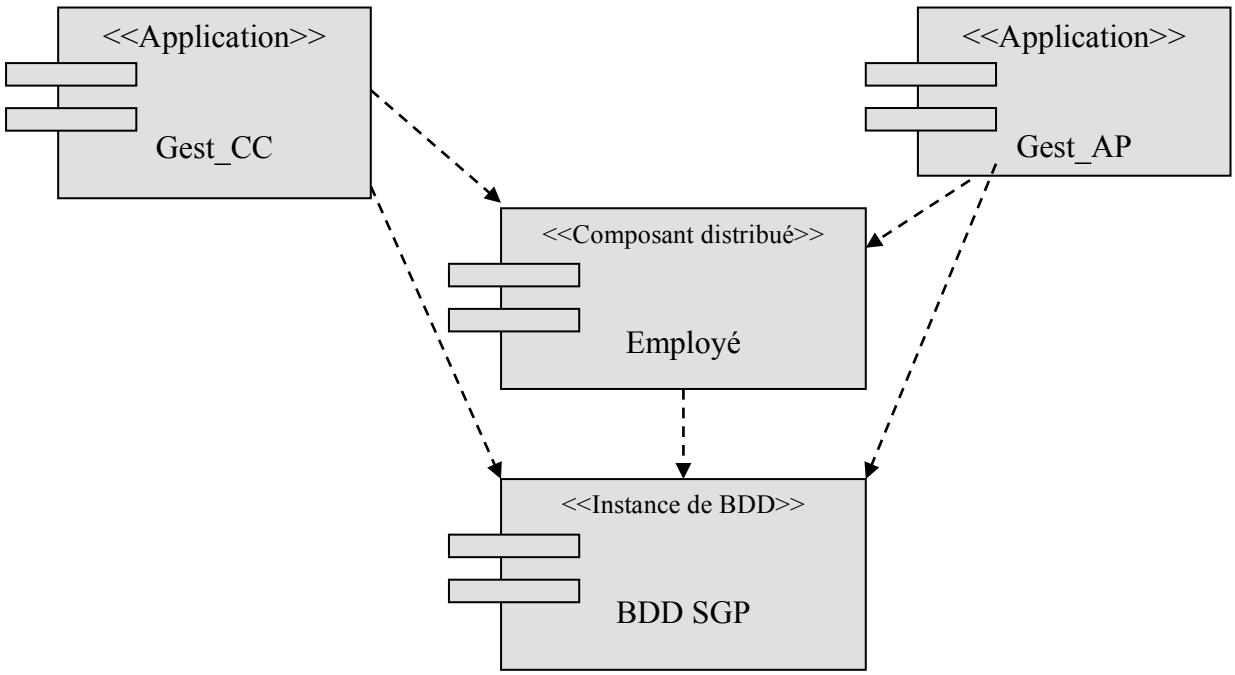

Figure 43. Exemple de composants distribués

L'étape suivante consiste à décrire les interfaces d'accès aux composants réutilisables. Ces interfaces sont identifiées à l'intérieur des catégories contenues dans chaque composant métier. Ainsi, chaque classe de catégorie d'analyse sera représentée par une interface. Chaque interface est décrite par ses responsabilités.

Exemple : dans le cas de SGP, le seul composant distribué étant Employé. Ce composant contient une catégorie qui contient pour sa part sept classes. Chaque classe donne lieu à une interface. Le tableau suivant décrit ces interfaces avec des noms précédés d'un « I » ainsi que leurs responsabilités.

Composant distribué	Interface	Responsabilité
Employé	IEmployé	Distribuer les informations générales sur les employés
	IType	Association des employés à leurs types
	IDiplôme	Recensement des diplômes
	IA_diplome	Connaître et distribuer les fonctions

Tableau. Exemple d'interfaces

3.3. Enumération des interfaces utilisateurs (IHM) des applications

Par interface, nous signifions les interfaces de présentation des applications par opposition aux interfaces objet. La définition des interfaces objets peut être prévue dès l'étape de capture de besoins fonctionnels et étape d'analyse, mais c'est à ce niveau là que les maquettes peuvent être dessinées et distribuées sur les applications. La définition des interfaces liées aux applications permet, plus tard, d'identifier les objets de la couche présentation. Les IHM peuvent être décrits textuellement dans un tableau comme illustré dans l'exemple suivant du système SGP pour l'application SGP_AP. La liste n'est pas exhaustive.

Vue IHM	Description
Edition Employé	Éditer les informations d'un employé. Création, modification, suppressions…
Sélection diplôme	Sélection d'un diplôme à assigner à un employé
Edition diplôme	Ajout d'un nouveau diplôme avec les différentes informations
Sélection Employé	Sélection d'un employé à partir d'une liste pour effectuer des traitements le concernant
Période	Sélection d'une date à partir d'un calendrier
Sélection fonction	Sélection d'une fonction
Edition fonction	Edition d'une formations : création, modification…
Sélection service	Sélection d'un service
Edition service	Edition d'un service : création, modification …

Tableau. Enumération d'interfaces homme-machine

4. Concevoir le modèle logique

A cette étape, nous développons le modèle logique de conception en tenant compte d'une part des catégories d'analyse, et d'autre part, des framework techniques et des composants d'exploitation. L'approche de conception du modèle logique consiste donc à procéder à un découpage en catégories de conception. Pour aboutir à ce découpage, on prend tour à tour chaque composant d'exploitation (applications, composants métier distribués, instance de

base de données) d'une part, et chaque catégorie d'analyse d'autre part et de la projeter sur les framework abstraits qui représentent les couches logicielles.
Naturellement, certains composants d'exploitation en seront concernés que par certaines couches logicielles. Par exemple, l'application SGP_CC est concernée par plusieurs catégories d'analyse (Employé, Période, Position, Carrière). Par contre, elle sera concernée par les couches Présentation et Application. Par contre, le composant d'exploitation Instance de base de données, il sera quant à lui concerné par les couches Métier, Accès aux données, et Stockage de données. A l'issue de ce croisement, il sera possible d'énumérer les catégories au niveau de chaque couche logicielle, avec leurs associations aux composants du modèle d'exploitation.

5. Concevoir les structures de la présentation

Cette étape ne fait pas partie d'UML car concerne les choix ergonomique en matière de présentation de chaque composant concerné par la couche présentation. Cette étape permet de déduire les classes correspondant aux vues de la couche présentation comme les panneaux, les boutons…

6. Organiser la configuration logicielle

La dernière étape de la conception préliminaire consiste à réorganiser le modèle logique de conception constitué de catégories de conceptions en sous-systèmes. Initialement, chaque catégorie de conception donne lieu à un sous-système à développer. Cependant, il est possible d'isoler certaines classes en sous-systèmes afin de permettre la réutilisation.

Exemple : la sélection d'employés est utilisée à chaque fois qu'on veut faire un traitement à un employé. Les classes nécessaires à cette sélection peuvent être isolées en sous-système. Aussi, les catégories d'une même catégorie d'analyse relevant des couches application et présentation sont souvent associées car généralement, ces deux couches ne sont pas isolées.

Un sous-système est modélisé sous la forme de paquetage UML. 

Exemple : dans le système SGP, on aura les sous-systèmes Employé, selection_employé, Période, Diplôme…

7. Conclusion

L'étape de conception préliminaire constitue le premier point de rencontre entre la branche fonctionnelle et la branche technique du processus 2TUP. Dans cette étape, le modèle de configuration matérielle donnera lieu au déploiement des postes de travail. Aussi, le modèle d'exploitation est concrétisé par l'identification des applications de découpage du système et les composants distribués. Les aspects d'interface sont également traités par l'énoncé des interfaces homme machine de chaque application.

Conception détaillée

1. Introduction

L'étape de conception détaillée est la dernière étape avant le codage. A cette étape, on commence à raffiner les choix de conception en incluant les spécificités du langage ou environnement de développement. Les étapes de conception détaillées sont présentées dans ce qui suit.

2. Concevoir le modèle logique

2.1. Concevoir les classes

La conception détaillée des classes est relativement systématique, car les concepts de l'orienté-objet trouvent leur équivalence au niveau des langages de programmation, comme Java. Cependant, certains concepts ne sont pas explicités dans certains langages de programmation.

2.2. Concevoir les associations

Lorsque le concept d'association n'est pas pris en charge par le langage de codage, il convient de procéder à la transformation de l'association. Cette transformation dépend du type et des multiplicités de l'association. **Exemple :** soit les classes représentées par le diagramme ci-dessous et la transformation qui lui est associés.

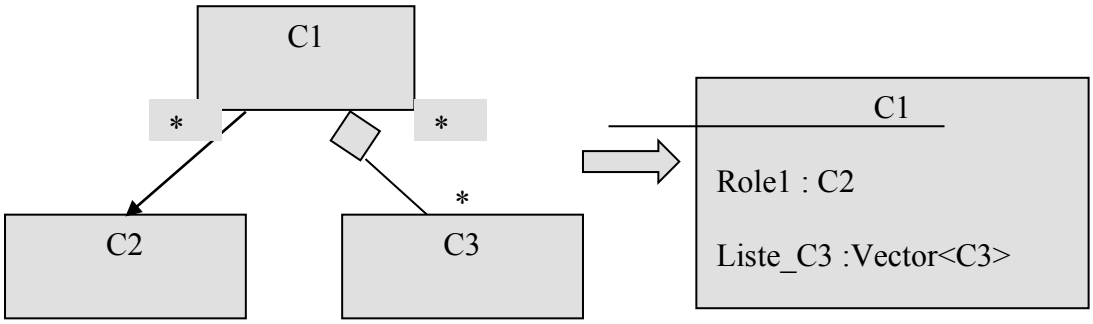

Figure. Exemple de transformation d'association

La conception des associations mène à penser à plusieurs situations représentées dans le diagramme et devant être conservées lors de la transformation de l'association. Parmi les éléments à gérer figurent :

- les multiplicités supérieures à 1 (attributs obligatoires)
- les contraintes *lecture seule* et d'*ordonnancement*
- la gestion des associations de type composition lors de la destruction de l'agrégat
- les associations bidirectionnelles donnent lieu chacune à la copie des attributs et des opérations de part et d'autre. La synchronisation entre les deux classes en termes de valeur est de fait à gérer.
- la classe d'association se transforme d'abord en deux associations puis chaque association sera gérée de sa part par les mêmes principes d'une association normale.
- transformer les relations n-aires en relations binaires.

2.3. Concevoir les attributs

La conception des attributs concerne les opérations suivantes

- Typage avec des types scalaires ou complexes
- Prise en compte des contraintes de type lecture seule ou lecture écriture
- Conservation de la visibilité de chaque attribut
- Ajout, si nécessaire, des méthodes d'accès en écriture ou lecture *set<nom d'attribut>* ou *get<nom d'attribut>*
- Spécifier les méthodes de mise à jour des attributs dérivés lorsque les attributs de base changent de valeur.

2.4. Concevoir les opérations

La conception des opérations consiste à détailler l'algorithme nécessaire à sa mise en œuvre.

3. Conception des couches logicielles

Les différentes couches logicielles vont subir une conception encore détaillée, que nous ne présentons pas dans ce cours. Toutefois, la couche de stockage mérite d'être étayée à ce niveau. En effet, le diagramme des classes métier est généralement mis en œuvre en relationnel, d'où la nécessité de procéder à une description des données en relationnel.

Le passage du modèle objet au relationnel suit un ensemble de règles illustrées dans le tableau suivant :

Modèle objet	Modèle relationnel
Classe	Table
Attribut de type simple	Colonne
Attribut de type composé	Colonnes ou clé étrangère

Instance	T-uplet
OID	Clé primaire
Association	Clé étrangère ou table de liens
Héritage	Clé primaire identique sur plusieurs tables

Tableau. Règles de passage du modèle d'objet au relationnel

4. Conclusion

La conception détaillée représente la dernière étape de conception qui précède le codage et les tests. C'est durant cette étape que le modèle statique sera détaillé. Il s'agit de concevoir les classes selon l'environnement de mise en œuvre (développement sous Java, par exemple) et d'affiner le contenu des classes en termes d'attributs et d'opérations.

Durant cette étape également, on procède à la traduction du diagramme de classes vers le modèle relationnel vu que les SGBD relationnels sont les plus utilisés actuellement.