Stratégies par couches

Peter Eeles

Livre blanc de Rational Software

TP 199, 08/01

Rational®
the software development company

Table des matières

Résumé

Un certain nombre de techniques existe, permettant de décomposer les systèmes logiciels. L'utilisation de couches en est un exemple et est décrite dans ce document. De telles techniques répondent à deux questions principales : la plupart des systèmes sont trop complexes pour les comprendre entièrement et différentes perspectives d'un système sont requises pour différents publics.

Les couches ont été adoptées dans de nombreux systèmes logiciels et sont suivies dans de nombreux textes, y compris dans le processus RUP (Rational Unified Process). Cependant, cette technique est souvent mal comprise et appliquée de manière incorrecte. Ce document vise à clarifier la signification de la technique par couches et à expliquer l'impact de l'application de ces différentes techniques.

Qu'est ce que la technique par “couches” ?

Commençons par définir ce que ce que signifie la technique par “couches.” Le terme *couche* fait référence à l'application d'un schéma architectural généralement en tant que pattern de *“couches”* , décrit dans un certain nombre de textes ([Buschmann], [Herzum], [PloP2]), et également dans le processus RUP. Un *pattern* représente une solution à un problème courant, présent dans un contexte particulier. Le tableau 1 présente le pattern des couches.

Tableau 1 : Présentation du pattern de “couches”

	Pattern des couches
Contexte	Système nécessitant une décomposition.
Problème	Système trop complexe pour pouvoir être compris dans son intégralité. Système difficile à gérer. Système dont les éléments les moins stables ne sont pas isolés. Système dont les éléments les plus réutilisables sont difficiles à identifier. Système devant être créé par différentes équipes, dotées vraisemblablement de compétences différentes.
Solution	Structurer le système en couches

L'un des exemples les plus courants de l'utilisation de couches est le modèle OSI 7 couches, définis par l'ISO (International Standardization Organization). Ce modèle, illustré à la figure 1, définit un ensemble de protocoles réseau dans lequel chaque couche met l'accent sur un aspect spécifique de la communication et se fonde sur les fonctions de la couche située en dessous d'elle. Le modèle OSI 7 couches utilise une stratégie par couches basées sur la responsabilité, chaque couche ayant une responsabilité particulière. Cette stratégie est expliquée en détails dans une prochaine section de ce document.

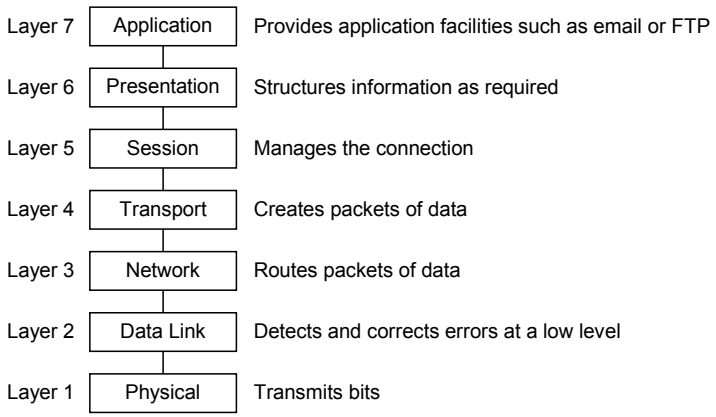

Figure 1 : Modèle OSI 7 couches (couches basées sur la responsabilité)

La figure 2 illustre un autre exemple de couches basées sur la responsabilité.

- La *couche Logique de présentation* contient des éléments responsables d'une certaine forme de rendu pour un être humain, tel qu'un élément de l'interface utilisateur.
- La *couche Logique applicative* contient des éléments responsables d'une certaine forme de traitement métier et de l'application des règles métier.
- La *couche Logique d'accès aux données* contient des éléments responsables de l'accès à une source d'informations, telle qu'une base de données relationnelle.

Il est à noter que les couches peuvent être modélisées de nombreuses façons, comme indiqué plus loin dans ce document. Pour le moment, nous allons représenter une couche de manière explicite à l'aide d'un package UML avec le stéréotype «layer».

Figure 2 : Couches basées sur la responsabilité

Les couches illustrées dans cet exemple particulier de couches basées sur la responsabilité sont souvent appelées “niveaux” et sont un concept connu dans le développement des systèmes distribués dans lesquels on trouve des systèmes à *2 niveaux*, *3 niveaux* et *n niveaux*.

Un aspect important de la figure 2 réside dans le *sens des dépendances* indiquées, du fait qu'il implique une certaine règle qui est une caractéristique des systèmes en couches : un élément dans une couche spécifique ne peut accéder qu'aux éléments de la même couche ou des couches inférieures[1]. Dans l'exemple présent, il se peut que des éléments de la *couche Logique applicative* n'accèdent pas aux éléments de la *couche Logique de présentation*. Il se peut également que des éléments de la *couche Logique d'accès aux données* n'accèdent pas à des éléments de la *couche Logique applicative*. Cette structure est souvent appelée graphe acyclique orienté. Elle est *orientée* de par le fait que les dépendances sont unidirectionnelles et *acycliques* de par le fait qu'un chemin de dépendances n'est jamais circulaire.

Plus spécifiquement, il est important d'être précis au sujet de la signification de chaque couche lors de la définition d'une stratégie par couches de sorte que les éléments soient correctement placés dans la couche appropriée. Tout manquement à l'affectation correcte d'un élément dans la couche appropriée diminue la valeur d'application de la stratégie initiale. Comme

[1] Bien qu'une notification d'événement puisse aboutir à un message d'un élément d'une couche, envoyé à un élément d'une couche supérieure, aucune dépendance explicite n'existe dans ce sens.

chaque stratégie par couches est détaillée, des informations générales sont données sur la signification de chacune des couches.

Modélisation des couches

A mesure que nous avançons dans les différentes stratégies par couches, il devient évident qu'il est nécessaire de communiquer chaque stratégie à l'aide de *modèles* spécifiques (et de fait d'éléments UML spécifiques). Un *modèle* représente une description complète d'un système d'un point de vue spécifique. La figure 3 montre un exemple de quatre modèles représentant différentes perspectives du système considéré :

- Modèle de cas d'utilisation : présente les exigences du système.
- Modèle d'analyse : présente l'analyse des exigences du système.
- Modèle de conception : présente la conception du système.
- Modèle d'implémentation : présente l'implémentation du système.

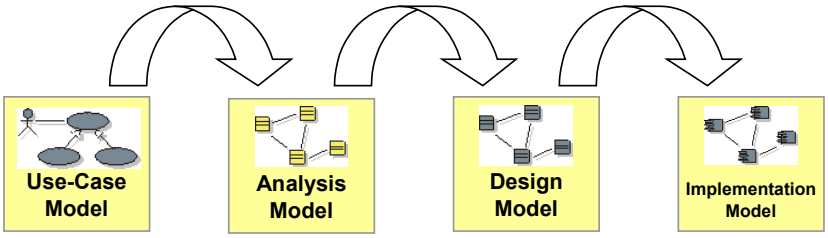

Figure 3 : Quatre modèles représentant un affinement graduel

Modèles supplémentaires :

- *Modèle de déploiement :* présente les aspects de distribution d'un système.
- *Modèle de données :* présente les aspects persistants d'un système.

Stratégies par couches

Les couches peuvent être basées sur un certain nombre de caractéristiques. Cette section aborde les couches basées sur les caractéristiques suivantes :

- responsabilité
- réutilisation

La représentation de chaque stratégie va être abordée, du fait que chaque stratégie est détaillée.

Couches basées sur la responsabilité

La stratégie par couches probablement la plus communément utilisée est celle basée sur la responsabilité. Cette stratégie particulière peut améliorer le développement et la maintenance d'un système du fait que les responsabilités système sont isolées les unes des autres. Par exemple (voir figure 2), un système peut être structuré par couches basées sur la responsabilité suivantes :

- logique de présentation
- logique applicative

- logique d'accès aux données

Chacune de ces responsabilités peut être représentée en tant que couche, comme le montre la figure 4 qui présente un exemple de contenu pour chaque couche. Dans le cas présent, trois concepts sont pris en considération dans un système de traitement des commandes : Client, Commande et Produit. Par exemple, le concept *Client* englobe les éléments suivants :

- *Classe CustomerView :* responsable de la *logique de présentation* associée à un client, comme l'affichage d'un client dans l'interface utilisateur.
- *Classe Customer* : responsable de la *logique métier* associée à un client, comme la validation de détails qui lui sont relatifs.
- *Classe CustomerData* : responsable de la *logique d'accès aux données* associée à un client, comme rendre l'état d'un client permanent.

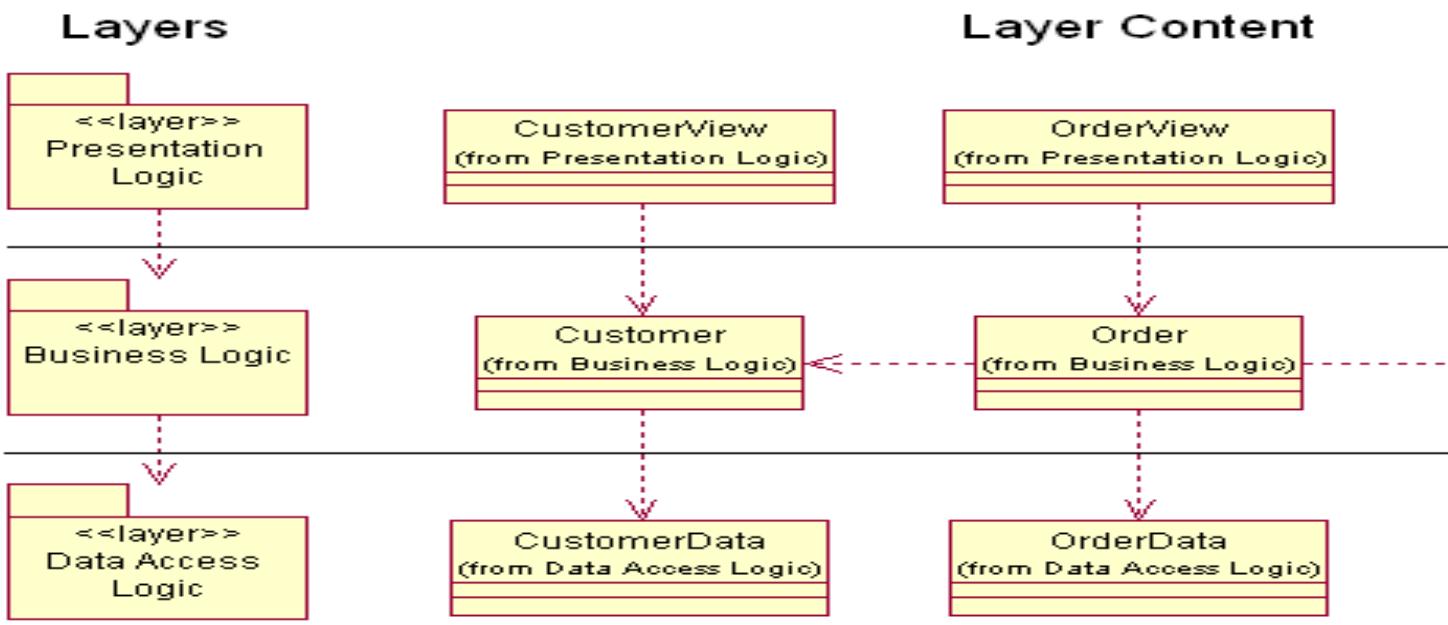

Figure 4 : Couches et contenu pour des couches basées sur la responsabilité

Etudions à présent quelques “mythes” concernant cette stratégie par couches particulière.

Mythe 1 : Couches et niveaux sont des choses différentes

Ce mythe spécifique est une source de confusion fréquente. Le fait est qu'un niveau *est* une couche, mais une couche basée sur une stratégie particulière : celle de la responsabilité. La confusion est due au fait que les concepts de niveaux peuvent être appliqués de nombreuses façons, comme indiqué dans le tableau 2.

Tableau 2 : Définitions des niveaux

Application	Couches (Niveaux)
2 niveaux	logique de présentation et logique applicative associées logique d'accès aux données

3 niveaux	logique de présentation logique applicative logique d'accès aux données
n niveaux	logique de présentation logique applicative (*répartie*) logique d'accès aux données

Mythe 2 : Les couches (niveaux) impliquent une répartition physique

Une autre mauvaise interprétation courante consiste à dire que les couches logiques impliquent une répartition physique. Examinons l'exemple de couches à 3 niveaux. Bien que divers éléments soient appelés à résider dans l'une des couches, chaque couche en elle-même peut être appliquée de nombreuses façons comme le montre le tableau 3 qui utilise des noms souvent usités pour caractériser une répartition physique particulière (tel que “client partiel”).

Tableau 3 : Application de couches à 3 niveaux

Application	Couches	
	Côté client	Côté serveur
Système unique	logique de présentation logique applicative logique d'accès aux données	
Client partiel	logique de présentation	logique applicative logique d'accès aux données
Client complet	logique de présentation logique applicative	logique d'accès aux données

On peut également dire qu'un seul système peut employer plusieurs stratégies de distribution physique, dans lesquelles certains éléments sont classifiés comme l'incarnation de la distribution d'un “client partiel” et d'autres de la distribution d'un “client complet”. En règle générale, le choix se base sur des exigences non fonctionnelles, telles que les performances.

Modélisation des couches basées sur la responsabilité

Comme nous allons le voir, l'application de cette stratégie peut influencer le *modèle de conception*, le *modèle d'implémentation* et le *modèle de déploiement*. Le *modèle de conception* est généralement structuré à l'aide de deux approches.

La **première approche** montre les éléments “contenus” dans la couche. Le résultat est illustré à la figure 5, une capture du navigateur Rational Rose qui montre :

- *des classes de présentation* (CustomerView, OrderView et ProductView) résidant dans un package de logique de présentation ;
- *des classes de logique applicative* (Customer, Order et Product) résidant dans un package de logique applicative ;
- *des classes de logique d'accès aux données* (CustomerData, OrderData et ProductData) résidant dans un package de logique d'accès aux données.

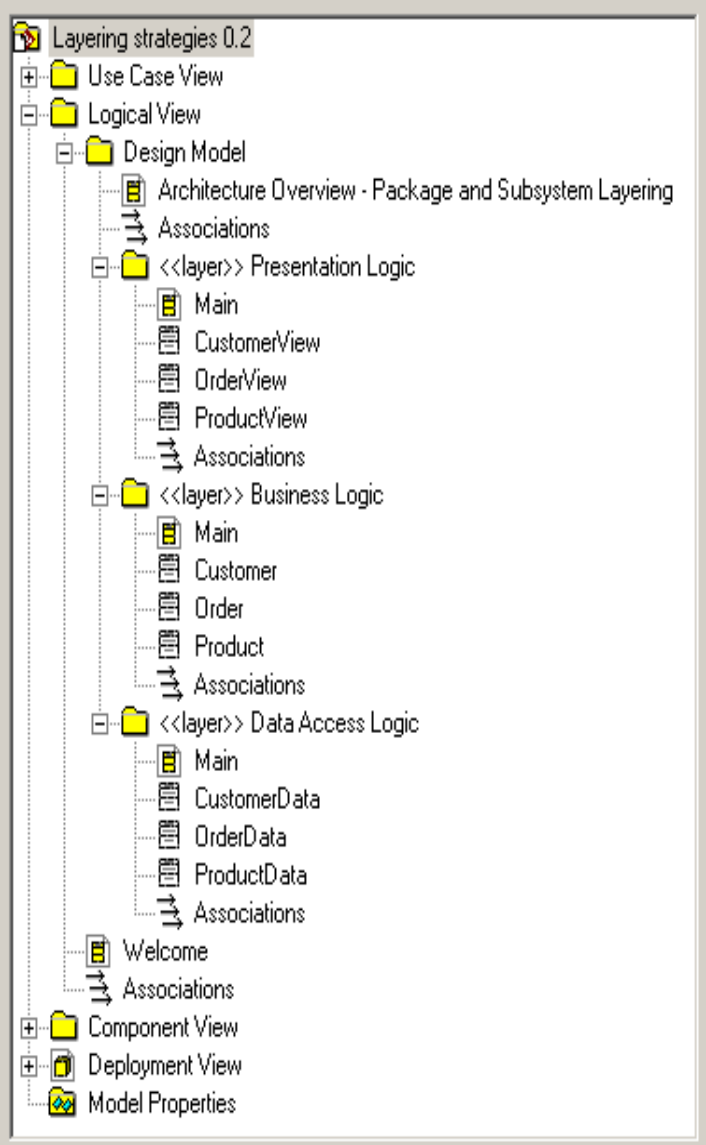

Figure 5 : Eléments contenus dans des couches

La seconde approche intègre le concept de *composant métier* (dans le cas présent, Customer, Order et Product) comme citoyen de premier ordre, dans lequel les principaux éléments sont les concepts liés au domaine et pris en charge par le système. Par exemple, le concept *Customer* peut disposer d'éléments associés de logiques de présentation, applicative et d'accès aux données. Ce concept de composant métier est détaillé dans [Eeles] et [Herzum]. Cette façon de penser aboutit à la structure de modèle illustrée à la figure 6. Dans cet exemple, les couches sont *induites* par les noms d'élément. Par exemple, toutes les classes *View* (telle que *CustomerView*) impliquent une couche de logique de présentation et toutes les classes *Data* (telle que *CustomerData*) impliquent une couche de logique d'accès aux données. Les noms de classe non qualifiés (tel que *Customer*) impliquent une couche de logique applicative.

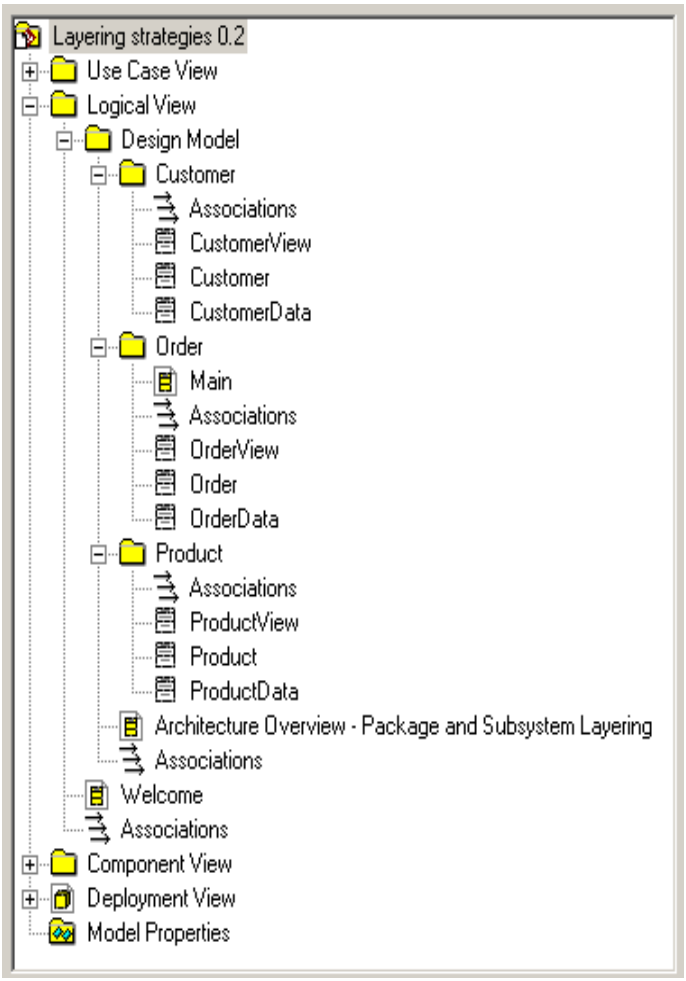

Figure 6 : Couches *implicites* dans chaque package de composant métier

Les couches peuvent être également représentées de manière explicite au sein de chaque package représentant un composant métier, comme le montre la figure 7. *Cette structuration est préférable lorsqu'un certain nombre d'éléments est impliqué dans chaque couche d'un composant métier donné*. Bien que seul le package de composant métier Customer ait été développé dans cet exemple, les packages Order et Product ont une structure similaire.

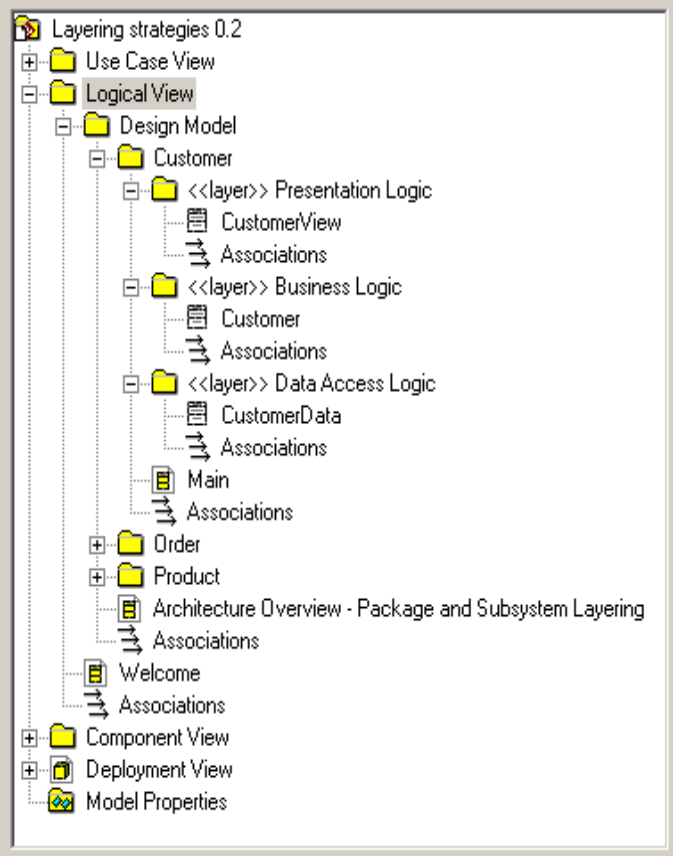

Figure 7 : Couches *explicites* au sein d'un package de composant métier

Une stratégie par couches basées sur la responsabilité influence le *modèle d'implémentation* en plus du modèle de conception, lorsqu'il est nécessaire de partitionner physiquement les éléments qui mettent en oeuvre chaque responsabilité. Prenons par exemple, un système qui présente une distribution physique de “client partiel”. Il est utile d'identifier les unités de mise en oeuvre requises pour prendre en charge l'exécution sur un client et celles requises pour prendre en charge l'exécution sur le serveur. Dans cet exemple, les éléments de la *couche de logique de présentation* résident dans une application déployée sur un client et tous les éléments de la *couche de logique applicative* et de la *couche de logique des données* résident dans une autre application déployée sur un serveur.

Ce scénario implique un *modèle d'implémentation* comme le montre la figure 8 qui présente une image du navigateur Rational Rose et un diagramme de composants affichant les éléments de l'application déployée sur le client. Dans cet exemple, il existe un mappage un à un entre une classe du modèle de conception et un composant UML dans le modèle d'implémentation. Notez, cependant, que ce mappage dépend généralement de la technologie d'implémentation utilisée.

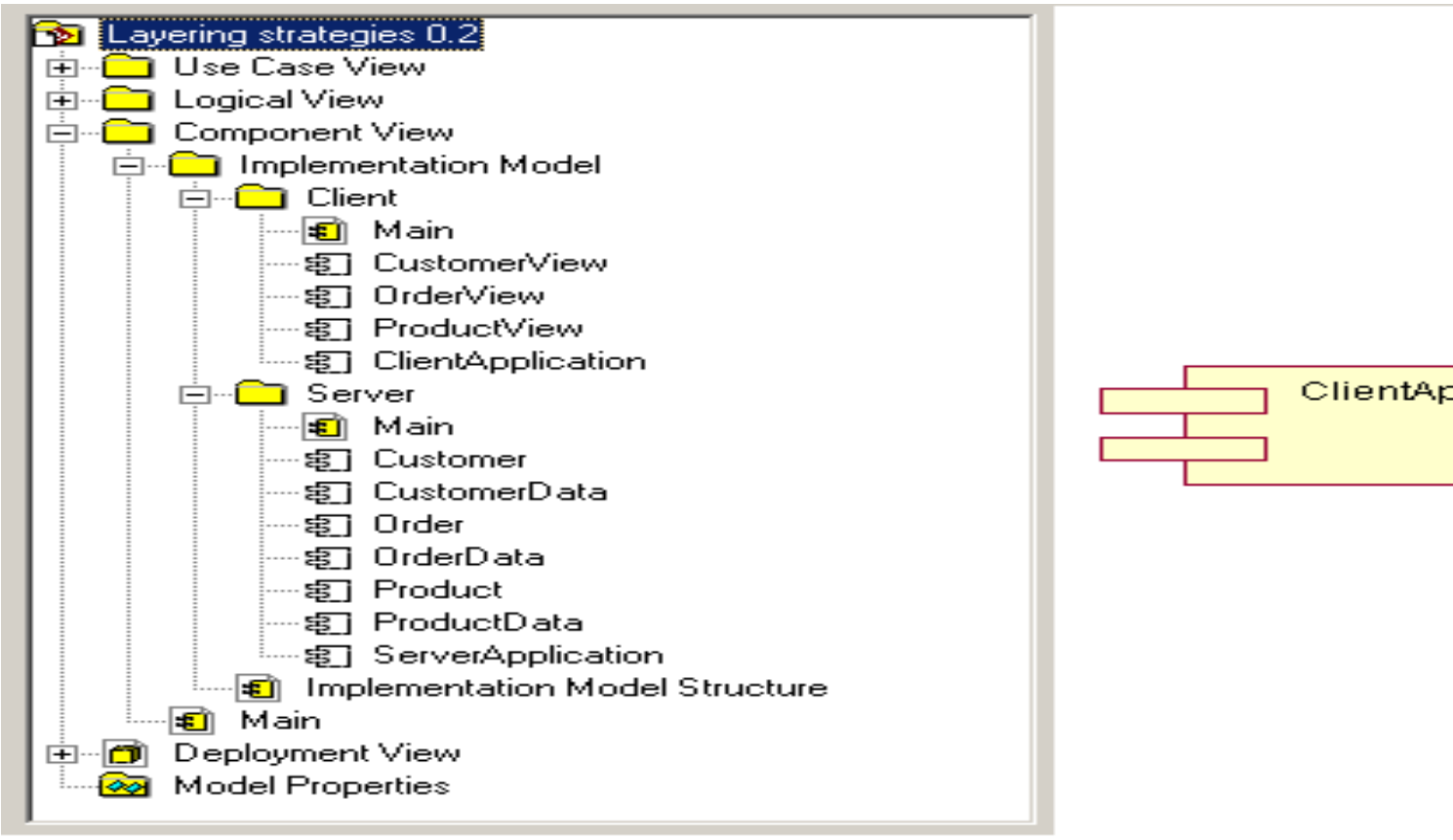

Figure 8 : Couches *implicites* au sein d'un modèle d'implémentation

De même manière, une stratégie par couches basées sur la responsabilité peut également influencer le *modèle de déploiement* lorsqu'il est nécessaire de décrire la répartition physique des responsabilités. A la figure 9, et à l'aide de l'exemple ci-dessus, nous pouvons voir que six noeuds ont été définis. Chacun des trois *noeuds Client* héberge un processus d'application client (ClientApplication). Le noeud du serveur frontal (FrontEndServer) héberge un processus LoadBalancer, responsable de la distribution des demandes client à l'un des deux noeuds serveur. Chaque *noeud Serveur* héberge un processus d'application serveur (ServerApplication).

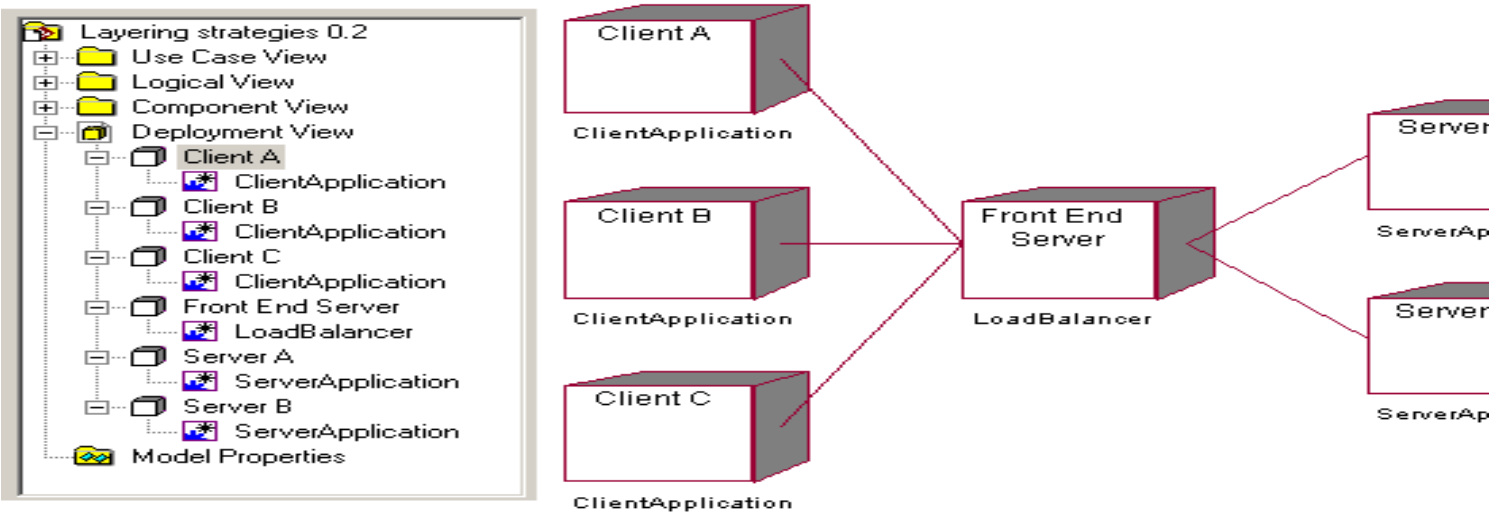

Figure 9 : Déploiement d'un modèle décrivant la répartition physique des responsabilités

Modélisation basée sur la réutilisation

Une autre couche utilisée communément est celle basée sur la réutilisation. Cette stratégie est particulièrement adaptée aux organisations ayant un objectif identifiable de réutilisation des composants dans toute l'organisation. L'impact lors de l'utilisation d'une telle stratégie par couches est que la capacité de réutilisation des composants est hautement visible, du fait que les composants sont explicitement groupés en fonction de leur niveau de réutilisation. Un exemple de couches, dérivé d'une stratégie décrite dans [Jacobson], est illustré à la figure 10. Dans ce cas, trois couches sont visibles : de base (Base), propre au métier (Business-Specific) et propre à l'application (Application-Specific).

- La *couche Base* contient des éléments qui peuvent s'appliquer à plusieurs organisations (tel que Math). De tels éléments seront largement réutilisés.
- La *couche Business-Specific* contient les éléments qui s'appliquent à une organisation particulière, mais qui ne dépendent pas de l'application (tel que le carnet d'adresses). De tels éléments seront réutilisés au sein des applications de la même organisation.
- La *couche Application-Specific* contient des éléments qui concernent une application ou un projet spécifique (l'organiseur personnel par exemple). Ces éléments sont les moins réutilisables.

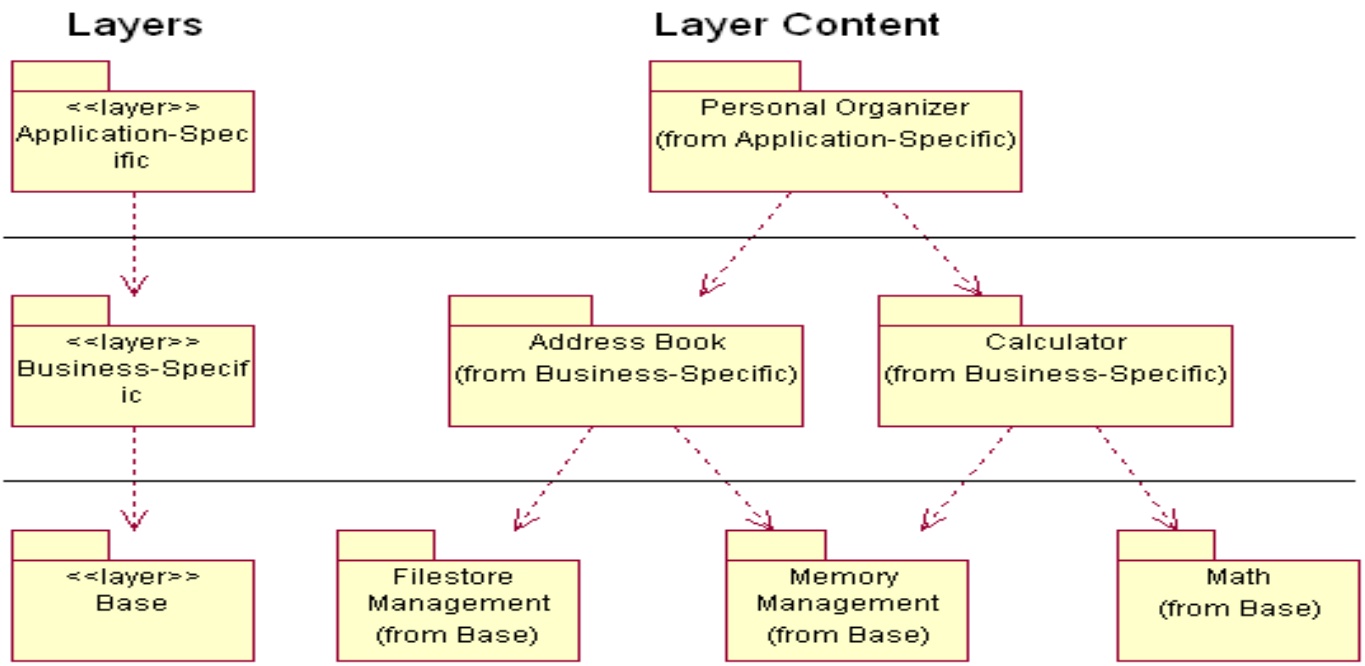

Figure 10 : Exemple de couches basées sur la réutilisation

On peut donc en déduire que les éléments de la couche de base sont les plus réutilisables, tandis que ceux de la couche spécifique à l'application sont plus spécifiques au projet, et donc moins réutilisables.

Modélisation des couches basées sur la réutilisation

L'application d'une stratégie de réutilisation influence principalement le *modèle de conception*. La structure d'un modèle de conception qui intègre des couches basées sur la réutilisation est simple à envisager et est illustrée à la figure 11 qui reprend l'exemple de la figure 10.

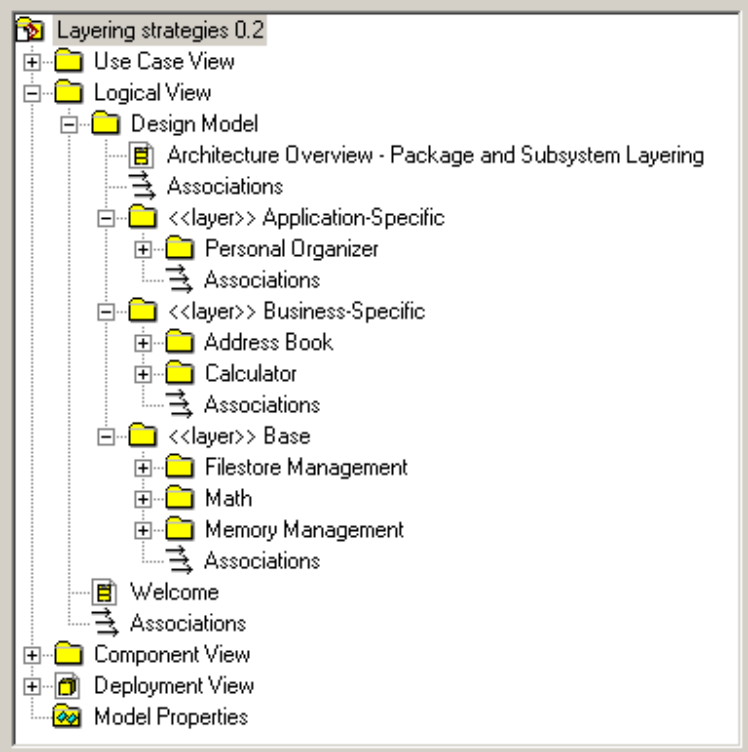

Figure 11 : Modèle de conception intégrant des couches basées sur la réutilisation

Autres stratégies par couches

Ce document est destiné à donner une simple “idée” des différentes stratégies par couches existantes, en prenant comme exemple les deux stratégies les plus largement utilisées. Cependant, des approches similaires peuvent être utilisées pour des stratégies présentant des caractéristiques, telles que sécurité, propriété et ensemble de compétences.

Couches multidimensionnelles

Les stratégies précédemment décrites peuvent être également associées pour créer de nouvelles stratégies par couches. L'exemple de la figure 12 montre :

- deux des couches basées sur la réutilisation de l'exemple précédent :
  - spécifique à l'application
  - spécifique métier
- trois couches basées sur la responsabilité (niveaux) :
  - logique de présentation
  - logique applicative
  - logique d'accès aux données

Les dépendances présentes dans la stratégie par couches basées sur la réutilisation émanent généralement de dépendances entre éléments des couches de logique applicative, comme le montre la figure 12, dans laquelle nous pouvons voir la dépendance entre l'organiseur personnel (PersonalOrganizer) et le carnet d'adresses (AddressBook).

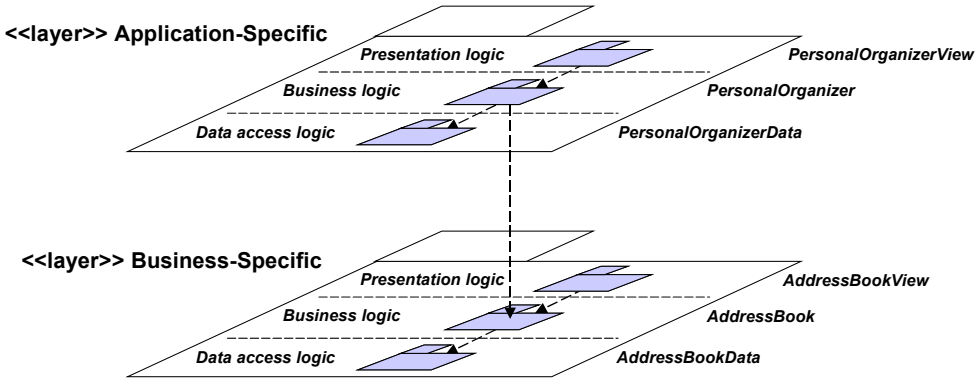

Figure 12 : Couches multidimensionnelles

Modélisation des couches multidimensionnelles

Dans ce cas, nous examinons la représentation des aspects multidimensionnels des couches dans un *modèle de conception* bi-dimensionnel. Nous envisageons également une structure dans laquelle est intégré le concept de *composant métier*.

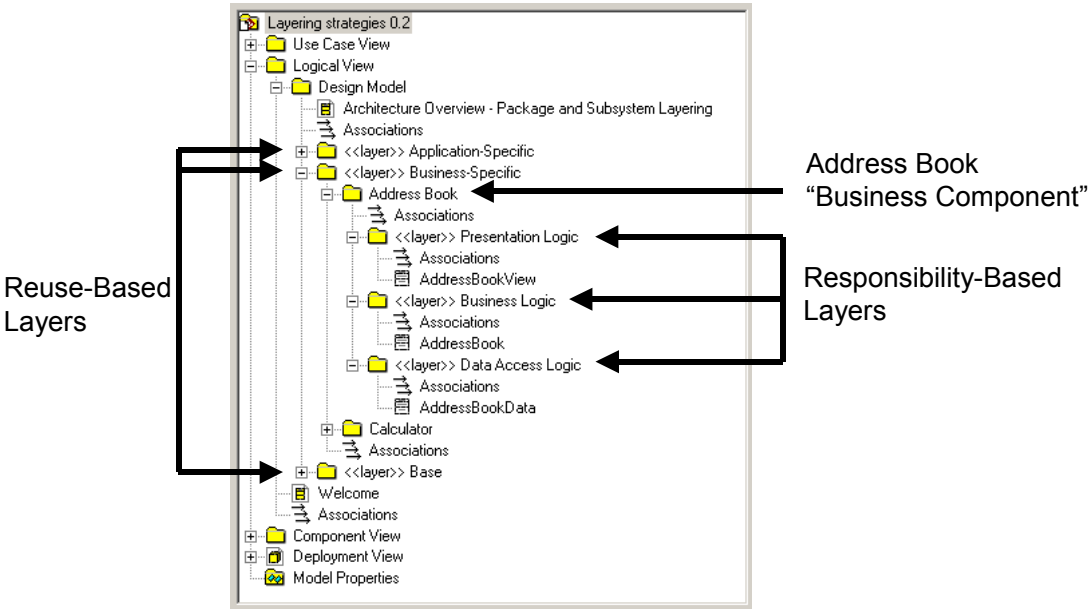

Figure 13 : Modèle de conception intégrant des couches multidimensionnelles

L'adoption d'une stratégie par couches multidimensionnelles nécessite qu'une stratégie primaire soit identifiée. Dans notre exemple, la principale stratégie par couches est basée sur la réutilisation. Le modèle de conception est tout d'abord organisé en fonction de cette stratégie, aboutissant aux couches *spécifique à l'application (Application-Specific)*, *spécifique métier (Business-Specific)* et *de base (Base)*. Chacune de ces couches est ensuite organisée par les éléments qui y résident. Par exemple, la figure 13 montre la couche spécifique métier contenant le *carnet d'adresses (Address Book)* et la *calculatrice (Calculator)*. Chacun de ces éléments est ensuite à nouveau organisé en fonction d'une stratégie secondaire : les couches basées sur la responsabilité. Par exemple, le package *Carnet d'adresses* contient les trois couches *Logique de présentation*, *Logique applicative* et *Logique d'accès aux données*.

Chacune de ces couches contient des éléments qui y résident :

- la *couche Logique de présentation* contient la classe AddressBookView

- la *couche Logique applicative* contient la classe AddressBook
- la *couche Logique d'accès aux données* contient la classe AddressBookData

Conclusion

L'une des décisions les plus importantes qu'un architecte doive prendre est celle de choisir une stratégie par couches appropriée, du fait que ceci aura une influence majeure sur la structure des modèles produits. Plus important encore cependant, est le fait que les avantages de l'entreprise, tels que la maintenabilité et la réutilisation, peuvent être directement pris en charge par la stratégie par couches choisie. Par exemple, davantage de systèmes gérables sont susceptibles d'être développés et donc les différentes responsabilités du système doivent être isolées les unes des autres via l'adoption d'une stratégie par couches basée sur la responsabilité. En outre, des éléments système réutilisables peuvent être facilement identifiés à l'aide d'une stratégie par couches basée sur la réutilisation.

Remerciements

L'auteur souhaite remercier Kelli Houston, Wojtek Kozaczynski, Philippe Kruchten, Bran Selic et Catherine Southwood (tous membres de Rational Software) pour leurs commentaires éclairés tout au long de la préparation de ce document.

Bibliographie

[Buschmann] Buschmann, Frank, et al. *A System of Patterns*. 1996. New York : John Wiley & Sons. ISBN 0-471-95869-7.

[Edwards] Edwards, Jeri. *3-Tier Client/Server at Work*. 1999. New York : John Wiley & Sons. ISBN 0-471-31502-8.

[Eeles] Eeles, Peter et Oliver Sims. *Building Business Objects*. 1998. New York : John Wiley & Sons. ISBN 0-471-19176-0.

[Herzum] Herzum, Peter et Oliver Sims. *The Business Component Factory*. 2000. New York : John Wiley & Sons.

[Jacobson] Jacobson, Ivar, et al. *Software Reuse*. 1997. Reading, Massachusetts : Addison-Wesley. ISBN 0-201-92476-5.

[PLoP2] Vlissides, John, James Coplien et Norman Kerth. *Pattern Languages of Program Design 2*. 1996. Reading, Massachusetts : Addison-Wesley. ISBN 0-201-89527-7.

Rational
the software development company

Sièges :

Rational Software
18880 Homestead Road
Cupertino, CA 95014
Tél : (408) 863-9900

Rational Software
20 Maguire Road
Lexington, MA 02421
Tél : (781) 676-2400

Appel gratuit : (800) 728-1212
Adresse électronique : info@rational.com
Site Web : www.rational.com
Sites internationaux : www.rational.com/worldwide