IAG

Working paper 16/01

GERER LA TAILLE DES REPRESENTATIONS CONCEPTUELLES DES DONNEES

Etat de la question

David Massart

UCL
Université catholique de Louvain

IAG
Institut d'administration et de gestion

Gérer la taille des représentations conceptuelles des données: État de la question

David Massart
Université catholique de Louvain
Institut d'Administration et de Gestion
Place des Doyens 1
B - 1348 Louvain-la-Neuve, Belgique
massart@isys.ucl.ac.be.

Résumé

L'absence de moyen efficace pour gérer les schémas conceptuels de grande taille est probablement l'un des freins les plus importants à l'utilisation des méthodes de modélisation conceptuelle des données. Cet article passe en revue les principales propositions qui ont été faites pour dépasser cette limite et tente de comprendre pourquoi elles ne sont pas employées.

1 Introduction

Cet article passe en revue les principales techniques qui ont été proposées pour résoudre les problèmes posés par la taille des représentations conceptuelles des données (schémas entités-associations, diagrammes de classes UML, etc.) – dans la suite nous appelons ces représentations : schémas conceptuels. Ces techniques se sont fort inspirées des hiérarchies de diagrammes de flux de données (DFD) [You89], où chaque processus d'un niveau donné peut être remplacé par un nouveau DFD au niveau inférieur.

La plupart de ces techniques s'appliquent lorsque le processus de modélisation conceptuelle est terminé. Elles prennent donc pour point de départ un schéma conceptuel considéré comme complet et correct qu'elles essaient de structurer en une hiérarchie de diagrammes.

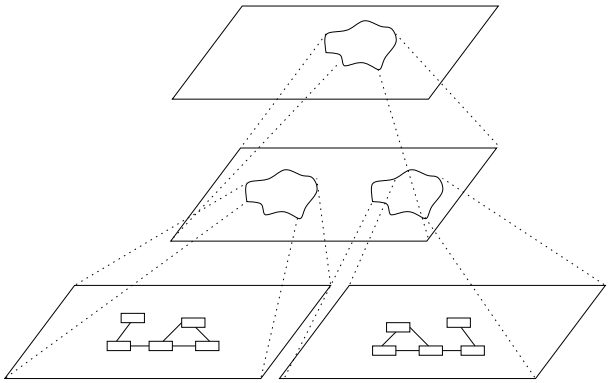

FIG. 1 – Les diagrammes d'un niveau correspondent à un composant de base du niveau supérieur.

Comme le suggère la figure 1, dans ces hiérarchies, les diagrammes du niveau le plus bas, dont les éléments de base sont des classes et des associations "classiques", sont eux-mêmes les éléments de base des diagrammes du niveau supérieur [MP92].

Généralement, on ne fait pas de différence entre un schéma, c'est-à-dire la description complète d'un domaine d'application, et un diagramme qui est la représentation graphique de ce schéma. Le schéma se présente sous la forme d'un diagramme unique reprenant l'ensemble des classes et associations pertinentes pour représenter le domaine d'application.

Avec les techniques de gestion des schémas de grande taille, le schéma se retrouve dispersé entre différents diagrammes qui constituent le niveau le plus bas de la hiérarchie tandis que les diagrammes de plus haut niveau fournissent des vues de plus en plus générales sur le schéma au fur et à mesure qu'on grimpe dans la hiérarchie. Finalement, le diagramme du sommet fournit une vue d'ensemble, peu détaillée, du schéma. Si les diagrammes du niveau le plus bas sont composés de classes et d'associations, la nature des constructions qui constituent les éléments de base des niveaux supérieurs n'est pas claire. Dans cet article, nous allons examiner comment les différentes techniques :

- définissent les composants de base des niveaux supérieurs,
- identifient ou dérivent ces composants et
- spécifient la représentation graphique de ces constructions, y compris leurs propriétés et relations avec les constructions des niveaux inférieurs.

Ces trois éléments sont, d'après [MP92], nécessaires pour gérer un schéma de grande taille. Chacune de ces techniques sera évaluée sur base des critères suivant : la simplicité et/ou la

possibilité d'automatiser la technique, la nécessité d'utiliser un modèle conceptuel spécifique et la qualité des diagrammes obtenus.

De nombreuses techniques de gestion des schémas de grande taille nécessitent de connaître les classes principales d'un schéma, c'est-à-dire les classes correspondant aux concepts les plus importants du domaine d'application représenté. La section 2 présente les principales méthodes utilisées pour identifier ces classes. La section 3 s'intéresse

- aux critères qui permettent de vérifier la cohésion des éléments qui composent un diagramme de bas niveau et
- aux méthodes qui permettent de découper un diagramme de grande taille en sous-diagrammes respectant ces critères.

La section 4 passe en revue différentes définitions qui ont été proposées pour les éléments de base des diagrammes de haut niveau. La section 5 présente d'autres champs de la modélisation conceptuelle dans lesquels des techniques développées à l'origine pour gérer les schémas de grande taille se sont montrées utiles. La section 6 examine la contribution des algorithmes d'optimisation de graphes à la gestion des schémas de grande taille. Finalement, la section 7 conclut ce chapitre par une discussion critique des résultats, limites et faiblesses des approches présentées.

2 Lutte des classes

Les classes d'un schéma conceptuel représentent les concepts d'un domaine d'application perçus comme pertinents. Dans un schéma, il n'est pas possible de distinguer les classes en fonction du rôle, plus ou moins important, que jouent les concepts qu'elles modélisent dans le domaine d'application. Cette homogénéité est à la fois une bonne et une mauvaise chose. Une bonne chose parce qu'elle assure la simplicité qui fait l'efficacité des modèles à objets. Une mauvaise chose parce qu'en ne permettant pas de distinguer l'essentiel de l'accessoire elle complique l'appréhension des schémas conceptuels.

De nombreuses techniques de gestion des schémas de grande taille nécessitent d'identifier les *classes principales* d'un schéma, c'est-à-dire les classes qui représentent les concepts jugés les plus "importants" d'un domaine d'application. Si la plupart des techniques utilisent des méthodes d'extraction des classes principales d'un schéma, d'autres (par exemple [EKW92, Moo97]) préfèrent laisser cette responsabilité aux utilisateurs du schéma. Si les tenants de cette dernière approche la justifient par les mauvais résultats qu'obtiennent les méthodes automatiques, on est en droit de se poser la question de l'intérêt d'une méthode de gestion de la complexité (c'est-à-dire d'une méthode d'aide à la compréhension d'un schéma) qui nécessite d'avoir compris un schéma avant de pouvoir être utilisée.

Cette section passe en revue les principales techniques qui ont été proposées pour identifier les classes principales d'un schéma. La figure 2 reprend un schéma qui, selon [Moo97], contient trois classes principales : CUSTOMER, ORDER et PRODUCT. Chacune des techniques présentées est appliquée à ce schéma. La *sensibilité* et la *spécificité* de chaque technique sont données à titre indicatif.

La sensibilité et la spécificité sont des mesures statistiques qui permettent d'évaluer la qualité d'une méthode de détection. Dans le cas présent, la sensibilité de la méthode est sa capacité à identifier comme principale une classe qui correspond à un concept important du domaine d'application d'un schéma. La spécificité de la méthode est sa capacité à identifier comme non-principale une classe qui correspond à un autre concept de ce domaine d'application. La

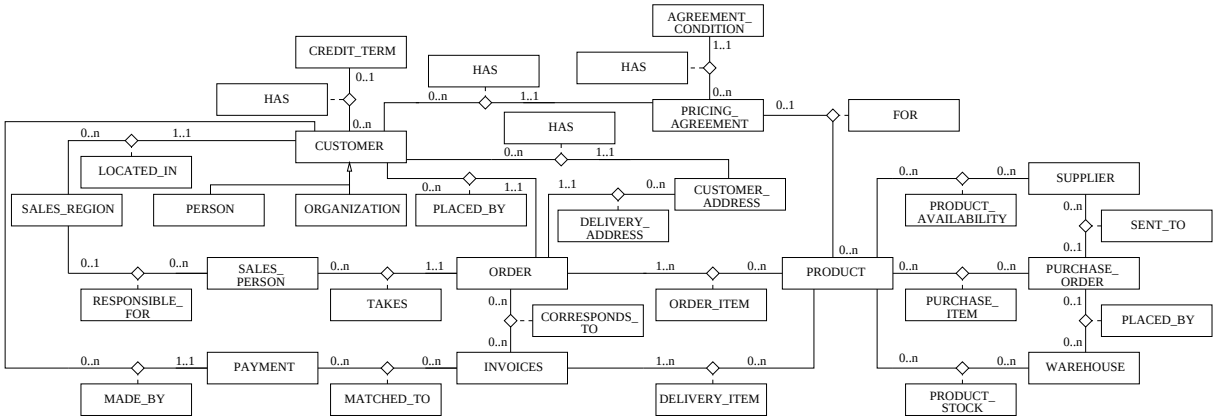

FIG. 2 – Schéma "la commande de produits" [Moo97].

sensibilité est donnée par la formule

$$sensibilité = \frac{nombre\ de\ classes\ principales\ correctement\ identifiées\ par\ la\ technique\ testée}{nombre\ réel\ de\ classes\ principales\ (ici\ 3)}$$

et la spécificité par la formule

$$spécificité = \frac{nombre\ de\ classes\ non\text{-}principales\ correctement\ identifiées\ par\ la\ technique\ testée}{nombre\ réel\ de\ classes\ non\text{-}principales\ (ici\ 13)}$$

2.1 Les concepts-clés

En 1983, Vermeir propose la notion de *concept-clé* [Ver83]. Un concept-clé est une classe considérée comme importante parce qu'elle permet de garder le schéma cohérent (connecté). Autrement dit, une classe est un concept-clé si elle participe à tous les chemins entre deux autres classes.

L'application de cette définition au schéma de la figure 2 permet d'identifier trois concepts-clés :

1. la classe CUSTOMER qui se trouve sur tous les chemins menant la classe CREDIT_TERM aux autres classes du schéma ;
2. la classe PRICING_AGREEMENT qui se trouve sur tous les chemins menant la classe AGREEMENT_CONDITION aux autres classes du schéma ;
3. la classe PRODUCT, qui, entre autres, est le point de passage obligé pour passer de la classe SUPPLIER à la classe ORDER.

soit une sensibilité de 0.67 et une spécificité de 0.92.

Comme le montre cet exemple, la sensibilité de la technique des concepts-clés est mauvaise. Celle-ci diminue encore lorsque le degré d'interconnexion, autrement dit la complexité du schéma augmente.

2.2 Les classes majeures

En 1984, Feldman et Miller proposent de distinguer parmi les classes d'un schéma des classes mineures et des classes majeures. Les classes majeures correspondent à des concepts fondamentaux du domaine d'application. Les classes qui ne sont pas majeures sont mineures.

Leur idée est la suivante : une classe est majeure si ses instances peuvent être identifiées de manière unique à partir des instances des classes qui lui sont associées. Autrement dit, la cardinalité de toutes les associations qui arrivent à une classe majeure ne peut pas dépasser 1 du coté opposé à cette classe.

L'application de cette définition au schéma de la figure 2 ne permet d'identifier qu'une classe majeure : SALES_PERSON[1] soit une sensibilité de 0 pour une spécificité de 0.92.

Comme le suggère l'exemple, cette technique donne de mauvais résultats.

2.3 Les classes dominantes

En 1989, Teorey et al. introduisent le concept de classe dominante [TWBK89]. Selon eux, certains types d'associations introduisent une relation dominé-dominant entre les classes qu'elles unissent. C'est le cas des associations suivantes :

- les associations porteuses d'une dépendance d'existence (cardinalité 1..1 du côté de la classe dépendante) ;
- les généralisations (une super-classe domine ses sous-classes) ;
- les agrégations (un composite domine ses composants).

L'application de ce critère au schéma de la figure 2 permet d'identifier cinq classes dominantes :

- la super-classe CUSTOMER dont dépendent les classes ORDER, CUSTOMER_ADRESS et PRICING_AGREEMENT ;
- la classe SALES_REGION dont dépend la classe CUSTOMER ;
- la classe SALES_PERSON dont dépend la classe ORDER ;
- la classe CUSTOMER_ADRESS dont dépend la classe ORDER et
- la classe PRICING_AGREEMENT dont dépend la classe AGREEMENT_CONDITION.

Soit une sensibilité de 0.33 pour une spécificité de 0.69.

Comme le suggère cet exemple, la technique n'est pas satisfaisante.

2.4 Les ancres

En 1994, Campbell et Halpin définissent le concept d'*ancre* pour distinguer les principales classes d'un schéma [CH94]. Une ancre est une association qui permet d'ancrer les instances d'une classe (autrement dit, les instances de cette classe sont obligatoirement associées à au moins une instance de la classe opposée), c'est-à-dire toutes les associations qui présentent une cardinalité minimum de 1.

D'après Campbell et Halpin, les classes importantes d'un schéma sont celles auxquelles s'attachent les ancres (du côté opposé à la cardinalité minimum de 1) soit les classes SALES_REGION, CUSTOMER, PRICING_AGREEMENT, SALES_PERSON, CUSTOMER_ADDRESS et PRODUCT dans le schéma de la figure 2. Ce qui donne une sensibilité de 0.66 et une spécificité de 0.69.

Ici encore, comme le suggère l'exemple, la technique n'est pas satisfaisante.

[1] La réduction des association N-N de ce schéma à des associations 1-N fait apparaître quatre autres classes majeures : INVOICES, PRODUCT, SUPPLIER et WAREHOUSE.

2.5 Les classes représentatives

En 1998, Castano et al. proposent une méthode de mesure de la pertinence des classes d'un schéma, les classes les plus pertinentes étant considérées comme les classes les plus représentatives du domaine d'application [CDAFP98].

Pour calculer la pertinence $P_S(c_i)$ d'une classe c_i d'un schéma S, ils considèrent les propriétés de c_i et les associations auxquelles elle participe. L'idée est que le nombre de propriétés et d'associations d'une classe peut être utilisé comme une mesure (heuristique) de la pertinence de cette classe dans le schéma. Plus cette mesure sera élevée, plus grande sera la pertinence de la classe.

Pour déterminer $P_S(c_i)$, une force $F_k \in [0,1]$ est assignée à chaque type d'association, de telle manière que plus la force est élevée, plus l'association est pertinente[2]. $P_S(c_i)$ est donnée par la formule

$$P_S(c_i) = \sum_{k=1}^{na} F_k.n_k(c_i)$$

où na est le nombre total de types d'associations représentés dans le schéma, F_k la force assignée au $k$ième type d'association et $n_k(c_i)$ est le nombre d'associations de type k auxquelles participe la classe c_i.

Etant donné la pertinence calculée à l'aide de la formule, les classes pertinentes sont obtenues en :

1. calculant la pertinence $P_S(c_i)$ de chaque classe du schéma,
2. définissant un seuil permettant de sélectionner les classes représentatives,
3. sélectionnant comme classes représentatives du schéma toutes les classes dont la pertinence est supérieure au seuil défini.

L'application de cette technique[3] au schéma de la figure 2 permet d'identifier les six classes représentatives suivantes (par ordre de pertinence décroissant) : CUSTOMER, PRODUCT, ORDER, INVOICES, PURCHASE_ORDER, et PRICING_AGREEMENT.

Avec une sensibilité de 1 et une spécificité de 0.77, il s'agit de loin de la meilleure des techniques examinées, d'autant que les trois classes principales du schéma correspondent aux trois classes représentatives dont la pertinence est la plus élevée.

3 De bas en haut

Dans cette section, nous allons passer en revue les principales stratégies qui ont été utilisées pour choisir les éléments d'un schéma de bas niveau qui correspondent à une construction du niveau supérieur, autrement dit, les critères utilisés pour découper le schéma conceptuel de départ en sous-diagrammes.

Que ce soit de manière implicite ou explicite, la plupart des techniques de gestion de la complexité utilisent les notions de *cohésion* et de *couplage* [You89] pour guider le processus de morcellement des schémas. La cohésion mesure la force des interconnexions entre les éléments d'un sous-diagramme tandis que le couplage mesure la force des interactions entre

[2] La force attribuée à chaque type d'association essaie de refléter les travaux de [TWBK89].

[3] En appliquant les valeurs conseillées (par Castano et al.) suivantes : $F_{association} = 0.4$, $F_{généralisation} = 0.6$, $F_{propriété} = 1.0$ et en prenant pour seuil la pertinence moyenne.

sous-diagrammes. Le principe est de minimiser le couplage des sous-diagrammes et de maximiser leur cohésion.

La différence entre ces stratégies tient essentiellement aux critères utilisés pour évaluer la cohésion et le couplage des sous-diagrammes. Ces critères peuvent être tant structurels, fonctionnels et dynamiques que sémantiques.

3.1 Les vues

Une vue décrit la portion d'un schéma conceptuel qui intéresse un ou plusieurs utilisateurs et leur cache le reste de ce schéma [EN00]. Le principe des vues est très simple et ne fait pas appel aux notions de cohésion et de couplage. Il a été utilisé par certains pour gérer la complexité (on n'accède jamais directement à l'entièreté du schéma mais à des vues partielles sur celui-ci) [EKW92, Sel93].

Le principal inconvénient de cette méthode tient essentiellement à l'absence de critère pour définir les vues.

3.2 La structure

3.2.1 Les associations

D'après Moody [Moo97], les associations entre les sous-diagrammes d'un schéma reflètent le couplage de ces diagrammes. Il suggère de mesurer le couplage obtenu lors de la décomposition d'un schéma en divisant le nombre d'associations à l'intérieur des différents diagrammes par le nombre d'associations entre ces diagrammes. Selon lui, le rapport ainsi obtenu doit être au moins de deux pour que la décomposition du schéma soit satisfaisante.

Il ne s'agit pas d'une méthode de décomposition de schémas, mais d'un critère qui permet d'évaluer a posteriori la qualité d'une décomposition.

3.2.2 Les associations 1-N

Les *associations 1-N* sont des associations binaires dont la cardinalité maximum est de 1 d'un côté et de N (un entier fini) de l'autre. Différentes méthodes de gestion de la complexité les utilisent pour regrouper classes et associations en sous-diagrammes.

Les associations 1-N permettent de définir l'*horizon logique* d'une classe [Ell82]. Une classe C possède un horizon logique si toutes les associations qui la quittent ont une cardinalité maximum de 1 du côté de C. L'horizon logique d'une telle classe C est constitué de toutes les classes dont les objets peuvent être identifiés de manière univoque à partir des objets de cette classe C de départ[4].

Feldman et Miller [FM86] proposent de regrouper en un élément de base du niveau supérieur les classes appartenant au même horizon logique à l'exception des classes majeures (cf. section 2.2) communes à plusieurs horizons logiques. Ces dernières se retrouvent telles quelles au niveau supérieur. Il en résulte qu'apparaissent dans le même diagramme des constructions de base de niveaux différents.

Une des faiblesses de cette méthode est qu'elle ne s'applique bien qu'aux schémas contenant de nombreuses associations 1-N. C'est ainsi qu'appliquée au schéma de la figure 2, la méthode

[4]Si donc à chaque instance de C est associée au plus une instance d'une autre classe. Autrement dit, la cardinalité de toutes les associations de C doit être de maximum 1 du côté de C (de cette manière, à chaque objet de C ne peut correspondre qu'un seul objet d'une classe associée à C).

ne permet de trouver que deux classes pouvant servir de point de départ à des horizons logiques : CREDIT_TERM et AGREEMENT_CONDITION (la figure 3 montre l'horizon logique de CREDIT_TERM). Ces deux horizons logiques possèdent une large intersection et ne reprennent que neuf des seize classes du schéma.

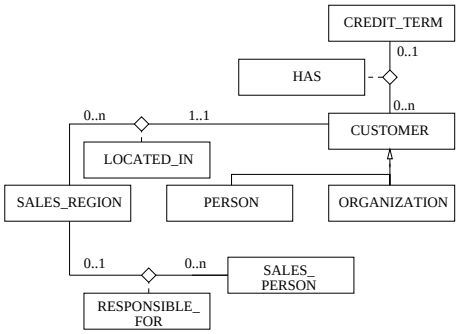

FIG. 3 – L'horizon logique de la classe CREDIT_TERM.

La méthode des *groupements par arbres* [RS92] est fort semblable à celle des horizons logiques si ce n'est qu'elle demande de décomposer les associations N-N en associations 1-N avant de découper le schéma en "arbres" (l'équivalent des horizons logiques). Ceci permet certes de découper[5] l'ensemble du schéma mais complique artificiellement le schéma de départ.

3.2.3 Le type des associations

D'après Teorey et al. [TWBK89], la cohésion entre deux classes dépend de la nature de l'association qui les lie. Ils ordonnent les différentes associations selon le spectre suivant (de l'association procurant la plus grande cohésion à celle en procurant le moins) :

1. association avec dépendance d'existence (cardinalité 1..1)
2. généralisation, agrégation
3. association binaire contrainte
4. association binaire non contrainte
5. association ternaire ou de niveau supérieur
6. pas d'association du tout

Un *groupe de classes* est le résultat d'une opération de groupement sur une collection de classes et d'associations. La méthode consiste à découper un schéma en zones fonctionnelles, puis, au sein de chacune de ces zones, à regrouper entre elles les classes possédant les associations les plus cohésives. Effectués de manière répétée, ces groupements permettent d'obtenir des schémas à plusieurs niveaux d'abstraction[6].

Appliquée au schéma de la figure 2, cette méthode ne fait intervenir que onze des seizes classes du schéma, soit les classes SALES_REGION, SALES_PERSON, PAYMENT, ORDER, ORGANIZATION, PERSON, CUSTOMER, CREDIT_TERM, AGREEMENT_CONDITION, PRICING_AGREEMENT et CUSTOMER_ADDRESS, qui peuvent être réparties en un à cinq groupes selon le nombre de classes dominantes (cf. section 2.3) utilisées pour construire ces groupes.

[5] La méthode des groupements par arbres peut être complètement automatisée.

[6] Ici non plus, l'élément de base d'un niveau ne doit pas obligatoirement participer à un élément de base du niveau supérieur.

En pratique, même si ce processus de groupement a pu être partiellement automatisé [HZ89], il nécessite un premier découpage "manuel" du schéma en zones fonctionnelles et donne des résultats peu satisfaisants.

3.3 L'agrégation et la généralisation

Certains auteurs, comme Carlson, Ji et Arora [CJA89], proposent d'utiliser l'agrégation et la généralisation comme critères de cohésion. Dans le cas de la généralisation, un diagramme de bas niveau n'est plus composé que de sous-classes et de leurs associations éventuelles tandis que la super-classe se retrouve au niveau supérieur. Dans le cas de l'agrégation, ce sont les classes "composants" et leurs associations éventuelles qui forment le diagramme de bas niveau tandis que la classe composite se retrouve sur le diagramme de niveau supérieur. Cette approche est limitée parce que, d'une part tous les schémas ne présentent pas des agrégations et des généralisations, d'autre part en définissant des niveaux de cette manière, il arrive fréquemment que des associations relient des classes appartenant à des niveaux différents.

3.4 La structure et la dynamique

Akoka et Comyn-Wattiau [ACW93] proposent d'utiliser un algorithme qui se base sur la *distance entre classes* pour optimiser la répartition des classes d'un schéma entre un nombre donné de sous-diagrammes. Cet algorithme maximise la somme des distances entre diagrammes et minimise la somme des distances à l'intérieur de chaque diagramme.

Trois types de distances sont proposées.

1. Une distance visuelle, appelée ainsi parce qu'elle est très proche de la représentation graphique des schémas. La distance visuelle entre deux classes correspond au nombre d'associations que contient le chemin le plus court entre ces deux classes.
2. Une distance hiérarchique, qui tient compte des cardinalités. Deux classes liées par des associations 1-N sont considérées comme plus proches que si elles l'étaient par des associations N-N. Comme pour la distance visuelle, le chemin le plus court entre deux classes est considéré. On compte une distance de 1 par association 1-N et de 2 par association N-N[7].
3. Une distance cohésive, qui correspond elle aussi au chemin le plus court entre deux classes, mais considère des segments de différentes longueurs sur ce chemin. C'est ainsi que la longueur d'une association avec dépendance d'existence est de 1, celle d'une généralisation ou d'une agrégation est de 10, celle d'une association binaire contrainte est de 100, celle d'une association binaire non contrainte de 1000 et celle d'une association ternaire ou de niveau supérieur est de 10000.

Utilisé avec la distance hiérarchique, cet algorithme donne des résultats semblables à ceux obtenus par les techniques basées sur les associations 1-N décrites à la section 3.2.2. Utilisé avec la distance cohésive, l'algorithme fournit des résultats comparables à ceux de la technique basée sur les types d'associations décrite à la section 3.2.3.

En plus des trois distances que nous venons de voir, Akoka et Comyn-Wattiau en proposent trois autres basées non plus sur des notions structurelles mais dynamiques : telles que le nombre et la fréquence des messages échangés entre objets.

[7] Akoka et Comyn-Wattiau justifient cela par le fait que les associations N-N sont décomposables en deux associations 1-N.

La première de ces distances est de 1 s'il existe des échanges de messages entre les objets de deux classes, sinon elle est égale au chemin le plus court entre ces classes.

La deuxième de ces distances tient compte du nombre de messages différents que peuvent s'échanger les instances de deux classes. La distance entre ces deux classes est $\frac{1}{n}$ où n est le nombre de messages différents échangeables, sinon elle est égale au chemin le plus court entre ces classes.

La troisième distance qui tient compte de la fréquence des messages n'est pas applicable à des schémas conceptuels.

3.5 La structure et le comportement

Auddino [Aud95] propose de mêler les critères structurels et comportementaux pour évaluer cohésion et couplage.

Elle définit la cohésion comme la mesure de la force des interconnexions qui existent entre un ensemble de classes et distingue trois degrés de cohésion :

- pas de cohésion : il n'y a ni associations entre les classes, ni propriétés qui les impliquent ;
- cohésion faible : les classes sont associées et il est possible de définir une propriété qui les implique ;
- cohésion forte : en plus des conditions nécessaires pour avoir une cohésion faible, il existe des contraintes d'intégrité qui impliquent les différentes classes comme un tout.

Les contraintes d'intégrité utilisées pour évaluer la cohésion peuvent être implicites si elles impliquent un mécanisme d'abstraction du modèle (par exemple une généralisation), ou explicites si elles sont explicitement déclarées.

De la même manière, Auddino définit le couplage comme la mesure de la force des interconnexions qui existent entre un ensemble de diagrammes et distingue trois degrés de couplage :

- pas de couplage : les différents diagrammes sont complètement disjoints ;
- couplage faible : les différents diagrammes ont une classe en commun ;
- couplage fort : les diagrammes sont associés par une relation composant-composite (certains diagrammes sont inclus dans d'autres).

Ici aussi , il s'agit moins d'une méthode de décomposition que d'un moyen relativement informel d'évaluer a posteriori la qualité de la décomposition d'un schéma.

3.6 La structure et la sémantique

Castano et al. [CDAFP98] proposent de définir le couplage entre diagrammes en termes de deux nouveaux concepts : l'*affinité* et la *proximité*.

Le sens précis de chaque classe et association d'un schéma peut être spécifié à l'aide de descripteurs tirés d'un vocabulaire contrôlé (du type d'un thésaurus). Castano et al. attribuent une valeur σ ($\sigma \in [0,1]$) à chaque type de relations existant dans le vocabulaire contrôlé et définissent la valeur de l'affinité entre deux termes t_i et t_j notée $\mathit{Affinité}(t_i, t_j)$ comme

$$\mathit{Affinité}(t_i, t_j) = \begin{cases} 1 & si\ t_i = t_j \\ \sigma & si\ t_i \neq t_j \wedge \exists\ \sigma\ pour\ la\ relation\ entre\ t_i\ et\ t_j \\ 0 & dans\ les\ autres\ cas \end{cases}$$

où ils proposent d'utiliser un σ de 0.8 entre synonymes[8] et de 0.5 entre hyponymes[9] et hyper-

[8]Synonymes : deux mots ou plus qui ont la même signification.

[9]Hyponyme : un mot plus spécifique qu'un mot donné.

nymes[10] et entre méronymes[11] et holonymes[12].

La proximité est définie à l'aide du nombre et du type d'association entre classes. Chaque type d'association se voyant attribué une certaine force (cf. section 2.5), la proximité de deux éléments varie en fonction de la force des associations qui les unissent.

La proximité, $Proximité(c_i, c_j)$, entre deux classes c_i et c_j est calculée par

$$Proximité(c_i, c_j) = \sum_{k=1}^{na} F_k.n_k(c_i, c_j)$$

où na est le nombre de types d'associations différents qui existent dans le modèle, F_k est la force des associations de type k et $n_k(c_i, c_j)$ est le nombre d'associations de type k entre les classes c_i et c_j.

La mesure du couplage $Couplage(D_x, D_y)$ entre deux diagrammes D_x et D_y contenant respectivement n_x et n_y classes est définie comme suit :

$$Couplage(D_x, D_y) = \frac{P_A \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} Affinité(t_i, t_j) + P_P \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} Proximité(c_i, c_j)}{n_x n_y}$$

où P_A, P_C, avec P_A, $P_C \in [0, 1]$ et $P_A + P_C = 1$, sont des poids introduits pour évaluer l'importance relative de l'affinité et de la proximité dans le calcul du couplage. Ces poids varient en fonction de l'importance relative attribuée à la sémantique par rapport à la structure du schéma[13].

Le découpage du schéma en γ diagrammes est basé sur la construction d'une matrice carrée d'ordre n, où n est le nombre de classes du schéma. Chaque classe est considérée comme un diagramme et la matrice est remplie en calculant le couplage de chaque paire de diagrammes puis, tant que le nombre de diagrammes est plus grand que γ, on sélectionne la paire de diagrammes dont le couplage est le plus grand[14], on combine ces deux diagrammes en un seul, on supprime de la matrice de couplage les rangées et les colonnes qui correspondent aux deux diagrammes, on ajoute une nouvelle rangée et une nouvelle colonne pour le nouveau diagramme et on calcule les nouveaux coefficients de couplage.

En pratique, la qualité du découpage obtenu dépend du γ choisi. Un γ de trois, par exemple, permet d'obtenir un découpage du schéma de la figure 2 très proche du découpage manuel proposé par l'auteur du schéma ([Moo97]). Cependant, il nous semble que l'affinité n'est pas nécessaire, au contraire. En effet, les résultats obtenus avec $P_A = 0.1$ ne diffèrent pas, dans les exemples que nous avons eu l'occasion de traiter, de ceux obtenus avec $P_A = 0$. Par contre, plus l'importance accordée à l'affinité augmente et plus la qualité des découpages obtenus devient discutable.

3.7 UML

Nous ne pouvions pas terminer cette revue des stratégies utilisées pour découper un schéma conceptuel en sous-diagrammes sans évoquer les recommandations d'UML pour regrouper les

[10] Hypernyme : un mot plus générique qu'un mot donné.

[11] Méronyme : un mot qui désigne une partie d'un tout désigné par un mot donné.

[12] Holonyme : un mot qui désigne un tout dont une partie est désignée par un mot donné.

[13] Avec $P_A = 0.1$ et $P_C = 0.9$, Castano et al. semblent considérer que le rôle de la structure est dominant dans la détermination du couplage.

[14] Pour augmenter la cohésion à l'intérieur d'un diagramme et diminuer le couplage entre diagrammes différents.

éléments d'un schéma en *packages*[15].

"A well-structured package :
- Is cohesive, providing a crisp boundary around a set of related elements.
- Is loosely coupled, exporting only those elements other packages really need to see, and importing only those elements necessary and sufficient for the elements in the package to do their job.
- Is not deeply nested, because there are limits to the human understanding of deeply nested structures.
- Owns a balanced set of contents ; relative to one another in a system, package should not be too large (split them up if necessary) or too small (combine elements that you manipulate as a group)." [RJB99]

S'il est difficile de ne pas être d'accord avec ces recommandations, on peut regretter l'absence de définition plus précise de la cohésion et du couplage des "packages".

4 Les éléments de base des diagrammes de haut niveau

Nous l'avons vu, les hiérarchies de diagrammes de flux de données (DFD) [You89], où chaque processus à un niveau donné peut être remplacé par un nouveau DFD au niveau inférieur, ont beaucoup influencé les différentes approches de la gestion de la complexité. Ces dernières tentent de représenter un schéma comme une hiérarchie de diagrammes dans laquelle les diagrammes d'un niveau constituent les composants de base des diagrammes du niveau supérieur. Le diagramme au sommet de la hiérarchie fournit une vue sur la totalité du schéma tandis que les diagrammes de plus bas niveau fournissent des vues partielles plus détaillées [Mas97a].

Si au niveau le plus bas, les diagrammes ne sont composés que de classes et d'associations "atomiques", la nature des constituants de base des niveaux supérieurs n'est pas claire. Il peut s'agir de classes [FM86, TWBK89, CDAFP98], d'associations [CJA89, EKW92, JOS93, Kri94] ou d'autres constructions comme les *domaines*[16] de [Moo97], les sous-systèmes de [Aud95] ou les packages de [RJB99].

Selon les approches, ces constructions sont de simples concepts visuels qui permettent d'organiser les classes et les associations d'un schéma conceptuel de manière à en augmenter la lisibilité [FM86, TWBK89, Moo97, CDAFP98] ou de véritables mécanismes d'abstraction que nous appelons *constructions complexes*[17].

La figure 4 est basée sur une version simplifiée du schéma de la figure 2. Elle représente un schéma à deux niveaux. Le niveau supérieur est constitué par un diagramme de haut niveau dans lequel des clients (CUSTOMER) commandent (ORDERS) des produits (PRODUCT). Le niveau inférieur est constitué de deux diagrammes : l'un, correspondant à l'association de haut niveau ORDERS, est constitué des classes de bas niveau PAYMENT, ORDER et INVOICE et de leurs associations ; l'autre, correspondant à la classe de haut niveau PRODUCT, est constitué des classes PRODUCT, SUPPLIER, PURCHASE_ORDER et WAREHOUSE et de leurs associations. La classe CUSTOMER, qui est pertinente aux deux niveaux de description, n'est impliquée dans aucun mécanisme de correspondance et appartient donc aux deux niveaux.

[15] Le choix des éléments qui composent un package est de la responsabilité de l'utilisateur.
[16] Subject areas.
[17] C'est-à-dire une classe ou une association dont la structure est décrite par une portion de schéma.

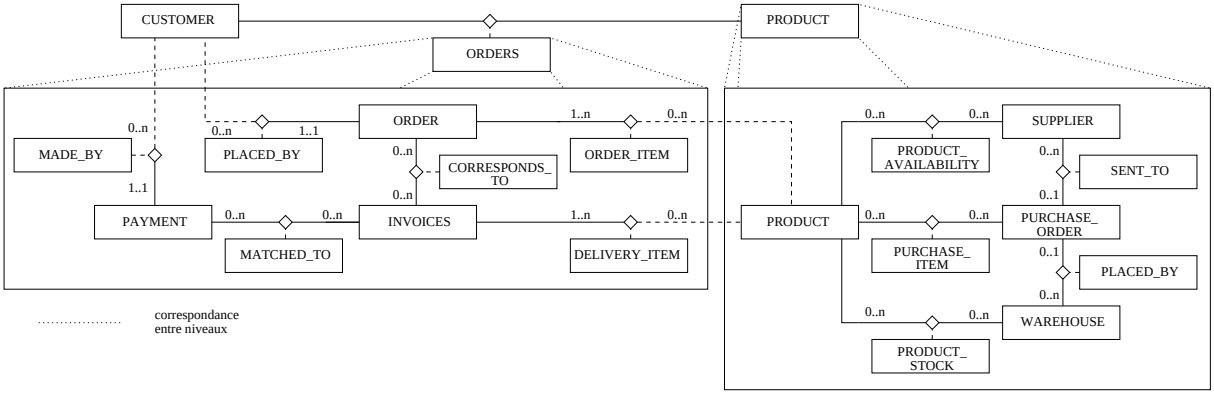

FIG. 4 – Une des approches possibles de la gestion de la complexité consiste à représenter un schéma comme une hiérarchie de diagrammes dans laquelle chaque diagramme de niveau supérieur est composé de diagrammes de niveau inférieur vus comme des classes et leurs associations.

Cette section passe en revue les principaux travaux sur les constructions complexes. Elle envisage successivement les raisons invoquées pour motiver l'utilisation de ces constructions, les mécanismes proposés pour faire correspondre ces constructions de haut niveau aux diagrammes des niveaux inférieurs, les propriétés de ces constructions, les manière dont elles peuvent s'associer et, finalement, leurs contrats.

4.1 Les motivations

Les trois principales raisons avancées pour enrichir les modèles conceptuels d'un mécanisme d'abstraction permettant de représenter et de manipuler une portion de schéma comme un tout sont :

- gérer la complexité,
- faciliter la réutilisation de portions de schéma et
- augmenter le pouvoir d'expression du modèle.

L'utilisation de mécanismes d'abstraction de haut niveau, c'est-à-dire de mécanismes impliquant plusieurs classes associées ou leur agrégation, aide à modéliser les différentes parties d'un domaine d'application de manière relativement isolée. Donc, d'un point de vue syntaxique, cette modularisation des schémas conceptuels peut améliorer leur lisibilité en les représentant de manière plus compacte [Aud95].

La représentation de parties complexes de schémas à un haut niveau d'abstraction permet de manipuler ces "morceaux" de schémas comme un tout, ce qui facilite leur réutilisation. D'une part cela permet de dériver des portions de schéma réutilisables, d'autre part l'organisation d'éléments réutilisables à différents niveaux d'abstraction facilite leur recherche en vue de la réutilisation [FP94]. Elle constitue aussi un bon moyen de les documenter. Comme la modélisation d'un domaine d'application peut impliquer la réutilisation de morceaux de schéma existants, les modèles conceptuels doivent être étendus pour pouvoir représenter à la fois la structure et le comportement de parties complexes d'un domaine d'application [Aud95].

Les mécanismes d'abstraction de haut niveau permettent de représenter de manière explicite les propriétés globales et les contraintes d'une portion de schéma [Aud95]. Par exemple, les contraintes d'intégrité peuvent constituer l'invariant d'une classe de haut niveau plutôt

que d'être annotées au schéma. De la même manière, ce changement de granularité permet d'expliciter des processus du domaine d'application[18] qui sinon n'existeraient que sous forme de combinaison implicite de requêtes et de commandes atomiques modélisées en dehors du schéma (c'est-à-dire comme des concepts indépendants de la structure du domaine) [HJ94].

4.2 La définition du diagramme sous-jacent

La représentation d'une portion de schéma par un mécanisme d'abstraction de plus haut niveau nécessite de définir la correspondance entre niveaux. Ceci amène trois questions :

- celle de la nature du lien qui associe les éléments d'un schéma de bas niveau à la classe ou à l'association complexe qui le représente au niveau supérieur,
- celle des associations entre les éléments d'un schéma de bas niveau et les classes de haut niveau autres que celles à la structure desquelles ils participent et
- celle de savoir s'il peut exister des intersections entre les schémas de bas niveau.

4.2.1 Dépendance et indépendance des concepts de haut niveau

Trois situations peuvent mener à la génération d'une classe (ou d'une association) complexe, qui peut être indépendante [EKW92] ou dépendre soit d'une des classes principales du niveau inférieur [EKW92] soit d'une association du niveau inférieur [Mas97a].

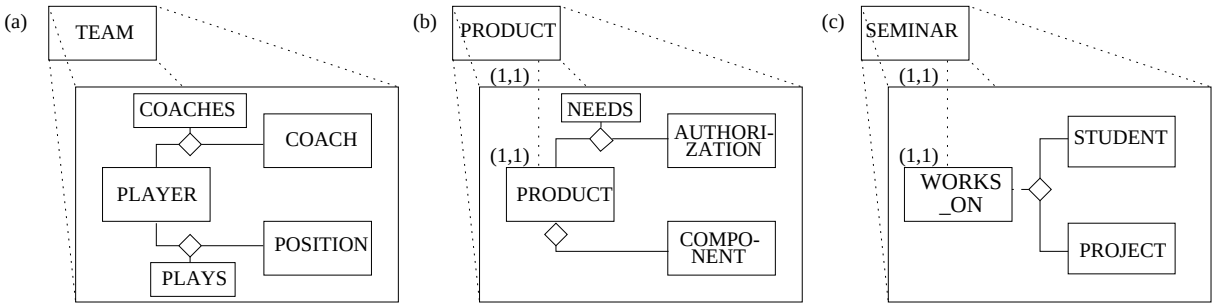

FIG. 5 – Classe de haut niveau (a) indépendante des constructions du niveau inférieur, (b) dépendante d'une classe de bas niveau, (c) dépendante d'une association de bas niveau.

La figure 5(a) illustre la première situation. La classe de haut niveau TEAM correspond à un diagramme composé des classes COACH, PLAYER et POSITION qui sont des concepts associés au concept d'équipe mais différents de lui. La classe de haut niveau TEAM représente un nouveau concept qui n'existe pas au niveau inférieur. Les instances de cette classe de haut niveau ne sont en correspondance avec aucun des objets du niveau inférieur dont ils sont donc indépendants.

La figure 5(b) illustre la deuxième situation. La classe complexe PRODUCT correspond à un schéma de bas niveau constitué de la classe principale PRODUCT et des classes AUTHORIZATION et COMPONENT qui lui sont associées. En fait, la classe PRODUCT de haut niveau et la classe PRODUCT de bas niveau représentent le même concept à différents niveaux d'abstraction. Il existe une correspondance un à un entre les instances de la classe principale du niveau inférieur et les instances de la classe de haut niveau. D'une certaine manière, la classe PRODUCT de

[18] Business processes.

haut niveau et la classe PRODUCT de bas niveau sont la même classe mais elles jouent des rôles différents dans le mécanisme.

La situation illustrée par la figure 5(c) est très similaire à celle de la figure 5(b) si ce n'est qu'ici ce sont les instances de l'association de bas niveau WORKS_ON qui sont en correspondance un à un avec les instances de la classe de haut niveau.

Pour Carlson et al., dans le premier cas, le mécanisme de correspondance entre niveaux doit être une généralisation ou une agrégation [CJA89].

4.2.2 Les associations inter-niveau

De Troyer [DT91] propose la notion de *classe schéma*[19]. Il s'agit d'une classe décrite par un schéma conceptuel. Ce schéma décrit l'intérieur de la classe, c'est-à-dire toutes les classes, associations et contraintes qui composent la classe ou existent dans son contexte (c'est-à-dire qui lui sont liées par une association d'agrégation). Cette correspondance est appelée abstraction de contexte parce que le schéma qui décrit l'intérieur de la classe schéma fait abstraction des contextes dans lesquels la classe schéma peut se trouver[20]. L'encapsulation de cette classe est donc totale.

Dans la plupart des autres approches, cette indépendance entre la portion de schéma que représente le mécanisme de haut niveau et son contexte n'existe pas. Ceci a pour conséquence qu'il peut exister des associations entre les classes de bas niveau qui constituent la classe de haut niveau et d'autres classes de haut niveau, ce qui brise l'encapsulation du mécanisme de haut niveau. Pour remédier à ce problème, Gandhi et al. proposent un mécanisme de requêtes (détaillé à la section 4.3) pour extérioriser les éléments de bas niveau d'une classe de haut niveau sans briser son encapsulation [GRVG94].

4.2.3 Les superpositions entre sous schémas

Deux schémas sont superposés s'ils partagent au moins une classe. A quelques exceptions près (dont [Aud95] qui représente les classes partagées à l'aide d'associations de groupement, voir section 4.4), la plupart des approches interdisent les superpositions de sous-schémas ce qui oblige à des décompositions strictement hiérarchiques des domaines d'application.

4.3 Les propriétés

Selon que l'on considère les éléments de haut niveau comme de simples vues ou comme des mécanismes d'abstraction à part entière, leurs propriétés peuvent être de simples requêtes qui se limitent à dériver de l'information des niveaux inférieurs ou des propriétés indépendantes de la structure sous-jacente. Cette section passe brièvement en revue les principaux types de requêtes et de commandes de haut niveau qui ont été décrites.

4.3.1 Les requêtes

Un premier type de requêtes se contente d'exporter (c'est-à-dire de rendre visible) des requêtes de plus bas niveau. C'est le mécanisme proposé par UML [RJB99]. Reprenons la

[19] Schema object class.

[20] Le contexte d'une classe schéma est constitué par l'ensemble des associations auxquelles cette classe schéma peut prendre part.

figure 5(b), on aurait, par exemple, une requête name (le nom du produit) de la classe PRODUCT de bas niveau qui se retrouverait telle quelle comme requête de la classe PRODUCT de haut niveau.

En 1994, Gandhi et al. introduisent un type particulier de requêtes qui permettent d'*extérioriser* de manière sélective certaines des classes qui composent la structure d'un élément de haut niveau. Ces requêtes un peu particulières sont utilisées pour donner accès aux classes de bas niveau sans briser l'encapsulation de la construction de haut niveau [GRVG94].

Une étape de plus est franchie en permettant de dériver l'information de la structure sous-jacente. Il peut s'agir :

- d'une information calculée à partir d'une requête de plus bas niveau. Par exemple, si on reprend la figure 4, une requête de l'association de haut niveau ORDERS pourrait être le montant total d'une commande calculé à partir du prix et des quantités de chaque produit commandé ;
- d'une sorte de compteur lié à la structure de la portion de diagramme sous-jacente [Aud95]. Par exemple, si on reprend la figure 5(a), une des requêtes de la classe de haut niveau TEAM pourrait être le nombre de joueurs de l'équipe.

Finalement, un dernier type de requête permet de représenter des aspects directement liés à une classe ou à une association de haut niveau, c'est-à-dire des informations indépendantes de la structure sous-jacente [CJA89, GRVG94, Aud95]. Par exemple, si on reprend la figure 5(a), des requêtes de la classe de haut niveau TEAM pourraient être le nom de l'équipe, la date de sa création, etc.

4.3.2 Les commandes

Tout comme les requêtes, les commandes de haut niveau peuvent être classées selon leur dépendance vis-à-vis des structures sous-jacentes.

Selon [Aud95], les commandes dépendantes permettent de profiter du changement de granularité que fournissent les constructions de haut niveau pour rendre explicites les grandes fonctions présentes dans le domaine d'application en les décrivant en termes des commandes des structures sous-jacentes. Par exemple, la commande "changer une roue" d'une construction de haut niveau représentant une voiture peut être décrit comme la séquence des actions suivantes : soulever la voiture, déboulonner, retirer la roue, placer une roue neuve, la boulonner, abaisser la voiture et vérifier la pression du pneu.

Les commandes indépendantes, quant à elles, servent à modifier les états des instances des constructions de haut niveau qui peuvent être définis indépendamment des requêtes des structures sous-jacentes. Par exemple, une telle commande permettrait de "changer le nom de l'équipe" sur le schéma de la figure 5(a).

Notons que Carlson et al. se sont intéressés aux conditions nécessaires pour pouvoir insérer et supprimer les objets d'une classe de haut niveau [CJA89].

4.4 Les associations

Les classes de haut niveau, tout comme les classes de bas niveau représentent des concepts du domaine d'application. Il semble donc naturel d'associer ces concepts. Cette section passe en revue les principales catégories d'associations qui ont été proposées pour associer les classes de haut niveau.

Généralisation/spécialisation. Certaines approches permettent de construire des hiérarchies de généralisation/spécialisation des classes de haut niveau [DTJ93, RJB99].

Associations de haut niveau. D'autres, par exemple [CJA89, EKW92, JOS93, Kri94], associent les classes de haut niveau à l'aide d'autres constructions de haut niveau. Celles-ci peuvent être classées dans un cadre général à l'intérieur duquel on peut identifier certaines situations particulières.

L'association de haut niveau ORDERS de la figure 4 est une bonne illustration du cas général. Il n'y a pas de restriction particulière quant au choix des classes et associations de bas niveau qui peuvent participer à l'association de haut niveau. La seule contrainte est que si une classe de bas niveau est incorporée dans une association de haut niveau, toutes les associations attachées à cette classe doivent aussi en faire partie [EKW92].

Une première situation spécifique se produit lorsqu'une association de haut niveau est basée sur des associations de bas niveau en *parallèle* [CJA89, EKW92, JOS93].

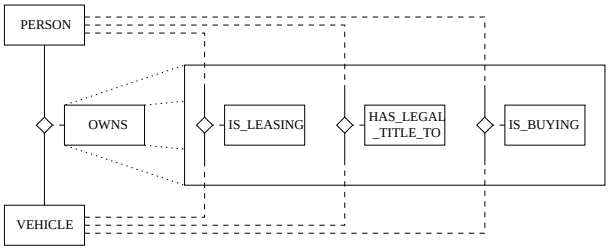

FIG. 6 – Généralisation d'associations de bas niveau en une association de niveau supérieur.

Cette situation est illustrée par la figure 6 où le fait qu'une personne puisse acheter, posséder en leasing ou avoir un titre de propriété sur un véhicule est *généralisé* à l'aide de l'association OWNS [EKW92]. Cette association de haut niveau peut être vue comme une super-association qui généralise un ensemble de sous-associations [CJA89]. En fonction des règles de correspondance entre niveaux, cette généralisation peut être totale ou partielle, exclusive ou non [Mas97b].

Une autre situation spécifique apparaît lorsque les classes et associations qui constituent le chemin entre deux classes sont utilisées pour décrire une association de haut niveau [CJA89].

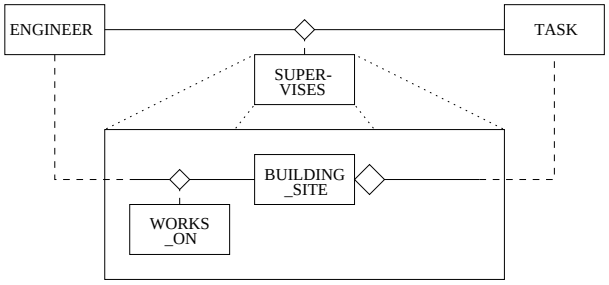

FIG. 7 – Agrégation d'associations de bas niveau en une association de niveau supérieur.

Cette situation est illustrée par la figure 7 où le fait qu'un ingénieur travaille sur un chantier qui implique différentes tâches est agrégé au niveau supérieur en l'association SUPERVISES.

Les associations de groupement. [Aud95] propose d'utiliser des *associations de groupement* pour représenter graphiquement les intersections qui peuvent exister entre les dia-

grammes qui définissent la structure des classes de haut niveau.

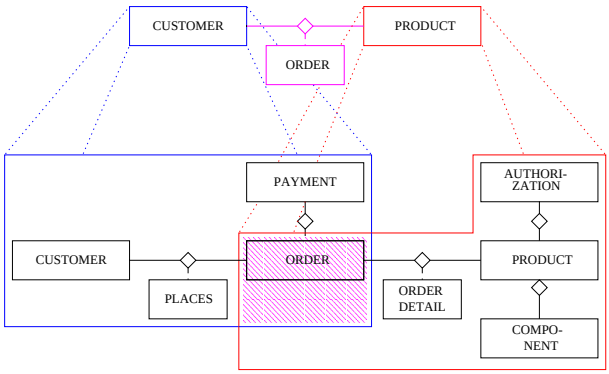

FIG. 8 – La classe de groupement ORDER signale l'existence d'une intersection entre les structures des classes de haut niveau CUSTOMER et PRODUCT.

La figure 8 illustre l'utilisation de ce type d'association. Sur le diagramme de haut niveau, l'association de groupement ORDER indique que les diagrammes de bas niveau qui définissent la structure des classes CUSTOMER et PRODUCT partagent la classe de bas niveau ORDER.

Les associations de bas niveau. Un dernier type d'associations qui peuvent apparaître sur un schéma de haut niveau sont les associations de bas niveau entre classes de bas niveau appartenant à des diagrammes représentés par des classes de haut niveau différentes.

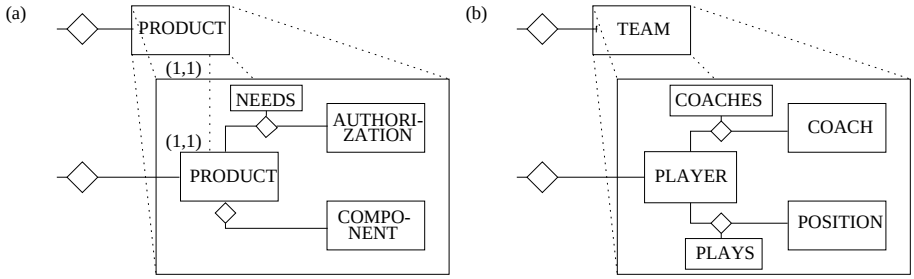

FIG. 9 – Associations de bas niveau entre classes de bas niveau appartenant à des diagrammes représentés par des classes de haut niveau différentes et soit (a) dépendant soit (b) ne dépendant pas de ces classes de bas niveau.

Deux cas peuvent se présenter : soit l'association de bas niveau aboutit à la classe de bas niveau dont dépend la classe de haut niveau, c'est le cas illustré par la figure 9(a), soit elle aboutit à une autre classe, c'est le cas illustré par la figure 9(b).

Dans le premier cas, du fait de la dépendance entre les classes du bas et du haut niveaux, la sémantique de l'association est préservée comme si elle se connectait directement à la classe de haut niveau [EKW92].

Le second cas nécessite l'utilisation du mécanisme de requête particulier évoqué à la section 4.3 pour éviter que l'encapsulation de la classe de haut niveau soit violée [GRVG94]. Dans l'exemple de la figure 9(b), cela consiste à extérioriser la classe PLAYER.

4.5 Les contrats

Les différents contrats (invariants, pré- et postconditions) des constructions de haut niveau peuvent se répartir en contrats privés et contrats publics. Les contrats publics s'appliquent aux clients extérieurs à la structure de la construction de haut niveau, tandis que les contrats privés ne s'appliquent qu'aux clients internes de cette structure.

Les invariants publics permettent de contraindre les états définis par les requêtes des constructions de haut niveau, que ces dernières dépendent ou non des structures sous-jacentes.

Les invariants privés permettent de modéliser sous forme d'invariants des contraintes sur la structure qui, en utilisant un modèle plat, auraient simplement été annotées sur le schéma.

Les pré- et postconditions publiques permettent de définir les conditions d'utilisation des propriétés d'une structure de haut niveau, tandis que les pré- et postconditions privées explicitent la manière dont ces propriétés dépendent des propriétés des classes et associations du niveau inférieur [Aud95].

5 Les autres applications

Les techniques développées pour gérer la complexité des schémas conceptuels se sont aussi montrées utiles dans des activités de modélisation de données telles que la modélisation descendante (top-down modeling), l'intégration de schémas ou la transformation de schémas, activités dans lesquelles il est souvent intéressant de représenter un concept à plusieurs niveaux d'abstraction. Cependant, si l'utilisation de techniques initialement conçues pour la gestion de la complexité donne généralement de bons résultats dans le domaine de la modélisation de données, cela n'a que peu contribué à la gestion de la complexité en tant que telle.

5.1 La modélisation descendante

Lors de la modélisation de l'information, un domaine d'application est examiné pour produire un schéma conceptuel. Une technique de modélisation consiste à commencer par la production d'une version abstraite du schéma qui est progressivement raffinée par l'utilisation d'opérations de transformation. C'est la modélisation descendante. Elle est basée sur un ensemble de primitives de transformation appliquées à un concept unique pour produire une description plus détaillée de ce concept [BCN92].

D'une certaine façon, ces primitives sont l'inverse des mécanismes de gestion de la complexité. Le schéma (de bas niveau) de départ du processus de gestion de la complexité peut être vu comme le résultat de l'activité de modélisation descendante (qui commence avec le schéma du plus haut niveau).

Batini et al. [BDBS93] suggèrent que les primitives de la modélisation descendante peuvent aussi être utilisées pour produire des schémas qui décrivent un domaine d'application à plusieurs niveaux d'abstraction.

De la même manière, les techniques de gestion de la complexité peuvent servir de base à une typologie des primitives de raffinement descendant. C'est ainsi que Jaeschke et al. [JOS93] proposent de classer ces primitives en trois catégories. La première catégorie, appelée *groupement de classes* (class clustering), se fonde sur le concept de classe principale. La seconde catégorie, appelée *groupement simple d'associations* (simple relationship clustering), correspond aux primitives qui permettent de regrouper en une seule des associations sémantiquement proches.

Quant à la troisième catégorie, appelée *groupement complexe d'associations* (complex relationship clustering), elle reprend les primitives qui permettent de regrouper des associations et des classes additionnelles en une association.

5.2 L'intégration de schémas

Lorsque la modélisation de l'information porte sur un domaine d'application très vaste, la construction du schéma conceptuel peut se dérouler en deux étapes. La première consiste à construire des schémas séparés pour chacune des catégories d'utilisateurs du futur système, la seconde consiste à intégrer ces différents schémas partiels en un schéma conceptuel global. En général, les techniques d'intégration de schémas permettent de résoudre des problèmes de terminologie, de recouvrement, de contradictions de contraintes et d'équivalences de représentations entre schémas partiels. Mais elles ont des difficultés à

- comparer deux à deux les différents concepts présents dans deux schémas partiels et
- valider l'intégration.

L'application de techniques automatiques de gestion de la complexité [ACW93] à l'intégration de schémas permet de limiter les problèmes d'explosion combinatoire rencontrés lors de la comparaison de deux schémas partiels et de vérifier que chacun des schémas partiels est bien représenté dans le schéma global [CWAK98].

5.3 Les transformations de schémas

Les transformations de schémas ont pour but de traduire un schéma d'un modèle dans un autre, lors de l'intégration de bases de données hétérogènes ou de la migration d'une base de données d'un système de gestion de bases de données à un autre.

Les techniques de gestion de la complexité se sont montrées utiles pour traduire des schémas entités associations en schémas à base d'objets complexes [RS92, MGG95]. En gros, différentes techniques de gestion de la complexité sont utilisées pour regrouper des entités atomiques en entités complexes qui servent de fondement à la conception des objets complexes.

La plupart du temps, ces modèles à base d'objets complexes sont relativement pauvres. S'ils permettent à une classe d'être l'attribut d'une autre classe, ils ignorent généralement le concept d'association.

6 Optimisation de graphes

De nombreux algorithmes ont été proposés ces dernières années pour assister les concepteurs de schémas conceptuels à base d'objets [Mut01, BETT99]. Ils permettent de disposer automatiquement les différentes classes et associations d'un schéma conceptuel dans le but d'en faciliter la lecture et la compréhension. Parmi les critères auxquels ces algorithmes recourent, on peut citer : éviter de superposer deux associations, éviter qu'une association traverse une classe, distribuer les classes de manière homogène sur la surface utilisable, minimiser le nombre d'intersections entre associations, etc.

Les algorithmes d'optimisation de graphes ne modifient ni la quantité d'information d'un diagramme ni le modèle dans lequel cette information est exprimée. Leur contribution à la gestion de la complexité se limite à améliorer de manière automatique la présentation des diagrammes. Leur principale faiblesse vient du fait qu'ils se servent de critères purement graphiques pour optimiser les graphes sans tenir compte de la sémantique des schémas. C'est

ainsi que, sous prétexte de distribuer les classes de manière homogène et de minimiser les intersections entre associations, des classes modélisant des concepts proches se retrouvent parfois fort éloignées l'une de l'autres sur le diagramme optimisé.

7 Discussion

Nous venons de passer en revue les principales propositions qui ont été faites pour gérer la complexité des schémas conceptuels. Force est de constater qu'en pratique, aucune d'entre elles n'a réussi à s'imposer alors que les problèmes liés à la complexité des schémas restent considérés comme le principal obstacle à l'utilisation des techniques de modélisation conceptuelle [Moo97]. Dans cette section, nous allons essayer de comprendre les raisons de cet échec.

A première vue, l'idée d'enrichir les modèles conceptuels d'un mécanisme d'abstraction de haut niveau qui permette de gérer les problèmes de complexité des schémas tout en augmentant leur pouvoir d'expression et en servant de base à leur réemploi est plutôt séduisante[21]. Paradoxalement, en pratique, les schémas à plusieurs niveaux que produisent ces techniques s'avèrent souvent plus difficiles à comprendre que les schémas plats de départ [Moo97].

Selon nous, l'explication de ce "paradoxe" tient au fait qu'aucune des techniques proposées ne respecte les trois conditions pour une solution efficace du problème de la gestion de la complexité [Mas01], à savoir qu'elles doivent :

1. être simples d'emploi (automatisables),
2. ne pas nécessiter de modifications des modèles conceptuels et
3. permettre d'organiser l'information de manière à en favoriser le traitement par un opérateur humain.

Même si certaines des techniques présentées sont automatisables, elles nécessitent le plus souvent des aménagements importants des modèles conceptuels et ne se soucient que peu, ou pas, des limites de la capacité de traitement de l'information des utilisateurs.

Comme nous l'avons vu à la section 4.1, les trois principaux arguments en faveur de l'enrichissement des modèles conceptuels par des mécanismes d'abstraction permettant de représenter et de manipuler des portions de schéma comme un tout sont :

- gérer la complexité,
- augmenter le pouvoir d'expression des modèles et
- faciliter la réutilisation de portions de schéma.

Selon nous, ces trois arguments sont faux.

Les difficultés éprouvées face à un schéma conceptuel tiennent aux limites de notre capacité à traiter l'information. Ces difficultés sont le reflet de la complexité du domaine d'application représenté et dépendent en partie du modèle conceptuel utilisé. N'ayant de prise ni sur la complexité des domaines d'application ni sur les faiblesses de la nature humaine, l'intervention est limitée aux modèles. Ces derniers peuvent produire des schémas d'autant plus simples qu'ils permettent de masquer la complexité non pertinente et qu'ils permettent d'exprimer d'une manière univoque la plus simple et la plus directe possible une correspondance claire entre la structure d'un domaine d'application et celle du schéma qui le décrit[22]. Or, les constructions

[21] Toute bonne extension d'un modèle doit permettre de résoudre plusieurs problèmes par le biais d'une simple addition [Mey01].

[22] Cette correspondance permet, entre autres, de respecter le principe de continuité [Mey97] qui veut que tout changement d'un domaine d'application entraîne une modification d'importance proportionnelle du schéma qui le représente.

de haut niveau compliquent de manière considérable les modèles conceptuels à base d'objets :

- elles changent l'interprétation de ces modèles de manière fondamentale : on passe de classes qui correspondent à des types de données abstraits à des classes complexes qui représentent des portions de schéma ;
- elles multiplient le nombre de concepts nouveaux : avec les classes complexes viennent les associations complexes, les associations entre classes complexes, les généralisations entre classes complexes et toutes les variétés d'associations entre classes de bas niveau appartenant à des classes complexes différentes.

Bref, on est loin de la simplicité et de la relative uniformité que procuraient les modèles de départ. Cette complication du modèle en rend la maîtrise plus difficile, ce qui a pour conséquence de compliquer la création, la compréhension et la maintenance des schémas.

De plus, l'augmentation du pouvoir d'expression des modèles promise par ces nouveaux mécanismes d'abstraction n'est pas au rendez-vous. En pratique les schémas multi-niveaux constituent des représentations moins directes des domaines d'application. En effet, l'aspect "gigogne" que présentent certains domaines d'application n'est généralement que superficiel. Prenons pour exemple la modélisation conceptuelle des macromolécules biologiques [MR98]. A première vue, il s'agit du domaine d'application idéal pour une représentation à plusieurs niveaux. Les manuels de base (par exemple [NC00]) comme les ouvrages plus spécialisés (par exemple [Les91]) proposent une description par niveau des protéines qui correspond au diagramme de la figure 10(a). Une protéine (PROTEIN) peut être vue comme un tout ou comme une association de chaînes peptidiques (PEPTIDIC_CHAIN). Ces dernières sont composées de résidus d'acides aminés (AA_RESIDUE) liés entre eux par des liaisons peptidiques (PEPTIDIC_BOUND) et les acides aminés sont eux-mêmes composés d'atomes (ATOM) liés entre eux par des liaisons covalentes (COVALENT_BOUND). Cette présentation par niveau est assez natu-

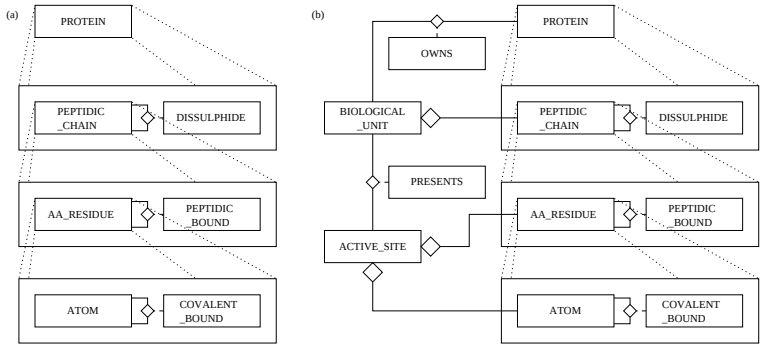

FIG. 10 – Représentation à l'aide de classes complexes (a) de la structure primaire des protéines, (b) de la structure primaire des protéines et des sites actifs de leurs unités biologiques.

relle et permet de bien comprendre ce qu'on appelle la structure primaire des protéines. Mais elle ne constitue qu'un aspect des protéines qui se déploient dans l'espace pour former des structures secondaires, tertiaires et quaternaires qui font apparaître, par exemple, des unités biologiques. Comme on peut le voir sur la figure 10(b), ces dernières ne se laissent pas enfermer dans un niveau de description. La structuration du schéma en niveaux est une simplification qui nous aide à comprendre les grandes lignes du domaine d'application mais dont la rigidité devient un handicap lorsqu'il s'agit d'appréhender la structure fine de ce domaine.

En ce qui concerne le réemploi des schémas conceptuels, là aussi la solution proposée par les

constructions complexes n'est, selon nous, pas adaptée. L'expérience des *patrons d'analyse*[23] [Fow97] montre que ce qui est réutilisé n'est pas des groupes de classes et d'associations avec l'ensemble complet de leurs propriétés et contraintes comme ceux qui définissent la structure des constructions complexes, mais bien, comme le montre l'exemple de la figure 11 des patrons, c'est-à-dire des portions de classes et d'associations, partiellement décrits par quelques propriétés et contraintes spécifiques d'un type de problème et devant être complétés pour s'adapter à chaque nouvelle situation.

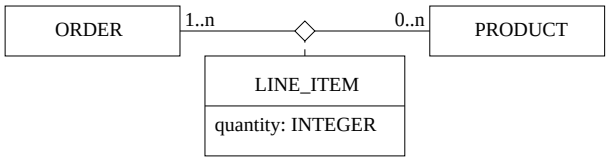

FIG. 11 – Patron d'analyse représentant la commande de produits [Fow97].

Selon nous, tout ceci plaide en faveur d'une contribution des modèles conceptuels à la gestion de la complexité qui se limite au choix du modèle conceptuel qui permet de représenter le domaine d'application à modéliser de la manière la plus directe et la plus naturelle possible. Essayer d'enrichir un tel modèle avec des mécanismes de gestion de la complexité est contreproductif. Du point de vue théorique, ce n'est pas fondé et, en pratique, ça ne marche pas.

Références

[ACW93] J. Akoka and I. Comyn-Wattiau. A framework for automatic clustering of semantic models. In Elmasri et al. [EKT93], pages 438–450.

[Aud95] A. Auddino. *A language for multilevel specification of information systems.* PhD thesis, Département d'informatique, École Polytechnique Fédérale de Lausanne, 1995.

[BCN92] C. Batini, S. Ceri, and S. Navathe. *Conceptual Database Design : An Entity-Relationship Approach.* Benjamin/Cummings, 1992.

[BDBS93] C. Batini, G. Di Battista, and G. Santucci. Stucturing primitives for a dictionary of Entity Relationship data schemas. *IEEE Trans. on Software Engineering*, 19(4) :344–365, April 1993.

[BETT99] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis. *Graph drawing.* Prentice Hall, 1999.

[CDAFP98] S. Castano, V. De Antonellis, M.G. Fugini, and B. Pernici. Conceptual schema analysis : Techniques and applications. *ACM Trans. on Database Systems*, 23(3) :286–333, 1998.

[CH94] L. Campbell and T. Halpin. Abstraction techniques for conceptual schemas. In *Proc. of the 5th Australasian Database Conference*, pages 374–388, 1994.

[CJA89] C.R. Carlson, W. Ji, and A.K. Arora. The nested entity-relationship model. In Lochovsky [Loc89], pages 43–57.

[23] Analysis Pattern.

[CWAK98] I. Comyn-Wattiau, J. Akoka, and Z. Kedad. Combining view integration and schema clustering to improve database design. In *Actes XIVèmes Journées Bases de Données Avancées*, Hammamet (Tunisie), October 1998.

[DT91] O. De Troyer. Schema object types : a new approach to modularization in conceptual modeling. In G. Saake and A. Sernadas, editors, *Information Systems - Correctness and Reusability*, September 1991.

[DTJ93] O. De Troyer and R. Janssen. On modularity for conceptual data models and the consequences for subtyping, inheritance & overriding. In *Proceedings of the 9th IEEE Conference on Data Engineering (ICDE 93)*. IEEE Computer Society Press, 1993.

[EKT93] R. Elmasri, V. Kouramajian, and B. Thalheim, editors. *Proc. of the 12th Int. Conf. on the Entity-Relationship Approach, ER'93*, LNCS 823, Dallas, Texas, 1993. Springer-Verlag.

[EKW92] D.W. Embley, B.D. Kurtz, and S.N. Woodfield. *Object-Oriented Systems Analysis : A Model-Driven Approach*. Yourdon Press, 1992.

[Ell82] H. Ellis. A refined model for the definition of systems requirements. *Database Journal*, 12(8), 1982.

[EN00] R. Elmasri and S. Navathe. *Fundamentals of Database Systems*. Addison-Wesley, third edition, 2000.

[FM86] P. Feldman and D. Miller. Entity model clustering : Structuring a data model by abstraction. *The Computer Journal*, 29(4) :348–360, 1986.

[Fow97] M. Fowler. *Analysis Patterns : Reusable Object Models*. Addison-Wesley, 1997.

[FP94] C. Francalanci and B. Pernici. Abstraction levels for entity-relationship schemas. In Loucopoulos [Lou94], pages 456–473.

[GRVG94] M. Gandhi, E. L. Robertson, and D. Van Gucht. Leveled entity-relationship model. In Loucopoulos [Lou94], pages 420–436.

[HJ94] P. Hartel and R. Jungclaus. Specifying business processes over objects. In Loucopoulos [Lou94], pages 10–27.

[HZ89] S. Huffman and R.V. Zoeller. A rule-based system tool for automated ER model clustering. In Lochovsky [Loc89], pages 221–236.

[JOS93] P. Jaeschke, A. Oberweis, and W. Stucky. Extending ER model clustering by relationship clustering. In Elmasri et al. [EKT93], pages 451–462.

[Kri94] B.B. Kristensen. Complex associations : Abstractions in object-oriented modeling. In *Proc. of the 9th Conf. on Object-Oriented Programming Systems, Languages and Applications, OOPSLA'94*, pages 272–286, Portland, Oregon, 1994. ACM SIGPLAN Notices 29(10), 1994.

[Les91] A.M. Lesk. *Protein Architecture : A Practical Approach*. IRL Press, 1991.

[Loc89] F.H. Lochovsky, editor. *Proc. of the 8th Int. Conf. on the Entity-Relationship Approach, ER'89*, Toronto, Canada, 1989.

[Lou94] P. Loucopoulos, editor. *Business Modeling and Re-Engineering, Proc. of the 13th Int. Conf. on the Entity-Relationship Approach, ER'94*, LNCS 881, Manchester, UK, 1994. Springer-Verlag.

[Mas97a] D. Massart. Complexity management of object schemas. Technical Report YEROOS TR-97/07, IAG-QANT, Université catholique de Louvain, Belgium, March 1997.

[Mas97b] D. Massart. On the status of high-level relationships in complexity management of conceptual schemas. Technical Report YEROOS TR-97/08, IAG-QANT, Université catholique de Louvain, Belgium, April 1997.

[Mas01] D. Massart. La complexité des schémas conceptuels à base d'objets. Technical Report 15/01, Institut d'Administration et de Gestion, Université catholique de Louvain, Belgium, October 2001.

[Mey97] B. Meyer. *Object-oriented Software Construction*. Prentice Hall, second edition, 1997.

[Mey01] B. Meyer. *Eiffel : The Language*. Prentice Hall, third edition, 2001. To appear (http://www.inf.ethz.ch/personal/meyer/ongoing/etl/).

[MGG95] R. Missaoui, J.M. Gagnon, and R. Godin. Mapping an extended entity-relationship schema into a schema of complex objects. In M. Papazoglou, editor, *Proc. of the 14th Int. Conf. on Object-Oriented and Entity-Relationship Modelling, OOER'95*, LNCS 1021, Gold Coast, Australia, 1995. Springer-Verlag.

[Moo97] D. Moody. A multi-level architecture for representing enterprise data models. In D.W. Embley and R.C. Goldstein, editors, *Proc. of the 16th Int. Conf. on Conceptual Modeling, ER'97*, LNCS 1331, pages 184–197, Los Angeles, California, 1997. Springer-Verlag.

[MP92] D.E. Monarchi and G.I. Puhr. A research typology for object-oriented analysis and design. *Communications of the ACM*, 35(9) :35–47, September 1992.

[MR98] D. Massart and J. Richelle. Object-oriented conceptual modeling of databases for macromolecular structures. In P.J. Charrel, H. Jaakkola, H. Kangassalo, and E. Kawaguchi, editors, *Information Modelling and Knowledge Bases IX*, pages 146–159. IOS Press, 1998.

[Mut01] P. Mutzel. Optimization in graph drawing. Technische Universität Wien, http://www.ads.tuwien.ac.at/, 2001.

[NC00] D.L. Nelson and M.M. Cox. *Lehninger Principles of Biochemistry*. Worth Publishers, third edition, 2000.

[RJB99] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language : Reference Manual*. Addison-Wesley, 1999.

[RS92] O. Rauh and E. Stickel. Entity tree clustering : A method for simplifying ER designs. In G. Pernul and A.M. Tjoa, editors, *Proc. of the 11th Int. Conf. on the Entity-Relationship Approach, ER'92*, LNCS 645, pages 62–78, Karlsruhe, Germany, 1992. Springer-Verlag.

[Sel93] H. Seltveit. An abstraction-based rule approach to large-scale information systems development. In C. Rolland, F. Bodart, and C. Cauvet, editors, *Proc. of the 5th Int. Conf. on Advanced Information Systems Engineering, CAiSE'93*, LNCS 685, pages 328–351, Paris, France, 1993. Springer-Verlag.

[TWBK89] T.J. Teorey, G. Wei, D.L. Bolton, and J.A. Koenig. ER model clustering as an aid for user communication and documentation in database design. *Communications of the ACM*, 32(8) :975–987, August 1989.

[Ver83] D. Vermeir. Semantic hierarchies and abstractions in conceptual schemata. *Information Systems*, 8(2) :117–124, 1983.

[You89] E. Yourdon. *Modern structured analysis*. Yourdon Press, 1989.