Principes et conseils de structure de modèle Pour Rational Software Modeler et Rational Software Architect (Version 2004)

Livre blanc

Bill Smith, Développement dirigé par modèle, IBM Rational Software

V1.0
8 septembre 2004

Table des matières

1. Introduction 3

Public ciblé 3

Objectif 3

Etendue 4

Conventions typographiques 4

Organisation du livre blanc 5

2. Concepts et terminologie de base 5

Modèles 5
Fichiers de modélisation 6
Types de modèle 6
Espaces de travail, projets et types de projets 7
Concepts examinés 8

3. Mappage du modèle RUP au modèle RSA 11

Types de modèle RSA 11
Modèle vierge 11
Modèle de cas d'utilisation 12
Modèle d'analyse 13
Modèle de conception Enterprise IT 14
Modèle de présentation de l'implémentation 15
Modèle d'implémentation 15
“Modèles croquis” 15

4. Recommandations et techniques générales pour l'organisation des structures internes de modèles 16

Représenter les points de vue en utilisant«des paquetages» de perspective 16

Créer des représentations à mise à jour automatique des préoccupations spécifiques en utilisant des diagrammes de rubriques 16

Examiner les modèles avec des diagrammes de parcours 17

Navigation inter-diagramme 17

5. Principes et conseils pour l'organisation interne du modèle de cas d'utilisation 18

Organisation de haut niveau du modèle de cas d'utilisation 18

Contenu du modèle de cas d'utilisation 20

6. Principes et conseils pour l'organisation interne du modèle d'analyse 23

Analyse Organisation de haut niveau du modèle 24

Contenu du modèle d'analyse 26

7. Principes et conseils pour l'organisation interne du modèle de conception 30

Conception Organisation de haut niveau du modèle 30

Contenu du modèle de conception 33

8. Principes et conseils pour l'organisation interne du modèle de présentation de l'implémentation 40

9. Principes et conseils pour l'organisation interne du modèle de déploiement 42

10. Utiliser un fichier de modélisation pour représenter le plan d'architecture logicielle 43

11. Considérations de développement coopératif 45

Modéliser en équipes 45

Deux approches au partitionnement de modèle 45

Approche planifiée : Décomposer les modèles dès le début 45

Approche ad-hoc : Réusinage de modèle 46

Références inter-fichiers 46

1. Introduction

Public ciblé

Ce livre assure le support des utilisateurs du produit Rational Software Architect (RSA) souhaitant appliquer les conseils du Rational Unified Process (RUP®) à leur utilisation de RSA. Si vous êtes utilisateur de Rational Software Modeler (RSM), ce livre blanc vous sera utile mais soyez conscients du fait que certaines sections expriment des capacités uniquement disponibles dans RSA et non RSM. Le livre blanc s'applique principalement au cas où vous créez un nouvel ensemble de modèles dans RSA. Si vous êtes nouvel utilisateur de RSA mais que vous connaissez Rational Rose ou Rational XDE et que vous avez importé des modèles de ces produits, ce livre blanc peut vous apporter des conseils pour la restructuration de vos modèles importés.

Ce livre blanc suppose que vous connaissez assez bien UML, ainsi que les concepts fondamentaux et le fonctionnement de RSA, en particulier la modélisation, les transformations et l'édition de code visuel.

Objectif

Le RUP décrit un ensemble de modèles représentant des points de vue bien définis sur le problème et les domaines de solution des systèmes. L'utilité de cet ensemble de structure de modèles a été prouvée dans bien des projets réels, et que vous suiviez un processus formel (comme le RUP) ou non, il est utile de tenir compte de ces structures de modèle. Ce document examine comment réaliser les artefacts de modèle RUP en utilisant RSA.

Comme le titre l'indique, les structures de projet et de modèle décrites dans ce livre blanc sont des conseils et non des impératifs. Que vous décidiez de modéliser un artefact RUP spécifique dans RSA dépend de votre propre processus de développement et cette décision est souvent spécifique au projet. Souvenez-vous que le RUP n'est pas un ensemble rigide de règles de processus. C'est une *infrastructure*de processus dans laquelle vous formulez des définitions de processus, qu'elles soient très formelles ou bien très simples.

La manière d'utiliser UML peut également avoir un degré de formalité varié. Vous pouvez décider de considérer vos modèles UML comme des dessins architecturaux formels devant être suivis à la lettre pendant la construction, ou bien comme des croquis évoquant les grandes lignes d'une conception, mais considérés comme jetables une fois que le projet passe en phase d'implémentation. RSA peut vous prendre en charge à tout stade du processus et de la modélisation. De ce point de vue, les principes et conseils présentés dans ce livre blanc n'ont pas pour but de brider votre raisonnement, mais de vous aider à comprendre comment utiliser les fonctions de RSA pour faciliter le processus qui semble vous correspondre le mieux.

Notez que RSA permet d'utiliser les modèles comme plan d'action, mais également en tant que spécification d'une solution à partir de laquelle les implémentations peuvent être générées automatiquement. On obtient ce résultat en utilisant des transformations RSA modèle-à-modèle et modèle-à-code. L'utilisation de transformations RSA pour pratiquer le développement dirigé par modèle (MDD) introduit des préoccupations spécifiques au sujet des structures de modèle. Si vous utilisez des modèles et des transformations RSA pour pratiquer le développement dirigé par modèle, vous devez également consulter les différentes ressources RSA spécifiques au développement dirigé par modèle disponibles sur Developer Works.

Etendue

Ce livre blanc décrit comment représenter les artefacts de modèle RUP dans RSA et propose des conseils pour les structures organisationnelles internes de ces artefacts. Il *n'essaie pas*de :

- Redéfinir les fondements conceptuels des artefacts de modèle RUP, ni fournir des descriptions détaillées de ces artefacts

- Décrire le processus ou les techniques de spécification de la sémantique détaillée ni du contenu en termes de diagrammes des artefacts RUP associés

Pour des informations adaptées à tous les outils de la manière de définir, développer et modéliser le contenu des artefacts RUP, voir le RUP.

Pour des informations sur des techniques spécifiques à l'outil pour développer le contenu des modèles RSA voir :

- La documentation produit (tutoriels, échantillons, aide en ligne)

- Les mentors outils dans la configuration RUP spécifique à RSA, dont ce livre blanc fait partie

- Les ressources liées au RSA dans Developer Works

Conventions typographiques

Des discussions intéressantes pour les utilisateurs de RSA qui migrent de Rational Rose ou XDE d'IBM sont présentées sous la forme d'une barre d'options latérale, dans une zone de texte encadrée avec un fond ombré :

> ***XDE/Rose***
> Discussions intéressantes pour les utilisateurs XDE ou Rose précédents.

Organisation du livre blanc

La section Concepts et terminologie de base qui suit établit un vocabulaire de travail et fournit des informations générales sur la façon dont les modèles sont implémentés dans le produit RSA.

Ensuite, la section Mappage du modèle RUP au modèle RSA définit comment RSA prend en charge les types de modèles définis par le RUP.

Les sections suivantes fournissent des conseils pour structurer des modèles de différents types dans RSA. Certaines de ces sections examinent plusieurs manières d'utiliser un type de modèle selon le degré de rigueur que vous préférez en termes de processus, d'approche de modélisation et de contrôle architectural.

Pour finir, une discussion traite de plusieurs problèmes liés à l'usinage de modèles en multiples fichiers de modélisation. Elle a trait aux stratégies de gestion de l'échelle et à l'activation du partage des modèles entre les membres de l'équipe afin de minimiser les conflits d'utilisation des fichiers et de fusions.

2. Concepts et terminologie de base

Modèles

Dans le RUP, un modèle est défini en tant que spécification complète d'un domaine de problème ou de solution d'un point de vue spécifique. Un domaine de problème ou un système peut être choisi par un certain nombre de modèles qui représentent différents points de vue sur le domaine ou le système. Le RUP propose un ensemble spécifique de modèles :

- Modèle de cas d'utilisation métier
- Modèle d'analyse métier
- Modèle de cas d'utilisation
- Modèle d'analyse (peut être inclut dans le modèle de conception)
- Modèle de conception
- Modèle d'implémentation
- Modèle de déploiement
- Modèle de données

Notez que le RUP lui-même est indifférent à l'outil. Du point de vue du RUP, un modèle peut être un dessin sur une serviette en papier ou un tableau, une partie d'un outil de modélisation ou même une image mentale. Ainsi, selon le RUP, un modèle est un concept logique. Dans le contexte du RSA, nous pouvons examiner les modèles en termes logiques, mais également en termes physiques.

Supposons que des équipes travaillent sur deux applications : une équipe de trois analystes travaillant sur une application de gestion des feuilles de présence, et une deuxième équipe de cinq analystes travaillant sur une application de gestion d'un central téléphonique. Les deux équipes travaillent actuellement afin de consigner des exigences et utilisent RSA pour la modélisation de cas d'utilisation. En termes RUP on peut dire qu'une équipe construit “le modèle de cas d'utilisation pour l'application de feuilles de présence” et que l'autre construit “le modèle de cas d'utilisation pour l'application de gestion d'un central téléphonique.” Cependant, si les équipes utilisent RSA, il est important de reconnaître que leurs modèles possèdent des manifestations physiques spécifiques. C'est l'objet de la prochaine section.

Fichiers de modélisation

Dans RSA, les modèles sont conservés en tant que fichiers. (Dans la terminologie Eclipse, on considère un fichier comme une "ressource" [1], ainsi, si vous trouvez le terme "ressource de modélisation" dans ce livre blanc ou tout autre source, cela désigne la même chose que "fichier de modélisation"). Dans un sens plus large, RSA prend en charge deux types de fichiers de modélisation :

- “Fichiers de modélisation de pré-implémentation”. Ces fichiers contiennent un contenu UML conceptuel qui ne reflète pas*directement*les artefacts d'implémentation. Ces fichiers contiennent à la fois la sémantique UML du modèle et les diagrammes UML qui décrivent les éléments sémantiques.

- Les fichiers de modélisation d'implémentation (ressources Eclipse), des artefacts d'implémentation normaux comme les fichiers source Java ou des pages Web qui résident dans un projet Eclipse. Dans ce cas, vous pouvez considérer que le projet représente l'étendue du contenu du fichier de modélisation d'implémentation. La sémantique de modèle se trouve dans les artefacts d'implémentation eux-mêmes. Chaque diagramme réside dans son propre fichier physique dans le projet. Ces diagrammes peuvent utiliser la notation UML, mais ils peuvent également utiliser d'autres notations (par exemple, IDEF1X ou notations d'ingénierie informatique pour la visualisation de données, ou bien une notation Rational utilisée pour la conception de niveaux Web). Les types de diagrammes UML pris en charge pour décrire les artefacts de code 3GL comportent les diagrammes de classe et les diagrammes de séquence (pour Java uniquement).

Ce livre blanc se focalise sur la manière d'organiser les structures internes des modèles de “pré-implémentation”, et **dans ce document le terme “fichier de modélisation” est réservé aux fichiers contenant un contenu de modèle de “pré-implémentation”** . Vous pouvez trouver des conseils sur l'organisation du contenu des projets d'implémentation dans d'autres sources comme l'aide en ligne pour Rational Software Architect, Rational Application Developer, et Rational Web Developer.

Un fichier de modélisation de “pré-implémentation” ne contient pas nécessairement toute l'information pour un modèle. En fait, un fichier de modélisation contiendra souvent uniquement un sous-ensemble d'un modèle. Par exemple, dans l'exemple ci-dessus dans lequel une équipe de trois personnes travaille sur un modèle de cas d'utilisation pour une application de feuille de temps, l'équipe peut décider de partitionner physiquement son modèle de cas d'utilisation en trois fichiers de modélisation pour que chaque membre de l'équipe puisse travailler sur un sous-ensemble différents des cas d'utilisation sans conflit de fichier. La dernière partie du livre blanc examine les problèmes liés au partitionnement des modèles et à la gestion de fichiers de modélisation.

Types de modèle

Dans le RUP, les modèles ont des types spécifiques comme le modèle de cas d'utilisation, le modèle d'analyse ou le modèle de données. Dans RSA vous pouvez considérer qu'un fichier de modélisation possède un type : le type du modèle (ou sous-ensemble de modèle) que le fichier contient. Le type d'un fichier de modélisation peut être établi de deux manières différentes :

- Commencez par un fichier de modélisation “vierge” (voir ci-dessous) puis établissez son type selon la manière dont vous choisissez de l'appeler et le type de contenu que vous y intégrez (y compris les profils UML que vous appliquez).

[1] Dans Eclipse une ressource est un fichier, mais possède également des propriétés et un comportement supplémentaire dans l'environnement Eclipse. Les fichiers de modélisations décrits ici sont traités comme des "ressources" par Eclipse.

- Créez-le en vous basant sur un “modèle” pré-défini qui représente un type de modèle spécifique. Vous remarquerez que le “type” d'un fichier de modélisation n'est qu'une convention concernant le contenu du fichier. Par exemple, l'outil n'empêchera pas un fichier contenant un modèle de cas d'utilisation de contenir les classes réalisant les cas d'utilisation. Cependant, ces principes et conseils vous recommandent de traiter les fichiers de modèles comme dactylographiés.

Espaces de travail, projets et types de projets

Les lecteurs connaissant bien Eclipse, les produits WebSphere Studio, ou Rational Application Developer savent déjà que les fichiers sont situés dans des Projets, que les Projets peuvent être de types différents, et que les Projets sont (virtuellement) regroupés et gérés dans des espaces de travail. Les fichiers de modélisation RSA se situent dans des projets comme les autres fichiers.

Pour mener à bien cette discussion, il n'est pas nécessaire d'expliquer en détails tous les types de projet disponibles dans Rational Software Architect, Rational Application Developer et Rational Software Modeler. De façon générale, nous nous intéressons à deux catégories de projets :

- **Projets UML**

- **Les projets d'implémentation**, qui incluent les types spécialisés comme les projets Enterprise, EJB, Web, et C++

Nous avons expliqué auparavant que RSA prend en charge deux types de fichiers de modélisation :

- Les fichiers contenant des modèles UML (éléments sémantiques ainsi que des diagrammes les décrivant) utilisés pour la modélisation à des niveaux d'abstraction au-delà de l'implémentation (comme les exigences, l'analyse et la conception)

- Les projets qui contiennent des artefacts d'implémentation comme le code Java ou les pages Web plus (facultatif) les fichiers de diagrammes contenant des diagrammes décrivant les artefacts d'implémentation.

La règle d'attribution des modèles aux projets est simple : **a)**déposez les fichiers de modélisation de“ pré-implémentation” dans les projets UML, et **b)** les modèles d'implémentation se gèrent eux-mêmes car, en essence :

[modèle d'implémentation] = [projet d'implémentation]

Il y a des exceptions à cette règle. Les fichiers de modélisations UML suivants peuvent être déposés dans un projet d'implémentation spécifique à la langue :

- "Modèles de croquis" de conception (qui seront examinés dans un paragraphe ultérieur)

- Modèles possédant des diagrammes de séquence décrivant les tests exécutés par rapport au code dans le projet

> ***XDE/Rose***
> Les principes de fonctionnement de Rose et XDE incluent le perfectionnement itératif du modèle de conception jusqu'à ce que vous atteignez un niveau d'abstraction équivalent au code, puis l'utilisation d'une technologie de synchronisation de code à modèle afin que la sémantique de ce modèle reste en phase avec le code lui-même. Par exemple, dans XDE, les modèles d'implémentation n'existent pas seulement en tant que code et diagrammes dans des projets, mais également en tant que fichiers "modèle de code" conservés indépendamment des artefacts d'implémentation et qui en essence représentent une copie redondante de leur sémantique.
>
> Les principes de fonctionnement de RSA encouragent l'utilisation de modèles indifférents à la plate-forme à des niveaux d'abstraction supérieurs au code (c'est-à-dire des modèles de conception comme un modèle de conception Enterprise IT) et l'utilisation des transformations pour générer le code à partir de ces modèles. Puis, au niveau de code de l'abstraction, RSA vous permet tout simplement de dessiner des diagrammes de code UML, sans avoir à utiliser un modèle sémantique conservé séparément.
>
> Notez que RSA ne vous*empêche* pas de définir des modèles UML à un niveau de code d'abstraction et de générer du code à partir de ces modèles ; en fait, ce type d'utilisation est prévu. Mais RSA ne fournit pas la technologie pour que ce type de modèle reste synchronisé avec le code.

Concepts examinés

Les illustrations suivantes résument les discussions précédentes. Les illustrations reflètent le scénario décrit plus tôt, dans lequel nous avons deux équipes, l'une travaillant sur une application de centre d'appels et l'autre sur une application de feuille de temps.

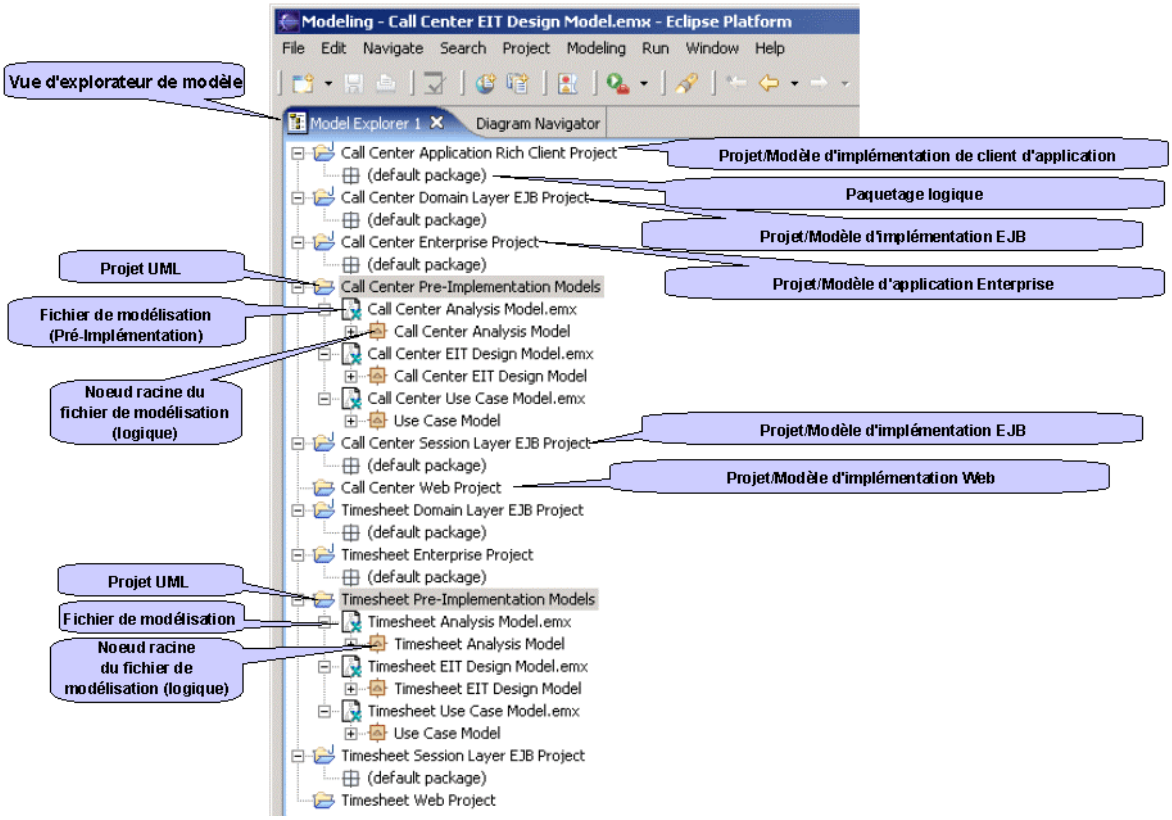


Figure 2-1

RSA fournit une vue d'explorateur de modèle qui apporte une vue de modèle à la fois physique et logique. Dans l'explorateur de modèle les projets de votre espace de travail sont décrits en tant que noeuds généraux, et dans chaque projet vous pouvez voir les ressources appartenant à ce projet. **Figure 2-1** décrit dans l'explorateur de modèle un assortiment de projets correspondant aux deux applications dans notre scénario. Les projets UML ont été utilisés pour les modèles de pré-implémentation. Un assortiment d'autres types de projets adaptés à la solution (comme les projets Enterprise Application, Web, etc.) a été utilisé pour les modèles d'implémentation.

> ***XDE/Rose***
> Contrairement à l'explorateur de modèle RSA, les explorateurs de modèle de Rose et XDE fournissent uniquement une vue logique des modèles. Notez que la vue de ressources fournie par l'explorateur de modèle RSA n'est pas la vue physique "pure" fournie par la vue Eclipse Navigator. Certaines ressources physiques sont disponibles dans l'explorateur de modèle, mais elles sont représentées principalement par des icônes indiquant des vues *logiques* des ressources.

La figure 2-2 montre comment le modèle de cas d'utilisation de feuille de temps peut être organisé de façon interne en paquetages représentant des sous-ensembles fonctionnellement cohésifs du domaine de problème.

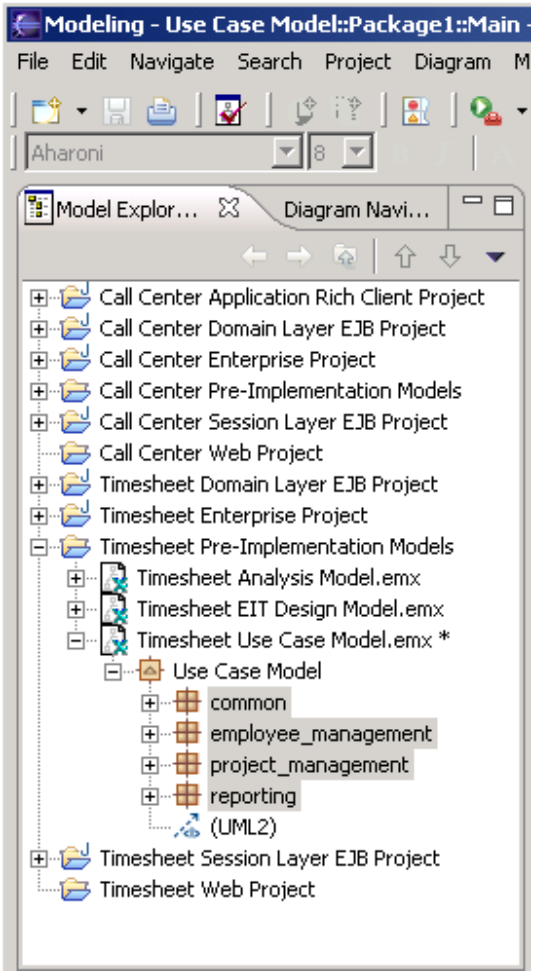


Figure 2-2

Entrée **Figure 2-1** et **2-2** chaque modèle de pré-implémentation réside dans un seul fichier de modélisation. Ils peuvent tous être réusinés en fichiers de modélisation multiples. Par exemple le modèle de cas d'utilisation de feuille de temps peut être réusiné en quatre fichiers de modélisation correspondant chacun à un des sous-ensembles décrits du domaine de problème (“général”, “gestion_du_personnel”, “gestion_de_projet”, et “compte-rendu”). Dans ce cas le noeud racine de chaque fichier de modélisation sera nommé afin de conserver un espace de nom cohérent dans tous les fichiers de modélisation qui forment le modèle de cas d'utilisation complet. Par exemple les noeuds racine des quatre fichiers de modélisation peuvent être “exigences.feuilled'heures.général”, “exigences.feuilled'heures.gestion_du_personnel”, “exigences.feuillesd'heures.gestion_de_projet” et “exigences.feuilled'heures.compte-rendu”.

3. Mappage de modèle RUP à modèle RSA

Le tableau suivant montre comment les modèles RUP les plus utilisés sont mappés à des types de modèles RSA. Le mappage est généralement direct, mais il permet d'utiliser ce livre blanc comme guide pour pratiquer le RUP avec RSA. Les types de modèle RSA mentionnés dans le tableau sont examinés directement après ce tableau. Nous vous proposons dans des paragraphes ultérieurs des conseils concernant l'organisation interne des différents types de modèle, et les types de projets pouvant les accueillir. Ces discussions ultérieures sont présentées en fonction des types de modèle RSA répertoriés ici.

Modèle RUP	Type de modèle RSA
Modèle de cas d'utilisation	Modèle de cas d'utilisation
Modèle d'analyse	Modèle d'analyse (autre possibilité : paquetages d'"analyse" dans le modèle de conception)
Modèle de conception	Pour les applications commerciales à plusieurs niveaux : Enterprise IT Design Model Pour d'autres types d'applications : Modèle vierge utilisé comme modèle de conception Pour les "croquis" de conception : Modèle vierge utilisé en tant que "croquis" de modèle de conception Ajout facultatif : Modèle vierge utilisé en tant que modèle de présentation de l'implémentation
Modèle d'implémentation	Projets d'implémentation contenant des artefacts d'implémentation et des fichiers de diagramme
Modèle de déploiement	Modèle vierge utilisé en tant que modèle de déploiement

Types de modèle RSA

Modèle vierge

RSA donne la possibilité de créer un (fichier) “modèle vierge” →Nouveau →Modèle UML→Modèle vierge). Un “modèle vierge” est un fichier de modélisation ne se basant pas sur un modèle. Il n'a aucun profil spécifique appliqué, et pas de contenu par défaut à l'exception d'un seul diagramme (libre) “Principal”. **Vous pouvez utiliser un fichier de modélisation vierge comme point de départ pour tout type de modèle**. En choisissant la façon de le nommer, le contenu que vous définissez, et les profils que vous y appliquez, vous pouvez utiliser un fichier de modélisation vierge pour construire un modèle de cas d'utilisation, un modèle d'analyse, un modèle de conception, un modèle de déploiement ou tout autre type de modèle RUP.

Modèle de cas d'utilisation

RSA donne la possibilité de créer un fichier de “modèle de cas d'utilisation” basé sur un modèle. Le modèle apporte un contenu par défaut définit dans ***Figure 3-1***. (Ce document n'expliquera pas l'utilisation des chaînes de contenu et de recherche “ d'unité élémentaire de structure”. Les modèles contiennent des instructions, vous les trouverez donc globalement évidents.)

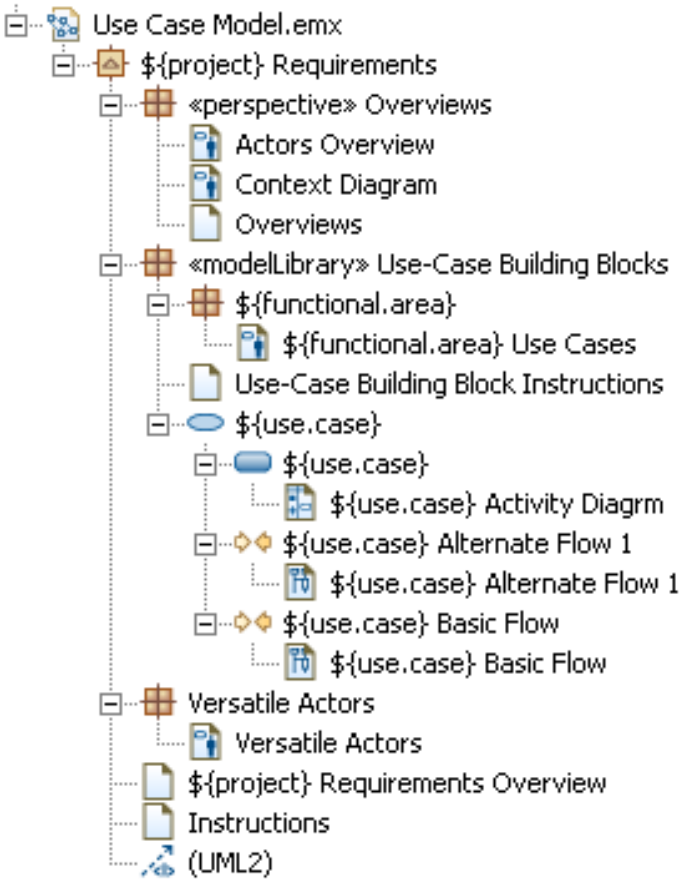


Figure 3-1

Modèle d'analyse

RSA donne la possibilité de créer un fichier de “modèle d'analyse” basé sur un modèle. Le modèle apporte un contenu par défaut définit dans ***Figure 3-2***. De plus, un profil d'“analyse” est appliqué aux fichiers de modélisation créés à partir de ce modèle :

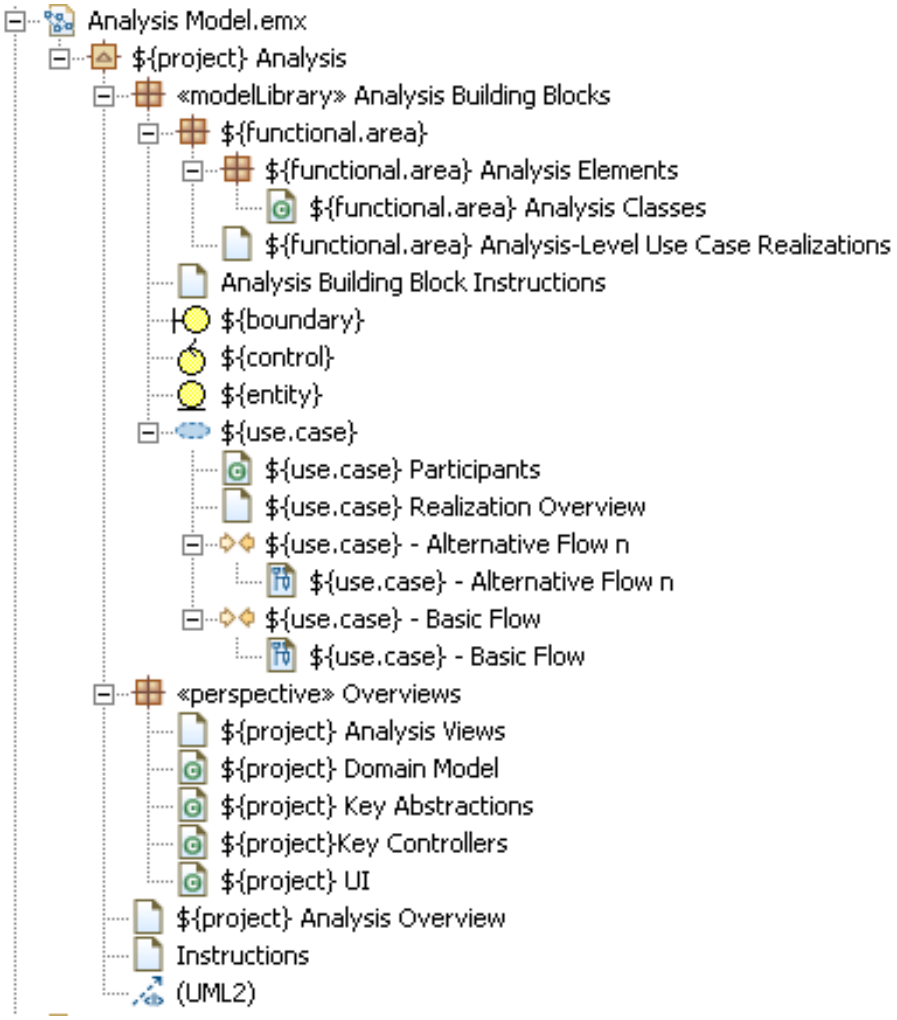


Figure 3-2

Modèle de conception Enterprise IT

RSA donne la possibilité de créer un fichier de “modèle de conception Enterprise IT” basé sur un modèle. Le modèle apporte un contenu par défaut définit dans **Figure 3-3**. Egalement un profil “Transformation EJB” [2] sera appliqué aux fichiers de modélisation créés à partir de ce modèle. C'est le modèle approprié à utiliser pour la conception (et éventuellement pour l'analyse) lorsque l'on cible les applications métier et qu'on utilise des transformations RSA générant du code pour prendre en charge la création de ce type d'applications.

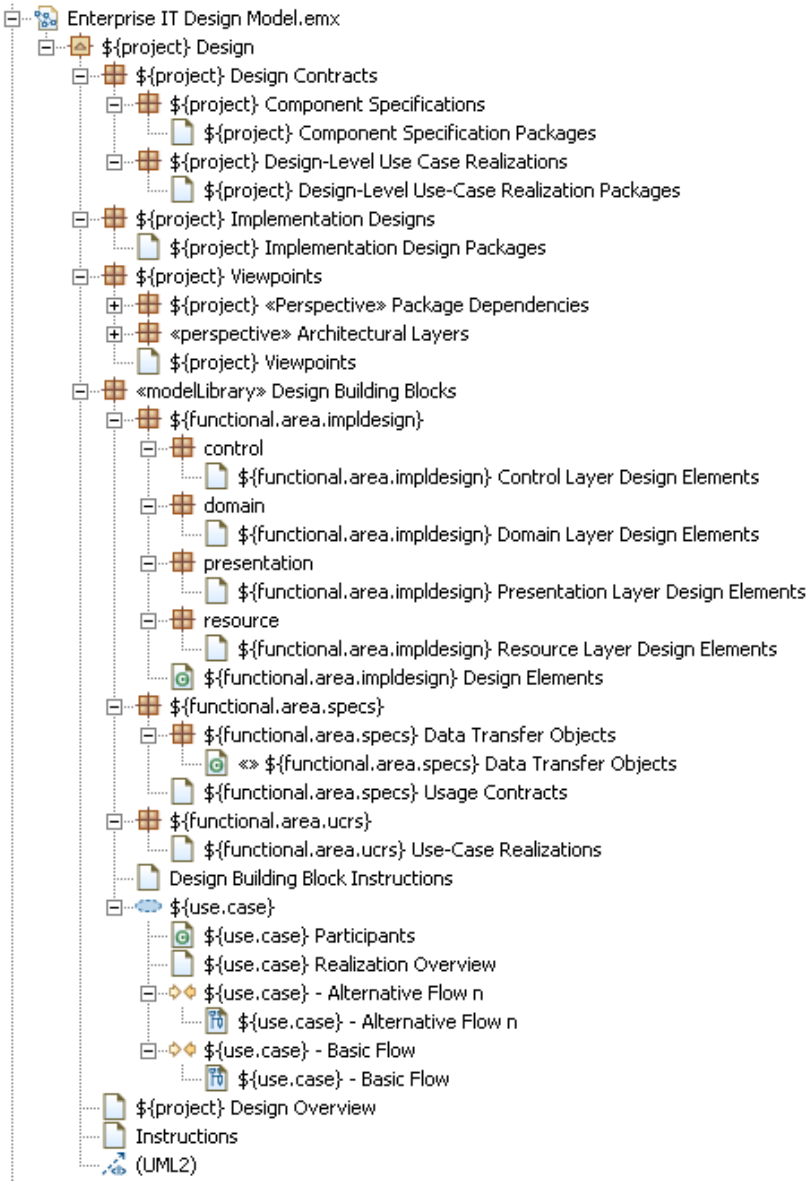


Figure 3-3

[2] L'ensemble de profils permettant une transformation fournie dans le cadre du modèle de conception EIT évoluera probablement au fur et à mesure de la mise sur le marché de mises à jour du produit.

Modèle de présentation de l'implémentation

Dans le cadre de votre modèle de conception, vous trouverez peut-être utile de définir un “modèle de présentation d'implémentation” pour consigner une vue générale de la façon dont l'implémentation doit être organisée. Le “modèle de présentation de l'implémentation” serait utilisé tôt dans la phase de conception—avant que le code ne soit généré ou écrit—afin de représenter les véritables projets et dossiers/paquetages RSA où vous pensez que le code et les fichiers associés (méta-données, descripteurs de déploiement, etc.) se trouvent. Vous pouvez également utiliser ce modèle pour montrer les dépendances prévues entre ces projets et paquetages, ce qui peut être utile pour identifier les exigences de la construction du système. Le modèle de présentation de l'implémentation peut également servir à conserver des diagrammes conceptuels informels de l'architecture de la solution.

Modèle d'implémentation

Comme nous l'avons dit précédemment, dans RSA un modèle d'implémentation consiste d'un projet contenant des artefacts d'implémentation et (éventuellement) des diagrammes décrivant ces artefacts [3].

“Croquis de ” modèles

Comme vous pouvez le voir dans la section “Concepts et terminologie de base”, vous pouvez choisir de traiter les modèles de conception comme des dessins architecturaux formels gérés pendant la durée de vie du système et utilisés pour prendre en charge/appliquer le contrôle architectural. Ou bien, vous pouvez les traiter comme des croquis servant à suggérer une conception et à aider à la clarifier et à la communiquer, mais considérées comme jetables une fois que la véritable implémentation a commencé à s'éloigner de la conception d'origine. RSA prend en charge les deux approches. Ses fonctions ne ciblent pas généralement une approche ou une autre, mais vos choix d'utilisation des modèles de conception permettent à l'évidence de déterminer les fonctions RSA que vous utilisez et votre façon de les utiliser. Lorsque la distinction est importante dans le contexte des principes et conseils présentés dans ce livre blanc, le terme “modèle croquis” sera utilisé pour indiquer que le modèle est utilisée d'une façon plus "jetable".

[3] Pour créer ces diagrammes, plutôt que d'utiliser Fichier→Nouveau→Modèle UML pour créer un modèle, utilisez Fichier→Nouveau→Diagramme de classe pour créer un diagramme dans lequel vous pouvez composer des "vues" de code dans la notation UML (ou autre). Chaque diagramme individuel est conservé en tant que fichier séparéavec une extension .dnx, et peut être géré selon la version principalement de la même façon qu'un fichier de code. Ces diagrammes ne contiennent pas d'information sémantique, mais uniquement une notation. Toutes les informatiques sémantiques pertinentes se trouvent dans le code lui-même. Lorsque vous changez quelque chose comme un nom de classe ou une signature d'opération dans l'un de ces diagrammes, vous changez en fait le code sous-jacent lui-même. Lorsque vous faites ce type de changement dans le code (en utilisant un éditeur de texte), les diagrammes dans lesquels les codes modifiés apparaissent sont automatiquement mis à jour.

4. Principes et techniques généraux pour l'organisation des structures internes de modèles

L'outil principal d'organisation du contenu des modèles UML est le paquetage. Les paquetages UML ont deux objectifs principaux :

- partitionner, organiser et étiqueter les informations de modèle
    - regrouper des éléments correspondant à une rubrique spécifique du domaine de problème ou de solution
    - séparer les différents types d'informations de modèle comme les interfaces, les implémentations, les diagrammes, etc
    - regrouper les éléments afin de définir et de contrôler leurs dépendances vis-à-vis d'autres éléments
    - regrouper les diagrammes fournissant des vues alternatives sur le même modèle

- établir les espaces de noms
    - pour les éléments de modèle
    - pour les artefacts d'implémentation générés à partir d'éléments de modèle (cela peut impliquer des mappages entre les espaces de noms des langues de modèle et d'implémentation
    - pour une unité de réutilisation

De manière générale, le RUP a proposé des stratégies spécifiques de paquetage pour les différents types de modèle. Ces stratégies sont reflétées dans les sections de ce livre blanc traitant des types de modèle. RSA introduit également quelques outils organisationnels supplémentaires, décrits ici :

Représenter des points de vue en utilisant les paquetages de«perspective»

Lorsque l'on souhaite que les éléments soient organisés de plus d'une seule façon, vous pouvez créer des paquetages supplémentaires avec des diagrammes décrivant les schémas organisationnels alternatifs. Cette même technique peut servir à chaque fois qu'il est nécessaire de représenter une vue spécifique du contenu du modèle recoupant le schéma de paquetage du modèle. RSA prend cette technique en charge en fournissant un stéréotype de paquetage de «perspective» dans le cadre du "profil de base" UML. Un paquetage de "perspective" peut être considéré comme équivalent à un “point de vue” RUP for Systems Engineering ou IEEE 1417.

Ne placez pas les éléments sémantiques (classes, paquetages, associations, etc.) dans des paquetages de "perspective". Placez-y seulement des diagrammes décrivant les vues basées sur la préoccupation organisationnelle ou le point de vue d'application alternatifs. L'application du stéréotype de "perspective" à un paquetage permet d'accomplir plusieurs choses. Elle identifie visuellement ce paquetage comme représentant un point de vue spécifique. Elle prend également en charge une règle de validation de modèle qui vous prévient lorsque les éléments sémantiques sont placés dans un paquetage de «perspective». Elle permet également de désigner les paquetages pouvant être court-circuités par les transformations RSA

Créer des représentations se mettant elles-mêmes à jour des préoccupations spécifiques en utilisant les diagrammes de rubriques

Contrairement aux diagrammes "normaux" dans lesquels vous placez manuellement les éléments que vous souhaitez décrire, le contenu d'un Diagramme de rubrique est déterminé par une requête exécutée par rapport au contenu de modèle existant. Pour créer un diagramme de rubrique, vous sélectionnez un élément de modèle "par rubrique", puis définissez les autres éléments que vous souhaitez voir apparaître dans le diagramme en fonction des types de relation(s) qu'ils ont à l'élément "par sujet". Au fur et à mesure que le contenu sémantique du modèle change, les diagrammes de rubrique s'ajustent.

Examiner les modèles avec les diagrammes de parcours

Les diagrammes de parcours ne sont pas des outils*spécifiques*à l'organisation de modèles. Ils ont pour objectif de faciliter la découverte et la compréhension du contenu de modèle sans avoir à composer manuellement les diagrammes. Mais dans le contexte de l'organisation de modèle il est souhaitable de les connaître car ils peuvent réduire votre besoin de composer des diagrammes persistants. Cela peut ainsi réduire la taille et la complexité de vos modèles, les rendant plus faciles à organiser.

Les diagrammes de parcours sont similaires aux diagrammes de rubriques, mais la différence clé réside dans le fait que les diagrammes de parcours ne sont jamais persistants, ils sont toujours générés à la volée. Pour produire un diagramme de parcours, sélectionnez un élément de modèle (à partir d'un diagramme ou de l'explorateur de modèle), et utilisez le menu contextuel pour“ Explorer dans un diagramme de parcours”. Cela produira un diagramme décrivant l'élément sélectionné comme "point central", les éléments associés étant présentés dans une présentation en étoile autour du point central. Vous pouvez évidemment sélectionner l'un des éléments associés dans ce diagramme de parcours, en faire le point central d'un autre diagramme de parcours, et continuer ainsi aussi longtemps que vous le souhaiterez.

Navigation inter-diagramme

Dans RSA il y a deux mécanismes permettant la navigation inter-diagramme :

- Il est possible de faire glisser un noeud de diagramme de l'explorateur de modèle vers un autre diagramme "hôte". Puis vous pouvez double-cliquer sur l'icône obtenue sur le diagramme hôte pour ouvrir le diagramme référencé.

- A chaque fois que vous créez un nouveau paquetage UML dans un modèle, un diagramme “Principal” (diagramme libre) est automatiquement créé. Par défaut, ce diagramme “Principal” est créé en tant que diagramme "par défaut" du paquetage. Vous pouvez renommer le diagramme d'une façon autre que “Principal”, et il sera toujours traité comme "par défaut". Vous pouvez également sélectionner un autre diagramme dans le paquetage et en faire le diagramme "par défaut" de ce paquetage. L'objectif du diagramme "par défaut" est le suivant : si vous placez le paquetage lui-même dans un autre diagramme "hôte", vous pouvez ensuite double-cliquer sur le paquetage, qui ouvrira le diagramme par défaut.

Ces mécanismes prennent en charge les **principes organisationnels**suivants, pouvant s'appliquer aux modèles de tout type :

1. Composez le diagramme Principal (ou autre diagramme par défaut) de chaque fichier de modélisation à décrire :
    a. chaque paquetage de premier niveau dans le fichier de modélisation
    b. les icônes de diagramme pour tout autre diagramme situé dans le paquetage racine du fichier de modélisation (en d'autres termes, ne décrivez pas l'icône pour le diagramme par défaut lui-même)

2. Composer le diagramme Principal (ou autre diagramme par défaut) de chaque paquetage de premier niveau pour décrire :
    a. les paquetages qu'il contient directement
    b. les icônes de diagramme pour tout autre diagramme qu'il contient directement

3. Répétez ce schéma pour chaque niveau inférieur successif de paquetages.

5. Principes et conseils pour l'organisation interne du modèle de cas d'utilisation

REMARQUE : Dans cette section et dans les autres sections traitant des types de modèle, les principes et conseils seront illustrés en utilisant des exemples pris dans l'exemple Enchères inclus dans RSA. Etudiez l'exemple pour voir des détails organisationnels supplémentaires et le contenu des diagrammes.

Organisation de haut niveau du modèle de cas d'utilisation

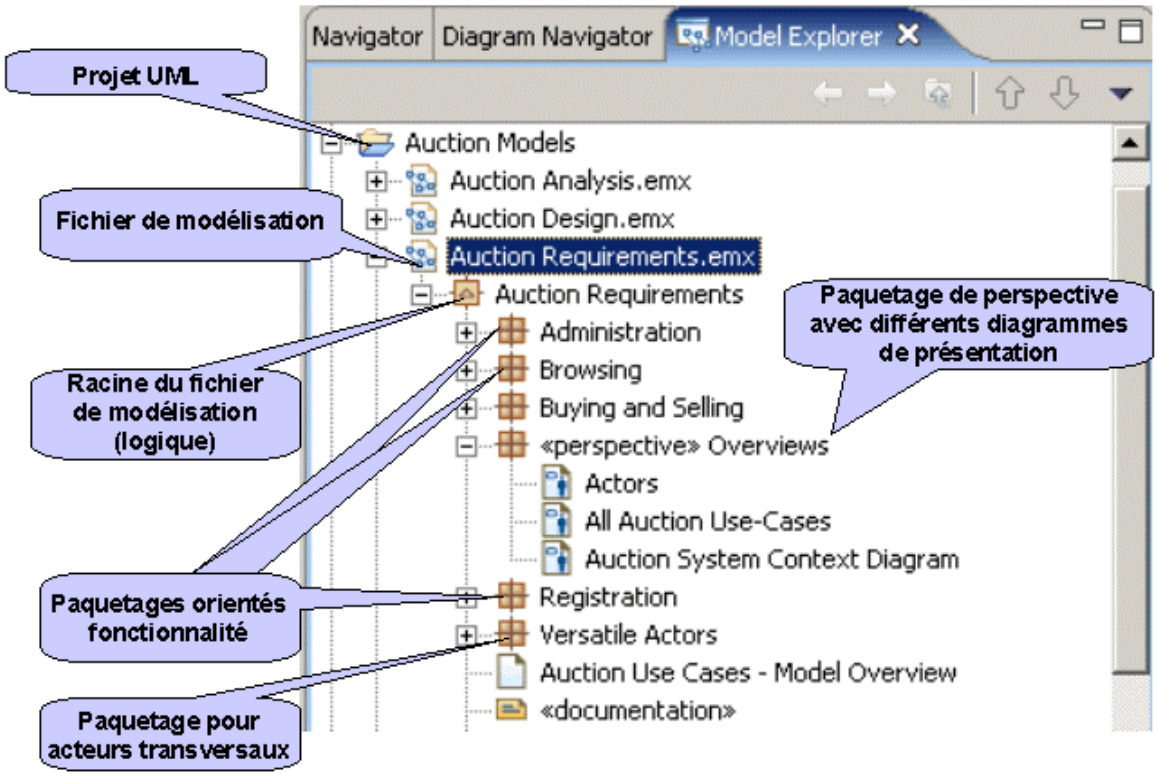


Figure 5-1

Figure 5-1 illustre les principes et conseils suivants pour structurer les modèles de cas d'utilisation :

1. Utiliser des paquetages globaux pour établir des regroupements orientés fonctionnalité.
   Justification:
   - Cela correspond généralement bien aux préoccupations de division du travail lorsqu'une équipe travaillera sur un modèle de cas d'utilisation. Cela vous facilite également la tâche si vous décidez de casser le modèle de cas d'utilisation en multiples fichiers de modélisation du fait de problèmes de concurrence d'accès aux fichiers (il vous suffit de créer un fichier de modélisation séparé par paquetage de premier niveau).

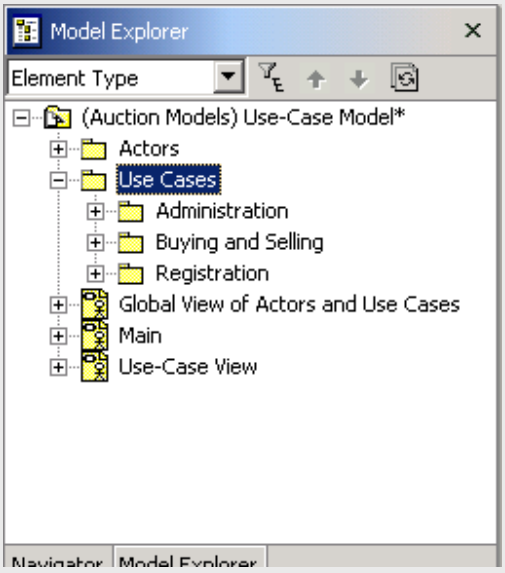
- Par rapport à d'autres approches organisationnelles, celle-ci correspondra généralement mieux à l'organisation de l'implémentation éventuelle. C'est important si vous utilisez des transformations pour débuter chaque niveau inférieur successif d'abstraction. Plus particulièrement, si vous générez du contenu de départ dans un modèle d'analyse basé sur le modèle de cas d'utilisation, vous souhaitez que la structure de paquetage du modèle de cas d'utilisation se mappe bien à la structure de paquetage souhaitée du modèle d'analyse cible. Vous souhaitez donc que la structure de paquetage du modèle d'analyse se mappe bien au modèle de conception, et que la structure de paquetage du modèle de conception se mappe bien à l'ensemble de projet qui formera l'implémentation. La simplicité des mappages va de pair avec la quantité d'effort nécessaire pour configurer les transformations d'un niveau d'abstraction à un autre.

2. Utilisez un autre paquetage de premier niveau pour consigner les acteurs "à grande habilité" ou "versatiles".

3. Utilisez des diagrammes dans des paquetages de «perspective» pour consigner des vues de haut niveau ou transversales des cas d'utilisation. **Justification**:

   - Fournir des vues transversales et des vues de cas d'utilisation "significatifs du point de vue architecturale" tout en conservant les éléments sémantiques du modèle organisés en regroupements orientés fonctionnalité.

XDE/Rose
Les recommandations de RSA sont en quelque sorte une révision des recommandations traditionnelles quant à l'organisation de haut niveau du modèle de cas d'utilisation, consistant à créer un paquetage pour les acteurs et un autre pour les cas d'utilisation. Ensuite, selon les exigences de la taille et de la complexité du modèle, vous utiliserez des paquetages de plus bas niveau pour établir des regroupements orientés fonctionnalité comme montré dans cet exemple basé sur XDE :



Contenu de modèle de cas d'utilisation

Ce document n'a pas pour but de fournir un tutoriel détaillé sur la façon d'écrire de bons cas d'utilisations ou sur ce qu'il faut faire ou ne pas faire dans la modélisation d'un bon cas d'utilisation. Cependant, voici une brève discussion sur ce qui peut être inclus dans un modèle de cas d'utilisation outre les acteurs et les cas d'utilisation.

- **Recommandé :** Créez un diagramme "principal" à la racine du modèle décrivant les autres paquetages du modèle et permet de faire un zoom avant sur ces paquetages et leurs diagrammes "principaux" respectifs.

- **Recommandé :** Dans chaque paquetage de cas d'utilisation, incluez un diagramme décrivant les cas d'utilisation du paquetage, leurs relations, et les acteurs qui y participent. (Si le nombre de cas est important, il peut être approprié d'effectuer plusieurs diagrammes.)

- **Recommandé :** Décrivez chaque flot principal et alternatif du cas d'utilisation dans le champ Documentation [4]. (Voir **Figure 5-2**.)

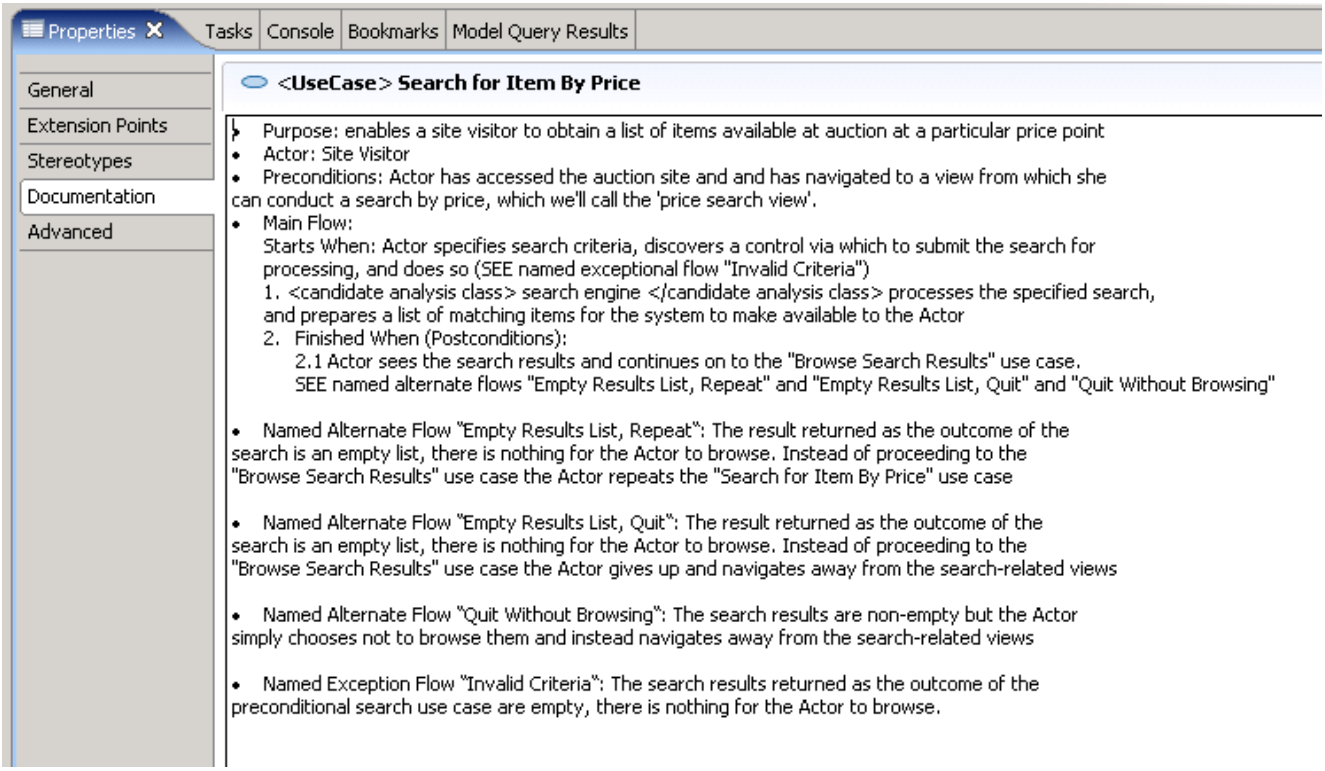


Figure 5-2

[4] Le formatage décrit dans l'exemple de description du cas d'utilisation a été obtenu en créant le "modèle" textuel pour une description de cas d'utilisation en utilisant un éditeur avec attribut RTF, puis en copiant-collant le modèle dans le champ de description du cas d'utilisation.

- **Facultatif :** Lorsque la complexité d'un cas d'utilisation le justifie, ajoutez un diagramme d'activité et composez-le pour refléter les flots d'activité généraux du cas d'utilisation. (Voir **Figure 5-3**.) **Justification :** Cela montre les conditions correspondant à chaque flot (principal et alternatifs/exceptionnels) et aide à s'assurer que tous les flots convergent au final. (L'ajout d'un diagramme d'activité dans RSM/RSA aura pour conséquence l'ajout automatique d'une activité dans le cas d'utilisation, le diagramme se trouvant sous l'activité.)

- **Facultatif :** Modélisez une réalisation "de boîte noire" pour chacun des flots nommés (principal, alternatifs et exceptionnels) du cas d'utilisation ; ajoutez une occurrence de collaboration au cas d'utilisation ; ajoutez-y une instance d'interaction correspondant au flot principal du cas d'utilisation ainsi qu'une instance d'interaction pour chacun des flots nommés alternatifs et exceptionnels ; composez un diagramme de séquence (ou bien un diagramme de communication) pour chaque instance d'interaction. Ces instances de collaboration de cas d'utilisation ne doivent pas être confondues avec les réalisations de cas d'utilisation de niveau d'analyse (décrites dans le modèle d'analyse) ni avec les réalisations de cas d'utilisation de niveau de conception décrites dans le modèle de conception). Ce sont des réalisations “de boîte blanche” du cas d'utilisation et elles décrivent les interactions entre les éléments internes d'une solution. Les occurrences de collaboration proposées ici pour le modèle de cas d'utilisation sont uniquement des interactions “de boîte noire” entre les acteurs et le système. (Voir **Figure 5-3**.) **Justification :** Cela donne aux intervenants non techniques une image de haut niveau de la manière dont les utilisateurs du système interagiront avec celui-ci. Cela peut également vous aider à identifier les différentes vues (écrans ou pages) requises pour l'implémentation. Cela établit également de façon formelle la dénomination des différents flots (scénarios) du cas d'utilisation en attribuant ces noms aux éléments de modèle sémantiques (c'est-à-dire, aux occurrences de collaboration).

> ***XDE/Rose***
> Dans UML 1.x, vous auriez utilisé des “instances de collaboration” plutôt que des “occurrences de collaboration” dans ce but.

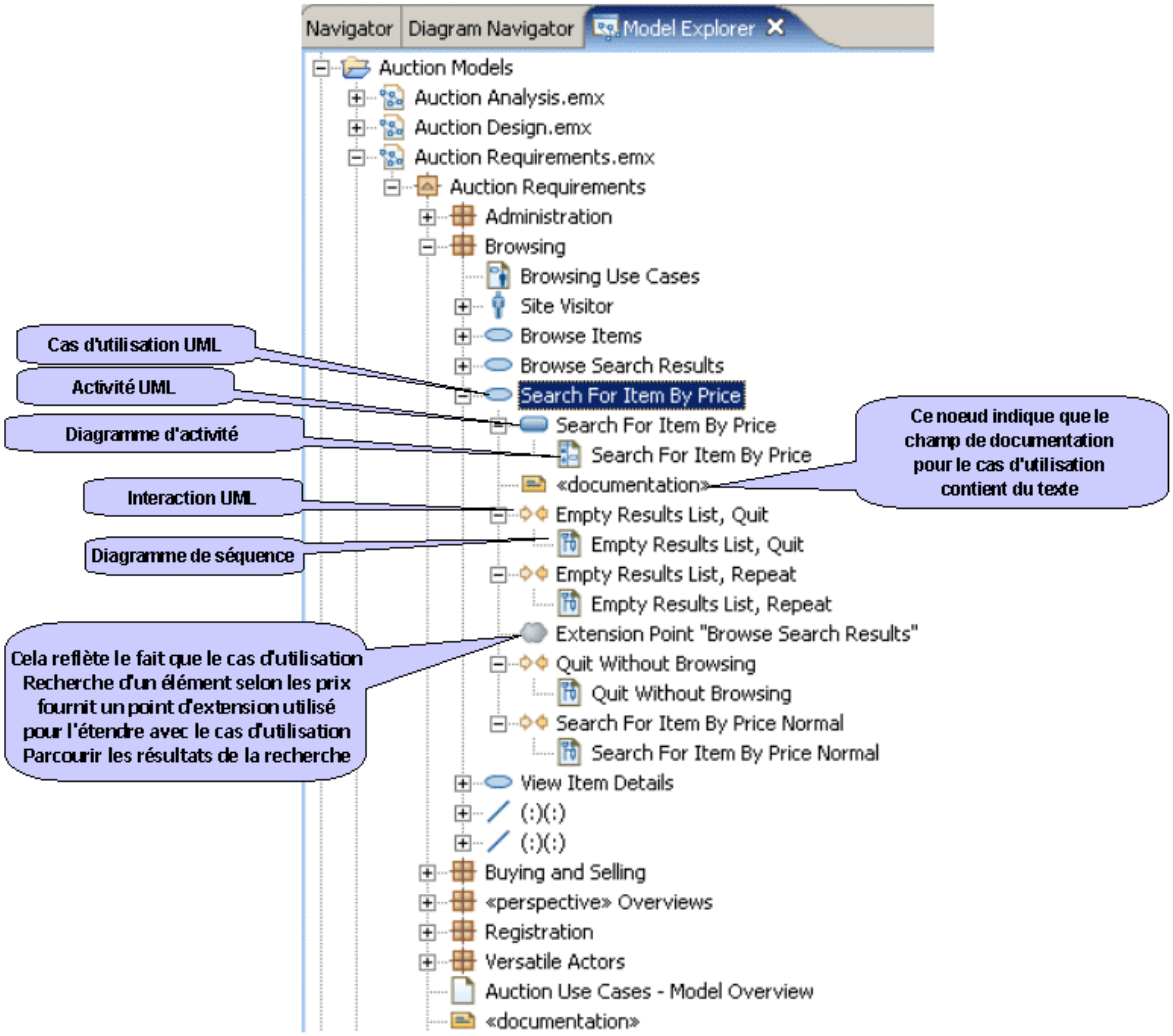


Figure 5-3

- **Facultatif :** Si vous suivez les recommandations RUP pour identifier les vues de votre architecture significatives du point de vue architectural", et en particulier si vous gérez un document d'architecture logicielle, ajoutez un paquetage de "perspective" de premier niveau pour contenir les diagrammes de cas d'utilisation décrivant les cas d'utilisations significatifs du point de vue architectural. Vous pouvez souhaiter nommer le paquetage "Vue de cas d'utilisation de l'architecture".

6. Principes et conseils pour l'organisation interne du modèle d'analyse

Le modèle d'analyse représente un "brouillon" d'une solution. C'est un marche-pied pour aller des exigences à la conception finale, qui consigne des informations au sujet du domaine de l'entreprise et montre les éléments de solution potentiels à un haut niveau d'abstraction proche de l'entreprise. C'est là que se situent lesclasses d'analyse et lesréalisations de cas d'utilisation de niveau d'analyse . Par le processus de modélisation des réalisations de cas d'utilisation (en utilisant principalement des diagrammes de séquence) vous commencez à *découvrir*quelles classes sont nécessaires pour la solution—en particulier, ce seront les classes correspondant aux lignes de vie dont vous avez besoin dans les diagrammes de séquence. Quelques règles empiriques peuvent également être appliquées pour suggérer le contenu de modèle d'analyse basé sur le contenu du modèle de cas d'utilisation. Elles seront examinés plus tard dans cette section.

Dans le RUP, le fait de gérer le modèle d'analyseindépendamment ou non du modèle de conception est une décision spécifique au projet, selon que la valeur de gestion du modèle d'analyse séparée justifie ou non selon vous le temps investi. Si un modèle d'analyseséparé est créé mais non géré, lesclasses d'analyse seront déplacées vers le modèle de conceptionet perfectionnées. Le modèle d'analyse peut également évoluer progressivement jusqu'à ce qu'il devienne un modèle de conception [5]. Dans des termes spécifiques au produit, voici quelques options que vous pouvez envisager:

1. Créez un modèle d'analyse résidant dans un fichier de modélisation (ou un ensemble de fichiers) basé sur le modèle d'analyse. Utilisez ensuite un processus manuel ou des transformations automatisées pour créer des versions perfectionnées des éléments d'analyse dans un deuxième fichier de modèle (ou ensemble de fichiers) basé sur le modèle de conception Enterprise IT, puis supprimez les fichiers de modélisation d'analyse. Vous aurez alors le choix entre gérer le modèle d'analyse séparé actuel, ou bien le jeter.
2. Effectuez la modélisation de niveau d'analyse dans un fichier de modélisation (ou ensemble de fichiers) basé sur le modèle de conception Enterprise IT, auquel vous appliquez le profil d'analyse. Cela vous permet de commencer à modéliser les réalisations de cas d'utilisation en utilisant des classes d'analyse, puis de les perfectionner progressivement afin que les interfaces de conception adoptent les rôles dans les comportements.
3. Une combinaison de la deuxième et troisième solution consiste à gérer un type de modèle d'analyse dans le(s) même(s) fichier(s) de modélisation que le modèle de conception. Pour ce faire vous devez séparer le contenu d'analyse en paquetages auxquels vous appliquez le mot clé "analyse". Cela vous permet de conserver des artefacts de niveau d'analyse dans les mêmes fichiers de modélisation que leurs équivalents de niveau de conception.

Vous devez être conscients, lorsque vous utilisez des transformations RSA pour générer des implémentations, que ces transformations peuvent dans bien des cas accepter des éléments de niveau d'analyse comme entrée, ce qui vous dispense de perfectionner manuellement ces éléments en éléments de conception. Lorsque vous utilisez RSA ainsi, privilégiez les solutions 2 ou 3 ci-dessous. Les transformations standards générant du code mises en paquetages dans RSA court-circuiteront les paquetages de modèle avec comme mot clé "analyse".

[5] En fait, le RUP fait appel à l'option de création de classes d'analyse et de réalisations de cas d'utilisation de niveau d'analyse dans le modèle de conception puis les fait évolue directement vers leurs formes de conception à partir de ce moment. . Cette approche vous permet, au fur et à mesure de la“ découverte” du modèle de conception, de créer des paquetages dans lesquels vous conservez certaines des perspectives d'“analyse pure”.

Organisation de haut niveau de modèle d'analyse

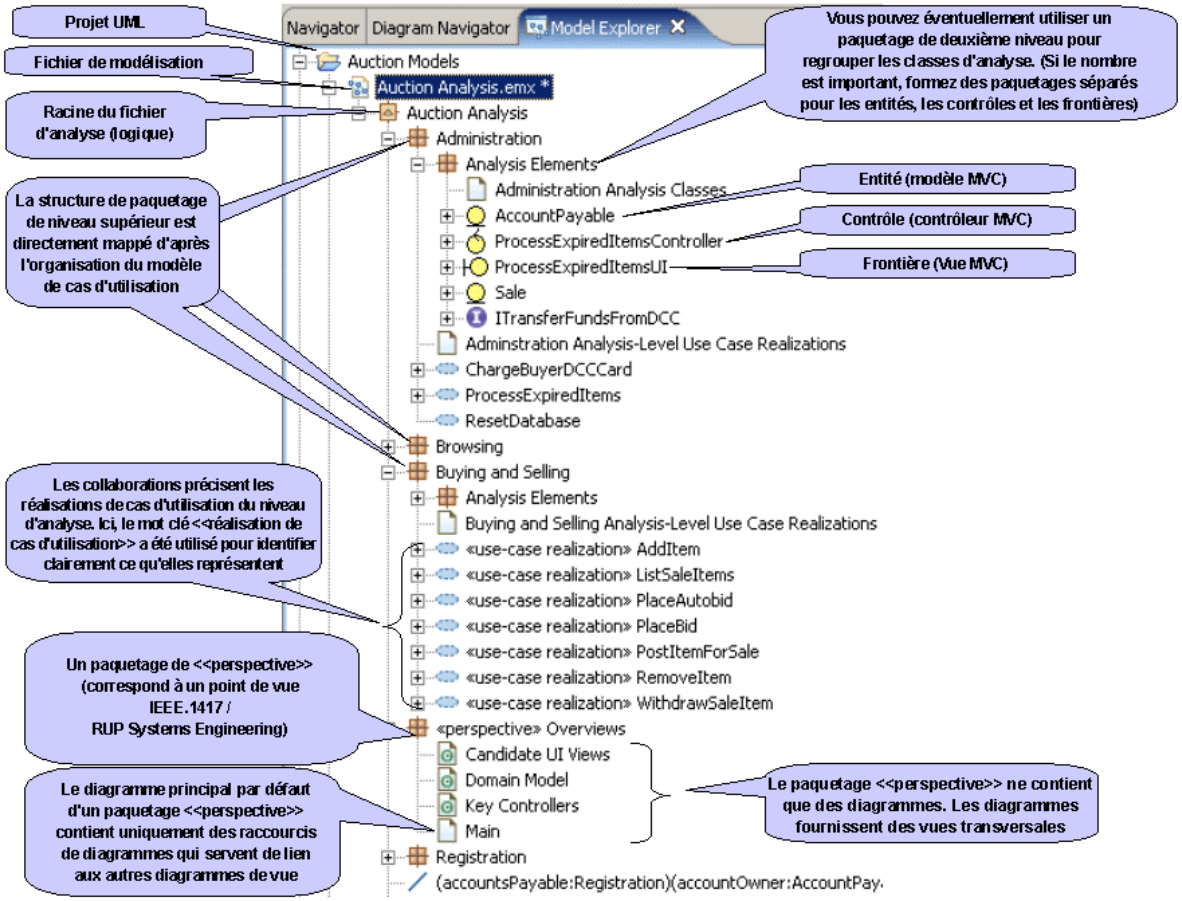


Figure 6-1

Figure 6-1 illustre les principes et conseils suivants pour structurer les modèles d'analyse :

1. Utilisez des paquetages de premier niveau pour établir des regroupements orientés fonctionnalité pour les classes d'analyse. **Justification**: même justification que pour le modèle de cas d'utilisation.

2. Vous pouvez éventuellement, dans les paquetages de premier niveau, utiliser les sous-paquetages pour recueillir et organiser les classes d'analyse.

3. Utilisez des diagrammes dans des paquetages de «perspective» afin de consigner des vues alternatives, de haut niveau ou transversales des éléments d'analyse. **Justification**: Fournir différentes perspectives pour différents intervenants tout en maintenant les éléments sémantiques du modèle organisés en regroupements orientés fonctionnalité.

Une légère variation de cette approche est décrite dans **Figure 6-2** montrant l'utilisation d'un paquetage de premier niveau pour séparer les réalisations de cas d'utilisation des classes d'analyse. Ce paquetage de premier niveau contient un ensemble de sous-paquetages orientés fonctionnalité correspondant à l'ensemble de paquetages de premier niveau. Isoler les réalisations de cas d'utilisation de cette manière permet de réusiner la structure de paquetage contenant les classes d'analyse, sans *forcément* affecter l'organisation des réalisations de cas d'utilisation. (En particulier dans le cas où le modèle d'analyse évolue vers la conception *in situ*, il est probable que l'organisation de paquetage pour les classes évoluera jusqu'à ce qu'elle ne corresponde plus à celle utilisée initialement pour les cas d'utilisation.)

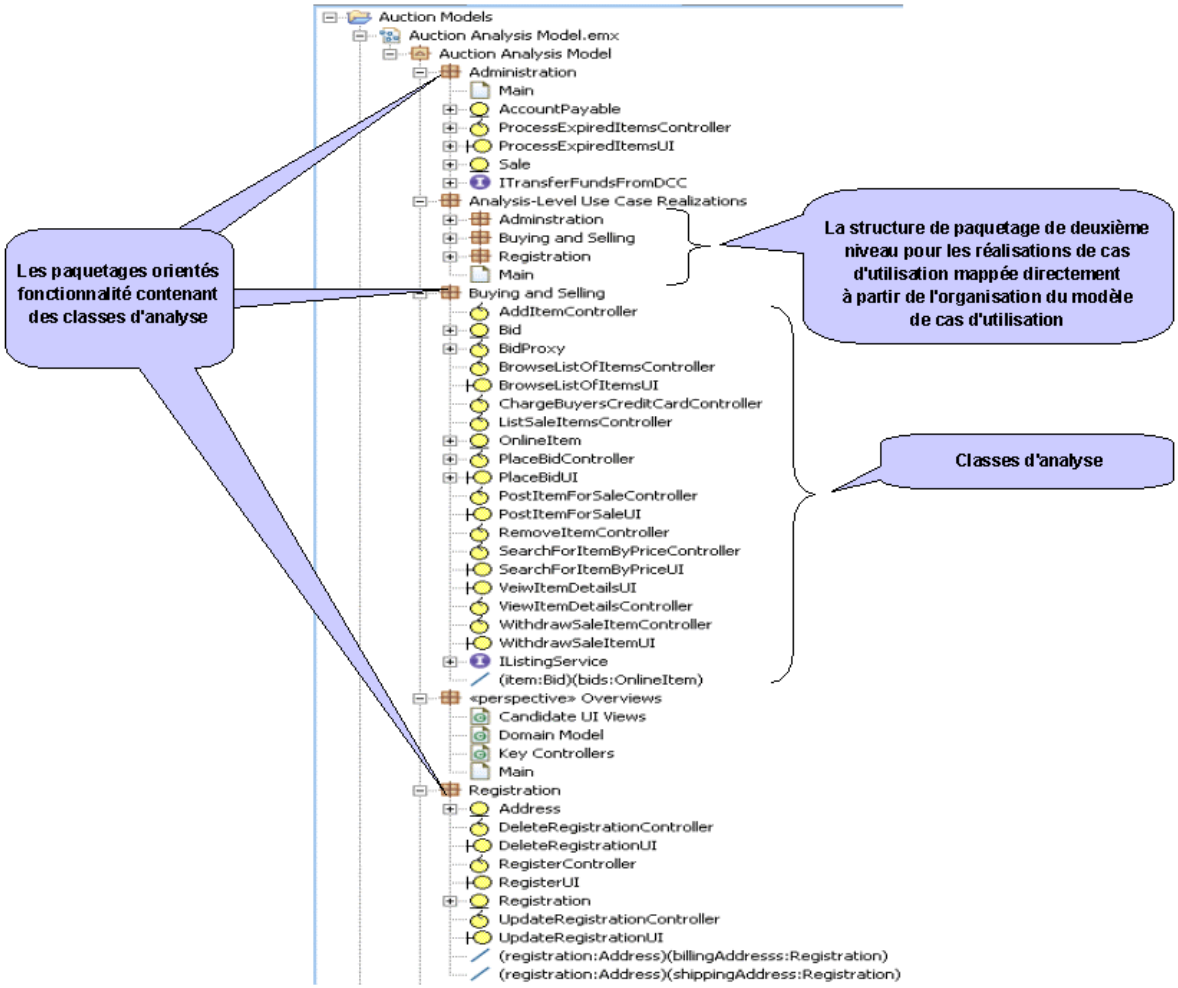


Figure 6-2

Selon votre situation, vous introduirez peut-être l'utilisation d'une convention de dénomination anticipant la fusion et la réutilisation du contenu de modèle créé par des groupes indépendants multiples, y compris les groupes d'autres entreprises (partenaires). Si ce problème se pose, envisagez d'utiliser un espace de nom de domaine Internet inversé décrit dans **Figure 6-3**. Notez que cette préoccupation n'est probablement pas très importante pour la modélisation d'analyse *per se*, mais si vous décidez que votre modèle d'analyse deviendra votre modèle de conception *in situ,* et que vous prévoyez une réutilisation ou une intégration métier au niveau de la conception, il vous serait utile de planifier par avance. Un autre avantage potentiel de cette approche : dans la mesure où elle se mappe bien à l'organisation du code généré à partir de l'analyse/conception, elle peut simplifier la configuration ultérieure des transformations générant du code.

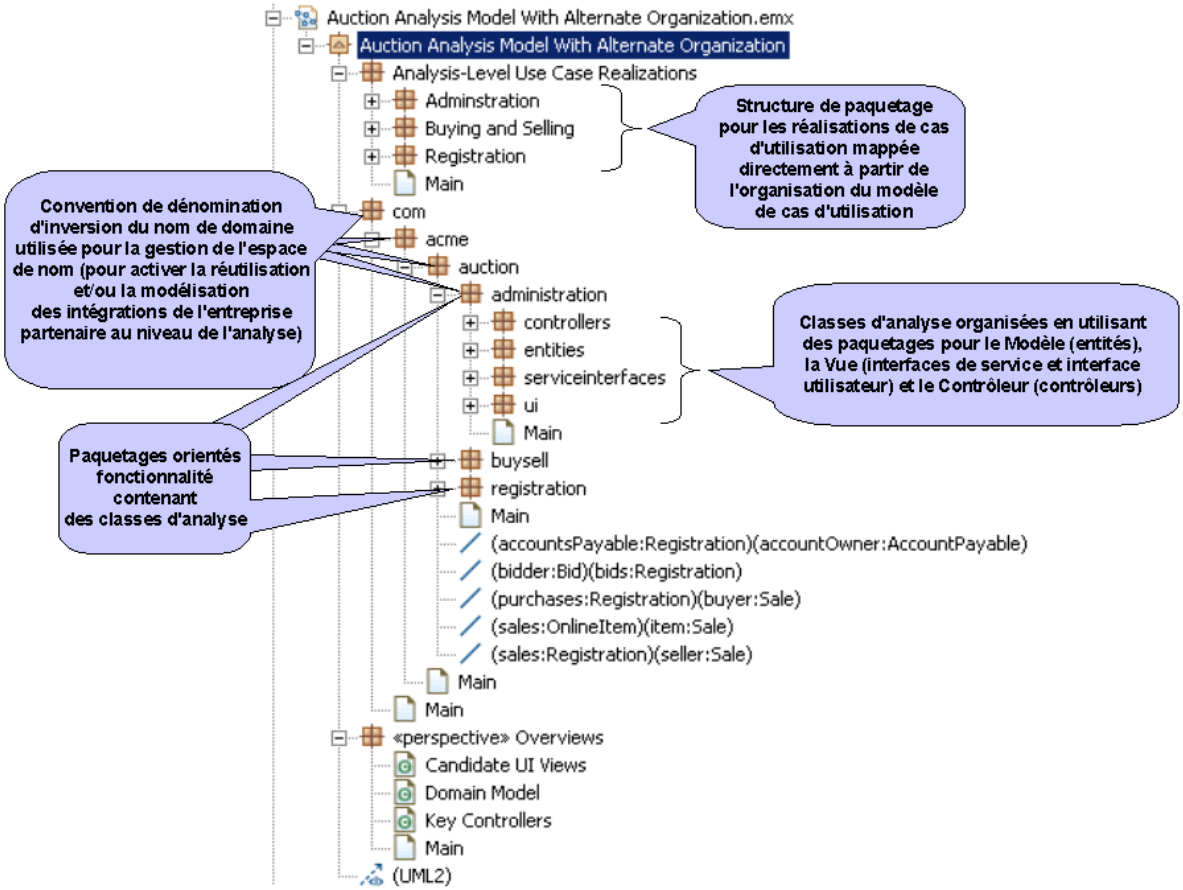


Figure 6-3

Contenu de modèle d'analyse

Il y a plusieurs manières de découvrir ce que sont les classes d'analyse. Une solution est de commencer à dessiner les diagrammes de séquence qui suggèrent les réalisations de cas d'utilisation. Vous découvrirez ainsi les lignes de vie dont vous avez besoin, et de manière générale, chaque ligne de vie correspondra à une classe d'analyse probable. Lorsque vous découvrez des classes de cette manière, vous pouvez les créer dans des paquetages de réalisation de cas d'utilisation du modèle d'analyse mais vous ne devez pas les y laisser. Vous devez "réusiner" le modèle pour transférer les classes d'analyse en paquetages orientés fonctionnalité comme nous l'avons expliqué dans les principes et conseils pour l'organisation de haut niveau du modèle d'analyse (voir ***Figure 6-1***).

Une autre approche utile pour la découverte de classes d'analyse : "débuter" le modèle d'analyse avec des classes basées sur ces règles empiriques :

- Pour chaque cas d'utilisation (dans le modèle de cas d'utilisation) ajoutez une classe de "contrôle" au modèle d'analyse. les classes de "contrôle" représentent la logique applicative liée au cas d'utilisation. (Plus tard, lors de la conception, elles se mapperont également aux préoccupations comme la gestion de session.)

- Pour chaque relation acteur/cas d'utilisation (dans le modèle de cas d'utilisation) ajoutez une classe "frontière" au modèle d'analyse. Les classes "frontières" représentent des interfaces entre la solution et un acteur humain ou entre la solution et un système externe. Les classes «frontières» correspondant à un acteur humain se mapperont sans doute finalement à un ou plusieurs artefacts d'interface dans la conception et l'implémentation. Les classes "frontière" qui correspondent à un système externe peuvent finalement se mapper à un type de couche adaptateur dans la conception et l'implémentation.

- Via un processus comme l'analyse de carte CRC, ou bien l'analyse par mot des descriptions de cas d'utilisation, identifiez les classes de "contrôles" supplémentaires (verbes) et les classes d'"entité" (noms).

Lorsque vous utilisez cette approche pour identifier les classes d'analyse, vous pouvez placer les classes directement dans des paquetages orientés fonctionnalité comme nous l'avons décrit précédemment dans les principes et conseils de l'organisation de haut niveau du modèle d'analyse (voir ***Figure 6-1***).

Quelle que soit votre manière de découvrir des classes d'analyse, il est pratiquement certain que vous vous rendiez compte que des changements à votre organisation de paquetage fonctionnel d'origine sont nécessaires.

Facultatif : Utilisez des paquetages de second niveau dans les paquetages de classe d'analyse pour organiser plus en détails le contenu de ces paquetages (voir **Figure 6-4**).

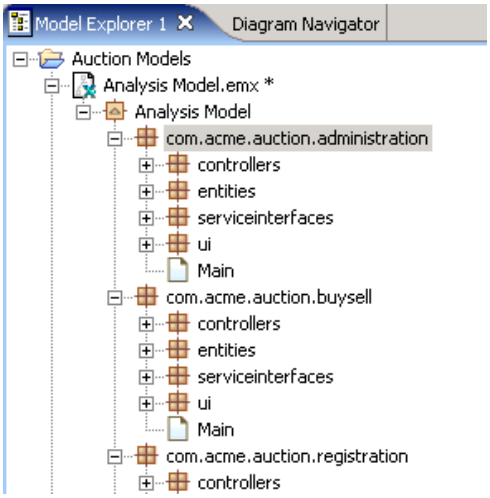


Figure 6-4

Recommandé : Le modèle d'analyse doit contenir desréalisations de cas d'utilisation de niveau d'analyse**,** décrivant comment lescas d'utilisationsont exécutés en termes de classes d'analyse. Chaque réalisation de cas d'utilisation d'analyse (représentée par une collaboration UML réalise un cas d'utilisationdans le modèle de cas d'utilisation et a le même nom que ce cas d'utilisation. Voir **Figure 6-5**. Pour chaque flot de cas d'utilisation nommé[6] qui selon vous doit être modélisé en tant que réalisation de niveau d'analyse, ajoutez un diagramme de séquence (qui ajoutera automatiquement une interaction propriétaire). **Figure 6-6** montre les types de contenu sémantique ajouté au modèle lorsque vous créez des diagrammes de séquence. (Notez que vous pouvez filtrer tout type d'élément UML à partir de la vue d'explorateur de modèle, et ainsi cacher une bonne partie de l'"encombrement" décrit dans **Figure 6-6**.)

[6] Comme établit précédemment dans le modèle de cas d'utilisation

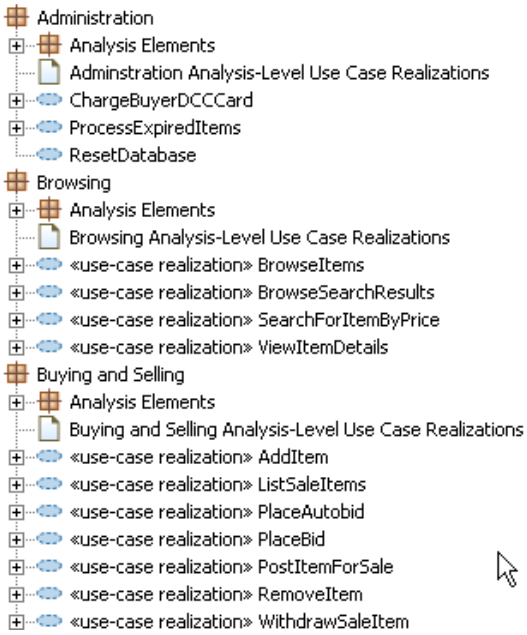


Figure 6-5

Facultatif : Une fois que vous avez créé le diagramme de séquence pour un flot de cas d'utilisation, vous pouvez sélectionner son interaction UML propriétaire dans l'explorateur de modèle et y ajouter un diagramme de communication. Le nouveau diagramme de communication sera rempli automatiquement avec les instances de classe d'analyse qui ont participé au diagramme de séquence.

Recommandé : Créez une relation de dépendance de réalisation entre chaque réalisation de cas d'utilisation (collaboration UML) et le cas d'utilisation correspondant dans le modèle de cas d'utilisation (voir **Figure 6-6**). Dans la mesure où vous pouvez utiliser des fonctions telles que les diagrammes de rubrique et l'analyse de la traçabilité pour comprendre les relations de traçabilité dans votre modèle, vous n'avez pas vraiment besoin de conserver des diagrammes permanents pour décrire les relations de traçabilité, il est donc recommandé de créer les relations en utilisant un type de diagramme "jetable", par exemple :

- Ajoutez un diagramme libre à la collaboration.
- Faites-y glisser la collaboration.
- Faites-y glisser le cas d'utilisation.
- Dessinez la relation de dépendance.
- Pour finir, (dans l'explorateur de modèle) supprimez le diagramme de la collaboration.

Recommandé : Incluez un diagramme “participants” pour chaque réalisation de cas d'utilisation afin de montrer les classes d'analyseparticipant à la réalisation(c'est-à-dire, les classes d'analyse dont les instances apparaissent dans les diagrammes d'interaction décrivant la réalisation du cas d'utilisation) et les relations entre les classes prenant en charge la collaboration décrite dans les diagrammes d'interaction. Voir **Figure 6-6**.

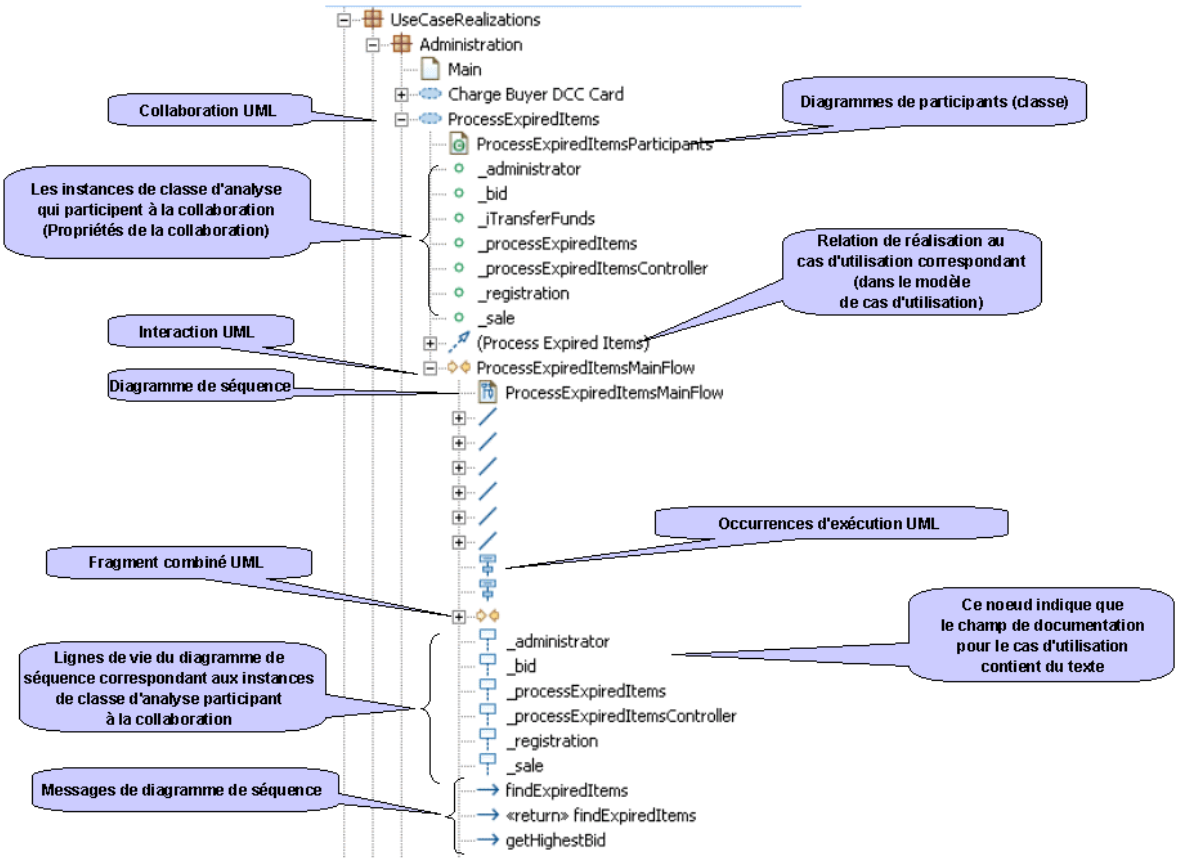
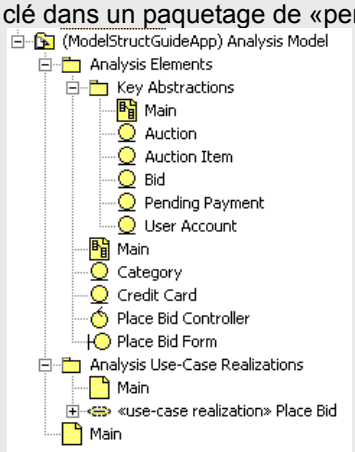


Figure 6-6

XDE/Rose

La structure généralement recommandée pour le **modèle d'analyse**décrite ci-dessous est modifiée pour RSA afin d'insister sur une organisation de paquetage orientée fonctionnalité pour les classes d'analyse. Notez également que l'utilisation d'un paquetage d'abstractions clé (qui mettrait en péril une approche de paquetage orientée fonctionnalité) est remplacée par l'utilisation d'un (ou de plusieurs) diagrammes d'abstractions clé dans un paquetage de «perspective».



7. Principes et conseils pour l'organisation interne du modèle de conception

Concevoir l'organisation de modèle de haut niveau

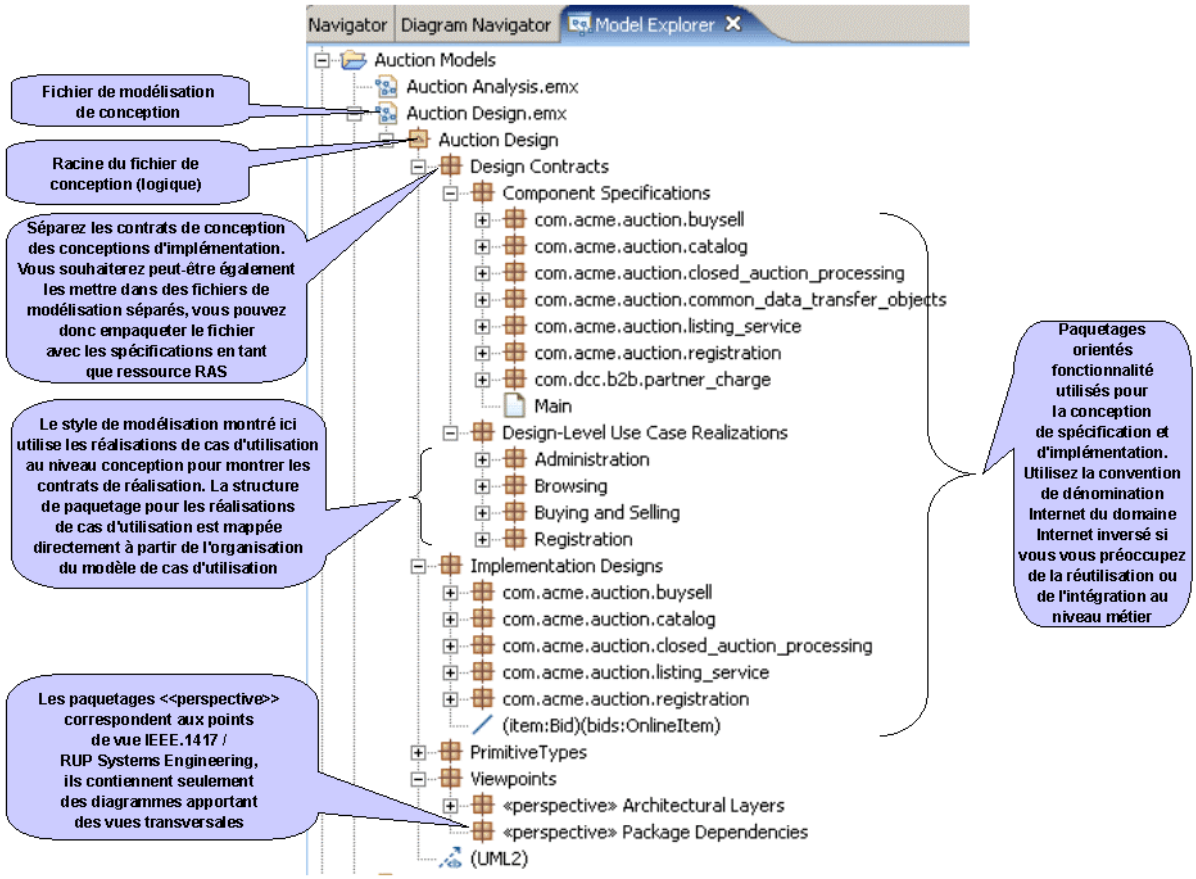


Figure 7-1

Figure 7-1 illustre les principes et conseils suivants pour structurer les modèles de conception :

1. Séparez les spécifications des conceptions d'implémentation. L'illustration montre l'utilisation de paquetages de “contrats de conception” et de“ conceptions d'implémentation” de premier niveau pour obtenir ce résultat.

2. Utilisez des paquetages de niveau inférieur pour établir des regroupements orientés fonctionnalité. Vous pouvez par exemple commencer avec l'organisation que vous avez utilisé pendant l'analyse, puis la laisser évoluer tout en prenant des décisions sur la façon dont les classes d'analyse se mappent aux véritables classes, composants et services de conception. (Tout schéma organisationnel initial est susceptible d'évoluer pendant la conception—voir discussion supplémentaire ci-dessous).

   Un mot sur les sous-systèmes pouvant fonctionner à ce stade. Dans les versions d'UML antérieures à 2, un sous-système est un type spécialisé de paquetage. Dans UML 2, un sous-système est un

type spécialisé de composant, et un Composant peut contenir des paquetages. Dans UML 2, les composants «sous-système» sont des alternatives viables organisationnelles/d'espace de nom aux paquetages, mais UML2 est vague au sujet de l'utilisation appropriée des sous-systèmes. Suggestion : Utilisez des paquetages à des niveaux de granularité semblables aux sous-systèmes de conception d'une application spécifique, et conservez les sous-systèmes pour représenter des applications complètes (par exemple CRM ou SCM) dans des vues d'architecture concernant l'entreprise globale.

> ***XDE/Rose***
> Au moment de la rédaction de cet article, il est prévu que les outils d'importation de modèle Rose et XDE donnent la possibilité de mapper les sous-systèmes UML 1.x aux sous-systèmes UML2 ou aux paquetages auxquels est appliqué le mot clé "sous-système".

3. Il est probable que l'organisation des éléments de conception s'éloignera de la manière dont les cas d'utilisation du système sont organisés (dans le modèle de cas d'utilisation et peut-être dans le modèle d'analyse si un modèle d'analyse séparé est géré). Utilisez des paquetages pour sub-diviser les contrats de conception dans les spécifications d'élément de conception (les contrats d'utilisation) et les réalisations de cas d'utilisation de niveau de conception (les contrats de réalisation) et gérer une sous-structure de paquetage pour les réalisations de cas d'utilisation qui reflètent toujours l'organisation des cas d'utilisation eux-mêmes.

4. *Considérez*l'utilisation de couches architecturales comme la base du schéma organisationnel de second niveau pour les éléments qui forment les conceptions de spécification et d'implémentation des domaines fonctionnels (voir également discussion ci-dessous).

5. Dans les composants et les paquetages UML regroupant les éléments de modèle sémantiques, placez des diagrammes fournissant des vues spécifiques à ce regroupement. Ces principes et conseils sont applicables que le regroupement soit basé sur des sous-ensembles orientés fonctionnalité du domaine commercial, sur une couche architecturale, etc. Appliquez au diagramme par "défaut" le même nom que le paquetage ou le composant lui-même, et composez-le pour présenter le contenu du paquetage. Cela permet à certains diagrammes de rester proches de ce qu'ils décrivent, ce qui facilite la navigation et la compréhension du modèle.

6. Vous souhaiterez peut-être introduire l'utilisation d'un espace de nom de domaine Internet inversé dans le modèle de conception. **Justification**:
   - Principalement car cette activité est importante par rapport aux implémentation spécifiques à la langue :
     a. scénarios impliquant des tâches d'intégration dans lesquels sont impliquées de multiples applications dirigées par modèle (particulièrement avec les entreprises partenaires).
     b. scénarios de réutilisation
   - Cela simplifiera probablement la configuration ultérieure des transformations vers l'implémentation (emplacement et mappage de noms de la source à la destination).

7. *Envisagez* d'utiliser des noms de paquetages qui seront valides dans la ou les plate-forme(s) d'implémentation cible(s) afin d'éviter la charge et la confusion potentielle du mappage d'espaces de noms. (Cela signifie principalement qu'il ne faut pas “utiliser des espaces ou une ponctuation autre que les traits de soulignement dans les noms”.)

8. Utilisez les minuscules pour les noms de paquetages afin de les distinguer des noms de classe dans un paquetage.

9. *Envisagez* d'utiliser différents noms pour les Interfaces et les Composants ou Classes qui les réalisent. Utilisez ILoan et Loan, ou bien Loan et LoanImpl pour les noms d'interface et d'implémentation. Ce n'est pas vraiment nécessaire dans le modèle, mais c'est souvent une bonne idée dans le code généré, c'est donc un autre domaine dans lequel vous pouvez éviter une partie importante du travail de configuration de transformation ultérieur.

10. Dans le scénario suivant, le contenu de niveau d'analyse à partir duquel aucun code ne doit être généré doit être séparé en paquetages stéréotypés en tant qu'"analyse"[7].

    - A) Vous devez court-circuiter l'utilisation d'un modèle d'analyse séparé, remplir le modèle de conception avec du contenu de niveau d'analyse et gérer ce contenu au niveau d'analyse d'abstraction tout en créant du contenu de niveau de conception dans le même modèle, et

    - B) Vous dirigerez des transformations de modèle à code à partir du modèle de conception EIT.

11. Utilisez des diagrammes dans des paquetages de «perspective» pour consigner des vues de haut niveau ou transversales des éléments de conception. **Justification**: Fournir des vues transversales, des vues du contenu significatif du point de vue architectural, et des vues qui correspondent aux différents types d'intervenants tout en maintenant les éléments sémantiques du modèle organisés en regroupements orientés fonctionnalité.

Il est important de reconnaître que les structures de paquetage des modèles de conception évolueront dans le temps. Au final, l'organisation doit correspondre à la façon dont vous structurez votre architecture en composants et en services. Cette approche à l'organisation *finale* de la conception utilisera généralement le meilleur potentiel de paquetage des ressources réutilisables et le mappage le plus direct de la conception à l'ensemble de projets et dossiers qui contiendront les artefacts d'implémentation (code, méta-données, documentation) générés à partir de la conception.

Cependant, l'organisation *initiale* doit correspondre plus ou moins directement à l'approche organisationnelle que vous avez utilisé pour le modèle de cas d'utilisation puis révisé pendant l'analyse [8]. En fait (comme décrit dans la section précédente “Principes et conseils pour l'organisation interne du modèle d'analyse”), vous pouvez choisir de laisser évoluer votre modèle d'analyse vers la conception existante. En d'autres termes, l'organisation initiale de la conception regroupera probablement des ensembles de préoccupations commerciales cohésifs et à couplage lâche et isolera les éléments transversaux ou réutilisables. Cette approche à l'organisation initiale est efficace car :

- Si vous espérez utiliser des transformations générant du contenu de modèle de conception à parti du contenu de modèle de cas d'utilisation, les mappages de paquetages source aux paquetages de destination seront simples et directs.

- Une approche organisationnelle initiale basée sur la cohésion fonctionnelle et le couplage lâche des paquetages sera la plus susceptible de correspondre à l'organisation finale orientée composant, ce qui signifie que la quantité de réusinage nécessaire dans le cadre du processus de conception sera réduite.

- Le couplage lâche des paquetages peut améliorer les enchaînements des activités de l'équipe et faciliter la réutilisation lorsque la conception est usinée en plusieurs fichiers de modélisation.

[7] Ces paquetages seront court-circuités par les transformations.

[8] L'empaquetage des classes d'analyse est souvent réusiné de manière significative au fur et à mesure de sa découverte ainsi de mieux prendre en charge la réutilisation et les exigences fonctionnelles non prévues.

D'autres approches sont bien sûr possibles et dans certains cas conseillées en tant qu'organisation *finale*:

- Si vous ciblez des applications Web basées sur J2EE, y compris les EJB, l'organisation de la conception peut prévoir les conventions de RSA et Rational Application Developer concernant les projets J2EE.[9] Nous pouvez notamment décider de définir des paquetages de conception de premier niveau correspondant aux couches architecturales (présentation et métier, la couche métier étant subdivisée en session et domaine). Il ne s'agit à l'évidence pas d'une approche indifférente à la plate-forme ; pour cette raison, elle n'est conseillée que si vous êtes certain que la solution que vous concevez ne sera pas implémentée sur une plate-forme autre que J2EE.

- Plus généralement, lors des applications à plusieurs niveaux sont construites, l'expertise du développeur et la division du travail correspondent souvent aux couches présentation et métier ; il vous faudra donc peut-être utiliser des paquetages de premier niveau correspondant à ces couches architecturales. Mais soyez prudent lorsque vous organisez des classes destinées à prendre en charge des *fonctions métier*spécifiques afin de prendre en charge une architecture *spécifique*. Cela rend ces deux éléments plus difficiles à modifier.

- Si vous pensez que l'utilisation d'une approche organisationnelle non orientée composant/service/sous-système est justifiée, vous devez pouvoir mapper l'organisation de la conception à un ensemble de projets et dossiers cibles en consacrant plus d'efforts à la configuration des transformations de génération de code. Vous pouvez utiliser un type spécifique de modèle associé appelé "modèle de mappage" pour définir des mappages particulièrement complexes.

Contenu du modèle de conception

Il n'existe pas de règle stricte définissant ce que doit contenir le modèle de conception, mais les suggestions suivantes peuvent s'avérer utiles.

[9] Approximativement : Un Enterprise Projet par système ou application ou sous-système de grande taille, et pour chaque Enterprise Project, un projet Web pour le niveau de présentation, et des projets EJB multiples où les projets EJB correspondent généralement aux composants ou aux sous-systèmes mineurs, et dans lesquels des EJB généralement séparés, couche de session (EJB de session) et de domaine (EJB d'entité) par composant ou sous-système. Voir la section 9 de cet article pour plus d'informations.

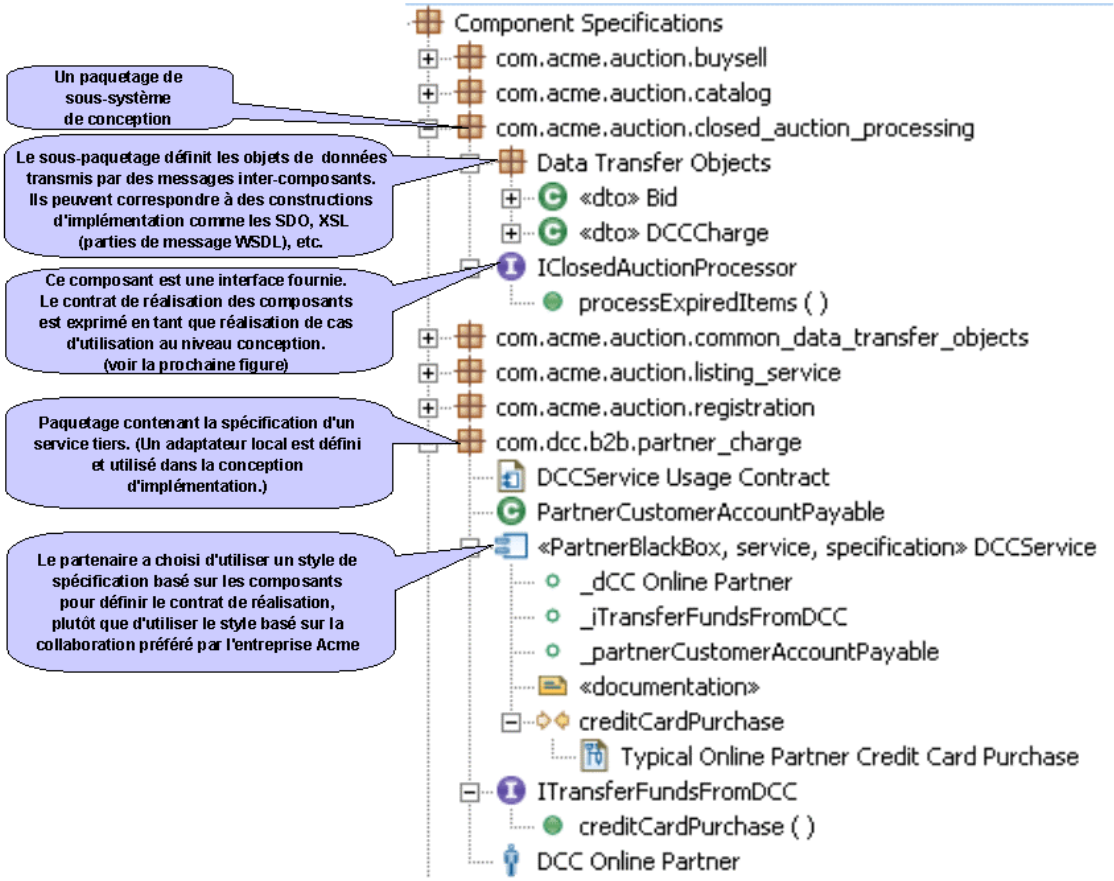

Un paquetage de sous-système de conception
Le sous-paquetage définit les objets de données transmis par des messages inter-composants. Ils peuvent correspondre à des constructions d'implémentation comme les SDO, XSL (parties de message WSDL), etc.
Ce composant est une interface fournie. Le contrat de réalisation des composants est exprimé en tant que réalisation de cas d'utilisation au niveau conception. (voir la prochaine figure)
Paquetage contenant la spécification d'un service tiers. (Un adaptateur local est défini et utilisé dans la conception d'implémentation.)
Le partenaire a choisi d'utiliser un style de spécification basé sur les composants pour définir le contrat de réalisation, plutôt que d'utiliser le style basé sur la collaboration préféré par l'entreprise Acme
Component Specifications
com.acme.auction.buysell
com.acme.auction.catalog
com.acme.auction.closed_auction_processing
Data Transfer Objects
«dto» Bid
«dto» DCCCharge
IClosedAuctionProcessor
processExpiredItems ( )
com.acme.auction.common_data_transfer_objects
com.acme.auction.listing_service
com.acme.auction.registration
com.dcc.b2b.partner_charge
DCCService Usage Contract
PartnerCustomerAccountPayable
«PartnerBlackBox, service, specification» DCCService
_dCC Online Partner
_iTransferFundsFromDCC
_partnerCustomerAccountPayable
«documentation»
creditCardPurchase
Typical Online Partner Credit Card Purchase
ITransferFundsFromDCC
creditCardPurchase ( )
DCC Online Partner

Figure 7-2

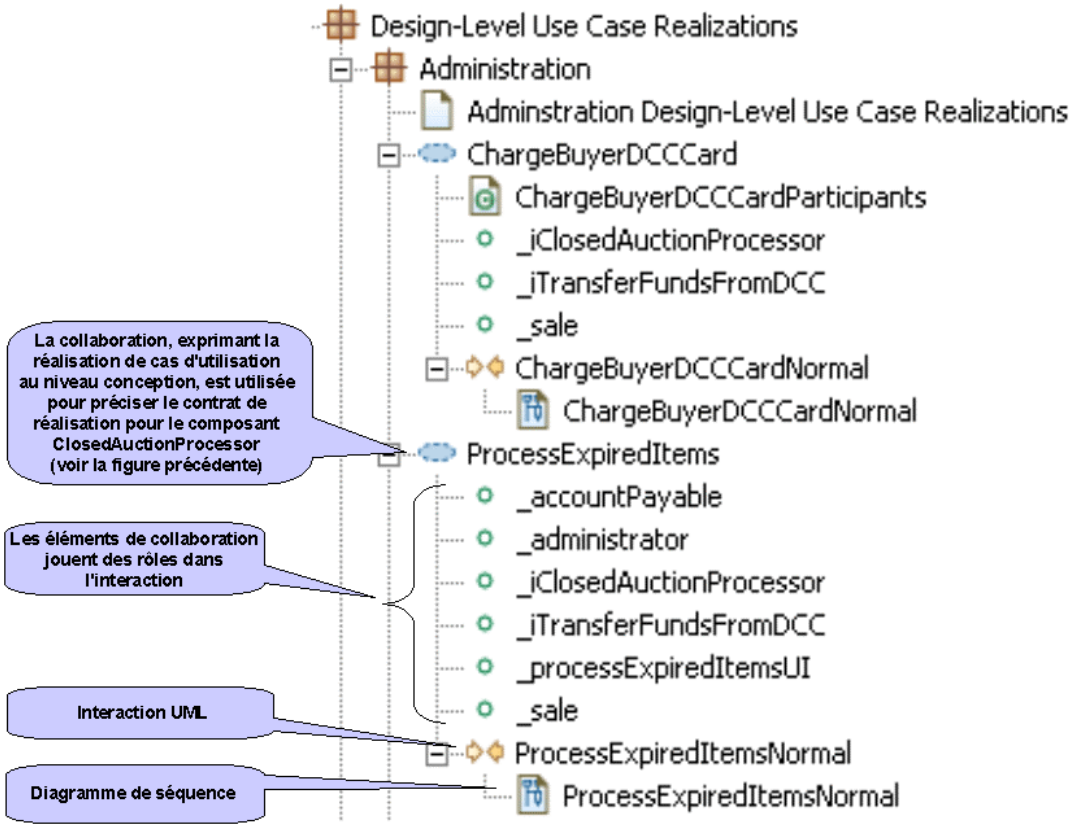


Figure 7-3

Figure 7-2 et **Figure 7-3** se conforme à la structure organisationnelle décrite dans **Figure 7-1** et décrit comment des contrats de conception peut être définis.

- Le contrat d'utilisation pour un composant “ProcesseurEnchèresTerminées” est exprimé en tant qu'interface simple[10] (**Figure 7-2**). Le contrat de réalisation correspondant est défini par une simple réalisation de cas d'utilisation de niveau de conception exprimée en tant que collaboration[11] (**Figure 7-3**) Notez que si les réalisations de cas d'utilisation de niveau d'analyse montrent les collaborations entre les classes d'analyse, les réalisations de niveau de conception montrent les collaborations entre les éléments de conceptions moins abstraits [12]. Si l'on souhaite que le sous-ensemble de spécification d'un modèle de conception soit empaqueté indépendamment du sous-ensemble de conception d'implémentation, il est important que les réalisations de cas d'utilisation de niveau de conception n'utilisent que des éléments de spécification de conception ou d'analyse—et jamais d'éléments de conception d'implémentation—dans leurs rôles.

[10] Les composants peuvent évidemment posséder de nombreuses interfaces fournies, mais dans cette exemple il n'y en a qu'une.

[11] D'autres composants peuvent participer à plusieurs cas d'utilisation système, c'est pourquoi leurs contrats de réalisation peuvent se situer dans des réalisations de cas d'utilisation multiples. Dans ces cas vous pouvez également inclure, dans le même paquetage que l'interface de composant, un diagramme appelé “{nom de composant} Analyse des utilisations” dans lequel vous placez des liens aux différents diagrammes qui forment les réalisations de cas d'utilisation pour ces cas d'utilisation.

[12] Autre différence probable : Certains des diagrammes “participants” des réalisations de niveau de conception peuvent être des diagrammes de composants décrivant les connexions entre composants, plutôt que (ou en plus) des diagrammes de classe participants comme nous l'avons conseillé pour les réalisations de cas d'utilisation de niveau d'analyse.

- Les contrats d'utilisation et de réalisations pour la“Fonction de conversion des devises” tiers sont regroupés dans un paquetage[13]. Une fois encore, nous pouvons voir que le contrat d'utilisation consiste d'une seule interface, mais dans ce cas le contrat de réalisation est exprimé en utilisant un composant de "spécification" (**Figure 7-2**). (La spécification du contrat de réalisation est similaire, exprimée en utilisant des comportements—dans ce cas, une interaction appelée “achatdecartedecrédit”). Vous trouverez un autre exemple utilisant des composants à la place des collaborations dans **Figure 7-4**.

- Les opérations sont définies dans des interfaces qui peuvent être réalisées par des composants de«spécification»(s'ils sont utilisés) ou bien des classifieurs dans la conception d'implémentation qui implémente les interfaces.

- La spécification des objets de transfert de données (qui serviront de types de paramètres des opérations fournies, et peuvent correspondre à des constructions d'implémentation comme le schéma XML ou les objets de données de service) peut également être inclue dans le cadre du contrat d'utilisation. Pour les composants qui ne sont pas conçus pour être distribuables, vous pouvez choisir ou non de définir des objets de transfert de données comme spécifications des types utilisés en tant que paramètres d'opération. Pour les services distribuables (par exemple, les services Web), les opérations du service ne doivent pas référencer les objets dans un espace d'adresse local, ce qui justifie l'utilisation d'objets de transfert de données.

> ***XDE/Rose***
> Dans les versions précédentes d'UML, les recommandations des réalisations de cas d'utilisation invitaient à utiliser une instance de collaboration par cas d'utilisation et un diagramme d'interaction et de séquence pour chaque flot significatif de la réalisation.
>
> Dans Rational Software Architect, vous devriez souvent pouvoir utiliser une seule interaction et un seul diagramme car les diagrammes de séquence UML2 prennent en charge des notations pour les chemins d'exécution alternatifs.
>
> De plus, dans UML 2 il n'y a plus d'"instance de collaboration". Il y a une "utilisation de collaboration" qui nécessite une collaboration comme type. Par conséquent, dans Rational Software Architect, utilisez des collaborations pour représenter des réalisations de cas d'utilisation.

- [13] Notez que dans la situation hypothétique actuelle, l'entreprise de conversion des devises a fournit à l'entreprise X la spécification UML, que X a intégré à son modèle de conception. L'utilisation d'espaces de domaines Internet inversés peut s'avérer utile dans ce type de scénario.
-

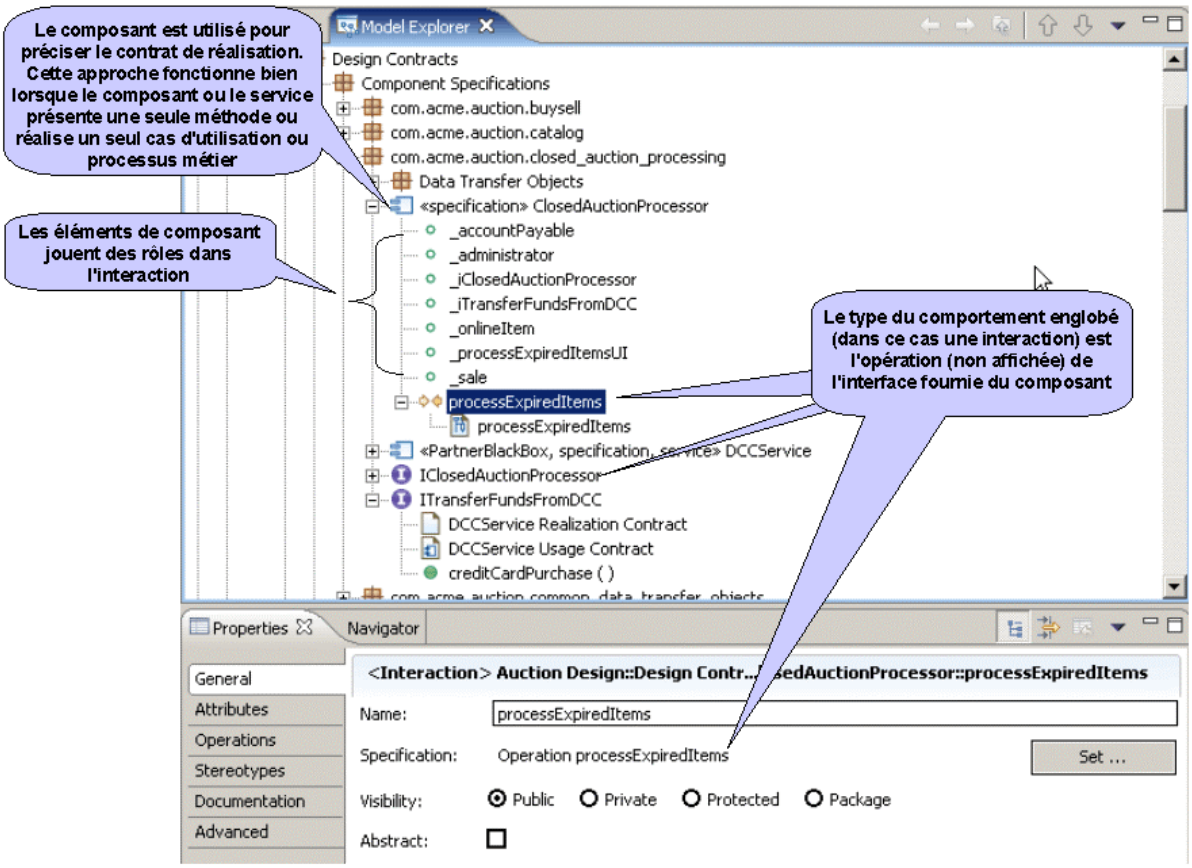


Figure 7-4

- Une approche possible pour définir les conceptions d'implémentations décrite dans **Figure 7-5**.La structure d'implémentation est définie en utilisant des classes simples contenant des opérations. Cette approche est assez répandue dans les modèles de conception créés en utilisant UML 1.x. Vous trouverez une autre approche possible correspondant peut-être aux objectifs d'UML2 dans **Figure 7-6**. Ici, nous voyons que nous utilisons des composants au lieu des classes, les composants n'étant pas propriétaires d'opérations mais de comportements (dans ce cas une interaction).

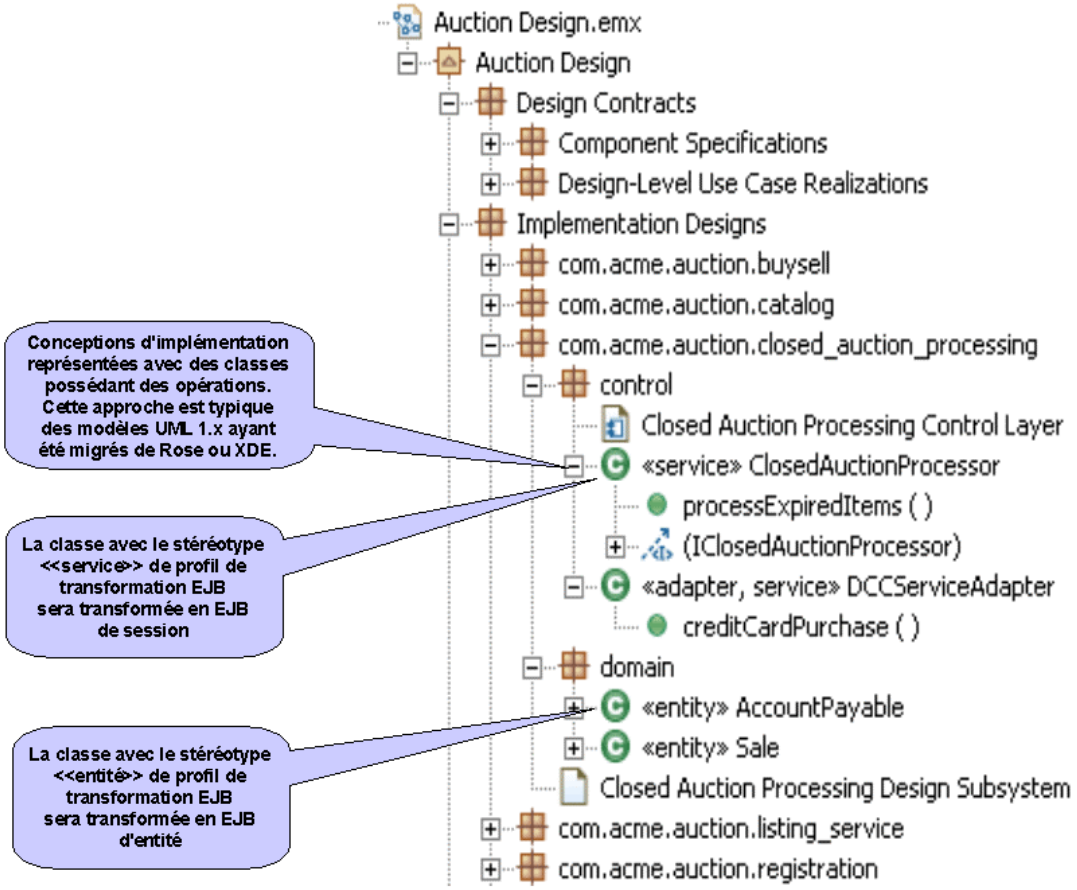

Auction Design.emx
Auction Design
Design Contracts
Component Specifications
Design-Level Use Case Realizations
Implementation Designs
com.acme.auction.buysell
com.acme.auction.catalog
com.acme.auction.closed_auction_processing
control
Closed Auction Processing Control Layer
«service» ClosedAuctionProcessor
processExpiredItems ( )
(IClosedAuctionProcessor)
«adapter, service» DCCServiceAdapter
creditCardPurchase ( )
domain
«entity» AccountPayable
«entity» Sale
Closed Auction Processing Design Subsystem
com.acme.auction.listing_service
com.acme.auction.registration
Conceptions d'implémentation représentées avec des classes possédant des opérations. Cette approche est typique des modèles UML 1.x ayant été migrés de Rose ou XDE.
La classe avec le stéréotype <<service>> de profil de transformation EJB sera transformée en EJB de session
La classe avec le stéréotype <<entité>> de profil de transformation EJB sera transformée en EJB d'entité

Figure 7-5

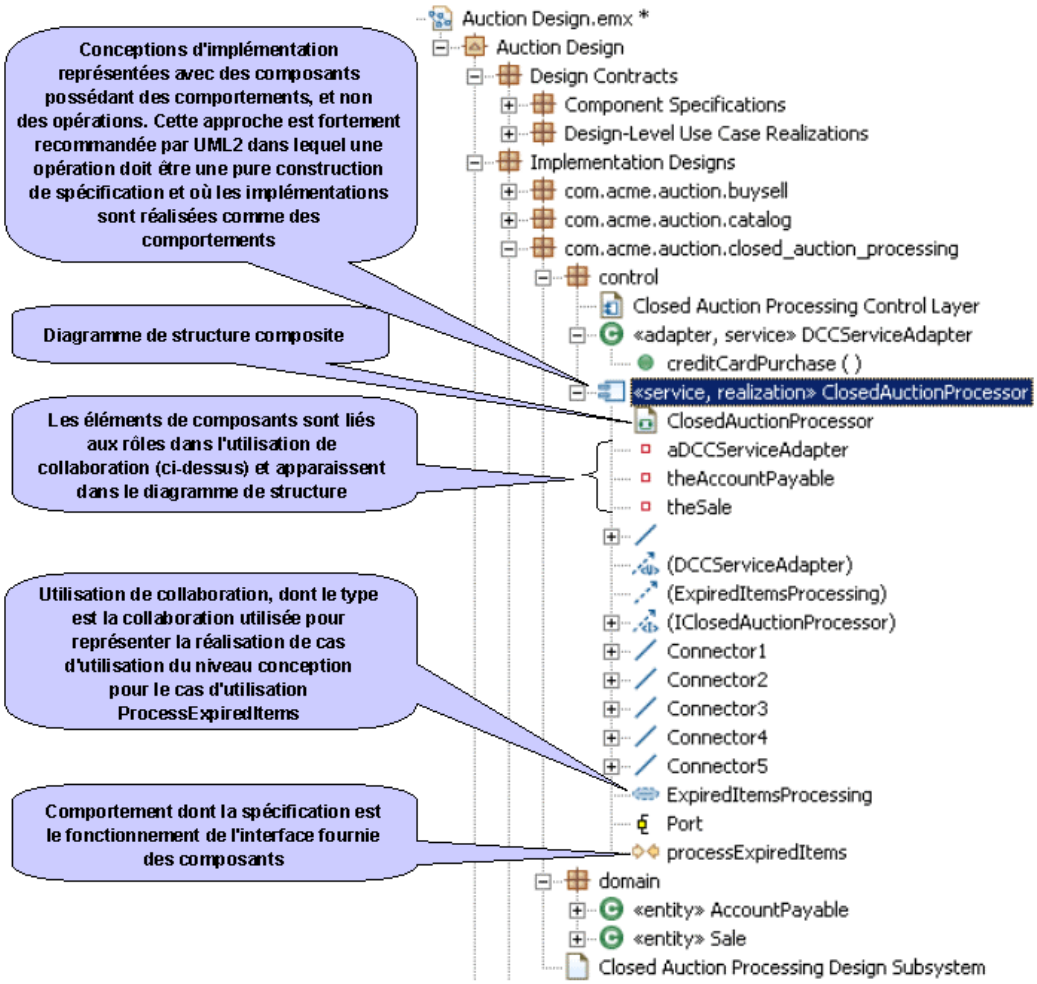

Conceptions d'implémentation représentées avec des composants possédant des comportements, et non des opérations. Cette approche est fortement recommandée par UML2 dans lequel une opération doit être une pure construction de spécification et où les implémentations sont réalisées comme des comportements
Diagramme de structure composite
Les éléments de composants sont liés aux rôles dans l'utilisation de collaboration (ci-dessus) et apparaissent dans le diagramme de structure
Utilisation de collaboration, dont le type est la collaboration utilisée pour représenter la réalisation de cas d'utilisation du niveau conception pour le cas d'utilisation ProcessExpiredItems
Comportement dont la spécification est le fonctionnement de l'interface fournie des composants
Auction Design.emx *
Auction Design
Design Contracts
Component Specifications
Design-Level Use Case Realizations
Implementation Designs
com.acme.auction.buysell
com.acme.auction.catalog
com.acme.auction.closed_auction_processing
control
Closed Auction Processing Control Layer
«adapter, service» DCCServiceAdapter
creditCardPurchase ( )
«service, realization» ClosedAuctionProcessor
ClosedAuctionProcessor
aDCCServiceAdapter
theAccountPayable
theSale
(DCCServiceAdapter)
(ExpiredItemsProcessing)
(IClosedAuctionProcessor)
Connector1
Connector2
Connector3
Connector4
Connector5
ExpiredItemsProcessing
Port
processExpiredItems
domain
«entity» AccountPayable
«entity» Sale
Closed Auction Processing Design Subsystem

Figure 7-6

8. Principes et conseils pour l'organisation interne du modèle de présentation de l'implémentation

> ***XDE/Rose***
> Dans les principes et conseils de structure de modèle XDE, un modèle de présentation d'implémentation était recommandé en tant que dispositif pour fournir une présentation de l'implémentation au niveau sous-système. Les détails de chaque sous-système étaient spécifiés dans le modèle de code du projet implémentant le sous-système.

Strictement parlant, il ne devrait pas être nécessaire d'utiliser un modèle de présentation d'implémentation dans RSA. Si les principes et conseils organisationnels du modèle de conception sont suivis, l'organisation (finale) du modèle de conception doit naturellement prendre forme autour des composants (y compris le "sous-système" plus lourd et des variétés de "service" distribuables). Ensuite, par le biais de transformations, les paquetages de la conception peuvent être mappés aux projets. Par exemple, dans le cas d'une implémentation J2EE, ils correspondront aux projets Java, EJB, Web, application J2EE, et aux autres projets dans lesquels l'implémentation est développée. (Ces projets représentent en fait le modèle d'implémentation pour la solution, comme nous le décrivons dans la section Concepts et terminologie de base de cet article.)

Cependant, vous souhaitez peut-être détailler votre structure de projet dès le début du projet, à moins que vous préfériez voir une représentation plus visuellement explicite de la structure de projet—par exemple, dans laquelle les artefacts représentant des projets et des dossiers sont dotées de mots clé comme «projet» et «dossier», voir «projet EJB» et «Projet Web». Tenez également compte du fait que la représentation d'artefacts d'implémentation à plus fine granularité (JAR, par exemple) ne serait pas appropriée pour le modèle de conception (qui, selon les principes de fonctionnement de Rational Software Architect, doit être indifférent à la plate-forme). Mais ces artefacts peuvent être inclus dans le modèle de présentation de l'implémentation. Ainsi, vous pouvez souhaiter utiliser un modèle de présentation d'implémentation pour plusieurs raisons. **Figure 8-1** représente un exemple de modèle de présentation de l'implémentation

Pour finir, un modèle de présentation de l'implémentation peut être un endroit adapté où consigner des diagrammes informels de différents aspects de la solution. **Figure 8-2** montre un diagramme informel de concept élevé du système d'enchères sur lequel la majorité des exemples de cet article sont basés.

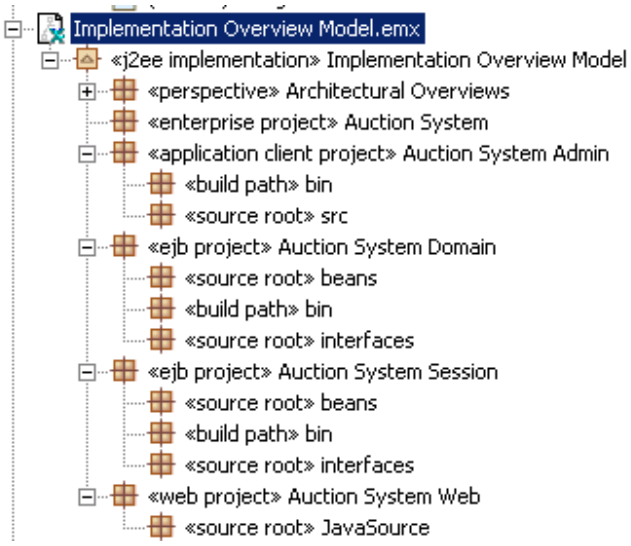


Figure 8-1

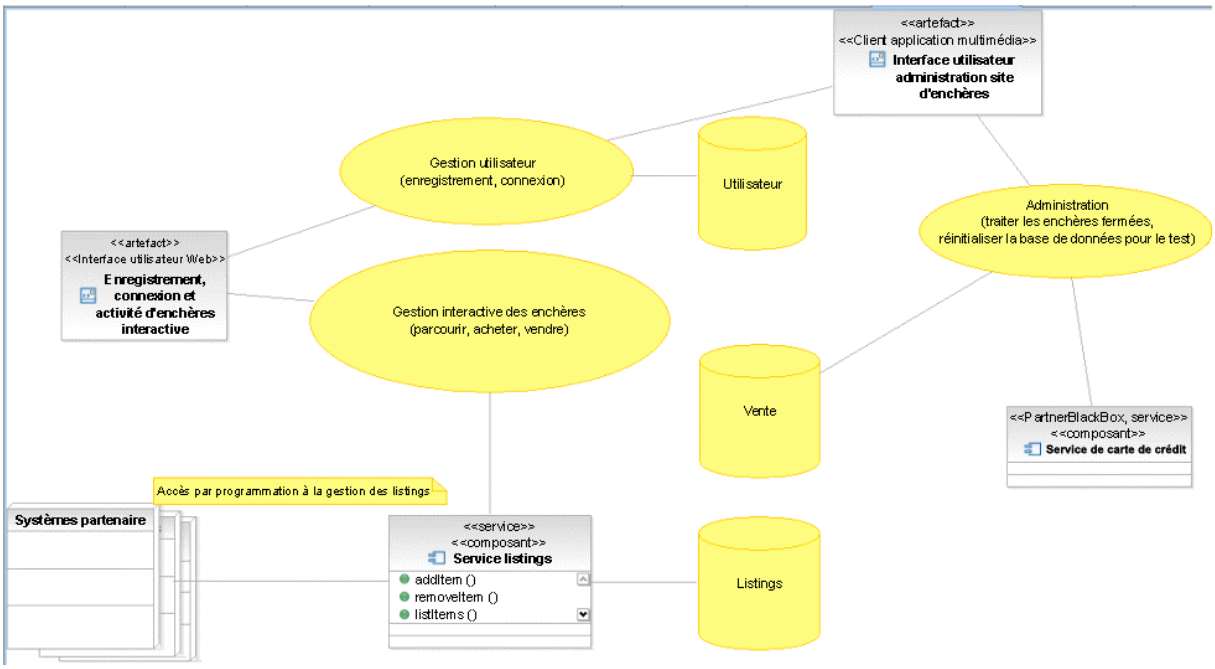


Figure 8-2

9. Principes et conseils pour l'organisation interne du modèle de déploiement

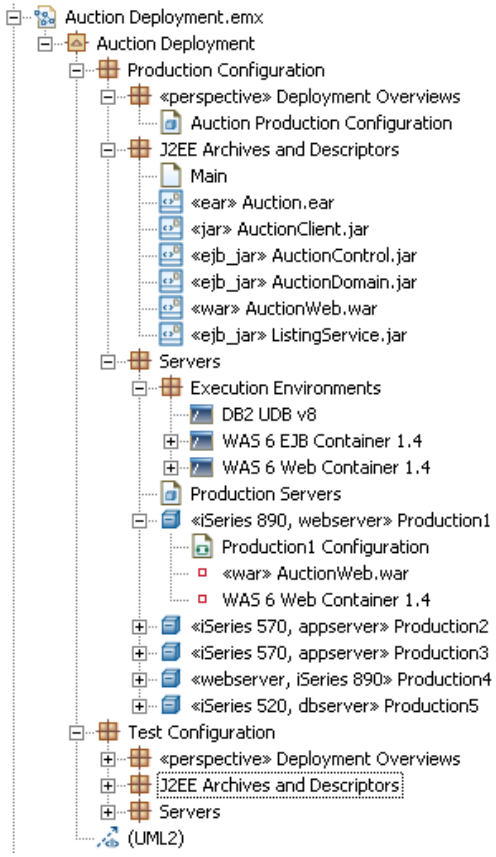


Figure 9-1

Il n'est probablement pas nécessaire d'insister autant sur le modèle de déploiement que sur les autres modèles traités dans cet article. Il n'y aura généralement que très peu d'implications en aval de votre organisation de modélisation de déploiement et de choix de contenu, donc agissez selon ce qui vous semble le plus logique. Cependant – juste pour débuter votre réflexion – vous trouverez ci-dessus une stratégie possible et un extrait de contenu représentatif dans **Figure 9-1**. Dans cette exemple, notez :

1. Les spécifications des configurations de production ont été séparées de celles des configurations test.

2. Les présentations (par exemple, des grappes, des centres de données, ou des "enterprises" sont gérées dans des paquetages de «perspective».

3. Une approche légère a été choisie par rapport à la spécialisation et à la classification des noeuds et des artefacts : une combinaison d'empaquetage et d'utilisation de mots clé. Une approche plus sophistiquée consisterait à développer un profil UML spécialisé définissant des stéréotypes et des propriétés spécialisés appropriés pour décrire et documenter les types de ressources utilisées dans votre propre environnement.

10. Utiliser un fichier de modélisation pour représenter le plan d'architecture logicielle

Du fait de ses outils d'organisation des modèles, comme les liens de diagramme et la prise en charge de multiples fichiers de modèle avec des références intermodèles, il est presque trivial de créer un modèle qui représente réellement le document d'architecture logicielle et les “vues d'architecture 4+1” du RUP.

Une solution simple consiste à **Figure 10-1**. Créer un fichier de modélisation et le remplir avec un ensemble simple de paquetages correspondant aux vues 4+1. (Cet exemple est montré sans paquetage pour la vue de processus, car le système dans cet exemple n'est pas très parlant en terme de simultanéité.)

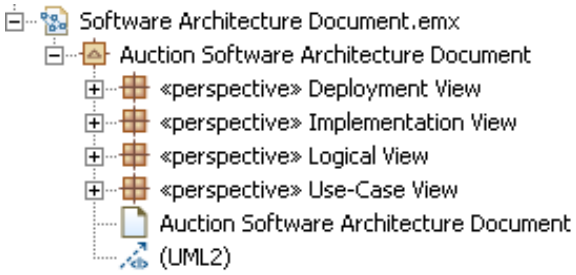


Figure 10-1

Puis vous pouvez composer le diagramme par défaut selon les indications contenues dans **Figure 10-2**. Vous pouvez également ajouter des notes ou des textes supplémentaires à ce diagramme.

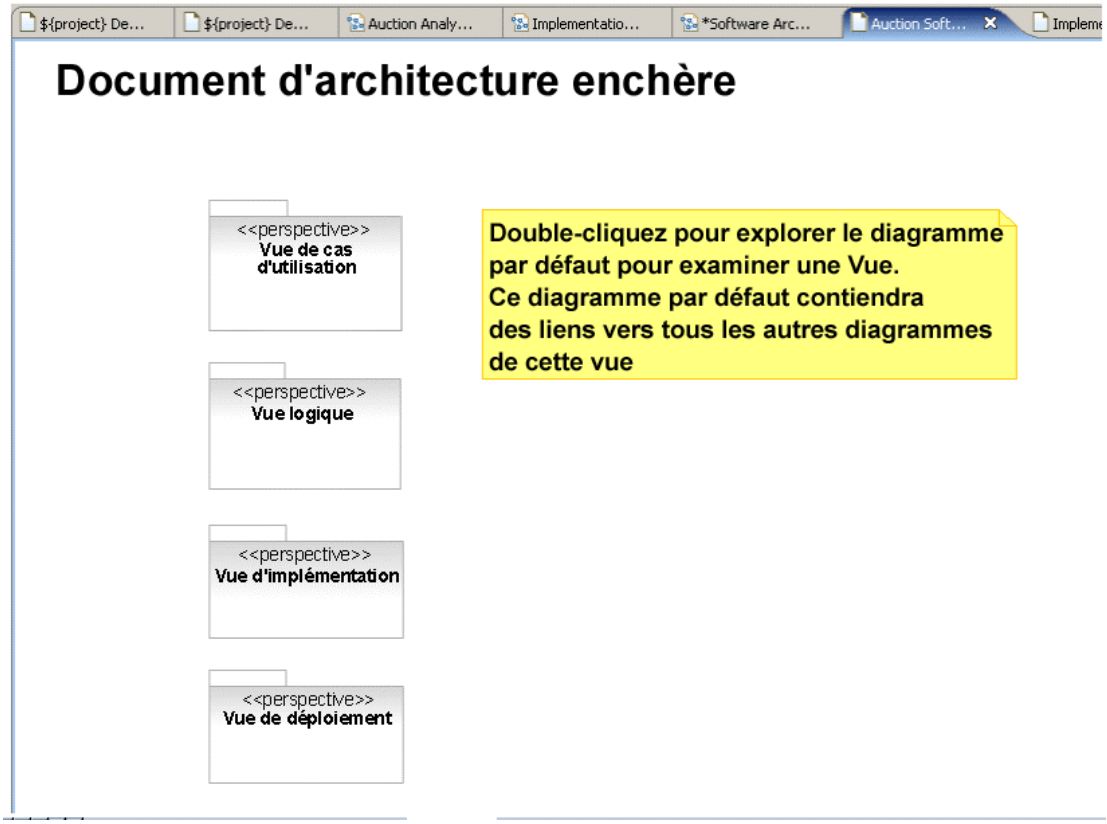


Figure 10-2

Puis créez simplement des diagrammes dans le fichier de modélisation du document d'architecture logicielle, en utilisant ces approches :

- Créez des diagrammes composés en utilisant des éléments sémantiques UML à partir d'autres fichiers de modélisation, et qui représentent des nouvelles vues qu'on ne trouve pas dans d'autres fichiers de modélisation mais qui sont nécessaires dans le cadre du document architectural.

- Créez des diagrammes composés de formes géométriques et/ou des éléments UML “ad-hoc” situés dans le fichier de modélisation du document d'architecture logicielle. (Ces éléments UML n'existent qu'à des fins de documentation ou d'éclaircissement et ne doivent avoir aucune signification sémantique pour la véritable implémentation de la solution décrite.)

- Créez des diagrammes qui contiennent simplement des liens vers les diagrammes existants dans d'autres fichiers de modélisation. (Cette technique fonctionnera bien si le fichier de modélisation du document d'architecture doit être distribué avec les autres fichiers de modélisation afin d'être utilisé par les lecteurs. Si, au contraire, le document d'architecture doit être publié sur le Web, suivez une autre approche.

11. Considérations de développement coopératif

La section “concepts et terminologie de base” a examiné les différents modèles comme le cas d'utilisation, l'analyse et la conception, reconnus par le RUP. Un exemple a également été donné pour illustrer l'argument selon lequel un modèle de “pré-implémentation” dans RSM ou RSA peut être conservé en tant qu'un ou plusieurs fichiers de modélisation, et également que vous pouvez gérer de multiples modèles de cas d'utilisation, etc., chacun de ces modèles étant conservé en tant que fichier de modélisation unique ou fichiers de modélisation multiples. Cette section présente certaines des considérations pour les cas où vous pouvez choisir de conserver un modèle en tant que multiples fichiers de modélisation. Vous pouvez trouver un traitement plus complet de ces préoccupations dans l'aide en ligne de RSA.

Modéliser en équipes

Lorsque plusieurs personnes doivent travailler en parallèle sur un modèle, il peut être plus efficace de découper le modèle en multiples fichiers de modélisation (fichiers .emx). Cela peut permettre à l'équipe d'effectuer son travail en examinant les fichiers de modélisation en accès exclusif. Cela permet de réduire la probabilité que des fusions doivent être effectuées si deux personnes ou plus modifient le même fichier de modélisation.

Cependant, il y a une contre-partie. Il faudrait toujours effectuer des fusions, même si elles seront moins souvent nécessaires lors de l'utilisation de multiples fichiers de modélisation. Or, les fusions traitent les *fichiers*de modélisation individuellement, et non en groupes. En d'autres termes, lorsqu'un modèle est stocké en plusieurs fichiers de modélisation, les fichiers individuels sont traités “hors du contexte” du modèle (logique) complet. Les fusions fonctionnent mieux lorsque la session de fusion a accès à un plus grand contenu informationnel du modèle complet—en d'autres termes, lorsque le modèle est partagé en *moins*de fichiers de modélisation.

Finalement, il est moins important de partitionner les modèles en de multiples fichiers que de structurer les modèles de façon à permettre à de multiples membres de l'équipe de travailler sur ces modèles sans introduire de changements conflictuels pouvant être résolus pendant les fusions. Nous vous recommandons de partitionner (*des instances logiques* de) modèles en multiples fichiers physiques lorsque cela peut réduire de façon significative le besoin d'effectuer des fusions. Vous pouvez généralement réduire le besoin d'effectuer des fusions lorsque les membres de l'équipe peuvent travailler avec les fichiers individuels à la fois *en exclusivité* (c'est-à-dire qu'un seul membre de l'équipe extrait un fichier à un moment donné) et *en isolation* (c'est-à-dire que chaque membre de l'équipe peut apporter ses changements aux fichiers individuels sans avoir besoin d'avoir accès aux autres fichiers qui forment le contexte logique complet du modèle).

Deux approches au partitionnement de modèle

Dans RSM et RSA vous pouvez prendre des décisions quant à la séparation d'instances de modèle logiques en fichiers physiques multiples sur une base ad-hoc, en utilisant la capacité des outils à réusiner les modèles. Cependant, nous vous recommandons de *planifier à l'avance.*Les paragraphes suivants comparent et opposent les deux approches.

Approche planifiée : Décomposer les modèles dès le début

Faites de votre mieux pour anticiper les besoins de votre équipe et usiner ainsi vos modèles en sous-ensembles. Par exemple :

- Usinez chaque modèle de cas d'utilisation en fichiers de modélisation séparés en examinant la façon dont l'expertise fonctionnelle est répartie entre les analystes de l'équipe, et ventilez cela en fonction (cela consiste à créer un fichier séparé plutôt que d'utiliser un paquetage pour organiser les véritables cas d'utilisation dans le modèle). Par exemple, dans une application de fournisseur de soins de santé

vous pouvez avoir un fichier de modélisation pour contenir les cas d'utilisation d'enregistrement du patient, et un autre pour les cas d'utilisation de traitement des commandes. Ou bien vous pouvez décomposer plus en détails les cas de traitement des commandes en fichiers de modélisation séparés pour les prescriptions de laboratoire, de radiologie, de pharmacie, etc.

- Usinez chaque modèle d'analyse en fichiers multiples, où, encore une fois, l'approche recommandée consiste à ventiler les données selon l'attribution d'expertise entre les membres de l'équipe ou les regroupements logiques se retrouvant naturellement dans votre domaine de problème (deux considérations qui se rejoignent très souvent).

- Usinez chaque modèle de conception en fichiers multiples selon les principaux sous-systèmes, services ou composants de votre architecture (en d'autres termes, selon vos prévisions quant à la façon dont le travail d'implémentation sera attribué aux équipes ou aux personnes).

Dans chacun de ces cas, pour chaque modèle vous devez également gérer un fichier de modélisation supplémentaire contenant des éléments et des diagrammes partagés/communs apportant des présentations traversant les frontières de partitionnement (sous-système/paquetage/service).

Pour plus de conseils sur le partitionnement de modèles voir l'aide en ligne de RSM/RSA

Approche ad-hoc : Réusinage de modèles

Dans RSM et RSA, vous pouvez créer des fichiers multiples pour un modèle en commençant par le modèle complet dans un fichier, puis en "découpant" des parties du modèle pour créer des fichiers supplémentaires. Même si vous avez prévu une stratégie de partitionnement par avance, cette approche peut être utile lorsque vous remarquez une nouvelle opportunité d'amélioration des enchaînements d'activités de l'équipe.

Dans le cadre de cette discussion nous nous intéressons au contenu logique des modèles, une partie coupée d'un modèle est donc appelée un “sous-modèle”. Lorsqu'un sous-modèle est coupé du modèle qui le contient, le modèle d'origine conserve un "raccourci" qui pointe vers le sous-modèle. Le sous-modèle ne conserve pas de raccourci pointant vers le modèle d'origine. Par défaut, les éléments du nouveau sous-modèle ne se situeront plus dans l'espace de nom du modèle mère d'origine. Cependant, il est possible, pendant la procédure de découpage, de saisir l'espace de nom du modèle d'origine dans le nouveau sous-modèle. Pour les étapes de création d'un sous-modèle en le coupant du modèle d'origine, voir l'aide en ligne RSM/RSA.

Références inter-fichiers

N'importe quelle paire de fichiers de modélisation RSA peut référencer les éléments de l'autre fichier. Cela peut inclure des références allant d'un projet à l'autre. Certaines de ces références représenteront des relations créées explicitement, comme les dépendances entre les cas d'utilisation ou les associations entre les classes. D'autres sont créées par RSA et sont parfois cachées. Par exemple, les liens de traçabilité peuvent être générés par l'application de transformations, et ces liens sont créés comme des éléments de modèle normaux et visibles. Autre exemple, un diagramme RSA contient des références pointant vers les éléments sémantiques décrits dans ce diagramme (tout élément sémantique peut apparaître dans de multiples diagrammes). Les références d'affichage de l'élément sont des références “cachées” (c'est-à-dire qu'elles n'apparaîtront pas dans l'explorateur de modèle).

Chaque référence inter-fichier représente un point de rupture potentiel comme lorsque :

- Un ou plusieurs fichiers pour lesquels des correspondances ont été établies ne peuvent pas être enregistrés correctement (par exemple, du fait d'une panne du système).

- Les fichiers sont déplacés dans le système de fichiers sans que le serveur de modèle RSM/RSA ne soit au courant.

Ainsi, lorsque vous prévoyez de décomposer un modèle en fichiers multiples, souvenez-vous qu'il est utile de tenir compte du fait d'un résultat peut être une prolifération de références croisées. Encore une fois, l'organisation de modèles pour favoriser la cohésion fonctionnelle et le couplage lâche des unités organisationnelles (paquetages ou fichiers de modélisation) peut favoriser une meilleure expérience de développement coopératif.