Linee guida sulla struttura del modello Per Rational Software Modeler, Rational Systems Developer e Rational Software Architect ("Orientamento Tradizionale di RUP")

White Paper

Bill Smith, Sviluppo basato sul modello, IBM Rational Software

V7.0
24 Marzo, 2006

Tabella dei contenuti

1. Introduzione

Destinatari

Questo documento è progettato per supportare gli utenti dei prodotti RSA (Rational Software Architect), RSD (Rational Systems Developer) e RSM (Rational Software Modeler) (nel complesso "**RS*x***"), specialmente quelli interessati all'applicazione della guida di modellazione trovata in RUP (Rational Unified Process®) per l'utilizzo di RS*x*. Chi utilizza RSM (Rational Software Modeler) può senz'altro considerare il documento utile ma deve essere consapevole del fatto che alcune sezioni riflettono capacità disponibili solo in RSA e RSD.

Il documento è incentrato sulla creazione di una nuova serie di modelli in RSx.Per chi non conosce RSx ma ha utilizzato Rational Rose o Rational XDE e progetta di importare modelli da questi prodotti, può anche fungere da buona guida per la ristrutturazione di questi modelli importati.

Si parte dal presupposto che si abbia qualche nozione di RUP e UML.E che si abbia familiarità con i concetti fondamentali e la teoria delle operazioni di RSx, consultabili in""**Nuovi prodotti di costruzione e progettazione di IBM Rational per Rose e utenti XDE**″

Scopo

RUP descrive una serie di modelli come quelli di caso d'uso, di analisi e di progettazione, che rappresentano prospettive ben definite sui domini di problemi e di soluzione dei sistemi.L'utilità di questa serie di modelli è stata verificata con progetti reali. Anche senza seguire RUP queste strutture sono da prendere in considerazione. Il presente documento descrive come realizzarli mediante RS*x*.

Possono essere presi in considerazione anche altri approcci alla modellazione, ma le relative guide di supporto vanno al di là dell'ambito del documento. Può essere usato ad esempio un approccio alla modellazione di sviluppo guidato dal business per le architetture "orientate al servizio". Viene avviato un modello di processo business forse descritto come un modello di attività UML 2 e si passa poi direttamente a delineare i contratti di una serie di servizi che automatizzano i compiti in esso. —Questo approccio suggerisce strutture di modello diverse da quelle descritte in questa sede.

È importante comprendere che il progetto e le strutture di modello qui descritte costituiscono linee guida, non imperativi.Decidere di modellare un particolare artefatto di RUP in RSx rientra in un processo di sviluppo specifico. È spesso una decisione relativa ad un progetto specifico. Va inoltre ricordato che RUP non rappresenta una serie di regole di processo rigide. È un framework all'interno del quale formulare definizioni del processo. Queste decisioni possono variare da molto formali a molto leggere.

Anche il modo in cui usare la modellazione di UML può variare da molto formale a molto informale. È possibile scegliere di trattare i modelli come formali disegni strutturali da seguire durante la costruzione. Oppure come sketch che suggeriscono ampi schemi di progettazione, ma che sono considerati eliminabili appena il progetto passa alla fase d'implementazione. RS*x* può fornire supporto sia alla fine del processo SE che per gli spettri della modellazione. Le linee guida qui presentate non intendono stravolgere il pensiero dell'utente. Intendono aiutarlo a comprendere come utilizzare le caratteristiche di RSx per semplificare il processo ritenuto migliore.

RS*x* rende possibile l'utilizzo di modelli non solo come progetti ma come precisazioni da cui poter generare significative porzioni di un'implementazione. Questo è possibile utilizzando le trasformazioni da modello a modello e da modello a codice. L'utilizzo delle trasformazioni per praticare MDD (Model-Driven Development) - sviluppo basato sul modello - introduce alcune considerazioni particolari riguardo alle strutture di modello. Utilizzando modelli e trasformazioni è necessario cercare anche risorse specifiche di MDD per RSx sull'area di progettazione e costruzione di Rational di developerWorks®.

Ambito

Il presente documento descrive come rappresentare gli artefatti del modello RUP in RSx.Offre inoltre linee guida per le strutture di organizzazione interna di questi artefatti. Non cerca di

- dichiarare le basi concettuali degli artefatti del modello RUP, né di fornirne descrizioni approfondite.
- descrivere il processo o le tecniche per la specifica della semantica dettagliata o del contenuto dei diagrammi degli artefatti di RUP associati.

Per informazioni generali del tool su come definire, sviluppare e modellare i contenuti degli artefatti di RUP, vedere RUP.

Per informazioni su tecniche specifiche del tool per lo sviluppo del contenuto dei modelli RSx vedere

- la documentazione di prodotto (esercitazioni, esempi, guida online)
- le guide al tool nelle configurazioni di RUP di cui il presente whitepaper rappresenta una parte
- risorse collegate a RSx su developerWorks

Convenzioni tipografiche

Gli argomenti d'interesse per gli utenti di RSx che stanno eseguendo la migrazione da IBM Rational Rose o XDE vengono presentati nella barra laterale, all'interno di una casella di testo delimitata con fondo grigio chiaro:

> ***XDE/Rose***
> Argomento d'interesse per precedenti utenti di XDE o Rose.

Organizzazione del documento

La seziona Concetti base e Terminologia stabilisce un vocabolario di lavoro e fornisce informazioni generali sulle modalità d'implementazione dei modelli nei prodotti di RSx.

Di seguito la sezione Modello RUP a mappatura di modello RS*x* tratta le modalità di supporto di RSx ai tipi di modelli definiti da RUP.

Continuando, sono presenti diverse sezioni che forniscono una guida per la strutturazione dei modelli di vario tipo. Alcune di queste sezioni trattano i differenti modi in cui è possibile utilizzare i modelli in accordo al rigore scelto in termini di processo, approccio alla modellazione e controllo strutturale.

Infine è presente una trattazione sulle questioni associate all'utilizzo dei modelli nei team. Tocca le strategie per la gestione di scala e l'abilitazione alla condivisione dei modelli tra i membri del team per minimizzare conflitti e unioni di file.

2. Concetti base e Terminologia

Modelli

In RUP un modello viene definito come completa specificazione di un problema o dominio della soluzione da una particolare prospettiva. “”Domini o sistemi del problema possono essere definiti da una quantità di

modelli che ne rappresentano diverse prospettive. RUP propone una serie specifica di modelli:

- Modello di caso d'uso del business
- Modello di analisi business
- Modello del caso d'uso
- Modello di analisi (può essere incluso nel Modello di progettazione)
- Modello di progettazione
- Modello di implementazione
- Deployment Model (Modello di distribuzione)
- Modellazione di dati

RUP è agnostico al tool. Finché è attinente a RUP, un modello può essere un disegno su un tovagliolo o su una lavagna, qualcosa in un tool di modellazione o anche un'immagine mentale.Dalla prospettiva di RUP un modello è un concetto logico.

Nel contesto di RS*x* è possibile discutere sui modelli in termini logici ma anche in termini fisici. Considerare l'ipotesi di team al lavoro su due applicazioni: uno composto da tre analisti al lavoro su un'applicazione di gestione time-sheet e un altro da 5 analisti al lavoro su un'applicazione di call center. Entrambi sono impegnati nella raccolta di requisiti e stanno utilizzando RSx per la modellazione di caso d'uso. In termini di RUP si può dire che un team sta costruendo il modello di caso d'uso per l'applicazione di time-sheet e un altro per l'applicazione di call center. “”“”Ma se i team utilizzano RSx è importante considerare che i rispettivi modelli hanno manifestazioni fisiche. Questo è l'argomento della prossima sezione.

Modellazione file

I modelli RS*x* sono persistenti come i file. (Nella terminologia di Eclipse un file viene considerato una ‘risorsa’[1], così nel caso di incontro del termine ‘risorsa di modellazione’ in questo documento o in altre fonti il suo significato è lo stesso di ‘file di modellazione’). In senso più ampio RSx supporta due tipi di questi file:

- **File di modellazione pre-implementazione memorizzati come file individuali nel file system OS host.** Hanno l'estensione “.emx”. Contengono
    - Elementi di semantica UML (classi, attività, relazioni, …)
    - Diagrammi UML che ne descrivono gli elementi di semantica (e i riferimenti visivi a cose presenti in altri domini semantici come Java, C++ o DDL).

- **I file di modellazione d'implementazione vengono memorizzati come progetti Eclipse in uno spazio di lavoro Eclipse.** Tali progetti contengono
    - artefatti d'implementazione (codice sorgente Java, pagine Web, file di metadati su base XML,)…
    - file di diagramma che descrivono e riflettono direttamente gli artefatti d'implementazione.

  La semantica del modello risiede negli stessi artefatti d'implementazione. Ad esempio, il modello semantico di un'implementazione Java viene serializzato e memorizzato come raccolta di file di codici sorgenti Java. Ogni diagramma risiede nel suo stesso file fisico all'interno del progetto. I file di diagramma possono avere dimensione variabile. Il più comune è “.dnx”. I diagrammi di modellazione d'implementazione usano spesso l'annotazione di UML, ma possono utilizzarne anche altre (come IDEF1X, quelle di Information Engineering per la visualizzazione dei dati o quelle proprietarie di IBM utilizzate per la progettazione di livelli Web).

L'attenzione nel presente documento è rivolta alle modalità d'organizzazione delle strutture interne dei modelli pre-implementazione. “”**Nella sua parte restante il file di modellazione del termine è riservato**

[1] In Eclipse una risorsa è un file che ha proprietà e funzioni aggiuntive all'interno del suo ambiente. I file di modellazione descritti qui sono gestiti come ‘risorse’ da Eclipse

ai file di modellazione pre-implementazione (ossia con estensione .emx). “”“”La guida per l'organizzazione dei contenuti dei progetti d'implementazione è disponibile in altre fonti come la guida online per Rational Software Architect, Rational Application Developer e Rational Web Developer Community Edition.

Un file di modellazione non contiene necessariamente tutte le informazioni per un modello (logico). Contiene spesso solo un sottoinsieme di un modello. Nell'esempio precedente un team di tre elementi è al lavoro su un modello di caso d'uso per un'applicazione time-sheet. Il team può scegliere di dividere fisicamente il modello in tre file di modellazione, in modo che ogni membro può lavorare su un sottoinsieme dei casi d'uso senza doversi contendere lo stesso file. La sezione finale del presente documento tratta le questioni associate alla separazione dei modelli e alla gestione dei file di modellazione.

Tipi di modello

In RUP i modelli sono di tipo specifico, ad esempio il modello di caso d'uso, di analisi o di dati. In RS*x* i file di modellazione non sono fortemente classificati ma è possibile, per convenzione, usarne una grande quantità. “”“”Per seguire questa convenzione è possibile stabilire una libera classificazione nei seguenti due modi:

- iniziare con un file di modellazione vuoto (vedere sotto) e, attraverso l'inserimento di nome e contenuto (profili UML inclusi)“”, stabilirne il tipo
- Creare un file di modellazione basato su un modello di maschera predefinito che rappresenta un particolare tipo di modello. “”I prodotti RSx forniscono una serie predefinita di tali maschere per i tipi di modello descritti nel presente documento. È anche possibile creare i propri modelli di maschera (fare riferimento alla guida del prodotto, ai forum e altre risorse su developerWorks).

In ogni caso, il tipo di un file di modellazione in RSx è, di fatto, una convenzione riguardante denominazione e contenuto di quel file. “”Ad esempio, il tool non impedisce ad un file che contiene un modello di caso d'uso di contenere anche le classi che realizzano in generale casi d'uso (cosa che, in termini logici, RUP dovrebbe considerare come parti del modello di analisi o di progettazione).

Queste linee guida suggeriscono di trattare i file di modellazione RSx come file classificati.

Spazi di lavoro, Progetti e Tipi di progetto

I lettori che hanno familiarità con Eclipse, i prodotti WebSphere Studio o Rational Application Developer sanno già che i file risiedono nei progetti, e che questi sono raggruppati e gestiti all'interno di spazi di lavoro. Per i file di modellazione di RS*x* è la stessa cosa.

Per gli scopi di questa discussione, non è necessario analizzare in dettaglio tutti i tipi di progetto disponibili in RSx e Rational Application Developer. L'attenzione può essere focalizzata su due categorie di progetti:

- **Progetti UML**
- **Progetti d'implementazione, che includono tipi specializzati come** i progetti d'azienda, EJB, Web e C++

RS*x* supporta, come detto in precedenza, due tipi di file di modellazione:

- I file con estensione .emx che contengono modelli UML utilizzati per modellazione a livelli di astrazione al di sopra dell'implementazione (requisiti, analisi, progettazione)
- I progetti d'azienda, che contengono la semantica d'implementazione (tipicamente serializzata come artefatto di file di codice d'origine) e diagrammi che riflettono tale semantica.

La regola di allocazione dei modelli ai progetti è semplice:

a) posizionare i file di modellazione pre-implementazione nei progetti UML“”

b) i modelli d'implementazione si gestiscono da soli per loro natura:

[modello d'implementazione] = [progetto d'implementazione]

Esistono eccezioni a questa regola. I seguenti file di modellazione UML sono idonei ad essere posizionati all'interno di un progetto d'implementazione con un linguaggio specifico:

- 'modelli sketch' di progettazione (trattati in una sezione successiva).
- Modelli con diagrammi di sequenza che descrivono test da eseguire sul codice nel progetto.

XDE/Rose

Le teorie di Rose e XDE sull'operazione includono la pratica di perfezionamento iterativo del Modello di progettazione fino al raggiungimento di un livello di astrazione equivalente al codice e quella di utilizzo della tecnologia di sincronizzazione su modello del codice per mantenere le semantiche del suddetto Modello in accordo con il codice stesso. Così come in XDE, dove i modelli d'implementazione esistono non solo in qualità di codice e diagrammi nei progetti, ma anche come file di 'modello di codice', permanenti indipendentemente dagli artefatti d'implementazione, che per natura rappresentano una copia ridondante delle loro semantiche.

La teoria sulle operazioni di RSx incoraggia l'uso di modelli non specifici di una piattaforma a livelli di astrazione maggiori del codice (come un modello di progettazione IT aziendale) e l'uso di trasformazioni per generare codice da tali modelli. Al livello di astrazione del codice, RSA, RSD e RAD consentono semplicemente di tracciare diagrammi di semantiche di codice espresse nelle annotazioni di UML, dispensando l'approccio dall'utilizzo di un modello semantico permanente separato a livello d'implementazione dell'astrazione.

Da notare che RSx non evita il fatto di dover definire i modelli UML a livello di codice di astrazione e generare codice da essi e infatti questo tipo di utilizzo è previsto. Ma RSx non fornisce la tecnologia per mantenere tali modelli in uno stato di auto-sincronizzazione con il codice.

Concetti in Revisione

Le seguenti illustrazioni riepilogano le discussioni precedenti. Riflettono lo scenario descritto in precedenza, quello con due team, uno al lavoro su un'applicazione call center e un altro su un'applicazione di gestione time-sheet.

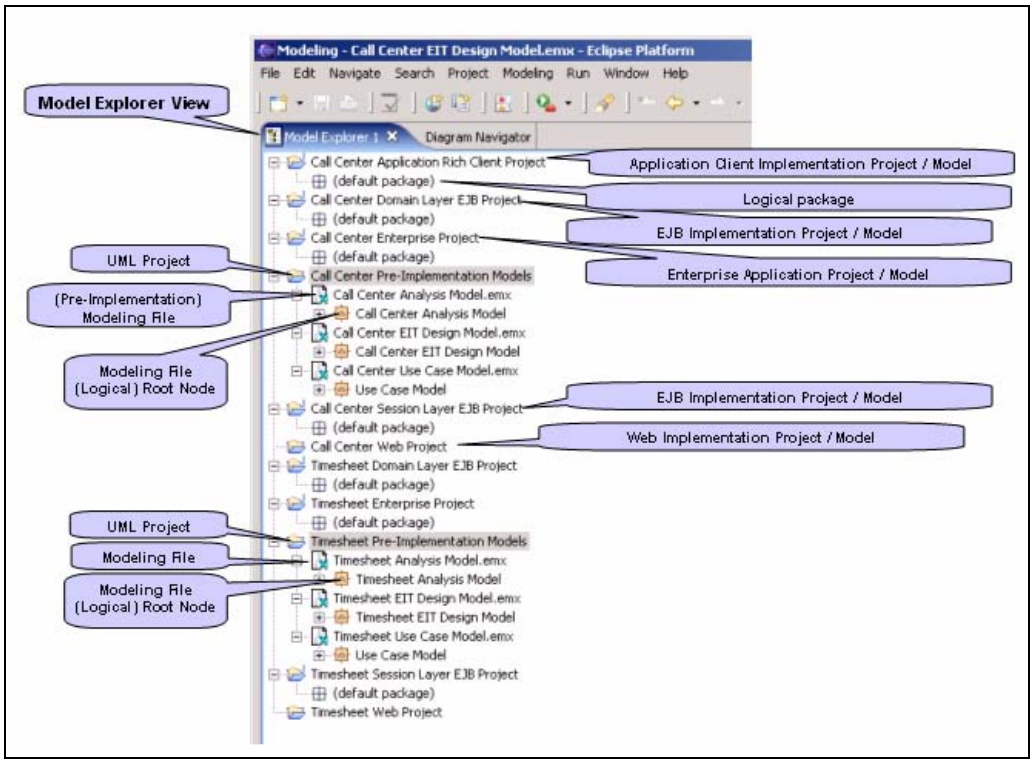


Figura 2-1

RSX fornisce una vista di navigazione[2] che fornisce una vista di modelli logica e fisica. Nell'Explorer i progetti visualizzati nello spazio di lavoro sono definiti nodi di alto livello e all'interno di ciascuno di essi è possibile visualizzarne le risorse. Così in **Figura 2-1** viene descritta in Model Explorer una raccolta di progetti corrispondenti alle due applicazioni nello scenario. Dunque, sono stati utilizzati progetti UML per i modelli pre-implementazione e una raccolta di progetti di tipi appropriati alla soluzione per i modelli d'implementazione.

> ***XDE/Rose***
> Rispetto al Model Explorer RSA quelli in Rose e XDE fornivano solo una vista logica dei modelli. Si noti che la vista di risorse fornita da RSA Model Explorer non è la vista fisica 'pura' fornita dal Navigator di Eclipse. Alcune risorse fisiche sono visibili nel Model Explorer, ma in gran parte vengono rappresentate da icone che ne indicano le viste logiche.

Nella figura 1-2 viene mostrato come è possibile organizzare il modello di caso d'uso Time-sheet in pacchetti rappresentanti alcuni sottoinsiemi funzionalmente coesivi del dominio del problema.

[2] Nelle versioni 6.x, era il "Model Explorer". In 7.0, i modelli sono visualizzati nell'Explorer generico. Le illustrazioni qui presenti sono basate sulla versione 6.0 e descrivono il "Model Explorer".

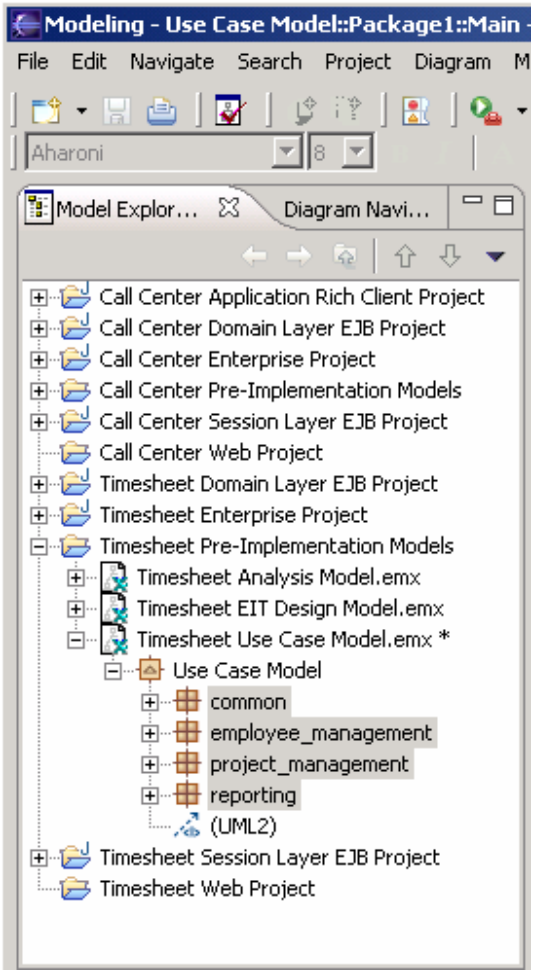


Figura 2-2

In **Figura 2-1 e 1-2** ogni modello di pre-implementazione risiede in un singolo file di modellazione. Nella figura 1-3 viene mostrato come è possibile modificare il modello di caso d'uso time-sheet in più file di modellazione corrispondenti a quegli stessi sottoinsiemi del dominio del problema. Da notare che la radice principale di ogni file di modellazione è stata denominata per mantenere uno spazio nomi coerente lungo tutti i file di modellazione che costituiscono il modello completo.

3. Modello RUP a Mappatura modello RS*x*

La seguente tabella mostra come poter associare, per convenzione, ai tipi di file di modellazione di RSx i modelli RUP più comunemente utilizzati.“”Tale mappatura è generalmente semplice, ma è la chiave per l'utilizzo del presente documento come guida alla pratica di RUP con RSx.I tipi di file di modellazione RSx citati nella tabella vengono trattati immediatamente dopo di essa.Le linee guida per l'organizzazione interna di questi file e i tipi di progetti da mantenere in essi, vengono fornite nelle sezioni successive. queste discussioni successive sono presentate nei termini dei tipi di file di modellazione RSx qui elencati.

Modello RUP	***Tipo di file di modellazione RSx***
Modello del caso d'uso	File di modellazione basati sulla maschera di modello del caso d'uso “” (in alternativa: iniziare con un file di modellazione vuoto e limitare il contenuto per la guida al modello di caso d'uso di RUP)
Modello di analisi	Modello di analisi (in alternativa: iniziare con un file di modellazione vuoto e limitare il contenuto per la guida al modello di analisi di RUP) (in alternativa: utilizzare pacchetti di «analisi» nel modello di progettazione)
Modello di progettazione	Per applicazioni business a n livelli: file di modellazione basato sulla maschera di modello di progettazione IT aziendale “” (in alternativa: iniziare con un file di modellazione vuoto e limitare il contenuto per la guida sul modello di progettazione di RUP) Per altri tipi di applicazioni: iniziare con un file di modellazione vuoto e limitare il contenuto per la guida al modello di progettazione di RUP Per lo ‘sketch’ di progettazione: file di modellazione vuoto Supplemento facoltativo: file di modellazione vuoto aggiuntivo come Modello della panoramica d'implementazione
Modello di implementazione	I progetti Eclipse contenenti artefatti d'implementazione e file di diagramma
Deployment Model (Modello di distribuzione)	Iniziare con un file di modellazione vuoto e limitare il contenuto per la guida sul modello di distribuzione di RUP

Tipi di file di modellazione RSx

File di modellazione vuoto

RSx fornisce l'opzione per creare un “modello”vuoto (File→New→UML Model→Blank Model). Un modello vuoto è un file di modellazione non basato su maschera di modello. “”Non ha profili speciali associati ad esso né contenuto predefinito diverso da un diagramma principale (in formato libero). “”**È possibile utilizzare un file di modellazione vuoto come punto di partenza per ogni tipo di modello.** Scegliendone il nome, il contenuto ed i profili da applicare, è possibile utilizzarlo per costruire un modello di caso d'uso, di analisi, di progettazione o tutti gli altri tipi di modello RUP.

File di modellazione di caso d'uso

RSx fornisce l'opzione per creare un file di modello di caso d'uso basato su una maschera. “”Tale maschera contribuisce al contenuto predefinito come descritto in ***Figura 3-1***. (È al di fuori dell'ambito del presente documento spiegare la modalità di utilizzo del contenuto di blocco costruito e delle stringhe di ricerca. “” Le maschere contengono istruzioni ampiamente esplicative.)

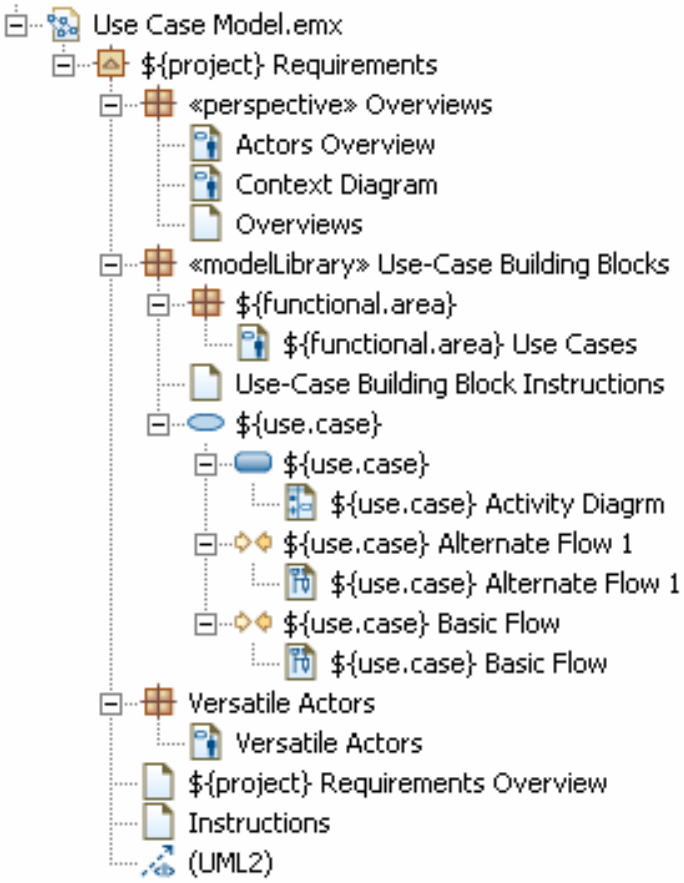


Figura 3-1

file di modellazione di analisi

RSx fornisce l'opzione per creare un file di modello di analisi basato su una maschera. “”Tale maschera contribuisce al contenuto predefinito come descritto in ***Figura 3-2***. In aggiunta viene applicato un profilo d'analisi ai file di modello creati da tale maschera“”:

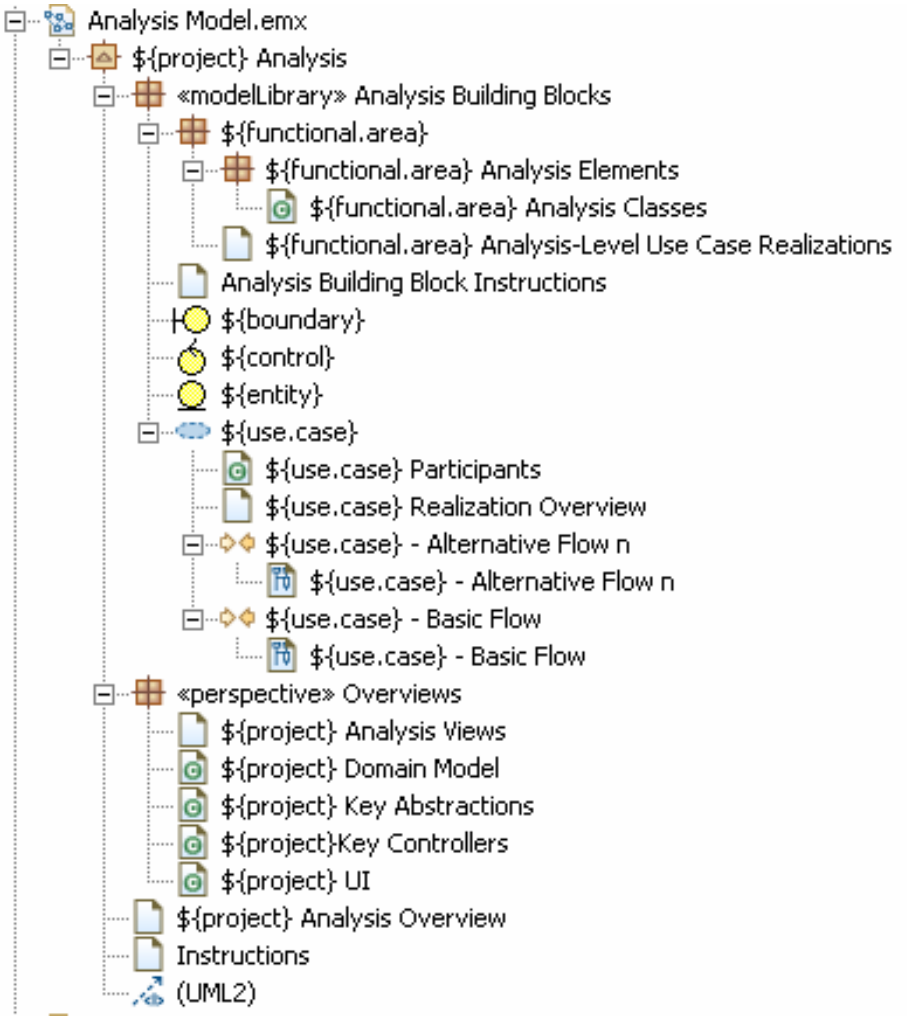


Figura 3-2

File di modellazione di progettazione IT aziendale

RSx fornisce l'opzione per creare un file EITDM (“Enterprise IT Design Model”) - modello di progettazione IT aziendale basato su una maschera. Tale maschera fornisce contenuto predefinito come descritto in **Figura 3-3** , In aggiunta un profilo di trasformazione “EJB ”[3] viene applicato per modellare file creati da questo modello.Questo è il modello appropriato da usare per la progettazione (e facoltativamente per l'analisi) quando sono identificate le applicazioni business e vengono utilizzate le trasformazioni che generano codice di RSx a supporto della creazione di tali applicazioni.

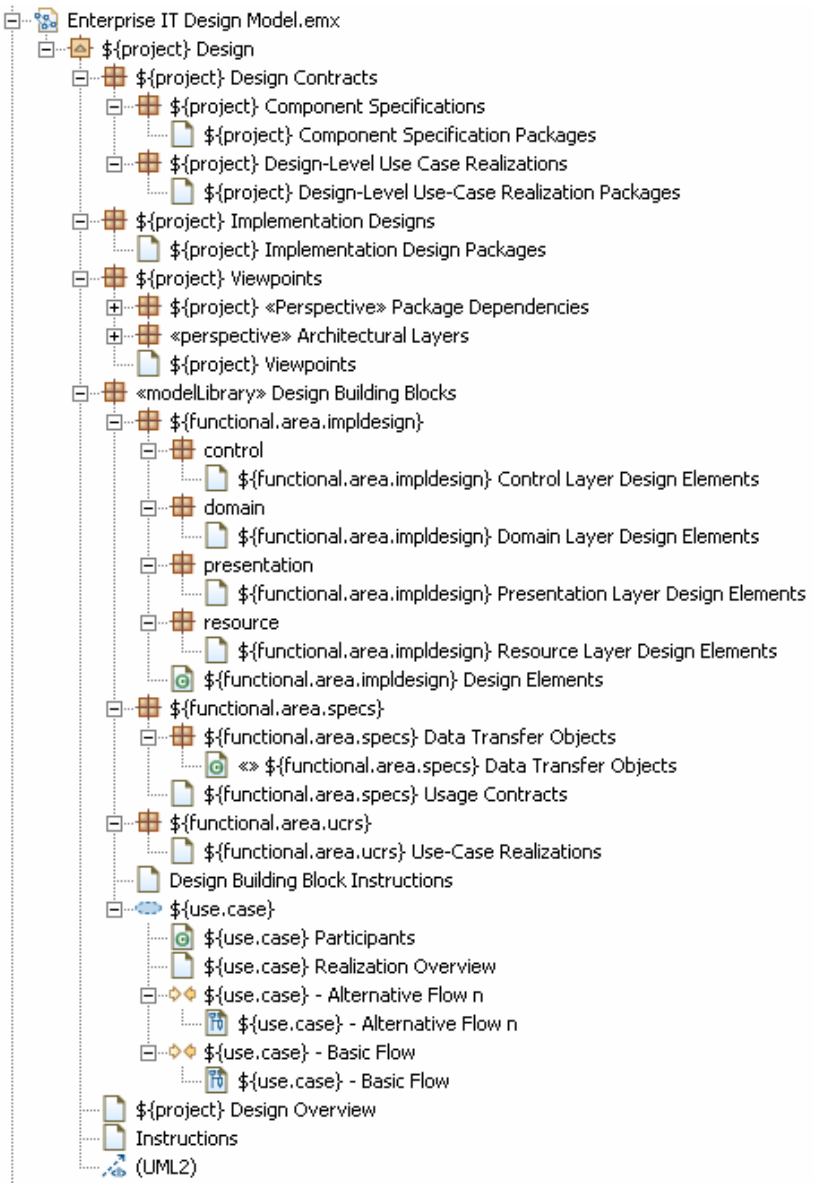


Figura 3-3

[3] La serie di profili che abilitano la trasformazione fornita come parte della maschera di modello di progettazione EIT evolve probabilmente con i rilasci degli aggiornamenti del prodotto.

File di modellazione della panoramica d'implementazione

Come parte del modello di progettazione, può essere utile, in aggiunta, definire un modello di panoramica d'implementazione per acquisire una vista di alto livello su come organizzare l'implementazione. “”Questo modello deve essere utilizzato all'inizio della fase di progettazione prima della generazione o della scrittura del codice -- per rappresentare gli attuali progetti e cartelle/pacchetti di RSx o “”–Rational Application dove sono previste le posizioni del codice e dei file collegati (metadati, descrittori di distribuzione, ecc..) . È possibile utilizzare questo modello per mostrare le dipendenze anticipate tra progetti e pacchetti, cosa che può rivelarsi utile per l'identificazione dei requisiti della build del sistema. Il modello di panoramica d'implementazione può anche essere utilizzato per mantenere diagrammi concettuali informali dell'architettura della soluzione.

Modello di implementazione

Come detto in precedenza, in RS*x* un modello di implementazione consta di un progetto contenente artefatti d'implementazione e (facoltativamente) diagrammi che descrivono tali artefatti [4].

“Modelli sketch”

Come rilevato nella sezione Concetti base e Terminologia, è possibile trattare modelli di progettazione come disegni strutturali formali mantenuti durante il periodo di funzionamento del sistema ed utilizzati per il supporto/rafforzamento del controllo strutturale. “”Oppure, è possibile trattarli come sketch per suggerire una progettazione e favorirne la chiarezza e la comunicazione, ma sono disponibili solo dopo l'inizio dell'implementazione. RSx supporta entrambi gli approcci. Le sue caratteristiche generalmente non tendono ad un approccio piuttosto che ad un altro, ma le scelte relative alla modalità di utilizzo dei modelli di progettazione di certo aiutano a determinare quali funzioni di RSx usare e in che modo. Se la distinzione diviene importante nel contesto delle linee guida qui presentate, viene utilizzato il termine modello di sketch ad indicare che un modello è utilizzato nel modo più “”‘disponibile’.

[4] Per creare questi diagrammi, invece di File →→Nuovo Modello UML per creare un modello usare File→→Nuovo Diagramma di classe per creare un diagramma in cui poter comporre ‘viste’ di codice nell'annotazione di UML (o altri). Ogni diagramma individuale è permanente come un file separato con estensione .dnx è può essere controllato come un file di codice. Questi diagrammi non contengono informazioni di semantica, solo annotazioni. Tutte le informazioni di semantica rilevanti risiedono all'interno del codice. Se viene modificato qualcosa, come il nome di una classe o la firma di un'operazione in uno di questi diagrammi, la modifica viene applicata allo stesso codice sottostante. Quando vengono effettuate tali modifiche nel codice (con un editor di testo) i diagrammi che lo contengono vengono automaticamente aggiornati.

4. Linee guida generali e tecniche per l'organizzazione delle strutture interne dei modelli

Il tool principale per l'organizzazione del contenuto dei modelli UML è il pacchetto. I pacchetti UML servono due scopi primari:

- informazioni sul modello di separazione, di organizzazione e di etichettatura
    - elementi di raggruppamento che corrispondono ad un argomento specifico nel dominio del problema o della soluzione
    - separazione dei differenti tipi di informazioni di modello come interfacce, implementazioni, diagrammi, ecc..
    - raggruppamento di elementi per definire e controllare la loro dipendenza su altri elementi
    - raggruppamento di diagrammi che forniscono viste alternative su uno stesso modello
- stabilire spazi nomi
    - per elementi di modello
    - per artefatti d'implementazione generati da elementi del modello (questo può coinvolgere mappatura tra modelli e spazio nomi del linguaggio d'implementazione)
    - per un'unità di riutilizzo

RUP ha sempre proposto specifiche strategie di impacchettamento per diversi tipi di modello. Sono riflesse nelle sezioni su specifici tipi di modello nel presente documento. RSx introduce alcuni tool organizzativi aggiuntivi qui descritti:

Rappresentazione dei punti di vista mediante pacchetti di «prospettiva»

Nei casi in cui è preferibile vedere gli elementi organizzati in diversi modi è possibile creare pacchetti aggiuntivi con diagrammi che descrivono gli schemi organizzativi alternativi. Questa stessa tecnica può essere utile ovunque sia necessario rappresentare una particolare vista sul contenuto del modello, vista che ne attraversa lo schema d'impacchettamento.RSx supporta tale tecnica fornendo uno stereotipo di pacchetto di «prospettiva» come parte del proprio 'profilo base' di UML. Tale pacchetto può essere visto come l'equivalente di un RUP per il punto di vista IEEE 1471- 2000 o SE (System Engineering).“”

Non posizionare elementi di semantica (classi, pacchetti, associazioni, ecc.) all'interno di pacchetti di «prospettiva». Posizionarvi solo i diagrammi che descrivono le viste basate sulla problematica organizzativa alternativa o sul punto di vista dell'applicazione. Applicare lo stereotipo di «prospettiva» ad un pacchetto comporta diverse cose.Identifica visivamente quel pacchetto come rappresentante di un particolare punto di vista. Supporta inoltre una regola di convalida del modello che avvisa di eventuali inserimenti di elementi di semantica in un pacchetto di «prospettiva». Funge anche da indicatore dei pacchetti da ignorare per le trasformazioni RSx.

Creazione di descrizioni auto aggiornanti per specifiche considerazioni mediante i Diagrammi di argomento

In contrasto con i 'normali' diagrammi in cui sono manualmente inseriti gli elementi da descrivere, i contenuti di un diagramma di argomento sono determinati da una query eseguita verso i contenuti del modello esistente. Per creare un diagramma di argomento selezionare un elemento di modello 'topico', poi definire gli altri elementi che devono apparire nel diagramma in base ai tipi di relazioni che hanno con esso. Se il contenuto della semantica viene modificato, il diagramma si regola di conseguenza.

Esame modelli attraverso diagrammi di navigazione

I diagrammi di navigazione non sono tool per l'organizzazione del modello. Il loro scopo è quello di semplificare la scoperta e la comprensione del contenuto di un modello senza dover comporre manualmente diagrammi. Ma nel contesto dell'organizzazione di un modello può essere utile considerarli poiché possono limitare la necessità di comporre diagrammi permanenti. Cosa che riduce la dimensione e la complessità di tali modelli, rendendoli così più facili da organizzare.

I diagrammi di navigazione sono in parte come quelli di argomento, ma con la differenza chiave che i primi non sono mai permanenti, sono sempre generati al momento. Per produrre un diagramma di navigazione selezionare un elemento del modello (da un diagramma o dal Model Explorer) e utilizzare il menu di scelta rapida su “Esplora nel diagramma di navigazione. ” Questo produce un diagramma che descrive l'elemento selezionato come ‘punto focale’ con elementi collegati presentati in un layout radiale attorno ad esso. Naturalmente è possibile selezionare uno di questi elementi collegati nel diagramma di navigazione e renderlo punto focale di un altro di questi diagrammi, e così via.

Navigazione tra diagrammi

In RSx esistono due meccanismi per la navigazione tra diagrammi:

- È possibile trascinare un nodo di diagramma dal Model Explorer ad un altro diagramma ‘host’. In seguito, facendo doppio clic sull'icona risultante nel diagramma host viene aperto il diagramma relativo.
- Se viene creato un nuovo pacchetto UML in un modello, viene generato automaticamente un diagramma “principale” (in formato libero). È il diagramma “”‘predefinito’ del pacchetto. È possibile rinominarlo in modo diverso, ma viene sempre considerato quello“” ‘predefinito’. È possibile anche selezionare un diagramma diverso nel pacchetto e renderlo ‘predefinito’ per quel pacchetto. Lo scopo del diagramma ‘predefinito’ è questo: se il pacchetto stesso viene posizionato in qualche altro diagramma ‘host’ è possibile poi fare doppio clic su di esso per aprirne il diagramma predefinito.

Questi meccanismi supportano la seguente linea guida organizzativa, applicabile a modelli di ogni tipo:

1. Comporre il diagramma principale (o altro diagramma predefinito) per ogni file di modellazione da descrivere.
    a. ogni pacchetto di alto livello nel file di modellazione
    b. le icone per tutti i diagrammi che risiedono nel pacchetto di radice principale del file di modellazione (in altre parole, non descrivere l'icona per il diagram ma predefinito)
2. Comporre il diagramma principale (o altro diagramma predefinito) per ogni pacchetto di alto livello da descrivere
    a. I pacchetti che contiene direttamente
    b. le icone per tutti i diagrammi che contiene direttamente
3. Ripetere questo pattern per tutti i successivi bassi livelli di pacchetti

5. Linee guida per l'organizzazione interna del modello di caso d'uso

Organizzazione di alto livello del modello di caso d'uso

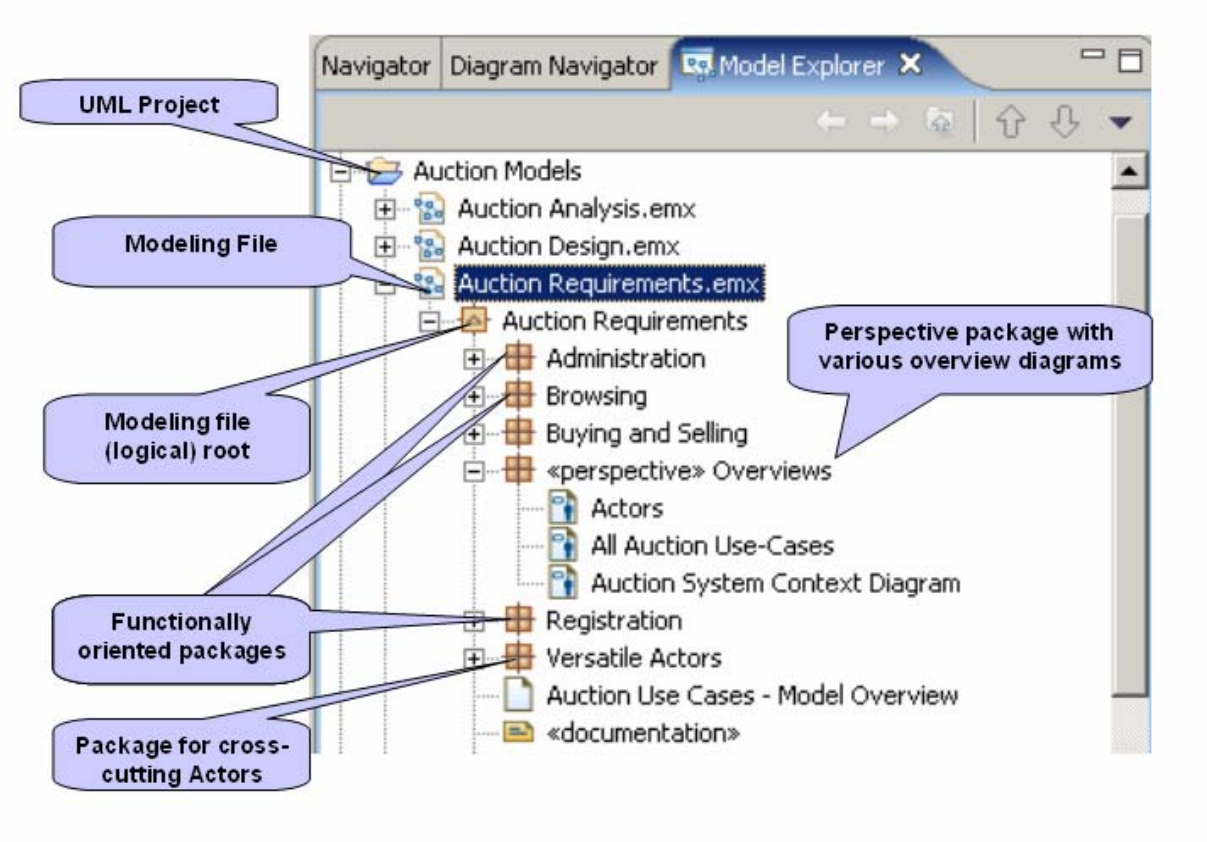


Figura 5-1

Figura 5-1 sono illustrate le seguenti linee guida per la strutturazione dei modelli di caso d'uso :

1. Utilizzare i pacchetti di alto livello per stabilire raggruppamenti orientati alla funzionalità. Logica:

    - Questo in genere si associa bene alle questioni di divisione del lavoro quando un team deve operare su un modello di caso d'uso. E prepara nel caso in cui si decida in un secondo momento di rompere il modello in più file di modellazione a causa di problematiche legate alla contesa di file (è possibile creare un file di modellazione separato per ogni pacchetto di alto livello).

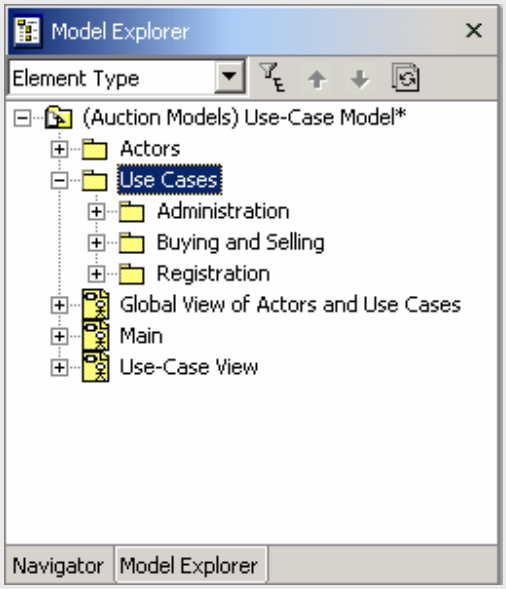
    - Comparato ad altri approcci organizzativi, questo si associa meglio all'organizzazione di un eventuale implementazione. È importante con l'utilizzo di trasformazioni per riempire tutti i successivi bassi livelli di astrazione. In modo specifico, creando contenuto di riempimento in un modello di analisi basato sul modello di caso d'uso, è preferibile che una struttura d'impacchettamento per quest'ultimo si associ bene a quella desiderata per il modello di analisi di destinazione. E a sua volta è preferibile che la struttura d'impacchettamento del modello di analisi si associ bene al modello di progettazione mentre quella di quest'ultimo all'insieme di progetti che comprendono l'implementazione. Più semplici sono le mappature, minore è il lavoro richiesto per la configurazione delle trasformazioni da un livello di astrazione all'altro.

2. Utilizzare un altro pacchetto di alto livello per acquisire attori versatili o ampiamente autorizzati.

3. Utilizzare i diagrammi in pacchetti di «prospettiva» per la cattura di viste di esplorazione di alto livello dei casi d'uso. Logica:

    o Fornire viste di esplorazione trasversale e di casi d'uso 'architettonicamente significativi' mentre vengono mantenuti elementi di semantica del modello organizzato in raggruppamenti orientati alla funzionalità.

XDE/Rose
La guida di RSX in qualche modo rivede quella tradizionale per l'organizzazione di alto livello del modello di caso d'uso, che consisteva nel creare un pacchetto per gli attori ed uno per i casi d'uso. Poi -- in base a quanto richiesto per la dimensione e la complessità del modello, è consigliato l'utilizzo di pacchetti di livello più basso per stabilire raggruppamenti orientati alla funzionalità come mostrato in questo esempio basato su XDE:



Contenuto del modello di caso d'uso

Fornire un esercitazione dettagliata sulle modalità di scrittura di buoni casi d'uso e su cosa fare o non fare della relativa modellazione va al di là dell'ambito del presente documento. Di seguito, tuttavia, viene riportata una breve discussione su cosa è possibile includere in un modello di caso d'uso oltre agli attori ed ai casi d'uso.

- **Consigliato**: creare un dominio 'principale' alla radice del modello che ne descriva gli altri pacchetti e che, di questi, supporti l'analisi ed i rispettivi diagrammi 'principali'

- **Consigliato: in ogni pacchetto di caso d'uso, includere un diagramma che ne descriva i casi d'uso, le relazioni tra di loro e gli attori che ne fanno parte.** (In caso di grandi quantità di casi, è consigliato più di un diagramma)

- **Consigliato:** descrivere i flussi principali ed alternativi di ogni caso d'uso nel proprio campo di documentazione[5] (vedere **Figura 5-2**)

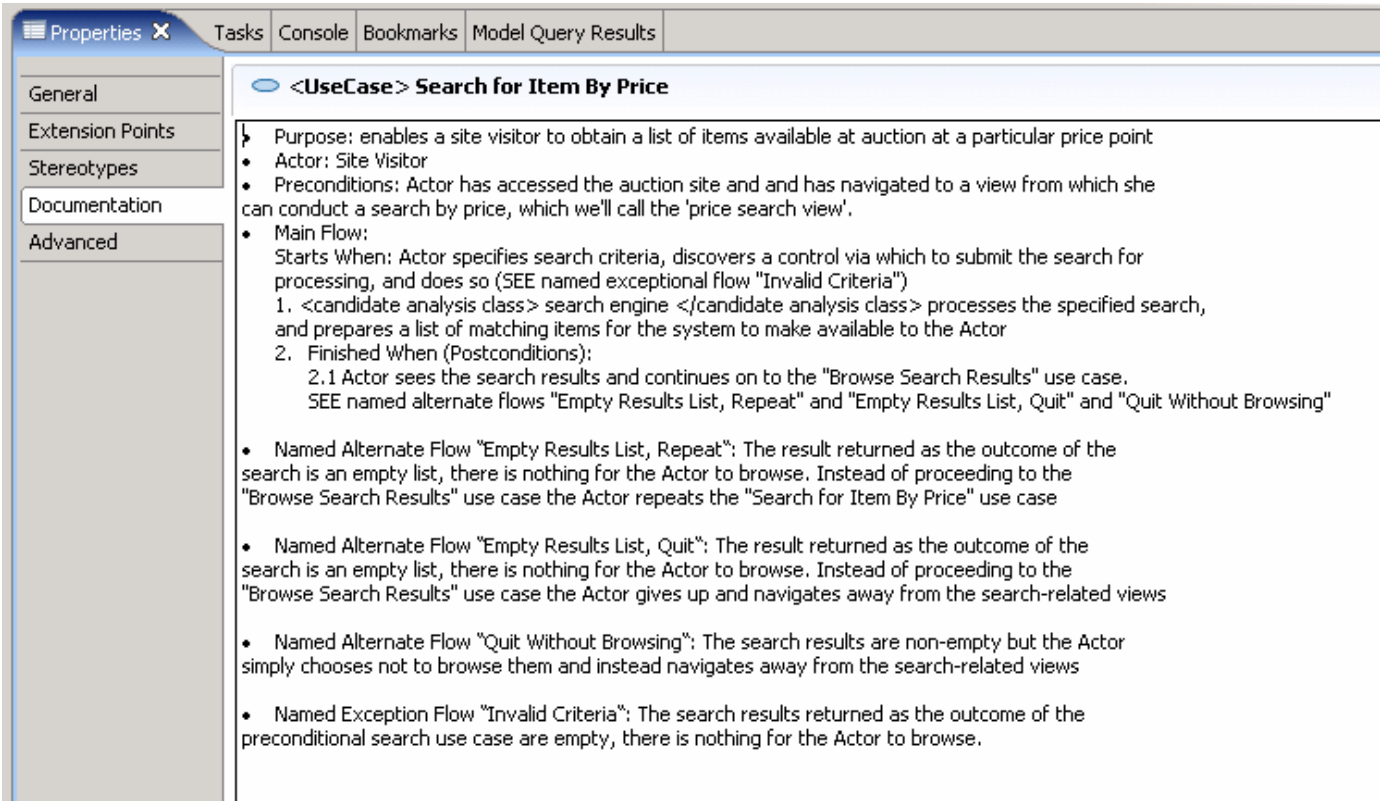


Figura 5-2

- **Facoltativo:** se consentito dalla complessità di un caso d'uso, aggiungere un diagramma di attività e comporlo affinché ne rifletta i flussi di attività complessiva. (vedere **Figura 5-3**) **Logica:** quest'azione aiuta a mostrare le condizioni che corrispondono ad ognuno dei flussi (principali e alternativi) e ad assicurare che tutti i diversi flussi alla fine ri-convergono. (Aggiungere un diagramma di attività in RSx dà come risultato un'attività automaticamente aggiunta e posta al di sotto del caso d'uso).

- **Facoltativo:** modellare una realizzazione a 'scatola nera' per ogni flusso denominato (principale, alternativo ed eccezionale) del caso d'uso: aggiungere un'occorrenza di collaborazione al caso d'uso; aggiungervi un'istanza d'iterazione corrispondente al suo flusso principale e una per ciascuno dei flussi denominati alternativi o eccezionali; comporre un diagramma di sequenza (o in alternativa, un diagramma di comunicazione) per ciascuna istanza d'iterazione. Queste istanze di collaborazione di caso d'uso non devono essere confuse con le realizzazioni di caso d'uso al livello dell'analisi (come descritto nel Modello d'analisi) o con quelle al livello della progettazione (come descritto nel Modello di progettazione). Queste sono realizzazioni a scatola bianca del caso d'uso e descrivono le interazioni tra gli elementi interni di una soluzione. “”Le occorrenze di collaborazione qui proposte sono interazioni a scatola nera tra gli attori ed il sistema. “”(Vedere **Figura 5-3**) **Logica:** quest'azione fornisce stakeholder non tecnici con immagine ad alto livello sulle modalità di interazione tra utenti e sistema. Può anche aiutare a identificare le diverse viste (schermate o pagine) richieste come parte dell'implementazione.Stabilisce poi formalmente la denominazione dei diversi flussi del caso d'uso (scenari) assegnando quei nomi ad elementi di modello semantico (come le

[5] La formattazione descritta nell'esempio di descrizione di caso d'uso è stata conseguita creandone il 'modello' testuale mediante un editor compatibile con RTF, poi copiando e unendo il modello nel campo di descrizione del caso d'uso.

occorrenze di collaborazione).

> ***XDE/Rose***
> In UML 1.x è preferibile usare “l'istanza di collaborazione piuttosto che l'occorrenza per tale scopo. ”“”

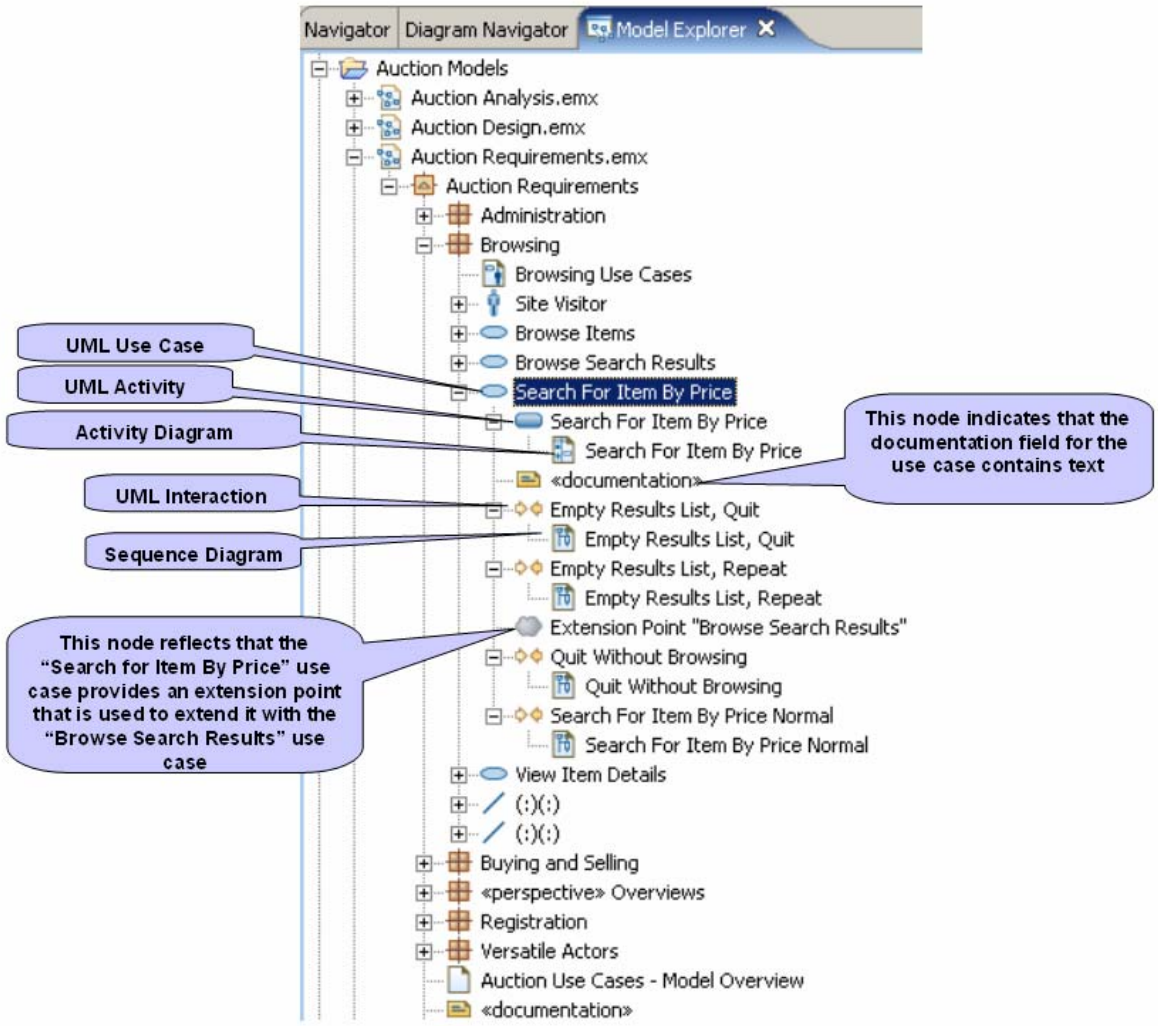


Figura 5-3

- **Facoltativo:** seguendo la guida di RUP per identificare viste di struttura ‘architettonicamente importanti” e in particolare per mantenere un documento di architettura software, aggiungere un pacchetto di «prospettiva» di alto livello per contenere i diagrammi di caso d'uso che descrivono quelli architettonicamente più significativi. È possibile scegliere di denominare la vista di caso d'uso del pacchetto dell'architettura. “”

6. Linee guida per l'organizzazione interna del Modello di analisi

Il modello di analisi rappresenta una ‘primo passo’ per una soluzione. È un passo cruciale per conseguire i requisiti di progettazione finale e si basa sulla cattura di informazioni sul dominio business e sul mostrare elementi di soluzione potenziali ad un alto livello di astrazione vicina al business. È dove risiedono le classi di analisi e le realizzazioni di caso d'uso del livello di analisi. È attraverso il processo di modellazione delle realizzazioni di caso d'uso (sopratutto mediante diagrammi di sequenza) che si inizia a scoprire quali classi sono necessarie per la soluzione, in particolare quelle che corrispondono al ciclo di vita necessario nei diagrammi di sequenza. –Esistono anche alcune regole pratiche che possono essere applicate per suggerire contenuto di modello di analisi basato su quello del modello di caso d'uso. Verranno trattate più avanti nella presente sezione.

In RUP, mantenere o meno un modello di analisi indipendente da quello di progettazione è una decisione specifica del progetto, da prendere in base a quanto sia ritenuta positiva un'azione del genere. Se viene creato un modello di analisi separato, ma non mantenuto, allora le classi di analisi vengono spostate nel modello di progettazione e poi perfezionate. Oppure, un modello di analisi può evolversi gradualmente in un modello di progettazione[6]. Nei termini specifici del prodotto, ecco di seguito alcune opzioni da considerare:

1. Creare un modello di analisi che risiede in un file di modellazione (o serie di file) in base alla maschera del modello di analisi. Utilizzare poi un processo manuale o trasformazioni automatizzate per creare versioni perfezionate degli elementi di analisi in un secondo file di modellazione (o serie di file) in base alla maschera del modello di progettazione IT aziendale, infine liberarsi dei file di modellazione di analisi. Questo lascia intatta l'opzione di mantenimento o di abbandono del modello di analisi separato.
2. Eseguire modellazione a livello di analisi in un file di modellazione (o serie di file) in base alla maschera del modello di progettazione IT aziendale, a cui applicare il profilo d'analisi. In questo modo è possibile avviare la modellazione delle realizzazioni di caso d'uso mediante le classi di analisi e poi, nel tempo, perfezionarle in modo che le interfacce della progettazione possano svolgere il proprio ruolo.
3. Un ibrido tra la seconda e terza opzione consiste nel mantenere un modello di analisi di tipi, all'interno degli stessi file di modellazione così come per il modello di progettazione. A tal scopo, dividere il contenuto di analisi in pacchetti a cui applicare il nome chiave «analisi». Questo dà l'opportunità di conservare artefatti a livello di analisi nell'ambito degli stessi file di modellazione, in qualità di loro controparti perfezionate a livello della progettazione.

Una questione da tener presente durante l'utilizzo delle trasformazioni di RSx per generare le implementazioni è che queste trasformazioni possono in molti casi accettare elementi a livello di analisi come loro input, salvando alcuni dei passi di perfezionamento manuale di tali elementi in quelli di progettazione. Utilizzando RSx in questo modo, le opzioni 2 o 3 precedentemente descritte sono da preferire. Le trasformazioni standard che generano codice impacchettate come parte di RSx saltano i pacchetti di modello con la parola chiave «analisi».

[6] RUP infatti utilizza l'opzione per la creazione di classi di analisi e di realizzazioni di caso d'uso a livello di analisi nel modello di progettazione per poi trasformarle da lì direttamente in forme della progettazione. Con questo approccio, appena viene scoperto il modello di progettazione è possibile creare pacchetti lungo il percorso in cui vengono conservate alcune delle prospettive di analisi pure. “”“”

Organizzazione di alto livello del modello di analisi

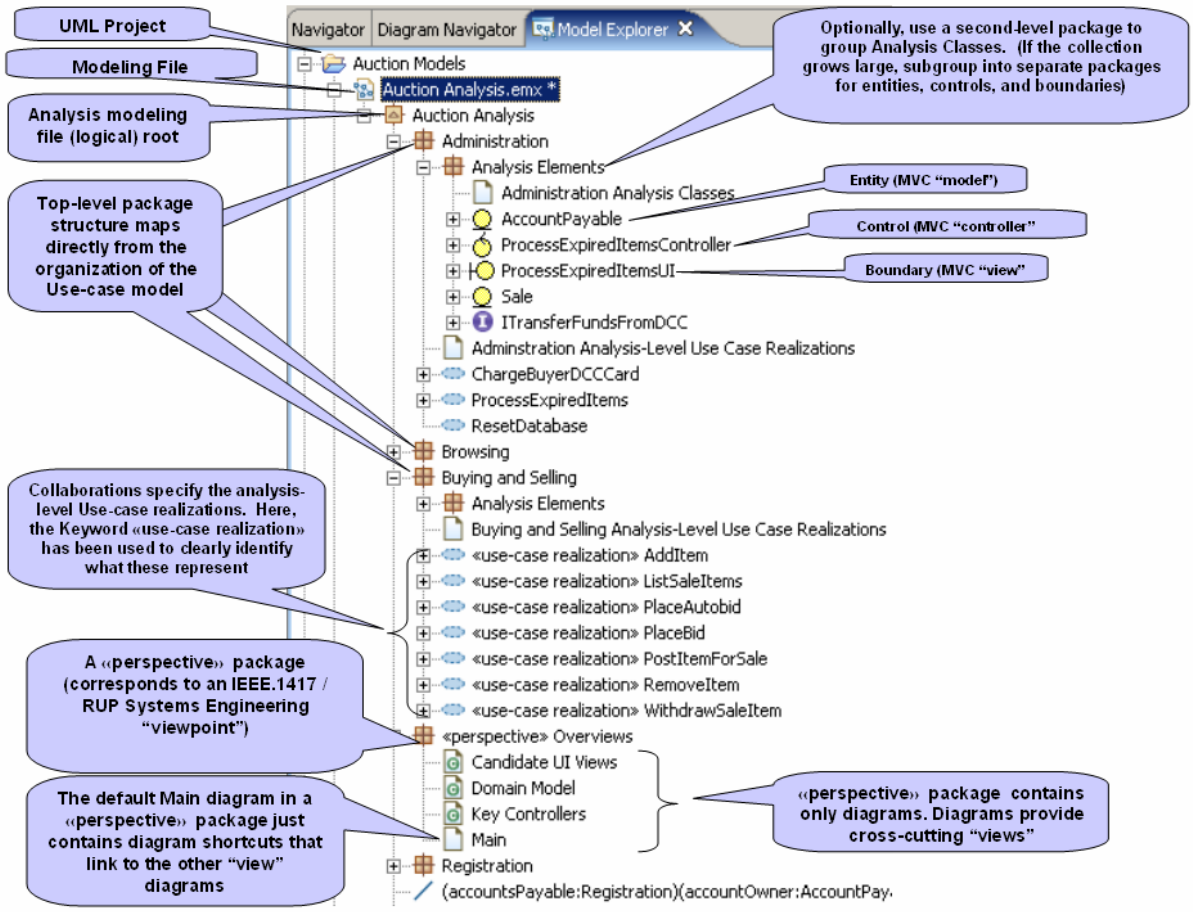


Figura 6-1

Figura 6-1 sono illustrate le seguenti linee guida per la strutturazione di modelli di analisi:

1. Utilizzare i pacchetti di alto livello per stabilire raggruppamenti orientati alla funzionalità per le classi di analisi. Logica: la stessa del modello di caso d'uso.

2. Facoltativamente, all'interno dei pacchetti di alto livello utilizzare pacchetti secondari per raccogliere ed organizzare le classi di analisi.

3. Utilizzare i diagrammi in pacchetti di «prospettiva» per acquisire viste alternative, di alto livello o di esplorazione trasversale degli elementi di analisi. Logica: fornire differenti prospettive per differenti stakeholder mentre vengono mantenuti elementi di semantica del modello organizzati in raggruppamenti orientati alla funzionalità.

Una leggera variazione di quest'approccio viene descritto di seguito in **Figura 6-2** dove è visibile l'utilizzo di un pacchetto di alto livello per dividere le realizzazioni di caso d'uso dalle classi di analisi. All'interno di tale pacchetto è visibile una serie di pacchetti secondari che si adattano a quelli di alto livello. Isolare le realizzazioni di caso d'uso in questo modo consente il refactoring della struttura del pacchetto che contiene le classi di analisi, senza necessariamente influenzare l'organizzazione delle realizzazioni suddette. (In particolare se il modello di analisi si evolve alla progettazione in situ, è come se l'organizzazione del pacchetto per le classi si evolvesse in modo da non potersi più adattare come quando veniva utilizzata per i casi d'uso).

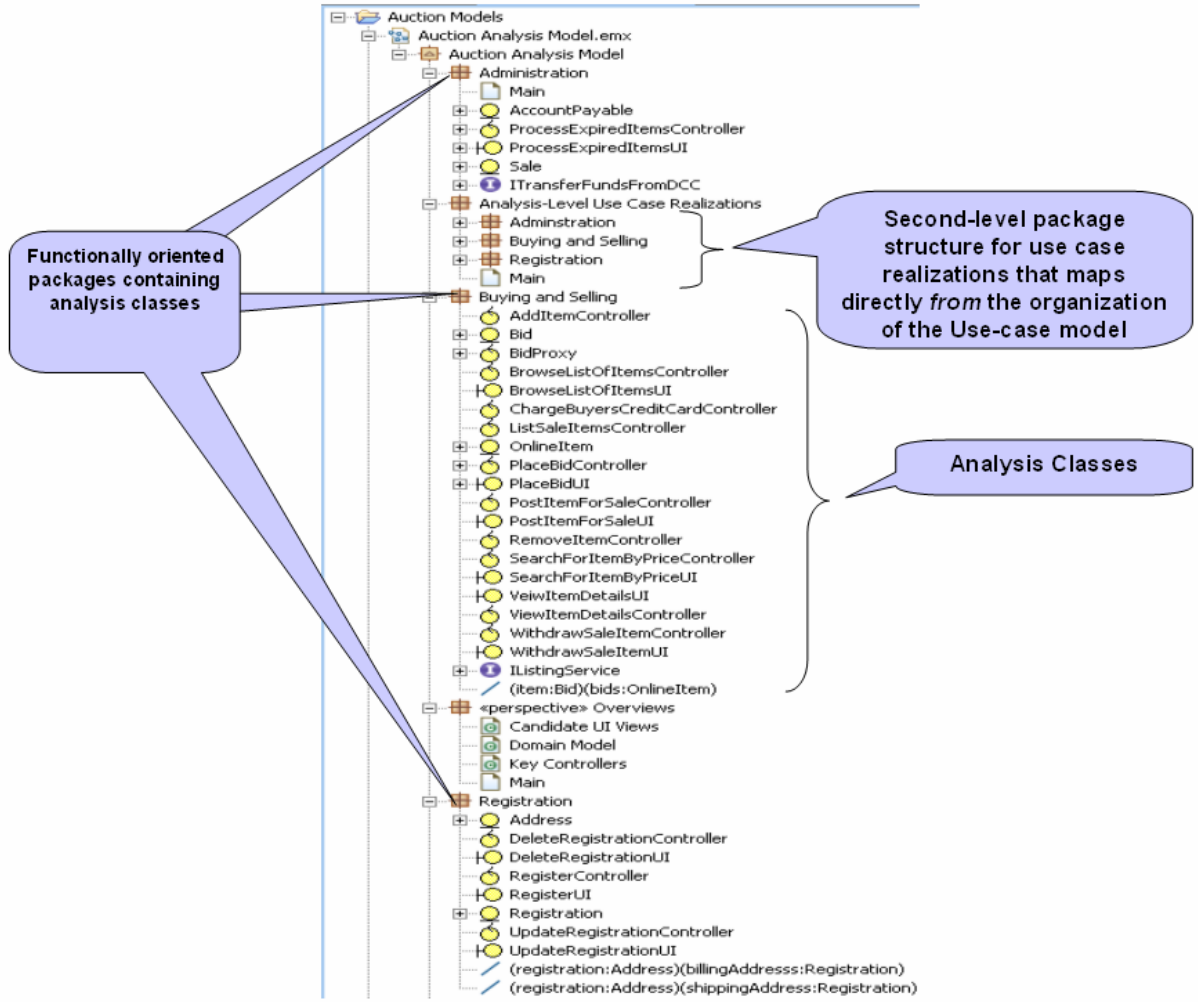


Figura 6-2

In base alla situazione, può essere ragionevole introdurre l'utilizzo di una convenzione di denominazione che anticipa la fusione ed il riutilizzo del contenuto di modello creato da più gruppi indipendenti, anche includendo gruppi in differenti business (partner). Se questo è un problema, considerare l'utilizzo di una convenzione di spazio nomi del dominio internet invertito come descritto di seguito in **Figura 6-3**. Da notare che questo non è probabilmente una grande questione per la modellazione di analisi di per se, ma nel caso in cui venga considerato l'approccio consistente nel lasciare evolvere il modello di analisi nel modello di progettazione sul posto e venga anticipato il riutilizzo o l'integrazione business a livello della progettazione, è possibile preferire una pianificazione anticipata.Un altro potenziale vantaggio nell'adottare quest'approccio: poiché esso si associa bene all'organizzazione di codice generato dalla progettazione/analisi, può semplificare la successiva

configurazione delle trasformazioni che generano codice.

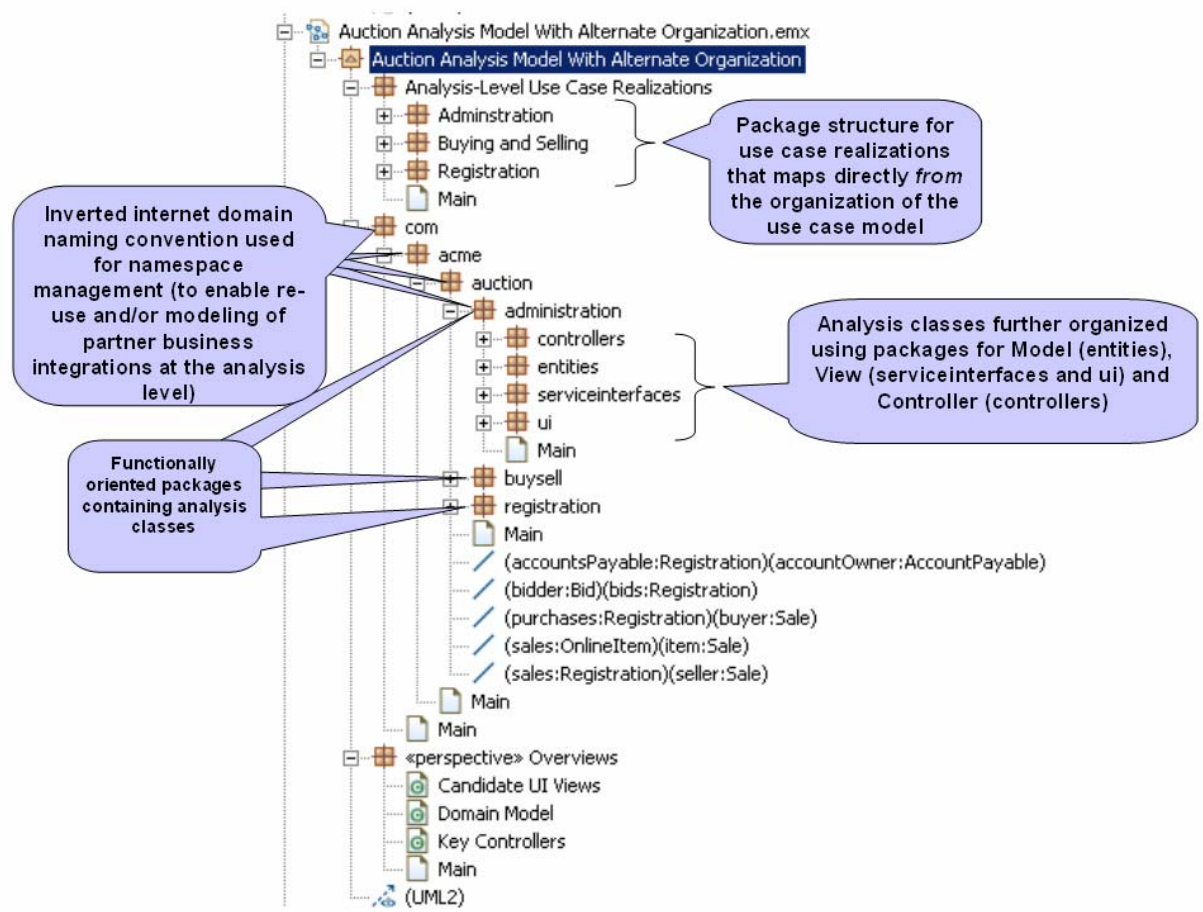


Figura 6-3

Contenuto del modello di analisi

Esistono molti modi per scoprire quali sono le classi di analisi. Uno consiste nell'iniziare a disegnare diagrammi di sequenza che suggeriscono realizzazioni di caso d'uso. Questo porta a scoprire quali sono i cicli vitali necessari e ognuno di questi corrisponde ad una probabile classe di analisi. Una volta scoperte tali classi, è possibile crearle nei pacchetti della realizzazione di caso d'uso del modello di analisi, ma non devono essere lasciate lì. È necessario eseguire il refactoring per spostare le classi di analisi in pacchetti orientati alla funzionalità, così come descritto in precedenza nelle linee guida per l'organizzazione di alto livello del modello di analisi (vedere ***Figura 6-1***)

Un altro utile approccio per scoprire classi di analisi: 'riempire' il modello di analisi con classi in base a queste regole pratiche:

- Per ogni caso d'uso (nel relativo modello) aggiungere una classe di «controllo» al modello di analisi. Queste rappresentano la logica di business associata al caso d'uso.(Più avanti, nella progettazione, si associano a questioni come la gestione di sessione).
- Per ogni relazione attore / caso d'uso (nel modello di caso d'uso) aggiungere una classe di «delimitazione» al modello di analisi. Queste rappresentano le interfacce tra la soluzione ed un attore umano o tra la soluzione ed un sistema esterno. Le classi di «delimitazione» che corrispondono ad

un attore umano vanno probabilmente associate ad uno o più artefatti dell'utente nella fase di progettazione ed implementazione. Quelle che corrispondono ad un sistema esterno possono eventualmente essere associate a qualche tipo di livello di adattatore nella fase di progettazione e implementazione.

- Attraverso un processo come l'analisi di carta CRC o della parola delle descrizioni di caso d'uso, identificare ulteriori classi di «controllo» (verbi) e di «entità» (sostantivi)

Con l'approccio di assegnazione per l'identificazione delle classi di analisi, è possibile sistemare tali classi direttamente in pacchetti orientati al funzionamento, come descritto in precedenza nelle linee guida per l'organizzazione di alto livello del modello di analisi (vedere ***Figura 6-1***)

Tuttavia, più si scopre delle classi di analisi e più si è certi che le modifiche all'organizzazione del pacchetto funzionale originale sono necessarie.

Facoltativo: utilizzare pacchetti di secondo livello all'interno di quelli delle classi di analisi per organizzarne ulteriormente il contenuto (vedere **Figura 6-4**)

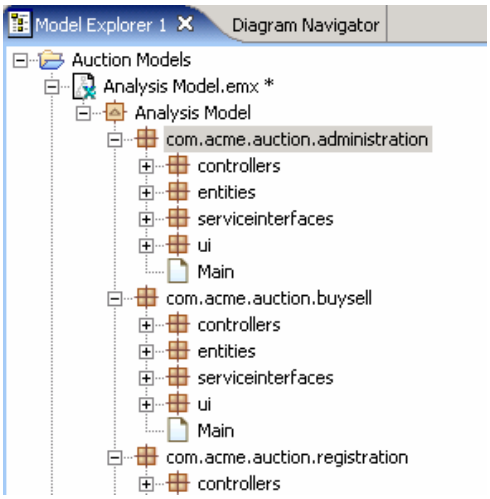


Figura 6-4

Consigliato: il modello di analisi deve contenere realizzazioni di caso d'uso a livello di analisi che ne descrivono le modalità di esecuzione nei termini delle classi di analisi. Ognuna di queste (rappresentate da una collaborazione UML) realizza un caso d'uso nel relativo modello, mantenendone il nome. Consultare **Figura 6-5**. Per ogni flusso di caso d'uso denominato[7] che si presume debba essere modellato come una realizzazione a livello di analisi, aggiungere un diagramma di sequenza (che automaticamente aggiunge un'interazione proprietaria). **Figura 6-6** mostra i tipi di contenuto di semantica da aggiungere al modello una volta creati i diagrammi di sequenza. (È possibile filtrare ogni tipo di elemento UML dalla vista Model Explorer e così nascondere buona parte della 'confusione' descritta in **Figura 6-6**)

[7] Come precedentemente stabilito nel Modello di caso d'uso

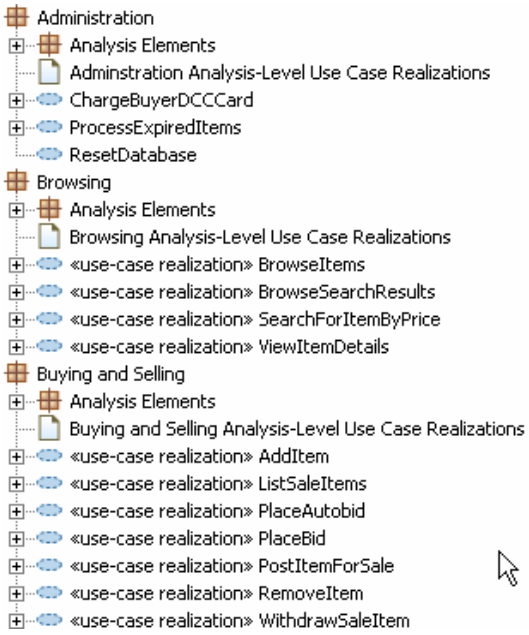


Figura 6-5

Facoltativo: una volta creato il diagramma di sequenza per un flusso di caso d'uso, è possibile selezionare la propria interazione UML nel Model Explorer e aggiungervi un diagramma di comunicazione. Questo diagramma aggiunto viene automaticamente popolato con le istanze di classe di analisi che hanno partecipato in esso.

Consigliato: creare una relazione di dipendenza da ogni realizzazione di caso d'uso (collaborazione UML) ed il corrispondente caso d'uso dal relativo modello (vedere **Figura 6-6**). Poiché è possibile utilizzare funzioni come i Diagrammi di argomento e l'Analisi di tracciabilità per comprendere le relazioni di traccia in un modello, non è necessario mantenere diagrammi permanenti per descriverle ma è consigliata la creazione di relazioni mediante un tipo di diagramma da 'buttare via' come:

- aggiungere un diagramma in formato libero alla collaborazione
- trascinarvi la collaborazione
- trascinarvi il caso d'uso
- tracciare la relazione di dipendenza
- infine (nel Model Explorer) eliminare il diagramma dalla collaborazione

Consigliato: includere un diagramma di partecipanti per ogni realizzazione di caso d'uso, per mostrare le classi di analisi che nefanno parte (ossia, quelle le cui istanze appaiono sui diagrammi d'interazione che descrivono la realizzazione del caso d'uso) e le loro relazioni che supportano la collaborazione descritta nei diagrammi d'interazione. ""Consultare **Figura 6-6**

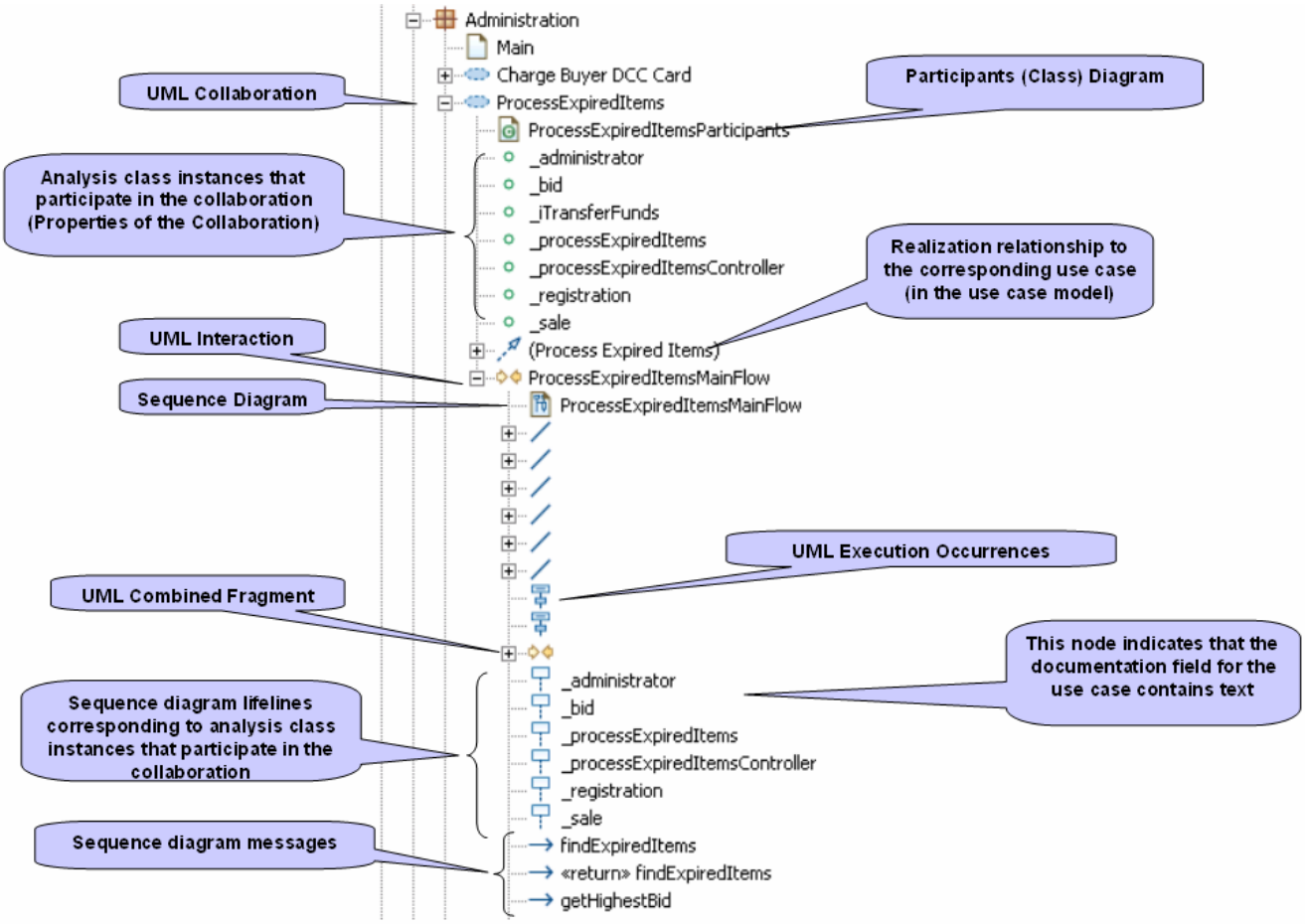
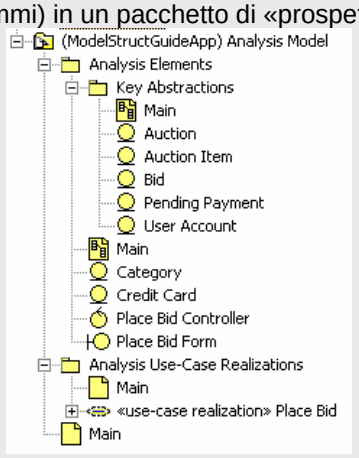


Figura 6-6

XDE/Rose

La struttura comunemente consigliata per il modello di analisi così come mostrato di seguito viene modificata per RSx al fine di porre enfasi su di un'organizzazione di pacchetto orientata alla funzionalità per le classi di analisi. Notare inoltre che l'uso di un pacchetto di astrazioni chiave (che comprometterebbe un'approccio alla creazione di pacchetti orientati alla funzionalità) viene sostituito dall'uso di un diagramma di astrazioni chiave (o diagrammi) in un pacchetto di «prospettiva».



7. Linee guida per l'organizzazione interna del modello di progettazione

Organizzazione di alto livello del modello di progettazione

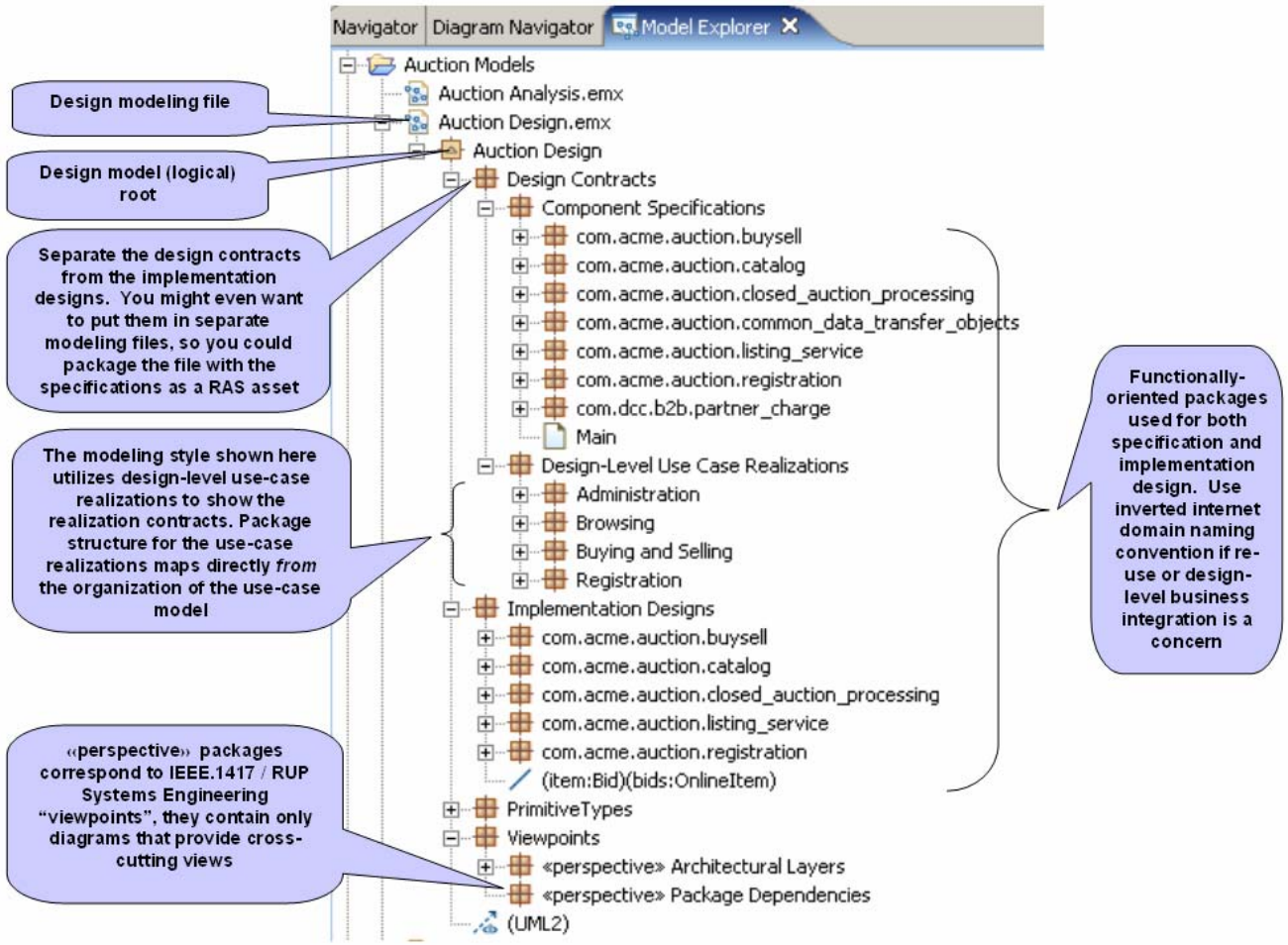


Figura 7-1

Figura 7-1 sono illustrate le seguenti linee guida per la strutturazione dei modelli di progettazione :

1. Separare specifiche dalle progettazioni d'implementazione. L'illustrazione mostra l'uso di contratti di progettazione di alto livello e pacchetti di progettazioni d'implementazione per realizzarli. “”“”

2. Utilizzare pacchetti di basso livello per stabilire raggruppamenti orientati alla funzionalità. È possibile ad esempio iniziare con l'organizzazione utilizzata durante l'analisi e lasciarla evolvere mentre vengono prese decisioni su come associare le classi di analisi alle attuali classi di progettazione, componenti e servizi. (Ogni schema organizzativo iniziale si modifica con ogni probabilità durante la progettazione -- vedere l'ulteriore discussione di seguito).

   Può essere opportuno a questo punto spendere qualche parola per i sistemi secondari. Nelle versioni di UML precedenti alla 2 un sistema secondario è un tipo di pacchetto specializzato. In UML 2 è un tipo specializzato di componente, che può contenere molti pacchetti. Dunque in UML 2 i componenti

di «sistema secondario» sono valide alternative organizzative/di spazio nomi ai pacchetti, anche se UML2 resta vago circa un loro appropriato utilizzo rispetto al pacchetto. Suggerimento: utilizzare i pacchetti a livelli di granularità come i sistemi secondari di progettazione di una particolare applicazione e riservarne alcuni in rappresentanza di intere applicazioni (come CRM o SCM) nelle viste generali aziendali dell'architettura.

> ***XDE/Rose***
> *Durante la composizione del presente documento era previsto che i tool d'importazione del modello XDE e Rose offrissero l'opzione di associazione dei sistemi secondari di UML 1.x a quelli di UML2 o a pacchetti con la parola chiave «sistema secondario»*

3. È probabile che l'organizzazione degli elementi di progettazione evolva diversamente dalla modalità di organizzazione dei casi d'uso del sistema (nel modello di caso d'uso e forse in quello di analisi, se ne viene mantenuto uno separato). Utilizzare i pacchetti per dividere ulteriormente i contratti di progettazione in specifiche dei suoi elementi (contratti d'uso) e in realizzazioni di caso d'uso al suo livello (contratti di realizzazione)e mantenere una struttura secondaria del pacchetto per quelle realizzazioni di caso d'uso che continuano a rispecchiare l'organizzazione degli stessi casi d'uso.

4. *Considerare l'utilizzo di livelli strutturali come base per lo schema organizzativo di secondo livello per gli elementi che compongono le progettazioni d'implementazione e di specifiche delle aree funzionali (e vedere l'ulteriore discussione di seguito)*

5. All'interno di componenti e pacchetti che raggruppano elementi di modello di semantica, posizionare i diagrammi che forniscono specifiche viste di quel raggruppamento. Questa linea guida riferisce se il raggruppamento si basa su sottoinsiemi orientati alla funzionalità del dominio business, su un livello strutturale o meno. Fare in modo che il diagramma ‘predefinito’ abbia lo stesso nome del pacchetto o del contenuto e comporlo per mostrare una panoramica dei contenuti del pacchetto.Questo mantiene i diagrammi vicino a ciò che descrivono, semplificando la navigazione e la comprensione del modello.

6. Può essere preferibile introdurre l'uso di spazio nomi del dominio Internet invertito nel modello di progettazione. Logica:
   - Fondamentalmente le stesse ragioni relative alle implementazioni di linguaggio specifico:
     a. gli scenari che comprendono il lavoro d'integrazione dove sono coinvolte multiple applicazioni guidate dal modello (specie con le compagnie partner)
     b. riutilizzo di scenari
   - Questo probabilmente semplifica la successiva configurazione delle trasformazioni nell'implementazione (posizione da fonte a destinazione e mappatura nome).

7. *considerare l'utilizzo di noi di pacchetto validi nelle piattaforme d'implementazione di destinazione, per evitare l'onere e la potenziale confusione della mappatura dello spazio nomi.* (Il più delle volte con ciò si consiglia di non utilizzare spazi o punteggiatura diversi dai caratteri di sottolineatura nei nomi.)“”

8. Utilizzare la lettera minuscola per i nomi di pacchetto, al fine di distinguerli meglio dai nomi di classe in esso contenuta.

9. *Considerare l'utilizzo di nomi differenti per le interfacce, i componenti o le classi che li realizzano.*Utilizzare sia ILoan che Loan o Loan e LoanImpl per i nomi d'implementazione e d'interfaccia. Non è necessario nel modello, ma spesso si rivela una buona scelta in un codice generato, perciò questa è un'altra area dove è possibile liberarsi di parte del successivo lavoro di configurazione della trasformazione.

10. Nel seguente scenario, tutto il contenuto a livello di analisi che si presume non produca codice deve essere diviso all'interno di pacchetti stereotipati come «analisi»[8].
    - A) si è scelto di saltare l'utilizzo di un modello di analisi separato, di popolare quello di progettazione con contenuti a livello di analisi e di mantenere tali contenuti ad un livello analitico di astrazione, mentre ne vengono creati anche a livello di progettazione nello stesso modello e
    - B) le trasformazioni da modello a codice vengono guidate a partire dal modello di progettazione EIT

11. Utilizzare i diagrammi in pacchetti di «prospettiva» per acquisire viste di alto livello e di esplorazione trasversale degli elementi di progettazione. Logica: fornire viste di esplorazione trasversale, di contenuto 'architettonicamente significativo' e viste che si rivolgono a differenti tipi di stakeholder mentre mantengono gli elementi della semantica del modello organizzati in raggruppamenti orientati alla funzionalità.

È importante riconoscere che le strutture di creazione di pacchetti dei modelli di progettazione evolvono nel tempo. Ultimamente l'organizzazione deve corrispondere al modo in cui l'architettura viene strutturata in componenti e servizi. Questo approccio all'organizzazione finale della progettazione deve poi in genere offrire il miglior potenziale di risorse riutilizzabili per la creazione di pacchetti e la mappatura più semplice dalla progettazione alla serie di progetti e cartelle che contengono gli artefatti d'implementazione (codici, metadati, documentazione) generati da essa.

Tuttavia l'organizzazione iniziale deve corrispondere più o meno direttamente all'approccio utilizzato per il modello di caso d'uso poi rivisto in sede di analisi[9]. Infatti (come descritto nella precedente sezione Linee guida per l'organizzazione interna del modello di analisi) è possibile decidere di far evolvere il modello di analisi in progettazione sul posto. “”In altre parole, l'organizzazione iniziale della progettazione tende a raggruppare serie accoppiate coesive e libere di questioni business ed a isolare elementi di esplorazione trasversale o riutilizzabili. Quest'approccio all'organizzazione iniziale si dimostra efficace perché

- Nella speranza di utilizzare le trasformazioni che generano contenuti di modello di progettazione da quelli del modello di caso d'uso o di analisi, le mappature da pacchetto a destinazione di origine saranno semplici e dirette
- Un approccio organizzativo iniziale basato su una coesione funzionale ed il libero accoppiamento dei pacchetti è chiaramente la migliore possibilità per la mappatura all'organizzazione finale orientata ai componenti, che può ridurre la quantità di refactoring da eseguire come parte del processo di progettazione.
- Il libero accoppiamento di pacchetti possiede il potenziale per migliorare i flussi di lavoro del team e semplificare il riutilizzo nei casi in cui la progettazione viene scomposta in più file di modellazione

Approcci alternativi sono sicuramente possibili ed in alcuni casi consigliabili come organizzazione finale:

- Se ci si indirizza verso applicazioni Web basate su J2EE inclusi EJB, l'organizzazione della progettazione può anticipare le convenzioni di RSA e Rational Application Developer relative ai progetti J2EE.[10] In particolare è possibile scegliere di definire pacchetti di progettazione di alto livello che corrispondono a livelli strutturali (presentazione e business, con quest'ultimo posto ad un livello secondario nella sessione e nel dominio). Ovviamente questo non è un approccio per

[8] Tali pacchetti devono essere saltati dalle trasformazioni.

[9] La creazione di pacchetti delle classi di analisi viene spesso modificata in modo significativo una volta scoperta, al fine di supportare meglio il riutilizzo e i requisiti funzionali non previsti.

[10] Libera discussione: Un progetto aziendale per ogni sistema, applicazione o grande sottosistema e per ognuno di questi progetti un progetto Web per il livello di presentazione e più progetti EJB, che corrispondono in genere ai sottosistemi minori o componenti, alcuni dei quali vengono utilizzati per il livello di sessione (EJB di sessione) e di dominio (EJB d'entità) per ogni componente o sottosistema. Vedere la sezione 9 del presente documento per ulteriori informazioni.

piattaforma generico e dunque è auspicabile solo nella certezza che non verrà utilizzato per piattaforme diverse da J2EE.

- Più in generale, si verifica più spesso il caso in cui nella creazione di applicazioni di livello-n lo sviluppatore esperto e la divisione del lavoro corrispondono ai livelli di presentazione e di business perciò, di nuovo, è possibile scegliere di utilizzare pacchetti di alto livello che corrispondono a questi livelli strutturali. Ma bisogna fare attenzione all'organizzazione delle classi previste per il supporto di particolari funzioni business a sostegno di una particolare architettura.Sono entrambe più difficili da modificare.
- Se c'è motivo di utilizzare un approccio organizzativo orientato al sottosistema/servizio/non componente, è ancora possibile associare l'organizzazione della progettazione ad una serie di progetti di destinazione e cartelle investendo uno sforzo aggiuntivo nella configurazione delle trasformazioni di generazione di codice.Può essere utilizzato un tipo speciale di modello di accompagnamento a cui si fa riferimento come 'modello di mappatura' per definire associazioni particolarmente complesse.

Contenuto del modello di progettazione

Non esistono regole forti e rapide per ciò che risiede nel modello di progettazione, ma i seguenti suggerimenti possono rivelarsi utili.

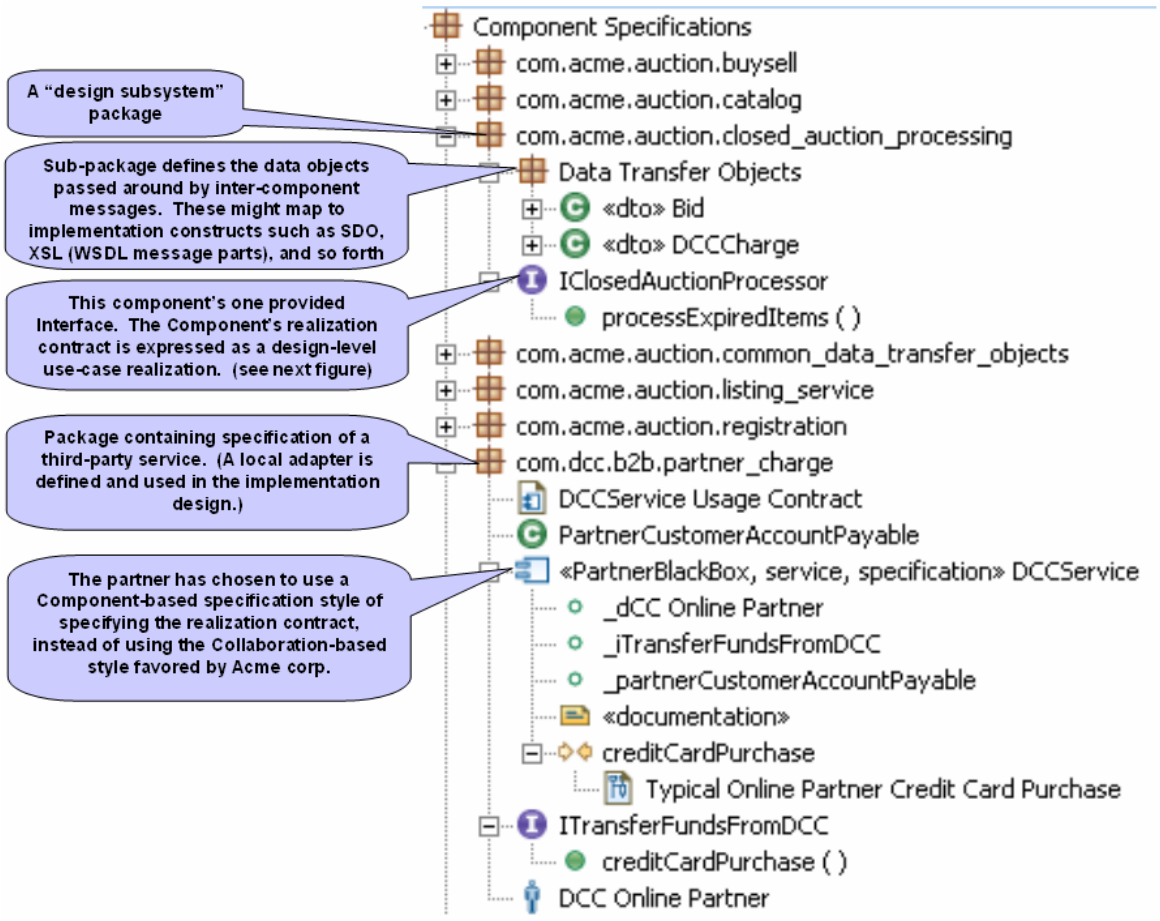


Figura 7-2

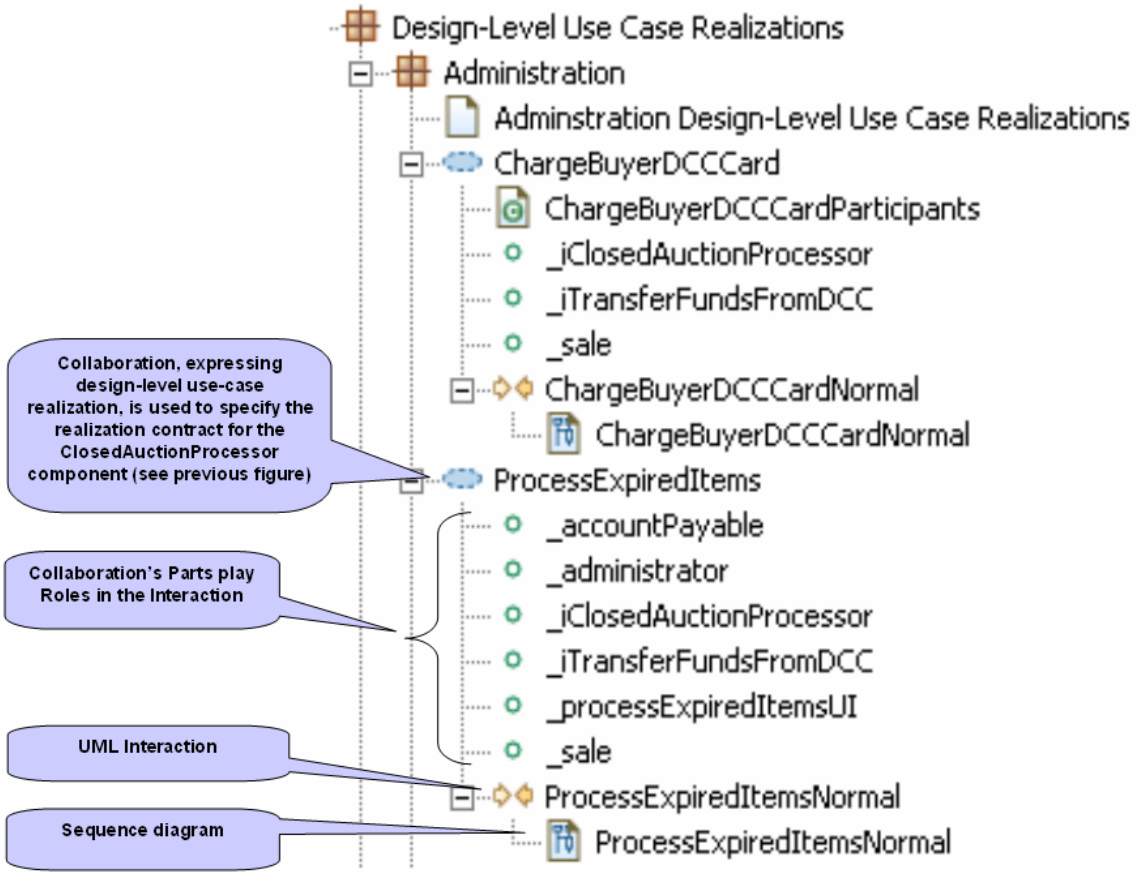


Figura 7-3

Figura 7-2 e **Figura 7-3** aderire alla struttura organizzativa descritta in **Figura 7-1** e descrivere come specificare i contratti di progettazione.

- Il contratto d'uso per un componente "ClosedAuctionProcessor" viene espresso come una singola interfaccia[11] (**Figura 7-2**). Il contratto di realizzazione corrispondente viene specificato da una singola realizzazione di caso d'uso a livello di progettazione espressa come una collaborazione[12] (**Figura 7-3**) Da notare che mentre le realizzazioni di caso d'uso a livello di analisi mostrano collaborazioni tra classi di analisi, quelle a livello di progettazioni mostrano collaborazioni tra elementi di progettazione meno astratti.[13]. Nel caso in cui sia preferibile che il sottoinsieme di specifica di un modello di progettazione sia impacchettato indipendentemente da quello della progettazione d'implementazione, allora è importante che le realizzazioni di caso d'uso a livello di progettazione utilizzino solo elementi di specifica o di analisi–mai elementi di progettazione d'implementazione -- nei propri ruoli.

[11] I componenti possono ovviamente avere più interfacce fornite, ma questo esempio ne possiede solo una

[12] Altri componenti possono partecipare a più casi d'uso del sistema, in modo che i loro contratti di realizzazione possono risiedere in più realizzazioni di caso d'uso. In tali casi è possibile anche includere, nello stesso pacchetto con l'interfaccia del componente, un diagramma chiamato nome componente ove utilizzato su cui è possibile inserire link ai diversi diagrammi che costituiscono le realizzazioni per quei casi d'uso. "{}"

[13] Altra probabile differenza: alcuni dei diagrammi che partecipano alle realizzazioni a livello di progettazione possono essere diagrammi che descrivono il collegamento tra componenti, piuttosto che (o in aggiunta a) diagrammi di classe partecipanti, come suggerito per le realizzazioni di caso d'uso a livello di analisi. ""

- I contratti d'uso e di realizzazione per “DCCService” di terze parti sono tenuti insieme in un pacchetto[14]. Di nuovo, il contratto d'uso consiste di una singola interfaccia, ma in questo caso il contratto di realizzazione viene espresso mediante un componente di «specifica» (**Figura 7-2**). (Altrimenti la specifica del contratto di reqalizzazione è quasi la stessa, espressa mediante i comportamenti – in questo caso un'interazione chiamata “creditCardPurchase”).Un altro esempio che utilizza i componenti invece delle collaborazioni viene mostrato in **Figura 7-4**

- Le operazioni sono definite in interfacce che possono essere realizzate da componenti di «specifica» (se utilizzati) o da classificatori nel progetto d'implementazione che inseriscono le interfacce.

- La specifica di oggetti di trasferimento dati (che fungono da tipi di parametri delle operazioni fornite e possono essere associate a costruzioni d'implementazione come uno schema XML o SDO) può anche essere inclusa come parte del contratto d'uso. Per i componenti non progettati per essere distribuibili, è possibile scegliere o meno di indicare gli oggetti di trasferimento dati come specifiche dei tipi usati come parametri dell'operazione. Per i servizi distribuibili (come i servizi Web) è obbligatorio che le loro operazioni non facciano riferimento ad oggetti in uno spazio d'indirizzo locale, perciò devono essere utilizzati i DTO.

> ***XDE/Rose***
> Nelle versioni precedenti di UML, la guida per le realizzazioni di caso d'uso prevedeva di usare un'istanza di collaborazione per caso d'uso ed un diagramma di sequenza e d'interazione per ogni flusso significativo della realizzazione.
>
> In RSx bisogna essere capaci di utilizzare solo un'interazione ed un diagramma perché i diagrammi di sequenza di UML adesso supportano le annotazioni per percorsi di esecuzione alternativi.
>
> Inoltre, in UML 2 non esiste più una ‘Istanza di collaborazione’ . Esiste invece un ‘uso della collaborazione’, che richiede una collaborazione come proprio tipo. Perciò, utilizzare in RSx le collaborazioni per rappresentare le realizzazioni di caso d'uso)

- [14] Da notare che la situazione ipotetica è che la compagnia DCC abbia fornito la Acme con specifiche UML, che Acme ha poi incorporato nel proprio modello di progettazione. È il tipo di scenario in cui l'utilizzo di spazi di dominio internet invertiti può rivelarsi utile.

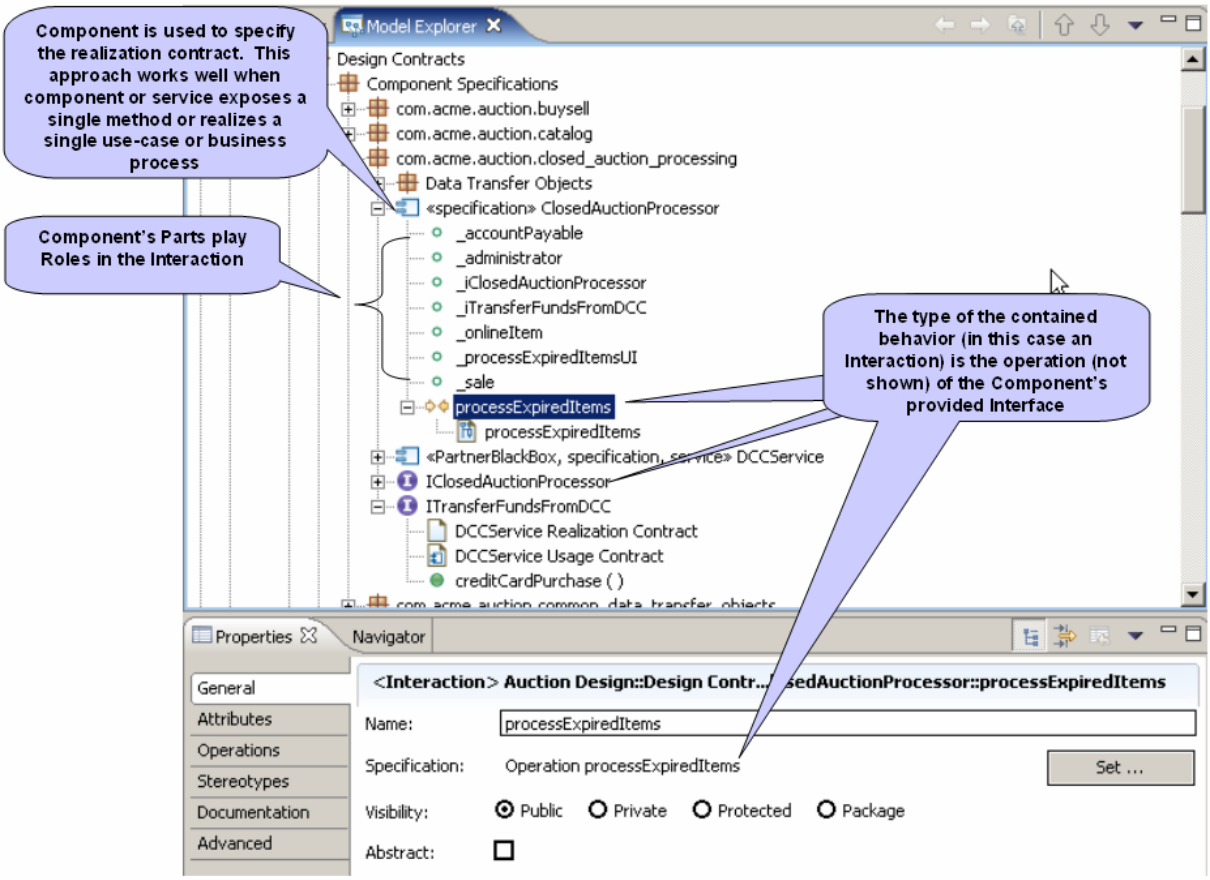


Figura 7-4

- Un possibile approccio alla specifica delle progettazioni d'implementazione viene mostrato in **Figura 7-5** la struttura d'implementazione viene definita mediante semplici classi che contengono le operazioni. Questo approccio è tipico dei modelli di progettazione creati con UML 1.x Un secondo possibile approccio che può risultare più adatto agli obiettivi di UML2 viene mostrato in **Figura 7-6**. Qui, invece delle classi vengono usati i componenti, che non posseggono le operazioni ma i comportamenti (in questo caso un'interazione).

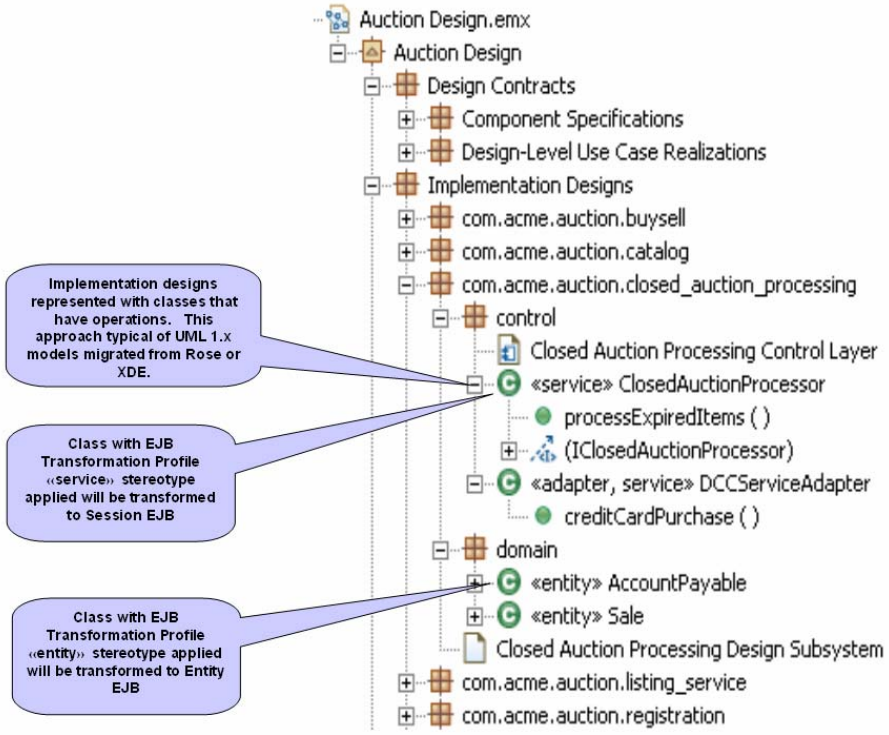


Figura 7-5

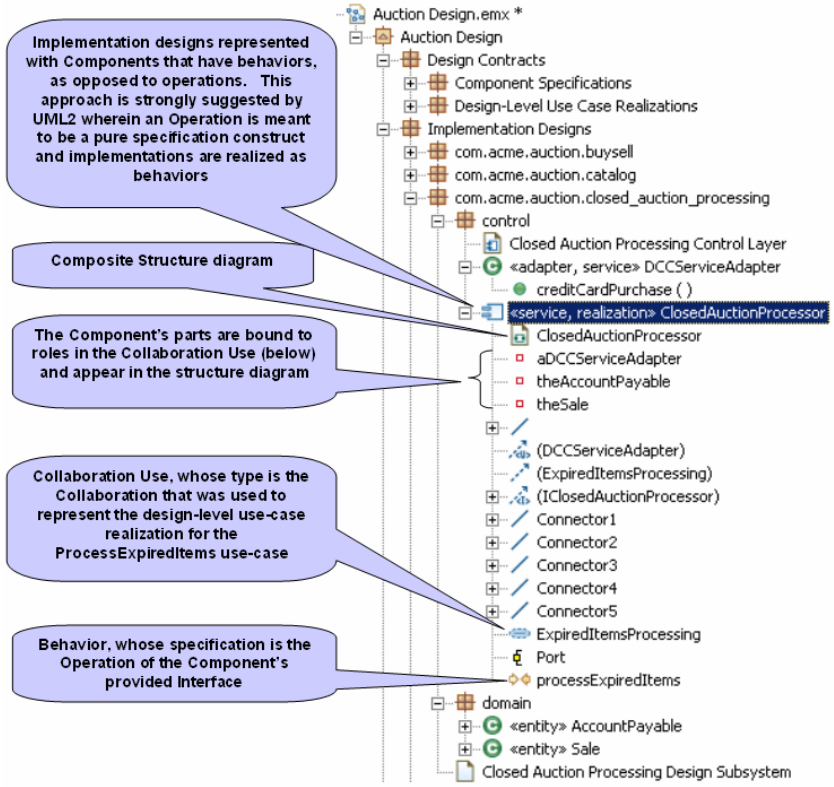


Figura 7-6

8. Linee guida per l'organizzazione interna del modello di panoramica dell'implementazione

> ***XDE/Rose***
> Nelle linee guida della struttura del modello XDE, viene consigliato un modello di panoramica d'implementazione come unità per fornire una panoramica a livello dei sottosistemi dell'implementazione.I dettagli di ogni sottosistema sono poi specificati nel modello di codice del progetto che lo ha implementato.

Francamente, non è necessario utilizzare un modello di panoramica d'implementazione in RSx.Se vengono seguite le linee guida organizzative del modello di progettazione, allora l'organizzazione (finale) di tale modello deve formarsi attorno ai componenti (incluso il «sottosistema» heftier e una varietà di «servizi» maggiormente distribuibili). Poi, attraverso le trasformazioni, i pacchetti della progettazione possono essere associati ai progetti.Ad esempio nel caso di un'implementazione J2EE vengono associati ai vari progetti Java, EJB, Web, applicazione J2EE ed altri in cui viene sviluppata l'implementazione. (e questi progetti rappresentano infatti il modello d'implementazione per la soluzione, come rilevato nella sezione del presente documento Concetti base e Terminologia.)

Pensando a tutto ciò in termini di "basso verso l'alto", bisogna considerare in che modo vada organizzata l'implementazione in termini di progetti e cartelle e convertire il tutto nell'organizzazione del modello di progettazione in modo che le mappature della trasformazione finiscano con l'essere più semplici.In una prospettiva "dall'alto verso il basso" l'organizzazione del modello di progettazione, evoluto nel corso dell'impegno speso per essa, deve determinare la serie di modelli d'implementazione (progetti) necessari. In un modo o nell'altro l'organizzazione finale del modello deve naturalmente andare incontro alle necessità di una panoramica di progetti primari e secondari, così, ad esempio, un diagramma che descrive i pacchetti nel modello deve essere equivalente ad una panoramica di progetti e sotto-cartelle.

Comunque, può ancora essere preferibile abbozzare la struttura di un progetto in una fase iniziale o vedere una descrizione di strutture di un progetto più visivamente esplicito – ad esempio, quello dove gli artefatti che rappresentano progetti e cartelle sono definiti in modo chiave come «progetto» e «cartella» o anche «progetto EJB» e «progetto Web». Un'altra considerazione è che descrivere artefatti d'implementazione più dettagliati (JAR ad esempio) può essere inappropriato per il modello di progettazione (che nella teoria dell'operazione di Rational Software Architect viene considerato generico). Ma tali artefatti sono perfettamente accettabili nel caso di inclusione in un modello di panoramica d'implementazione. Dunque esistono ragioni per cui preferire l'utilizzo di un modello di panoramica d'implementazione. **Figura 8-1** Di seguito viene riportato un esempio di tale modello

Un'ultima considerazione sul modello di panoramica è che può essere un buon posto per acquisire diagrammi informali dei diversi aspetti di una soluzione. **Figura 8-2** sotto viene mostrato un diagramma informale di alto concetto sul sistema di asta su cui sono stati impostati tutti gli esempi del presente documento.

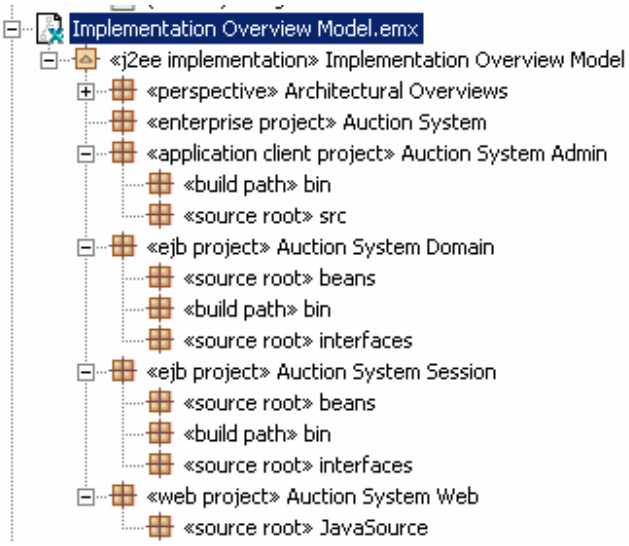


Figura 8-1

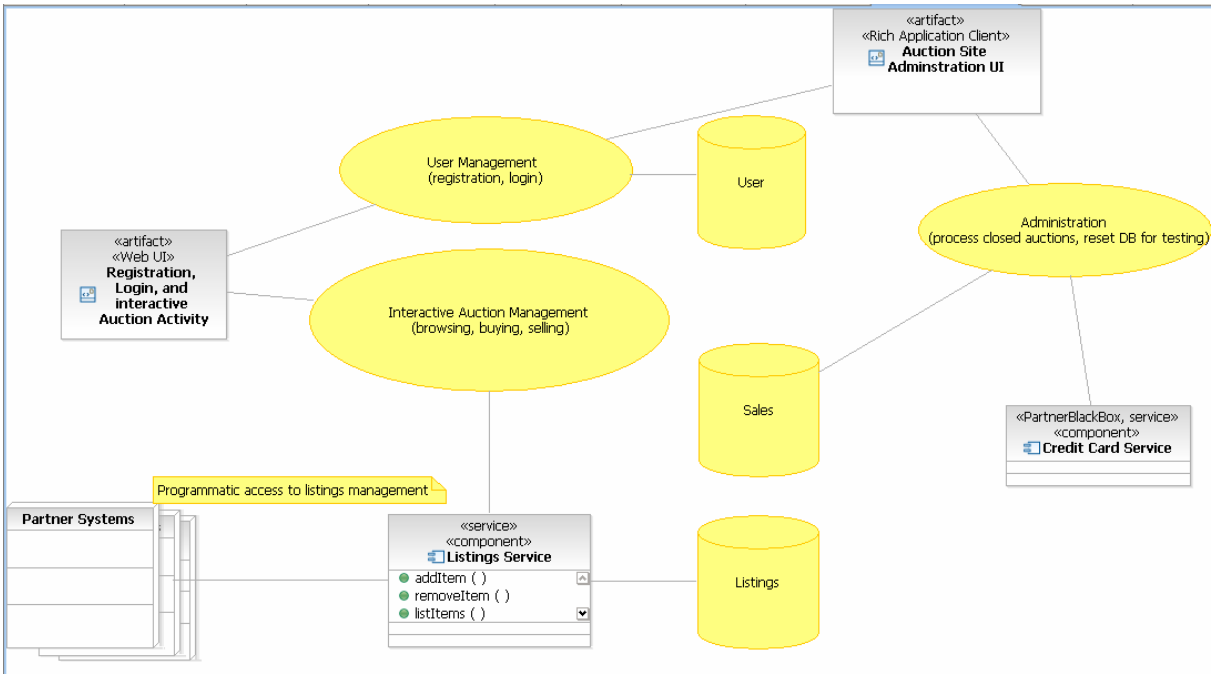


Figura 8-2

9. Linee guida per l'organizzazione interna del modello di distribuzione

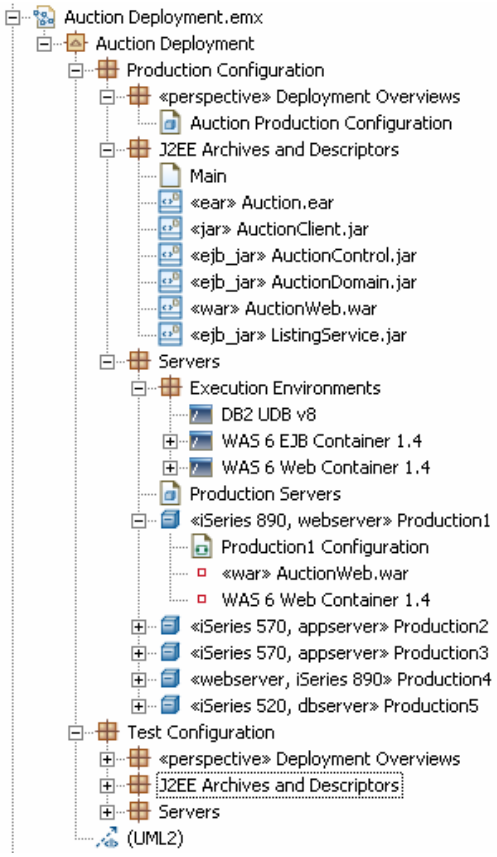


Figura 9-1

Probabilmente non è necessario dire molto sul modello di distribuzione rispetto a tutti gli altri modelli qui trattati.Scelte di contenuto e di organizzazione della modellazione della distribuzione non comportano quasi alcuna implicazione a livello di flusso successivo, dunqe limitarsi a fare quanto ritenuto più sensato.Pensare– soltanto – che una possibile strategia ed un pò di contenuti rappresentativi vengono descritti sopra in **Figura 9-1**. Solo poche cose da notare in quest'esempio:

1. Le specifiche delle configurazioni di produzione sono state separate da quelle di test

2. Le panoramiche (come quelle di cluster, centri dati o aziendali) sono mantenute in pacchetti di «prospettiva»

3. È stato scelto un approccio leggero con riferimento alla specializzazine e la classificazione di nodi e artefatti: una combinazione d'impacchettamento e di utilizzo di parole chiave. Un approccio più sofisticato deve sviluppare un profilo UML specializzato che definisce stereotipi specializzati e proprietà appropriate alla descrizione e alla documentazione dei tipi di risorse utilizzate in un ambiente.

10. Utilizzare un file di modellazione per rappresentare il documento di architettura del software

Dati i tool per l'organizzazione dei modelli, come link di diagrammi e supporto per file di modello multipli con riferimenti al di sopra di esso, diventa quasi una questione superficiale creare un modello di rappresentanza del documento di architettura software di RUP e le 4+1 viste dell'architettura.“”

Nella forma più semplice, bisogna fare qualcosa nelle linee di **Figura 10-1**. Creare un file di modellazione e popolarlo con una serie semplice di pacchetti corrispondenti alle viste 4+1. (L'esempio viene mostrato senza un pacchetto per la vista processo, poiché il sistema qui non mostra molto in termini di concorrenza.)

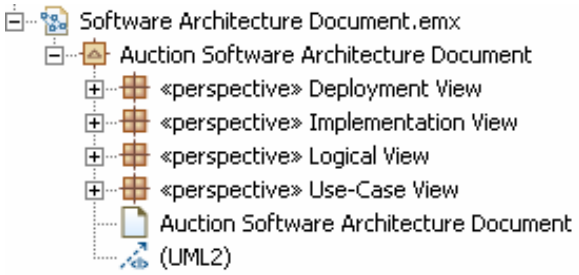


Figura 10-1

Poi, nel caso, comporre il diagramma predefinito sulle linee suggerite in **Figura 10-2** . È anche possibile aggiungere note ulteriori o testo al diagramma

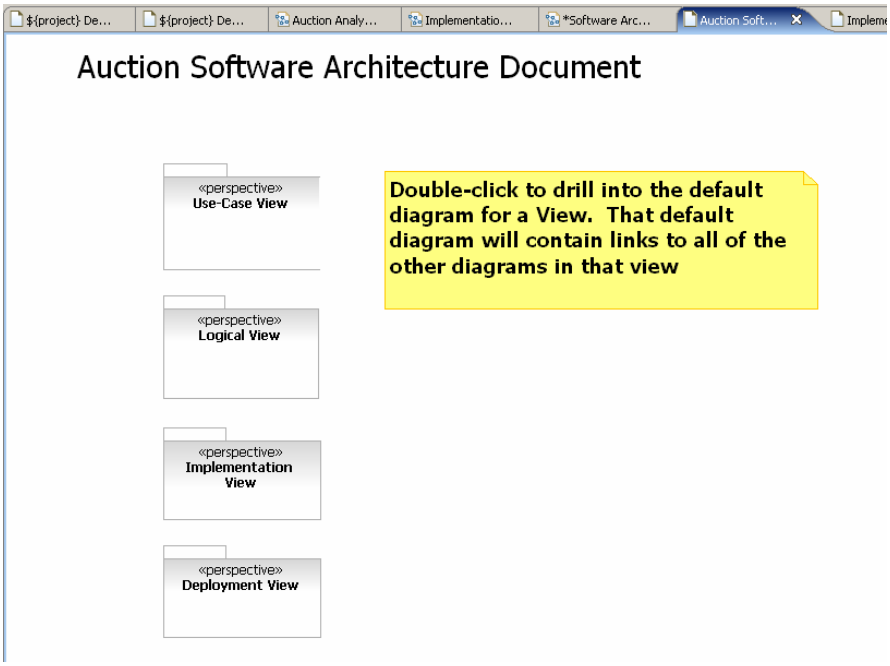


Figura 10-2

Creare poi diagrammi nel file di modellazione del documento di architettura del software, utilizzando i seguenti approcci:

- Creare diagrammi composti mediante elementi di semantica UML da altri file di modellazione che descrivono nuove viste non trovate in precedenza ma che risultano necessarie come parte del documento di architettura

- Creare diagrammi composti da forme geometriche e/o elementi UML ad hoc che risiedono nel file di modellazione del documento di architettura del software.“” (Tali elementi UML servono solo per scopi di documentazione o chiarimento e non devono avere significato di semantica per l'attuale implementazione della soluzione descritta)

- Creare diagrammi che contengono semplicemente link a quelli esistenti negli altri file. (Questa tecnica funziona bene se il file di modellazione del documento di architettura del software deve essere distribuito insieme agli altri file ad uso dei lettori. Se il documento va pubblicato sul Web, seguire invece uno degli altri approcci)

11. Considerazioni sullo sviluppo del team e sulla gestione del modello

Questa sezione introduce alcune delle considerazioni su quando e perché scegliere di suddividere un modello in più file di modellazione. Un trattamento più comprensivo di queste questioni può essere trovata nella guida online di RSx.Si presume che il lettore abbia familiarità con i concetti di sviluppo parallelo e la nozione di modifiche di fusione fatte in parallelo su più copie di un artefatto.

Per prima cosa, una veloce analisi: nella sezione Concetti base e Terminologia sono stati discussi i diversi tipi di modello, come il caso d'uso, l'analisi e la progettazione, così come riconosciuti in RUP.“”Sono stati mostrati esempi per illustrare ciò in RSx.…

- Nella costruzione di multiple applicazioni è possibile avere multipli modelli di ogni tipo (ad esempio multipli modelli di caso d'uso, di analisi e così via)

- un modello (in senso logico) può essere permanente come uno o più file di modellazione, ad esempio quello di progettazione per l'applicazione “‘X’”) può essere permanente come un singolo file di modellazione o una raccolta di più file.

Modelli di partizionamento

Le tecniche di suddivisione dei modelli in più file di modellazione sono presenti nella guida online di RSx e non qui.Qui l'interesse è sul quando e perché eseguire la suddivisione. Le circostanze per cui scegliere di mantenere un particolare modello in più file di modellazione sono due:

1. Il modello è cresciuto in maniera imprevista o la sua struttura d'impacchettamento è aumentata di profondità [15]

2. Si iniziano a sperimentare troppe distribuzioni concorrenti di modifiche ad un file di modellazione, con il risultato che è necessario eseguire fusioni con > 2 fattori che contribuiscono alla modifica[16]. (Se l'istruzione non è chiara, continuare a leggere.)

Modellazione nei team

Quando la politica di gestione della configurazione abilita lo sviluppo parallelo dei modelli[17], modifiche non coordinate vengono eseguite sul file (e sul modello logico o il sottoinsieme del modello da esso rappresentato). Ad un certo punto le modifiche vanno unite.Se viene fuori che alcune modifiche vanno in conflitto, la fusione viene considerata non superficiale perché qualcuno deve prendere una decisione in merito a quale delle modifiche in conflitto deve prevalere.“”“”(Una fusione ‘superficiale’ è quella in cui le modifiche non coordinate non sono in conflitto e il motore della fusione del modello può eseguire le fusioni senza l'intervento umano.)
Le fusioni non superficiali possono essere difficili da gestire.Come minimizzarle?

[15] Il partizionamento è necessario sopratutto quando i file diventano troppo grandi per le macchine in uso nella comunità utenti. Ad esempio, un modello che cresce fino a 30MB su disco risulta difficile da utilizzare quotidianamente su una macchina da 1GB di RAM. In questa situazione, è preferibile suddividere il modello con l'obiettivo di mantenere il valore da 5 a 10MB dei modelli nella RAM ogni volta. Un'alternativa consiste nell'aumentare la RAM (una soluzione relativamente economica, ma che funziona molto bene -- una macchina con 2GB di RAM senza file di swap è più veloce su quasi tutte le operazioni di Eclipse, consentendo così una nuova esecuzione migliore per modelli molto grandi.)

[16] il tool di fusione del modello RSx supporta fusioni fino ad un massimo di 3 contribuenti: 2 di ‘modifica’ e uno di ‘base’ (antenato comune). Con più di 2 fattori che contribuiscono alla ‘modifica’ da gestire, le fusioni iniziano la cascata e da quel punto uno dei due fattori è il risultato di precedenti fusioni completate nella sessione.

[17] Esempi di politiche di sviluppo parallelo:“”

- un file di modellazione può essere esaminato per un accesso non esclusivo
- un file di modellazione è sotto il lavoro in parallelo di flussi di sviluppo di più esperti)

Esistono di base due armi per evitare il conflitto e le risultanti fusioni non superficiali.Una è una forte architettura.“”L'altra è una forte proprietà.“” Vanno a braccetto: una forte architettura abilita una forte proprietà.

Arma #1: forte architettura

“Una forte architettura in questo contesto fa riferimento primariamente alla decomposizione.” I principi di decomposizione architettonica applicati qui sono gli stessi che guidano lo sviluppo orientato all'oggetto, la progettazione in base al componente e le architetture orientate al servizio:

- Cercare la massima separazione delle funzioni business
- Raggruppare tutto ciò che va mantenuto unitoIsolare i raggruppamenti gli uni dagli altri
- Se la decomposizione risultante ha una grande quantità di 'granuli', allora in base al modello del personale (ricordare che forte architettura e forte proprietà vanno a braccetto) può essere preferibile avviarne il raggruppamento in aggregati ad alta affinità (che in termini di modellazione significa pacchetti UML)
- Esiste sempre qualcosa che deve essere toccata da molte (o tutte in alcuni casi) unità di decomposizione.Raggrupparlo in un pacchetto ‘comune’ e pianificare ogni iterazione di sviluppo in modo che includa una piccola cascata all'avvio dell'iterazione che ponga attenzione ala stabilizzazione della roba “”"comune".
- Esiste anche un elemento di tempo.Passando da una comprensione astratta ad una più concreta di una soluzione, migliora il proprio senso in merito alla migliore organizzazione da adottare. È necessario pianificare un'attività (riorganizzazione) di refactoring di un modello nella transizione da una fase all'altra (analisi business, requisiti, analisi di applicazione progettazione di alto livello...).

Se nell'analisi di una soluzione tutto appare altamente interdipendente ed unito, o l'architettura necessita di lavoro o qualcosa nella natura del dominio del problema non ne consente la decomposizione.In ogni caso, esistono delle possibilità di scelta:

- Decidere di assegnare il progetto ad un team molto piccolo che condivide uno spazio fisico e comunica in modo molto attivo su tutte le modifiche apportate da ognuno che possono avere effetto su altri artefatti.
- Prepararsi ad eseguire molte fusioni non superficiali

XDE/Rose

Un utente Rose o XDE ha forse sperimentato la creazione di fusioni di modello non superficiali. La buona notizia è che in RSx tali fusioni sono più semplici da affrontare. Una principale differenza è che RS*x* opera su un tessuto di file di modellazione collegati contro una gerarchia di file di petalo o sotto-unità.Significa che la decomposizione logica di alto livello viene supportata molto meglio a livello fisico in RSx.

È anche supportato il confronto della fusione con migliori tool – infatti, con una esperienza di fusione enormemente migliore.

- Il motore di fusione di back-end è 10.000 (si, diecimila) volte più veloce di quello di XDE.
- Implementa anche una quantità di tecnologie come delta compositi (un meccanismo di raggruppamento che ordina le modifiche per diagramma e raggruppa grandi gesti di diagramma insieme per abilitare il gruppo e/o operazioni atomiche), protezione dell'integrità del modello, atomicità e elaborazioni delta a cascata.“”“”“”“” Riducono la probabilità di corruzione di un file per fusione quasi come avviene per modifica.
- Esiste ora un reale diagramma visivo di fusione di GUI per la risoluzione di conflitti di livello o ornamentali

Arma #2: forte proprietà

Una volta stabilita una forte decomposizione strutturale dovrebbe essere abbastanza semplice (abilità speciali a parte) associare forti proprietà di componenti strutturali ad individui esperti o piccoli team.“”Quando ogni pacchetto logico (o ramo) in un modello può essere elaborato solo da un esperto, allora le fusioni eseguite con quel modello saranno per lo più superficiali (indipendentemente dal fatto che il modello sia conservato come un singolo file o multipli file di modellazione).Lo stesso vale se ogni ramo può essere elaborato esclusivamente da un piccolo team i cui membri sono inclini alla comunicazione su ciò che stanno facendo.

È possibile evitare fusioni non superficiali dividendo i modelli in più file di modellazione? No. “”**Le interdipendenze architettoniche sono un fenomeno logico, non fisico.**Nella divisione di un modello, le rappresentazioni delle interdipendenze dell'elemento diventano riferimenti lungo il file piuttosto che nel file.Non viene fatto niente di più facile per risolvere conflitti (infatti viene reso più difficile). E quando vengono introdotti riferimenti lungo il file questi diventano anche punti di rottura (vedere la barra laterale).

Barra laterale: riferimento al file di modellazione incrociata

Se due elementi di modellazione risiedono in file di modellazione diversi e viene creata una relazione tra di loro, allora viene creato un riferimento al file di modellazione incrociata.“” Poiché i file di modellazione (file .emx) sono esposti nel file system OS host e possono essere spostati, rinominati o modificati al di fuori dell'ambiente di Eclipse, i riferimenti rappresentano potenziali punti di rottura. Tuttavia, se i file di modellazione vengono modificati e gestiti lungo l'ambiente di Eclipse e vengono seguite le seguenti linee guida, le rotture non dovrebbero verificarsi.

Qualora si lavori con una ‘chiusura’ dei file di modellazione (ossia, una raccolta di file che fanno riferimento l'uno agli altri) è necessario avere t utti questi file nello spazio di lavoro. Questo non implica strettamente che debbano risiedere tutti nello stesso progetto, ma questo garantisce la presenza di tutti i modelli, in quanto nei flussi di lavoro tipici di CM tutti i file viaggiano insieme nello stesso progetto.

Dunque, ecco il riepilogo:

- Se manca una forte architettura o una forte proprietà, è possibile sperimentare frequenti fusioni non superficiali che nn possono essere risolte da divisioni del modello.
- Con una forte proprietà e una forte architettura, viene ridotta (ma non eliminata) la frequenza di tali fusioni.Non possono mai essere eliminate perché esistono sempre interdipendenze di componenti.gli elementi ‘comuni’ menzionati sono solo un esempio tra tanti.
- I modelli di separazione in più file non sono importanti come la logica strutturazione dei modelli per abilitare più esperti al lavoro in parallelo senza introdurre modifiche conflittuali.
- La buona notizia è che RSx gestisce la fusione del modello in modo più rapido ed efficace di ogni altro tool disponibile.

Quando eseguire modelli di separazione? Cercare di evitarli, tranne nei casi in cui la dimensione di un modello ha iniziato a verificare i limiti del proprio hardware e i file di modellazione creati possono essere elaborati esclusivamente (ad esempio, solo un membro del team ha un file verificato in ogni momento) o in isolamento (ad esempio, la maggior parte delle modifiche possono essere eseguite sul file senza richiedere l'accesso ad altri file contenenti elementi di modello collegati).

appendice: modifiche tra le versioni 6.x e 7.x di RSx

Nel primo rilascio di RSx (versioni 6.x), y non supportava il concetto di ‘sotto-unità’ come Rose e XDE (una sotto-unità è un file di modellazione che contiene un sottoinsieme di un modello ed è ‘transparente’ nel senso che non appare nella vista logica del modello visibile nella sua vista di navigazione). Invece, la separazione del modello è stata limitata alla definizione di più modelli di alto livello (nel senso che ogni file di modellazione appare come immissione separata, di alto livello nella vista di navigazione del modello). “”

RSx fornisce anche la capacità di selezionare un pacchetto UML di un modello esistente e di creare un modello basato su tale pacchetto. “” È possibile credere che ciò significhi tagliare il pacchetto per creare un nuovo file di modellazione. “”Il risultato è che nella vista di navigazione del modello il pacchetto sembra diventare un nuovo modello logico di alto livello. La visibilità dell'originale struttura di contenimento gerarchico è perduta, ma se viene aperto un file di modellazione da cui sono stati tagliati in questo modo i pacchetti, tutti i modelli tagliati vengono aperti. “”“”Questo può risultare da controlli non necessari, come la congestione nella vista di navigazione del modello e l'apparizione di grandi quantità di schede nel pannello di editor, che insieme rendono difficile la navigazione.

L'aggiunta del supporto di sotto-unità nelle versioni 7.0 di RSx abilita la separazione dei modelli in più file senza perdita di visibilità nella struttura gerarchica e, insieme all'eliminazione delle schede di editor del modello nel relativo pannello, vengono superati gli altri problemi di navigazione. “”

Tuttavia, una lezione sulla strutturazione del modello risalente all'esperienza delle versioni precedenti di RSx viene ancora applicata oggi. Come sopra rilevato, aprire un file di modellazione da cui i pacchetti sono stati tagliati via comporta l'apertura di tutti i modelli tagliati. Questo può essere evitato pianificando una strategia di separazione in anticipo, invece di utilizzare la funzione Make a Model. “”

Chi segue la funzione Make a Model inizia con un singolo file di modellazione e crea pacchetti per rappresentare l'organizzazione dell'architettura della soluzione. “”Ad esempio può esserci un pacchetto per ogni componente principale o sottosistema della soluzione (dove ‘componente’ e ‘sottosistema’ riflettono l'uso informale di quei termini che risalgono ai primi giorni di calcolo, non le loro definizioni formali di semantica UML). Possono anche esserci pacchetti per componenti comuni, di utilità o di“”“” “framework”.Poi, con la crescita del modello i pacchetti che corrispondevano a componenti specifici di un'applicazione possono essere tagliati via come file di modellazione separati, in modo che i team che costruiscono tali componenti possono (teoricamente) lavorare su quei file di modello in isolamento. “” Ma in pratica aprire uno di questi file di modello specifico del componente comporta l'apertura del modello originale ‘principale’ (dove risiedono i pezzi ‘comuni’) che a turno apre tutti gli altri modelli specifici del componente dell'applicazione. I risultati sono, come detto in precedenza, superflui controlli e difficoltà di navigazione.

Un miglior approccio consiste nel pianificare in anticipo di definire un modello separato per ogni componente specifico dell'applicazione, insieme ad un modello per i componenti ‘comuni’ (o forse più modelli che definiscono successivi livelli di componenti comuni/riutilizzabili, come ad esempio il riflettere i livelli di astrazione dell'architettura d'implementazione). Infine, tutti i modelli di componenti specifici dell'applicazione possono essere aperti dal team che li possiede e solo i modelli ‘comuni’ da cui dipende quello specifico dell'applicazione deve essere aperto.