Лекція 4. Принципи специфікації вимог

Специфікація вимог пов'язана з докладним моделюванням вимог за замовниками, визначених у процесі встановлення вимог. При цьому рассматрива ються тільки послуги, які прагнуть отримати від системи замовники (формулювання сервісів) (разд.3.1). На етапі специфікації вимог формулювання обмежень не підлягають подальшого опрацювання, хоча і можуть зазнати змін як результат звичайного циклу ітерації.

В якості вхідної інформації процесу специфікації вимог виступають неформальні вимоги замовників, а результатом цього процесу є моді чи специфікації проектних конструкцій. Ці моделі (розд. 2.2) дають більш фор мальное визначення різних сторін (подань) системи. Зазвичай требова ня користувачів в процесі специфікації поділяються на дві основні кате горії: функціональні вимоги і вимоги до даних.

Як результат етапу специфікації виступає розширений ("детально пророблений") документ опису вимог (розд. 3.6). Новий документ часто називають документом специфікації вимог (або просто "специфікацією" на жар гонах розробників). Структура вихідного документа не змінюється, однак содер жание значно розширюється за рахунок глав, які визначають вимоги замовлення чиков. Поступово для цілей проектування і реалізації документ специфікації вимог замінює документ опису вимог (на практиці, розширений до мент може як і раніше називатися документом опису вимог).

Моделі специфікацій можна розділити на три групи.

1. Моделі станів.

2. Моделі поведінки.

3. Моделі Зміни станів.

Моделі станів "деталізують" вимоги до даних. Моделі поведінки забезпе чивают деталізовані специфікації для функціональних вимог. Моделі зміни станів охоплюють два види вимог. Вони покликані пояснити, ка ким чином дію функцій призводить до зміни даних.

Моделі представляються у вигляді діаграм на мові візуального моделювання (візуальна модель ІНГ Language) - в нашому випадку це мова UML. Зазвичай діаграма служить цілям моделиро вання однієї зі сторін системи - станів, поведінки або зміни станів. За Метн виняток становить діаграма класів, яка визначає всі три аспекти - стан і поведінку об'єктів, і, побічно, зміни станів об'єктів.

Кожна діаграма дає уявлення про певну стороні системи. Взяті разом діаграми дають можливість розробникам і користувачам поглянути на пропоноване рішення з різних точок зору, виділяючи одні його сторони і ігнорувати руя інші. Жодна з діаграм окремо не дає повного визначення систе ми. Систему можна зрозуміти тільки через взаємопов'язаний набір діаграм.

Аналогічно нагоди інтерпретації завершених моделей конструювання діа грам - це не послідовний процес побудови однієї діаграми за одною. Діаграми розробляються в паралель, і в результаті кожної наступної Ітера ції до них додаються нові деталі. У той час, як розробники повинні следо вать строго певному процесу розробки, рішення про те, яка з моделей має відігравати роль "рушійної сили" розробки, значною мірою залежить від особистих переваг аналітика. Зазвичай діаграми прецедентів і моделі клас сов - як найбільш важливі типи моделей - конструюються паралельно, взаємно "збагачуючи" один одного ідеями.

З кожною новою итерацией розробки глибина і ступінь деталізації специфи кации зростає. Багато глибші властивості об'єктів моделі виражаються ско реї в текстовому, ніж графічному вигляді. Деякі властивості визначають задум об'єкта моделі, а не результат аналізу. Деякі інші властивості можуть відображати особливості CASEсредств.

4.2. специфікації станів

Стан об'єкта визначається значеннями його атрибутів і асоціацій. Напри заходів, BankAccount (об'єкт Банківський рахунок) може перебувати в стані "перевищення кредитного ліміту", якщо значення атрибута балансу (БАЛАНС) негативно кові. Оскільки стану об'єкта визначаються структурам даних, моделі структур даних називаються специфікаціями станів.

Специфікація станів дає статичний погляд на систему (тому моделиро вання станів часто називають статичним моделюванням). Тут основний за дачею є визначення класів проблемної області, їх атрибутів і відносин з іншими класами. Спочатку операції класів зазвичай не розглядаються. Вони виводи водяться з моделей специфікації поведінки.

У типовій ситуації спочатку визначаються классисущності, тобто класи, які визначають проблемну область і характеризуються постійною присутністю в базі даних системи. Подібні класи іноді називаються "бізнескласса". Класи, вича обслуговують системні події (керуючі класи) і класи, які перед становлять GUIінтерфейс (класи уявлення або прикордонні класи) НЕ встановлюється вають до тих пір, поки не стануть відомі поведінкові характеристики системи.

4.2.1. моделювання класів

Модель класів - це наріжний камінь розробки об'єктно ної системи. Класи лежать в основі спостережливості властивостей і поведінки систе ми. На жаль, класи завжди важко піддаються визначенню, а властивості класів не завжди очевидні. Дуже малоймовірно, щоб два аналітика прийшли до одного і того ж безлічі класів і їх властивостей для однієї і тієї ж нетривіальною проблем ної області. Хоча моделі класів можуть відрізнятися, кінцевий результат і ступінь задоволеності користувача можуть бути в рівній мірі достатніми (або в рав іншої мері недостатніми).

Моделювання класів - це не детермінований процес. Не існує про ного рецепта відшукання і визначення найкращих класів. Цей процес в значній котельної мірі носить ітеративний і покроковий нарощуваний характер. До факторів, що визначають успішне проектування класів, відноситься рівень кваліфікації і досвіду аналітика, зокрема, такі можливості.

1. Знання в області моделювання класів.

2. РОЗУМІННЯ області проблемною.

3. Досвід у сфері аналогічних і успішних проектів.

4. Здатність дивитися вперед і передбачати наслідки рішень.

5. Готовність до перегляду моделі з метою усунення недоліків і т.д.

Останній момент пов'язаний з використанням CASEсредств. Широкомасштабне застосування CASEтехнологіі може стати перешкодою на шляху розробки системи в технологічно незрілих організаціях (розд. 1.1.4.2). Однак використання ДЕЛО коштів для підвищення особистої продуктивності завжди виправдано.

4.2.1.1. виявлення класів

Два різних аналітика, як правило, не можуть прийти до ідентичним моделям клас сов для однієї і тієї ж проблемної області, і точно так само два різних аналітика не користуються одним і тим же розумовим процесом при виділенні класів. Літі ратура рясніє підходами, пропонованими для виявлення класів. Аналітики мо гут спочатку навіть слідувати одному з цих підходів, проте подальші ітерації, як правило, обов'язково призводять до використання нешаблонних і в чомусь навіть випадкових механізмів.

Барами (Бахрамі) [4] детально вивчив головні особливості чотирьох основних підходів до виявлення класів (відкриття класу). Нижче перераховані ці підходи.

1. Підхід на основі використання іменних груп.

2. Підхід на основі використання загальних шаблонів для класів.

3. Підхід на основі використання прецедентів.

4. Підхід CRC (клас-відповідальність-колабораціоністи - Клас-обязанності- "співробітники").

Барами [4] поставив у відповідність кожному з підходів опубліковані роботи, проте - на нашу мненію- тільки останній підхід має незаперечну ис джерелом. Тепер ми узагальнимо ці підходи, а потім приведемо приклади, в яких використовується комплексний підхід (змішаний підхід).

4.2.1.1.1. Підхід на основі використання іменних груп

Підхід на основі використання іменних груп (тобто іменників в предло жениях) передбачає, що аналітик читає формулювання документа опису тре мог в пошуках іменних груп. Кожне іменник розглядається як потенційний клас (кандидат клас). Потім список всіх класів поділяється на сле дмуть три групи.

1. Релевантні або відповідні класи.

2. Нечіткі або сумнівні класи.

3. Нерелевантні або невідповідні класи.

До нерелевантних (не має значення) відносяться класи, які виходять за рамки проблем ної області. Для них не вдається дати формулювання їх призначення. Досвідчені анали тікіпрактікі найімовірніше не включають невідповідні класи в початковий список потенційних класів. Таким чином вдається уникнути формальних кроків по ідентифікації і виключенню нерелевантних класів.

До релевантним (відношення) відносяться класи, які безумовно належать про проблемної області. Іменники, що представляють імена цих класів, час то зустрічаються в документі опису вимог. Крім того, значення і призначенням ня цих класів можна обґрунтувати на основі загальних знань про прикладну області, а також на основі вивчення аналогічних систем, керівництв, документів і патенто ванних програмних пакетів.

До нечітким (нечітка) відносяться класи, які не можна впевнено і беззастережно визнати придатними. Вони складають найбільшу проблему. Їх необхідно про аналізувати глибше, а потім або включити в список релевантних класів, або виключити зі списку нерелевантних. Остаточне віднесення цих класів до тієї чи іншої групи, власне, і проводить відмінність між хорошою і поганою моделлю класів.

Підхід на основі використання іменних груп передбачає наявність повного та коректного документа опису вимог. На практиці це припущення рідко відповідає дійсності. Але навіть якщо воно обгрунтовано, копітка изуче ня великих обсягів тексту не обов'язково має привести до отримання исчерп вающего і точного результату.

4.2.1.1.2. Підхід на основі використання загальних шаблонів для класів

Підхід на основі використання загальних шаблонів для класів дозволяє вивести потен соціальні класи на основі теорії родової класифікації об'єктів. Теорія класси фікації стосується поділу світу об'єктів на зручні групи, що дозволяє більш ефективно будувати міркування про них.

Барами [4] наводить такий перелік груп (шаблонів) для виявлення по потенційних класів.

- Понятійний (або концептуальний) клас (поняття клас) являє собою ідею, яку розділяє або з якої згодна значна спільність людей. За відсутності понять люди не здатні ефективно спілкуватися або навіть спілкуватися на якомусь задовільному рівні. Наприклад, Бронювання (Резервування) - це понятійний клас, що відноситься до системи резервування місць в авіакомпаніях.
- Событійний клас (клас подій). Подія - це щось, що не вимагає часу стосовно нашої часовій шкалі. Наприклад, Прибуття (Прибуття) - це подієвий клас, що відноситься до системи резервування місць в авіакомпаніях.
- Організаціонний клас (організація класу). Організація - це будь-який вид цілеспрямованого об'єднання або сукупності сутностей. Наприклад, турагент (Бюро подорожей) - це клас, що відноситься до системи резервування місць в авіакомпаніях.
- Класс "людей" (люди, клас). Під "людьми" тут розуміється скоріше роль, яку людина відіграє в тій чи іншій системі, а не фізична особа. Наприклад, Пасажир (Пасажир) - це клас, що відноситься до системи резервування місць в авіакомпаніях.
- Класс розташування (місця класу). Місцезнаходження визначає фізичне розташування об'єктів, пов'язаних з інформаційною системою. Наприклад, TravelOffice (Офіс бюро подорожей) - подібний клас, що відноситься до системи резервування місць в авіакомпаніях.
- Дж. Рамбана (J. Рамбо), А. Джекобсон (I. Jacobson) і Г. Буч (Г. Буч) [76] пропонують іншу схему класифікації.
- Фізіческій клас (фізичний клас) (наприклад, літак (Літак)).
- Бізнескласс (бізнес-клас) (наприклад, резервування).
- Логіческій клас (логічний клас) (наприклад, FlightTimetable (Розклад рейсів)).
- Прікладной клас (клас додатки) (наприклад, ReservationTransaction (Операція резервування)).
- Компьютерний клас (комп'ютерний клас) (наприклад, Index (Індекс)).
- Поведенческій клас (поведінковий клас) (наприклад, ReservationCancellation (Скасування резервування)).

Підхід на основі використання загальних шаблонів класів служить в якості поліз ного керівництва, але не визначає систематичного процесу, за допомогою якого можна було б виділити надійне і повне безліч класів. Цей підхід можна з успіхом використовувати для визначення початкового безлічі класів або для перев ки того, чи повинні деякі класи (отримані іншими способами) бути присутнім в нашому безлічі або, навпаки, бути відсутніми в ньому. Однак, підхід на основі ис користування загальних шаблонів класів занадто слабо пов'язаний зі специфічними требо ваниями користувачів, щоб претендувати на вичерпне рішення.

Особлива небезпека, пов'язана з підходом на основі використання загальних шаблонів класів, полягає в неправильному тлумаченні імен класів.

Наприклад, що означає Прибуття (Прибуття)? Чи означає це прибуття на взлетнопосадочную смугу (час приземлення), прибуття до терміналу (час висадки), прибуття в зал повернення багажу (час митного огляду) і т.д.? Аналогічно, слово Бронювання (в даному слу чаї резервація. Прим. Ред.) У середовищі північноамериканських індіанців має зовсім інше значення в порівнянні з тим, що малося на увазі досі.

4.2.1.1.3. Підхід на основі використання прецедентів

Підходу на основі використання прецедентів надається особливе значення в мові UML. Можна навіть сказати, що цей підхід рекомендується використовувати в рамках UML (якщо бути точним - то в рамках методології RUP (Rational Unified Process)). Графічна модель прецедентів супроводжується неформальними описами, а також діаграма ми послідовностей і кооперації для окремих прецедентів (розд. 2.2 і далі в цьому розділі). Ці додаткові описи і кроки визначення діаграм (і об'єктів) потрібно виконати для кожного прецеденту. На основі цієї інформації можна прийти до узагальнень, необхідним для виявлення потенційних класів.

Підхід, що направляється прецедентами, володіє особливостями, притаманними під ходу снізувверх. Після того, як прецеденти стають відомі, а уявлення про систему з точки зору взаємодії, щонайменше, частково визначено з по міццю діаграм послідовностей, об'єкти, що використовуються в цих діаграмах, призводять до виявлення класів.

Насправді цей підхід в чомусь схожий на підхід, який використовує іменні групи. Їх об'єднує те, що прецеденти специфікують вимоги. Обидва підходи спрямовані на вивчення формулювань, викладених в документі опису вимог, щоб виявити в результаті потенційні класи. Те, що ці формулювання викладаються в оповідної формі або представлені графічно, має второ статечне значення. У будь-якому випадку на цьому етапі ЖЦ розробки ПЗ більшу частину прецедентів можна описати тільки в текстовій формі без діаграм взаємодії.

Підхід, заснований на прецедентах, має ті ж вади, що й під хід, який використовує іменні групи. Будучи по суті підходом снізувверх в сенсі точності, він спирається на повноту і коректність моделей прецедентів. У результа ті, він навіть може привести до небажаного розбалансування ітеративного і нарощуємо процесу розробки ПО, при якому моделі прецедентів повинні бути завершені ще до побудови моделей класів. Загалом, хоч би якими були цілі і засоби, це призводить до функціонального підходу (functiondriven підхід) (прихильники об'єктно підходу вважають за краще називати його про блемноорієнтірованним (problemdriven)).

4.2.1.1.4. CRC Підхід

CRC Підхід (Клас-Відповідальність-Співавтори - клас-відповідальність-"співро ники") являє собою щось більше, ніж метод виявлення класів, - це при привабливості спосіб інтерпретації та вивчення об'єктів (а також і навчання об'єк проектних підходу). Найбільшу популярність підхід CRC отримав завдяки роботам Ребекки ВірфсБрок (Ребекка WirfsBrock) і її колег Б. Вилкерсон (Б. Вилкерсон) і Л. Вінер (L. Wiener) [94], [93].

CRC Підхід включає в себе сеанси "мозкового штурму", проведення яких обліг чає за рахунок використання спеціально підготовлених карток. Картки складаються з трьох відділень: ім'я класу записується в верхньому відділенні, "обов'язки" класу пере чисельні в лівому відділенні, а "співробітники" перераховані в правому відділенні. Зобов'язане сти - це послуги (операції), які клас готовий виконати в інтересах інших класів. Для виконання багатьох обов'язків необхідна участь (обслуговування) з боку інших класів. Такі класи перераховуються як "співробітники".

CRC Процес - це живий процес, під час якого розробники "грають в карти", вони заповнюють картки іменами класів і призначають їм "обов'язки" і "співробітників" в ході виконання сценарію обробки інформації (наприклад, сце Нарія прецеденту). У тих випадках, коли виникає потреба в якійсь послузі, а су Існуючі класи не покривають її, створюється новий клас, якому призначаються відповідні "обов'язки" і "співробітники". Якщо клас стає "занадто за прийнятим", він поділяється на кілька менших класів.

CRC Підхід відрізняється від інших підходів тим, що при його використанні виокрем лення класів є результатом аналізу повідомлень, переданих між об'єктивним тами для виконання завдань обробки інформації. Акцент робиться на уніфікує ванном методі розподілу "інтелекту" в системі, і деякі класи можуть бути скоріше отримані, виходячи з подібної технічної потреби, ніж виявлені в якості "бізнесоб'ектов" як таких. У цьому сенсі метод CRC може бути по над прийнятним для перевірки правильності вибору класів уже виявлених з по міццю інших методів. CRC Підхід також корисний при встановленні властивостей класів (які логічно випливають з "обов'язків" і типів "співробітників" Класу).

4.2.1.1.5. комплексний підхід

На практиці процес виявлення класів в різний час, найімовірніше, слід різним підходам. Найчастіше, він включає елементи всіх чотирьох підходів, рассмот корінних вище. Важливими чинниками при цьому виступають загальна ерудиція екс перта, його досвід і інтуїція. Процес в чистому вигляді не слід ні методу сверхувніз, ні методу снізувверх - він весь час йде "з середини". Подібний підхід до виявле ню класів ми називаємо комплексним підходом (змішаний підхід).

Ось один з можливих сценаріїв. Початкове безліч класів можна сформи вати на основі загальних знань і досвіду експерта. При цьому додатково можна керуватися підходом на основі загальних шаблонів класів. Решта класи можна

додати, грунтуючись на аналізі узагальненого опису проблемної облас ти з використанням підходу на основі іменних груп. Якщо в розпорядженні анали тика є прецеденти, можна скористатися підходом, які направляються Прец дентами, щоб додати нові і перевірити спроможність існуючих класів. Нарешті, CRC підхід дозволяє застосувати "мозковий штурм" для перевірки пра ності вибору виділеного до цього часу безлічі класів.

4.2.1.1.6. Деякі правила виявлення класів

Нижче наведено далеко не повний перелік керівних принципів або правил менту, котрим повинен слідувати аналітик при виборі потенційних класів. Знову напір міна про те, що тут ми маємо справу лише з классамісущностямі.

1. Для кожного класу має бути ясно сформульовано його призначення в системі.

2. Кожен клас - це шаблон опису безлічі об'єктів. Поодинокі класи, для яких можна уявити існування тільки одного об'єкта, досить малоймовірні серед "бізнесоб'ектов". Подібні класи зазвичай складають в додатку "загальне знання" і як правило жорстко запрограмовані в програмах додатки. Наприклад, якщо система спроектована для единст кої організації, існування класу Організація (Організація) мо же бути не виправдано.

3. Кожен клас (тобто класссущность) повинен містити набір атрибутів. Хоро шим прийомом є встановлення ідентифікують атрибутів (ключів), щоб допомогти нам судити про потужність (потужність) класу (тобто очікуваний коли честве об'єктів даного класу в базі даних). Слід, однак, пам'ятати про те, що клас не обов'язково повинен володіти призначеним для користувача ключем. Об'єкти класів ідентифікуються за допомогою ідентифікаторів об'єктів

(OID) (розд. 2.1.1.3).

4. Кожен клас повинен відрізнятися від атрибута. Чи подається поняття класом або атрибутом залежить від галузі застосування. Колір автомобіля зазвичай сприймається як атрибут класу Car (Автомобіль). Однак на фаб рике з виробництва фарб Color (Колір) - це безперечно клас зі своїми власними атрибутами (яскравістю, насиченістю, прозорістю і т.д.).

5. Кожен клас містить набір операцій. Однак на даному етапі ми не торкаємося питань ідентифікації операцій. Операції, що входять в інтерфейс класу (послуги, що надаються класом системі), є логічним наслідком їм формулювання призначення класу (пункт 1).

4.2.1.1.7. Приклади виявлення класів

Давайте проаналізуємо вимоги з метою виділення потенційних класів. У першому затвердження відповідними класами є класи Ступінь (Ступінь) і курс (Курс). Ці два класи задовольняють п'яти правилам, перерахованих раніше. Ми поки не впевнені, чи повинен і яким чином клас Курс бути звужений до класів CompulsoryCourse (Обов'язковий курс) і ElectiveCourse (Вибірковий курс). На

приклад, ясно, що курс є обов'язковим або вибірковим залежно від сте пені. Можливо, що відмінність між обов'язковими і вибірковими курсами може бути зафіксовано за допомогою асоціації або навіть атрибута класу. Таким чином,

CompulsoryCourse і ElectiveCourse розглядаються як нечіткі класи.

Друге твердження ідентифікує тільки атрибути класу для гольфу, а саме course_level (рівень курсу) і credit_point_value (кількість умовних оч ков). Третє твердження характеризує асоціацію між класами Курс і

Ступінь. Четверта формулювання вводить атрибут min_total_credit_points (мінімальна загальна кількість умовних очок) в якості атрибута класу ступеня.

Останнє твердження дозволяє нам виділити три нових класи: Студент (Студент), CourseOffering (Пропонований курс) і StudyProgram (Програма обу чення). Перші два, безумовно, є релевантними класами, ось StudyProgram а можна перетворити в асоціацію між класами Студент і CourseOffering. Тому StudyProgram класифікується як нечіткий клас. Наші міркування відображені в табл. 4.1.

Таблица 4.1. Потенциальные классы (Запись на университетские курсы)

Релевантные классы	***Нечеткие классы***
Course (Курс)	CompulsoryCourse (Обязательный курс)
Degree (Степень)	и ElectiveCourse (Выборочный курс)
Student (Студент)	StudyProgram (Программа обучения)
CourseOffering (Предлагаемый курс)	

Перше твердження містить кілька іменників, але тільки недо торие з них можна перетворити в потенційні класи. "Відеомагазинів" - це не клас, оскільки він становить всю систему (в базі даних може бути тільки один об'єкт цього класу - так званий поодинокий клас). Аналогічно, поняття "запасу" і "відеотеки" занадто загальні, щоб розглядати їх як класів, по край ній мері на цьому етапі. Релевантними класами представляються MovieTitle (Назва фільму), відеокасетах (Відеокасета) і відеодиск (Відеодиск).

Друге твердження вводить додаткову спеціалізацію відеопрокату як про ката відеокасет і дисків. Ми можемо запропонувати три нових класи: BetaTape, VHSTape і DVDDisk. Однак оскільки ми маємо справу лише з одним типом відео дисків, в кінцевій моделі можна не залишати обидва класи: відеодісковая і DVDDisk. Який з двох класів залишити, залежить від кінцевого рівня спеціалізації примі нительно до відеокасет і дисків. Зауважимо, що ми поки не впевнені в тому, чи будуть класи BetaTape і VHSTape володіти какимилибо відмітними атрибутами (за винятком того факту, що один з них належить типу Beta, а інший - VHS).

Третє твердження говорить про те, що кожне найменування фільму відрізняється умовами прокату, пов'язаними з ним. Однак не ясно, що мається на увазі під "фільмом" - найменування фільму або ж носій фільму (касета або

диск)? Нам необхідно прояснити цю вимогу з замовниками. Тим часом, ми можемо віддати перевагу оголосити клас RentalConditions (Умови прокату) як нечіткий, замість того, щоб зберігати інформацію про період прокату і плати за прокат в клас се для найменування фільму або для носія фільму.

Останнє формулювання переконує нас в тому, що класи BetaTape, VHSTape і DVDDisk (або відеодиск) - релевантні. Нам потрібно зберігати інформацію про ті кущіх умовах для кожної касети і диска. Однак атрибути на зразок video_condition (умови відеопрокату) або number_currently_available (кількість, яка є в наявності) можна в загальному оголосити в абстрактному класі верхнього рівня (назвемо його VideoMedium (відеоносіях)), після чого вони можуть бути успадковані конкретним подклассом (таким як VHSTape). Підсумки наших рас суджень відображені в табл. 4.2.

Таблиця 4.2. Потенциальные классы (Магазин видеопроката)

Релевантные классы	***Нечеткие классы***
MovieTitle (Название фильма) VideoMedium (Видеоноситель) VideoTape (Видеокассета) VideoDisk (или DVDDisk) BetaTape VHSTape	RentalConditions (Условия проката)

Перше твердження містить поняття "клієнт", "контракт" і "товар". Наша про щая ерудиція і досвід підказують нам, що це типові класи. Однак поняття контракту і товару не входять в рамки системи Управління контактами з клієнтами і повинні бути відкинуті.

Замовник (Клієнт) - це релевантний клас, проте ми можемо віддати перевагу на кликати його Контакт (Контакт), маючи на увазі, що не всі контакти здійснюються з на шими справжніми клієнтами. Різниця між готівковим і потенційним клієнтом може виправдати або не виправдати введення таких класів як CurrentCustomer (Готівковий клієнт) і ProspectiveCustomer (Потенційний клієнт). Оскільки впевненості у нас немає, ми оголошуємо ці класи нечіткими.

Друге твердження проливає нове світло на наведені вище міркування. Нам необхідно провести відмінність між контактної організацією і контактною особою. Ім'я клієнта не здається занадто зручним для назви класу. Крім іншого, Замовник поняття має на увазі тільки готівкового клієнта і, крім того, це може містити в собі як поняття організації, так і контактної особи. Наша нова пропозиція полягає в тому, щоб назвати класи наступним чином: Організація (Організація), Contact (Контакт) (мається на увазі контактна ліцо), CurrentOrg (Готівкова організація) (тобто організація, яка є нашим на особистим клієнтом) і ProspectiveOrg (потенційна організація) (тобто організує ція, що є нашим потенційним клієнтом).

У другому затвердження згадується кілька атрибутів класів. Однак звичайні ний і кур'єрський поштові адреси являють собою складові атрибути і при Меним до обох класів: Організація і контакт. Тому PostalAddress і CourierAddress - законні нечіткі класи.

Таблица 4.3. Потенциальные классы (Управление контактами с клиентами)

Релевантные классы	***Нечеткие классы***
Organization (Организация)	CurrentOrg
Contact (Контакт)	ProspectiveOrg
Employee (Сотрудник)	PostalAddress
Task (Задание)	CourierAddress
Event (Мероприятие)	

У третій формулюванні вводиться три релевантних класу Співробітник (Співробітник), Task (завдання) і Event (Захід). Ця пропозиція пояснює суть планової діяльності.

Останнє твердження присвячено подальшому проясненню сенсу і взаимоот відносин між класами, однак тут не вводиться жодного нового класу.

4.2.1.2. специфікація класів

Після того, як перелік потенційних класів сформований, необхідна їх подальша специфікація: класи потрібно включити в діаграму класів і визна ділити їх властивості. Деякі властивості можна ввести і відобразити всередині графічні ських піктограм, що представляють класи на діаграмі класів. Багато інші властивості, включені в специфікацію класу, мають тільки текстове представле ня. CASEсредства, як правило, володіють можливостями редагування, позво ляющая легко вводити або модифікувати подібну інформацію за допомогою діалогових вікон, забезпечених вкладками, або за допомогою аналогічних способів.

Як пояснювалося на початку цієї глави, класи задаються на певному рівні абстракції. Більш розвинені можливості моделювання мови UML тут не ис користуються - вони розглядаються в розділі 5 і наступних розділах.

4.2.1.2.1. іменування класів

Кожному класу необхідно присвоїти ім'я. При роботі з деякими ВИПАДОК засобами крім імені класу можна також привласнити код, можливо, відмінний від імені. Код може задовольняти угодам по іменування, необхідним целе вим мовою програмування або СУБД. Для генерації програмного коду з проектної моделі використовується саме код класу, а не ім'я.

Мимохідь, ми прийняли певна угода щодо імен класів. Ця угода укладається в тому, що ім'я класу починається з великої літери. Для складу них імен в якості першої літери кожного слова також використовується велика літера (замість того, щоб відокремлювати слова знаком підкреслення або

дефісом). Це всього лише реко вані угоду, проте серед розробників воно знайшло досить багато при верженцев. (В розд. 6.1.3.2 до імені кожного класу рекомендується додати однобуквен ний префікс для позначення програмного шару, до якого належить клас).

Ім'я класу має бути іменником в однині (наприклад, курс) або, при можливості, поєднанням прикметника і сущест вітельнимі в однині (наприклад, CompulsoryCourse). Ясно, що клас є шаблон для безлічі об'єктів і використання в якості іменників у множині не несе ніякої додаткової ін формації. Іноді іменник в однині не відображає справжнього призначення класу. У подібних ситуаціях допустимо використання існуючих тільних у множині (наприклад, RentalConditions (Умови прокату) в прикладі 4.2).

Ім'я класу має бути осмисленим. Вона повинна відбивати справжню природу класу. Воно повинно запозичувати з словника користувачів (а не жаргону розробників).

Краще використовувати довші імена, ніж приховувати їх сенс за шифром. Можна з упевненістю сказати, що імена довжиною понад тридцять символів занадто громіздкі (а деякі програмні середовища їх просто не сприймають, якщо ДЕЛО кошти працюють з іменами класів, а не кодами класів). Можливо також використання довших описових імен на додаток до імен та кодами класів.

4.2.1.2.2. Виявлення та специфікація атрибутів класів

Графічна піктограма, що представляє клас, складається з трьох відділень (ім'я класу, атрибути, операції) (розд. 2.1.2). Специфікація атрибутів класів належить до специфікації станів і розглядається в цьому розділі. Специфічні кация операцій розглядається пізніше в цьому розділі в підрозділі, присвяченому специфи кации поведінки (розд. 4.3).

Виділення атрибутів здійснюється паралельно з виділенням класів. Ідентифікації атрибутів свого роду "побічний ефект" встановлення класів. Це не означає, що виявлення атрибутів - просте завдання. Навпаки, це процес, требующій значних зусиль і багаторазових ітерацій.

Вихідні моделі специфікації визначають тільки атрибути, які є существ веннимі для розуміння станів, в яких можуть знаходитися об'єкти класу. Ос ментальні атрибути можна до пори до часу ігнорувати (проте аналітик повинен бути впевнений в тому, що встановлена, але проігнорована на певному етапі інформація не буде помилково загублена і буде зафіксована згодом). Мало ймовірно, щоб всі атрибути класу були приведені в документі опису охоронних вимог, однак важливо не включати в специфікацію ті атрибути, які не випливають з вимог. У наступних ітераціях можна додати більше атрибутів.

Для імен атрибутів ми рекомендуємо дотримуватися простого угоди: в іменах атрибутів використовувати тільки малі літери, а слова в складових іменах від делять підкресленням.

4.2.1.2.3. Приклади специфікації класів

У першому затвердження згадуються конфлікти розкладу, однак ми не знаємо дос товерно, як слід моделювати цю проблему. Можливо, що мова тут йде про Прец денте, який процедурно визначає конфлікти розкладу. Другу частину цієї ж формулювання можна змоделювати за рахунок ведення атрибута enrolment_quota (квота набору) в клас CourseOffering. Тепер також ясно, що клас повинен мати Атрібі тами рік (рік) і семестр (семестр).

Друге формулювання зміцнює нас на думці про необхідність введення класу StudyProgram. Можна бачити, що клас StudyProgram поєднує в собі ряд дисцип лін, пропонованих до вивчення в поточний момент. Тому клас StudyProgram як і повинен мати атрибутами рік і семестр.

Найближчий розгляд нечітких класів CompulsoryCourse і

ElectiveCourse призводить нас до висновку, що навчальний курс є обов'язковим або вибірковим щодо певної наукового ступеня. Один і той же курс мо же бути обов'язковим по відношенню до одного ступеня, вибірковим, що стосується іншого, і взагалі неприпустимим стосовно деяким іншим ступенями. Раз так, то CompulsoryCourse і ElectiveCourse не є класами в повному смислі ле слова. (Зауважимо, що тут ми не торкаємося область моделювання класів з ис користуванням узагальнення (2.1.5 розд.) - Моделювання узагальнення розглядається в розділі 4.2.4.)

На рис. 4.1 представлена модель класів, відповідна проведеним нами міркувань. Крім того, на малюнку використовуються символи (стереотипи (стереотипу)) << PK >> і << >> CK для позначення первинних ключів і потенційних ключів, з відповідально (розд. 8.4.1.2). Це унікальні ідентифікатори об'єктів для рассмот корінних класів. Тут же задані типи даних для атрибутів.

Класи StudyProgram і CourseOffering не мають поки що ідентифікують атрибутів. Вони будуть введені в ці класи після встановлення асоціативних зв'язків між класами (розд. 2.1.3 і 4.2.2).

Перше твердження роз'яснює, що умови прокату відрізняються для касети (відеокасетах) і диска (відеодиск), що містять один і той же фільм. Тому має сенс ввести окремий клас RentalConditions (який повинен асоціювати ся з класами відеоплівку і відеодиск).

З формулювання другого вимоги випливає необхідність співіснування обох класів: відеодісковая і DVDDisk. Ієрархія узагальнення з коренем VideoMedium тепер стає досить очевидною, однак ми відкладаємо обговорення ставлення ня узагальнення до розділу 4.2.4.

На підставі аналізу третього пропозиції ми вводимо в клас MovieTitle ат рібути коду фільму movie_code (як ключового атрибута) і режисера - режисер. Решта атрибути були розглянуті в прикладі 4.2.

На рис. 4.2 показана модель класів для додатка Магазин відеопрокату, по строєна в результаті обговорення вимог прикладів 4.2 і 4.3. VideoMedium.number_currently_available Атрибут є атрибутом з областю дейст віякласс (статичним атрибутом) (розд. 2.1.6). MovieTitle.is_in_stock Атрибут, що відображає наявність в запасі фільму з даними назвою, є похідним (отримані) атрибутом, тобто він може бути обчислений на основі наявного в наявності ті кущего кількості фільмів - VideoMedium.number_currently_available.

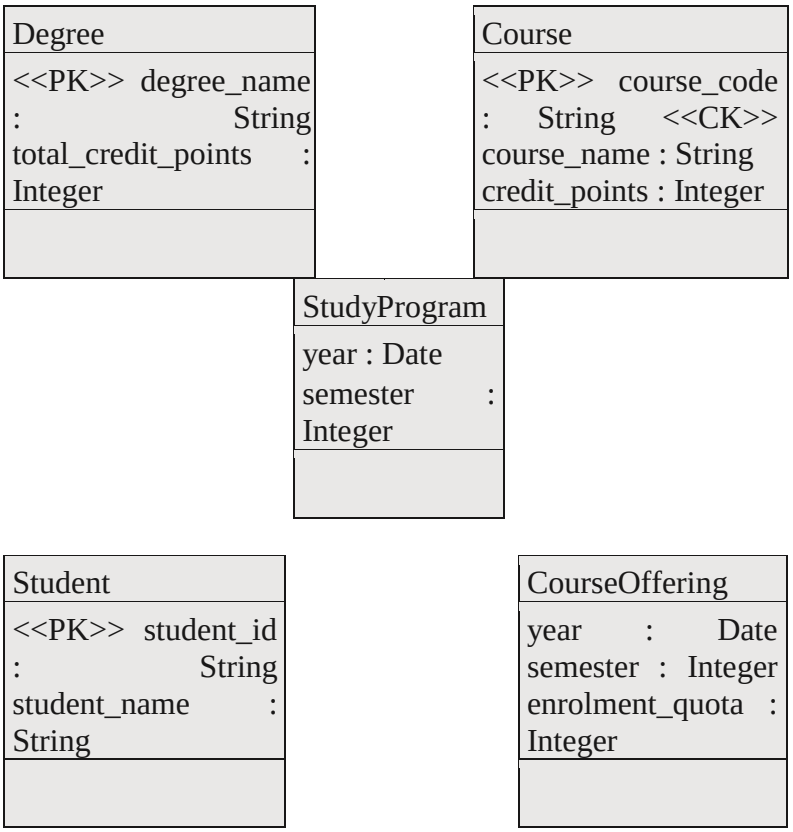
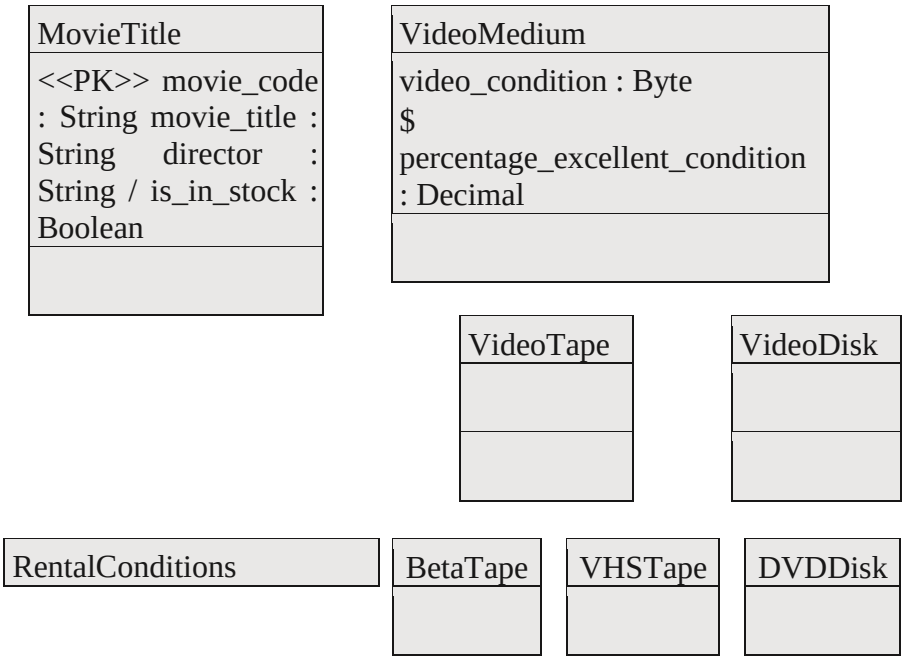


Рис. 4.1. Спецификация классов (Запись на университетские курсы)



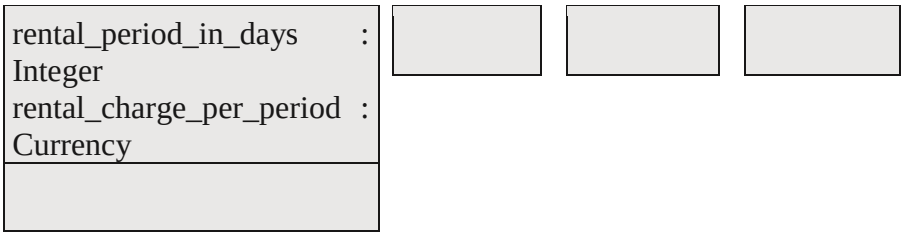


Рис. 4.2. Спецификация классов (Магазин видеопроката)

Аналіз першого твердження говорить нам про те, що поняття готівкового клієнта ви водиться на основі асоціації між класами організації і контракту. Ця асоціацію -ціативності зв'язок може бути досить динамічною. Отже, потреба в класах CurrentOrg і ProspectiveOrg відпадає. Більш того, наша система не стосується управ ління контрактами і не відповідає за підтримку класу контракту. Кращий варіант, до торий проглядається в даному випадку - змоделювати рішення за допомогою вироб водного атрибута Organization.is_current, який позначає готівкову органи зацию і при необхідності може змінюватися підсистемою управління контрактами.

Друге твердження наводить на думку про необхідність запровадження двох класів для адрес: PostalAddress і CourierAddress.

Решта формулювання вимог дають додаткову інформацію про зі триманні атрибутів класів. Крім того, тут можна висловити кілька сообра жень, пов'язаних з асоціативними відносинами та обмеженнями цілісності (вони будуть розглянуті пізніше).

Специфікація класів для додатка Управління контактами з клієнтами перед ставлена на рис. 4.3. Як видно з малюнка, відносини між класами в моделі від сутні. Тому, наприклад, восьме твердження, яке пов'язує співробітників із завданнями і заходами, не знайшло відображення в моделі.

Перша формулювання дозволяє нам встановити кілька атрибутів класу

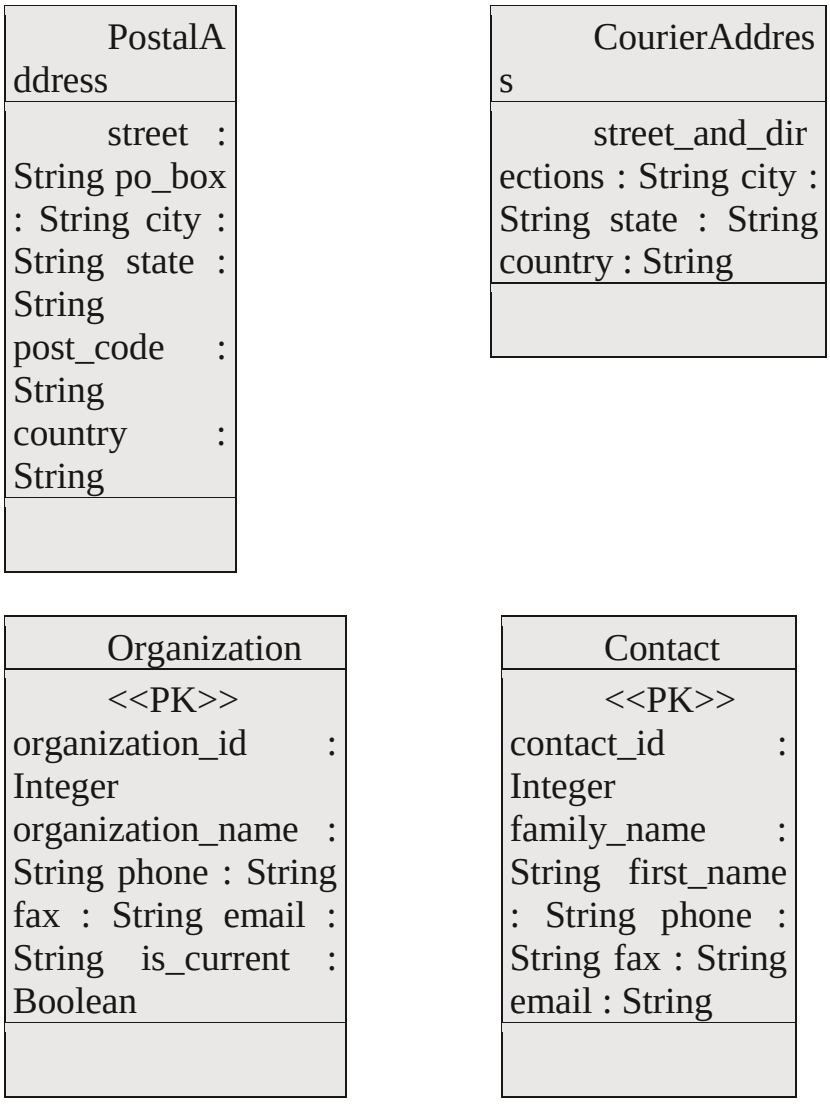
Кампанія. Клас кампанії містить наступні атрибути CAMPAIGN_CODE (код кам панії) (первинний ключ), campaign_title (назва кампанії), DATE_START (дата початку) і date_closed (дата закриття).

Остання пропозиція затвердження 1 стосується призів кампанії. При його бли жайшее розгляді можна прийти до висновку про те, що премія (Приз) - самостоя вальний клас: приз розігрується, йому притаманна така властивість як переможець, і він повинен володіти іншими явно не вираженими властивостями, такими як опис, цінність і місце в ряду інших призів кампанії.

Ми вводимо клас Приз в модель класів разом з атрибутами, про які говорилося вище: prize_descr, prize_value і prize_ranking. Ми помічаємо, що дата троянди Гриша призів збігається для всіх призів кампанії. Ми вводимо атрибут дати розигри ша date_drawn в клас кампанії. Приз виграє благодійник. Ми можемо зафік сіровать цей факт пізніше в зв'язку з введенням асоціації класів і премії Supporter.

2 Формулювання говорить, що у кожного квитка є номер, однак цей номер не унікальний серед усіх квитків (він унікальний тільки в рамках кампанії). Ми вводимо ат рібут ticket_number, однак, не надаємо йому статус первинного ключа для класу CampaignTicket. Два інших атрибута CampaignTicket представляють ціну квитка (ticket_value) і статус квитка (ticket_status). Загальна кількість квитків і до лічество проданих квитків - КЛАСУ кампанії Атрибути (num_tickets і num_tickets_sold).

3 Затвердження виявляє кілька відкритих питань. Які структури дан них потрібні нам для розрахунку продуктивності Телемаркетер? Щоб відповісти на це питання нам потрібно визначити деякі показники, за допомогою яких можна висловити продуктивність. Один з можливих варіантів полягає в тому, щоб підраховувати середня кількість дзвінків на годину і середня кількість успішних дзвін ков на годину. Потім можна обчислити показник продуктивності, розділивши кількість вус пешню дзвінків на загальну кількість дзвінків. Тепер введемо в клас Телепродавец відповідні атрибути: average_per_hour і success_per_hour.



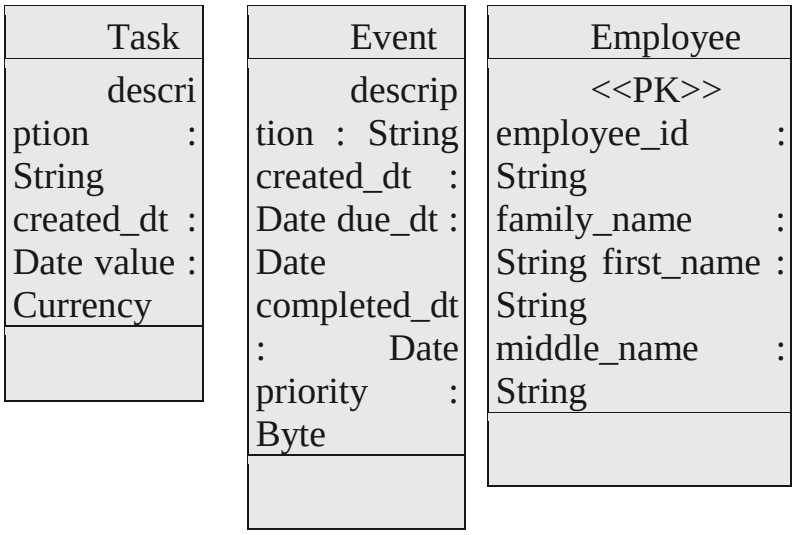


Рис. 4.3. Спецификация классов (Управление контактами с клиентами)

Для обчислення показників продуктивності Телемаркетер необхідно запоми нать тривалість кожного дзвінка. Сховищем для цієї інформації служить клас CallOutcome (Результат дзвінка). У цей клас ми вводимо атрибути для початку і закриття кампанії: start_time і END_TIME. Ми припускаємо, що результат каж дого дзвінка пов'язаний Телемаркетер допомогою асоціації.

Результатом аналізу затвердження 4 є включення ряду атрибутів в клас Прихильник. Ось ці атрибути: supporter_id (первинний ключ),. supporter_name, phone_number, mailing_address, date_first, date_last, campaign_count, preferred_hours і credit_card_attributes. Деякі з цих атрибутів (mailing_address, preferred_hours) досить складні для того, щоб превра тить їх в додаткові класи пізніше в процесі розробки. Поки ж ми зберігаємо їх як атрибути.

5 Затвердження стосується класу CallScheduled (Запланований дзвінок). Ми вводимо в нього атрибути phone_number, пріоритет і attempt_number. У нас немає підлогу ного розуміння того, як підтримувати вимогу про те, що наступні дзвінки повинні проводиться в різний час дня. Очевидно, що це повинно бути покладено на алгоритм планування, однак нам потрібна підтримка структур даних. До сча стю, певне світло на цю проблему проливає наступне формулювання.

Результатом аналізу затвердження 6 є введення класу CallType (Тип дзвін ка). Цей клас містить наступні атрибути: type_descr, call_attempt_limit, а також alternate_hours. Останній атрибут - це складна структура даних, анало гічно атрибуту preferred_hours Прихильник класу, яка врешті-решт стане окремим класом.

Останнє твердження класифікує результати дзвінків. Це пряма вказівка на необхідність введення відповідного класу - OutcomeType. Можливі типи результату можуть зберігатися в атрибуті outcome_type_descr. Не ясно, які ще атри пляшок можна включити в OutcomeType, проте ми переконані, що у міру вивчення де талізірованних вимог, ці атрибути будуть встановлені. Одним з них

може бути атрибут follow_up_action, призначення якого - зберігати інформацію про ти пічних наступні кроки стосовно кожного типу результату.

На рис. 4.4 представлена модель класів, завершальна наведені вище рассу ждения. Асоціативні відносини, вже встановлені в моделі бізнескласса (рис. 3.6), збережені. Нові асоціацію не введені.

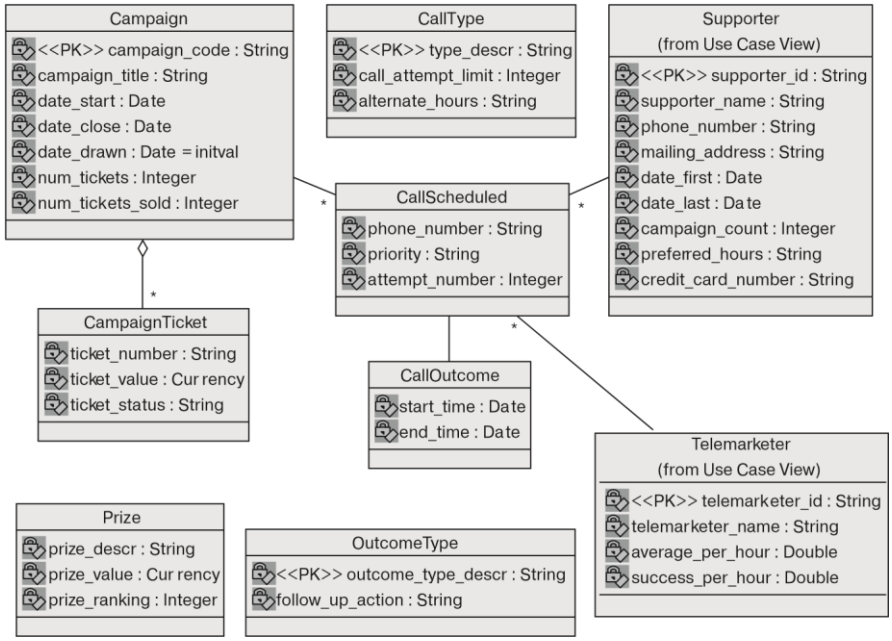


Рис. 4.4. Спецификация классов (Телемаркетинг)

4.2.2. моделювання асоціацій

Асоціації служать об'єднанню об'єктів в системі. Вони сприяють взаємо дії між об'єктами. Без асоціацій об'єкти можуть встановлювати зв'язок тіль ки під час прогону програми, якщо вони спільно використовують одні і ті ж Атрібі ти або вони мають доступ (за допомогою інших засобів, таких як глобальні змін ні) до ідентифікує об'єкти значеннями інших об'єктів.

Асоціації представляють собою найбільш істотний вид відносин моделей, зокрема, моделей постійних "бізнесоб'єктов". Асоціації підтримують ви конання прецедентів і, таким чином, забезпечують поєднання специфікації станів і поведінки.

4.2.2.1. виявлення асоціацій

Знаходження основних асоціацій є побічний ефект процес са виявлення класів. При визначенні класів аналітик приймає рішення про ат рібутах класів, і деякі з цих атрибутів є асоціаціями з іншими класами. Атрибути можуть ставитися до елементарних типів даних або можуть вво диться в якості інших класів, встановлюючи таким чином відносини з іншими класами. По суті, будь-який атрибут, що відноситься до Неелементарні типам даних, повинен моделюватися як асоціація (або агрегація) по класу, який представляє цей тип даних.

Виконання пробного прогону прецедентів дозволяє виявити залишаються асоціації цій. Встановлюються шляхи взаємодії між класами, необхідні для прогону прецедентів. Зазвичай асоціації повинні підтримувати ці шляхи взаємодії.

Кожна тернарного асоціація повинна бути замінена циклом або бінарної асоціації цією. Тернарние асоціації привносять ризик невірного семантичного тлумачення.

Іноді для того, щоб повністю виразити базову семантику, цикли, утворені асоціаціями, не повинні комутувати (бути замкнутими) [51]. Це означає, що щонайменше одна з асоціацій в циклі може бути похідною (похідний). Подібна асоціація є надлишкової в семантичному сенсі і повинна бути виключена (хороша семантична модель повинна бути позбавлена надмірності). Цілком допус Тімо, що багато похідні асоціації, тим не менш, все ж увійдуть в проектну модель (наприклад, з міркувань ефективності). 4.2.2.2. специфікація асоціацій

Специфікація асоціацій має на увазі виконання наступних дій.

1. Привласнення Імен асоціаціям.

2. Привласнення імен асоціативним ролям.

3. Встановлення кратності асоціації (розд. 2.1.3.2).

Правила іменування асоціацій повинні відповідати угодам по име нованіе атрибутів - імена асоціацій складаються з малих літер, окремі слова в імені асоціації розділяються підкресленням (розд 4.2.1.2.2.).

Якщо два класи пов'язані лише одним асоціативним відношенням, задавати ім'я асоціації та асоціативні рольові імена між цими класами необов'язково (розд. 2.1.2.1.1). CASEсредства можуть внутрішньо розрізняти кожну асоціацію через системні ідентифікаційні імена.

Рольові імена можна використовувати для розкриття більш складних асоціацій, в ча стности самоассоціативних відносин (самостійні об'єднання) (рекурсивних асоціацій, кото риє пов'язують об'єкти одного і того ж класу). При завданні рольових імен їх наслідком дме вибирати з урахуванням того, що в проектній моделі вони стануть атрибутами класів, розташованих на протилежних кінцях асоціативного зв'язку.

Кратність повинна бути задана для обох кінців (ролей) асоціації. Якщо питання кратності на цьому етапі не ясний, нижня і верхня межі кратності можна опустити.

4.2.2.3. Пример спецификации ассоциации

Модель ассоциаций для приложения *Управление контактами с клиентами* показана на рис. 4.5. Чтобы продемонстрировать гибкость ассоциативного моделирования, имена ассоциаций и ролевые имена используются бессистемно.

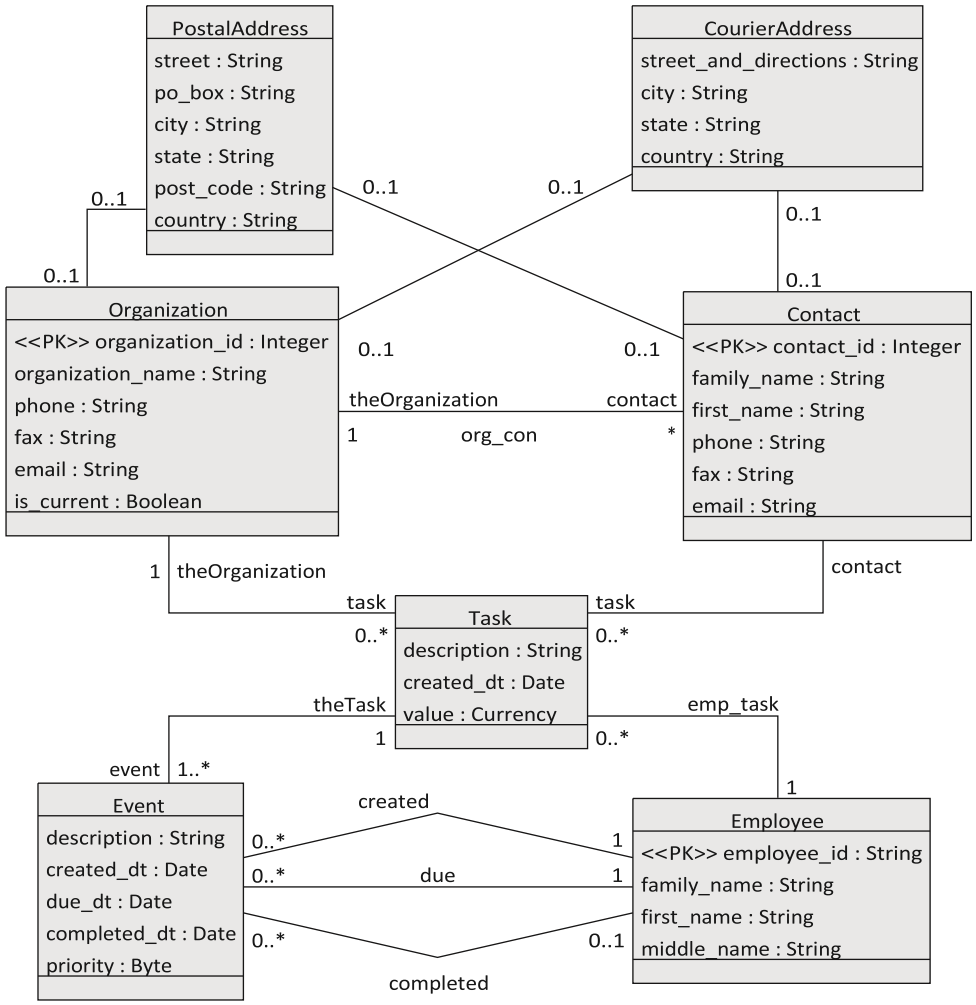


Рис. 4.5. Спецификация ассоциаций (Управление контактами с клиентами)

Кратність всіх асоціацій між класами і PostalAddress CourierAddress з одного боку і класами Організація і зв'язатися з з іншого дорівнює "нуль або один". Дамо пояснення до використання асоціацію між класами і організацією

Поштова адреса).

На одному кінці асоціації об'єкт Організація пов'язаний максимум з одним об'єктах тому PostalAddress, але тільки в тому випадку, якщо поштова адреса організації извес тен. На протилежному кінці асоціації конкретний об'єкт PostalAddress по в'я з об'єктом організації або об'єктом контакту. Отже, щоб удовле творять цим обмеженням, кратність повинна дорівнювати "нуль або

один". (Навіть за цих умов, саме обмеження необхідно окремо зафіксувати документально за допомогою засобів моделювання обмежень мови UML (розд. 5.1.2)).

Асоціація між класами і організації Зв'язатися демонструє використання у який спосіб асоціацій, так і рольових імен. Рольові імена перетворюються в імена атрибутів моделі реалізації програми Управління контактами з клієнтами. Модель реалізації містить наступні атрибути: Organization.contact і Contact.theOrganization. Префікс "The" означає, що роль theOrganization має кратність точно рівну "один". (Артикль "The" вживається в англійській мові для вказівки на певний, конкретний об'єкт. Прим. Ред.). Кратність ро чи контакт дорівнює "багато" (нижня і верхня межі не встановлені).

Кратність ролі контактного Між класами завдань і зв'язатися з не задана. Вимоги не пояснюють, чи повинно завдання бути безпосередньо пов'язано з контактом. Тому у нас немає впевненості в тому, чи може воно бути пов'язано більше, ніж з одним контактом.

Нарешті, існує три асоціації між класами і події співробітників. Ці ас асоціації встановлюють, хто із співробітників створює захід, хто відповідає за його виконання і хто, врешті-решт, завершує його. Під час створення мероприя ку співробітник, який працює над його завершенням, невідомий (тому крат ність на кінці асоціації завершена з боку працівника дорівнює "нуль або один").