Сьогодні процес створення складних програмних застосувань неможливо уявити без поділу на певні етапи життєвого циклу. Серед основних можна винести:

* Аналіз предметної сфери та створення технічного завдання
* Проектування структури програми
* Написання безпосередньо коду програми
* Тестування
* Впровадження програми
* Супровід програми

Сьогодні ми розглянемо процес проектування. В ході проектування архітектором створюється документація проекту, яка включає в себе текстовий опис, діаграми, моделі майбутньої програми. В цьому йому домагає мова UML.

UML – це графічна мова для візуалізації, опису параметрів, конструювання та документування різних систем (зокрема програм). На основі технології UML створюється єдина інформаційна модель. Редактори UML моделей можуть також генерувати код на різних об’єктно-орієнтованих мова.

Основні типи діаграм для візуалізації моделі:

* Діаграма варіантів використання (use case diagram)
* Діаграма діяльності (class diagram)
* Діаграма послідовності (sequence diagram)
* Діаграма стану (state diagram)
* Діаграма компонентів (component diagram)
* Діаграма розгортання (deployment diagram)
* Діаграма пакетів (package diagram)

Розглянемо деякі з них.

Діаграма варіантів використання

Проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою, так званих прецедентів. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою.

Діаграма класів

Діаграма класів служить для подання статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. Діаграма класів може відображати, зокрема, різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти і підсистеми, а також описує їх внутрішню структуру (поля, методи ...) і типи відносин (спадкування, реалізація інтерфейсів ...). На даній діаграмі не вказується інформація про тимчасові аспекти функціонування системи. З цієї точки зору діаграма класів є подальшим розвитком концептуальної моделі проектованої системи.

Діаграма станів

Головне призначення цієї діаграми - описати можливі послідовності станів і переходів, які в сукупності характеризують поведінку елемента моделі протягом його життєвого циклу. Діаграма станів представляє динамічну поведінку сутностей, на основі специфікації їх реакції на сприйняття деяких конкретних подій.

Діаграма послідовностей

Для моделювання взаємодії об'єктів в мові UML використовуються відповідні діаграми взаємодії. Взаємодії об'єктів можна розглядати в часі, і тоді для представлення тимчасових особливостей передачі і прийому повідомлень між об'єктами використовується діаграма послідовності. Взаємодіючі об'єкти обмінюються між собою деякою інформацією. При цьому інформація приймає форму закінчених повідомлень. Іншими словами, хоча повідомлення і має інформаційний зміст, воно набуває додаткового властивість надавати спрямований вплив на свого одержувача.

Діаграма компонентів

Діаграма компонентів, на відміну від раніше розглянутих діаграм, описує особливості фізичного представлення системи. Діаграма компонентів дозволяє визначити архітектуру розроблюваної системи, встановивши залежності між програмними компонентами, в ролі яких може виступати вихідний, бінарний і виконуваний код. У багатьох середовищах розробки модуль або компонент відповідає файлу. Пунктирні стрілки, що з'єднують модулі, показують відношення взаємозалежності, аналогічні тим, які мають місце при компіляції початкового програмного коду. Основними графічними елементами діаграми компонентів є компоненти, інтерфейси і залежності між ними.

Навіщо використовувати платформу JavaEE та UML разом?

Будь-який більш-менш досвідчений програміст може розробити програму, яка буде виконувати поставлену задачу – до якогось часу. Але розробка програмної системи підприємства, яка буде зручною в супроводі, масштабованою та спроможною до вдосконалення – це зовсім інша справа. В наші дні, коли система повинна вдосконалюватись в надзвичайно швидкому темпі, щоб не застаріти, важливо мати уявлення про майбутнє системи до того, як виникне потреба її обслуговувати, масштабувати і вдосконалювати.

Мова UML надає інструментальні можливості, необхідні для проектування і побудови складних систем, наприклад таких, що потрібні на підприємстві.

Також UML містить моделюючі конструкції, за допомогою яких розробники зможуть швидше зрозуміти, як будуть взаємодіяти великі частини системи на етапі виконання і як вони залежать один від одного на етапі компіляції.

Діаграми UML, такі як діаграми взаємодії, діяльності та класів можуть служити для кращого розуміння і документування складних взаємодій в системі. Це сприяє аналізу проблеми, а також полегшує детальну письмову фіксацію передбачуваної поведінки.

Використання мови UML дозволяє зробити процес розробки істинно візуальним. Візуальне представлення чудово підходить для обміну інформацією в групі розробників. Крім того, воно може бути використане для швидкого і ефективного введення нових членів групи в проект.

Спробуємо застосувати це на практиці, використовуючи для цього дослідження системи MyHomeBank.

Постановка задачі

Щоб мати кращі позиції в боротьбі з конкурентами банк SuperBank замовив систему MyHomeBank, яка буде надавати клієнтам повний набір електронних банківських послуг.

Послуги MyHomeBank будуть доступні безкоштовно всім клієнтам банку через інтернет.

В першу чергу цей набір буде включати послуги трьох основних типів:

* Довідкові послуги. Включають оновлення залишку на рахунку, перегляд операцій і загрузку протоколу операцій з рахунком
* Послуги по оплаті. Користувачі можуть оплачувати рахунки через інтернет.
* Операційні послуги. Включають також функції переказу грошей, купівлю та продаж цінних паперів.

Окремий механізм знадобиться для функцій адміністрування. Крім повного доступу до банківських функцій описаних раніше, функції адміністрування дозволяють створювати нові рахунки, закривати рахунки, дозволяти проводити операції певному клієнту, вручну коректувати залишок на рахунку та на вимогу власника рахунку відміняти майбутні операції по оплаті рахунків.

Електронні послуги MyHomeBank повинні працювати на основі існуючих баз даних і іншої банківської інфраструктури. Також повинна бути передбачена інтеграція с існуючою електронною кредитною системою CreditSystem та системою відстеження неоплачених рахунків BankAccountSystem.

Допущення

Звичайно неможливо описати повну реалізацію справжньої повнофункціональної електронної банківської системи за короткий час. Так як головна ціль нашої лекції проілюструвати концепції платформи JavaEE будуть внесені деякі допущення.

Для реалізації будуть використовуватись лише технології JavaEE.

Клієнт системи має мати хоча б один рахунок.

Рахунок може належати лише одному клієнту. Спільні рахунки не допускаються.

Вимоги до MyHomeSystem

Система в всіх випадках використовує існуючі бази даних. Вона повинна дозволяти швидкий перехід на іншу платформу, якщо виникне необхідність.

Інтерфейс користувача має бути таким, щоб клієнти могли легко робити запити і виконувати операції, роблячи мінімальну кількість кліків мишкою.

В будь-який момент часу користувачу мають бути доступні лише дозволені послуги. Наприклад, якщо у користувача немає рахунків, які потрібно оплатити, система повинна показати це і не дозволяти йому починати операції оплати рахунку.

Довідкові послуги

Користувачі повинні мати можливість переглянути інформацію по рахункам.

Попередньо користувачі повинні авторизуватись в системі. Для цього необхідні логін та пароль. Як тільки вони перевірені, користувач отримує доступ до всіх можливостей MyHomeBank.

Користувач, який успішно авторизувався, має мати можливість змінити пароль і обрати новий. При цьому, в якості мірі безпеки, користувач повинен ввести попередній пароль. Коли пароль буде перевірений, система повинна прийняти зміну паролю.

Користувач повинен мати можливість отримати з бази даних підсумки по залишкам на рахунках. Разом з цим користувачу повинна бути представлена детальна інформація про рахунок.

Користувач повинен мати можливість отримувати і переглядати список останній операцій по всім рахункам. Для кожної операції в списку повинні бути показані наступні дані: номер операції, задіяні рахунки, сума операції.

Послуги по оплаті

Користувач повинен мати можливість застосовувати послуги MyHomeBank для оплати своїх рахунків через інтернет. Користувачу також повинен надаватися список неоплачених рахунків через систему BankAccountSystem. Користувач повинен мати можливість вибрати зі списку неоплачений рахунок і виконати його оплату з будь-якого зі своїх зареєстрованих рахунків або кредитний карток, якщо залишок на рахунку достатній для покриття суми рахунку.

Також користувач повинен мати можливість оплачувати рахунки зареєстрованим продавцям. Після того, як продавець доданий в список отримувачів платежів, користувач повинен мати можливість вносити йому платежі, поки цей продавець зареєстрований в банку і не видалений зі списку отримувачів платежів.

Операційні послуги

Хоча користувачі системи MyHomeBank не можуть виконувати банківські операції звичайного виду, такі як розміщення вкладу або зняття з рахунку, вони повинні мати можливість проводити переказ грошей між звичайними та ощадними рахунками, кредитними картками та неоплаченими рахунками. При цьому користувач вказує рахунки і суму переказу, система повинна перевірити чи є на рахунку потрібна сума. Якщо так, сума знімається з одного рахунку та записується на інший.

Якщо користувач має рахунки для інвестиційних операцій, він повинен мати можливість отримувати підсумок з залишків на рахунках і список цінних паперів, а також виконувати покупку та продаж цінних паперів.

На початковій стадії проектування було визначено список основних дійових осіб системи та список основних прецедентів.

Дійові особи:

* Клієнт. За допомогою MyHomeBank отримує інформацію і виконує банківські операції.
* Адміністратор. Налаштовує користувачів системи. Отримує та модифікує інформацію.
* Продавець. Отримує перекази в результаті оплати рахунків.
* Поштова система. Отримує від MyHomeBank запити на відправку підтверджень по електронній пошті.
* Система CreditSystem. Взаємодіє з системою MyHomeBank для надання клієнтам послуг по кредитуванню.
* Система BankAccountSystem. Взаємодіє з системою MyHomeBank для надання клієнтам послуг автоматичного відслідковування неоплачених рахунків.

Найбільш важливі прецеденти:

* Перегляд рахунку
* Список транзакцій
* Переказ грошей
* Оплата рахунків
* Вхід в систему
* Вихід з системи
* Редагування профілю

Прецеденти, які помічені для подальшого дослідження і деталізації:

* Оплата рахунку продавця
* Зміна списку продавців
* Покупка цінних паперів
* Продаж цінних паперів
* Відміна оплати рахунків

На презентації показано діаграми першочергових та додаткових прецедентів.

Розглянемо потоки подій для першочергових прецедентів.

Прецедент «Перегляд рахунку»

Головний потік подій: прецедент починається, коли клієнт успішно ввійшов в систему і обирає перегляд залишку на своїх рахунках. Система знаходить інформацію для даного користувача і виводить її на екран.

Прецедент «Переказ грошей»

Головний потік подій: Клієнт входить в систему і обирає переказ грошей. Система виводить список наявних рахунків. Клієнт обирає рахунок, з якого будуть переведені гроші і рахунок, на який вони будуть переведені і вказує сему переказу. Система перевіряє перший з вказаних рахунків і підтверджує, що потрібна сума є на рахунку. Сума знімається з першого рахунку і записується на другий. Операція реєструється.

Прецедент «Оплати рахунку»

Головний потік подій: Клієнт входить в систему і вирішує оплатити один з неоплачених рахунків, які визначаються через систему BankAccountSystem. Клієнт обирає рахунки і вводить потрібну інформацію. Система перевіряє чи на рахунку міститься необхідна сума і пропонує клієнту підтвердити оплату. Потім система знімає суму платежу з рахунку клієнта і реєструє платіж.

На стадії розвитку дані діаграми взаємодії можуть бути розширені та доповнені. Для прикладу розглянемо додаткові діаграми взаємодії для прецедентів перегляду списку операцій, оплати рахунку, переказу грошей. На цій стадії визначено основні компоненти системи та показано взаємодію компонентів при виконанні певного прецеденту.

Також побудовано діаграму класів, на якій відображено основні класи системи.

Використовуючи можливості UML редактору, в даному випадку Enterprise Architect, ми маємо можливість згенерувати код на основі даної діаграми класів. Для цього в контекстному меню діаграми класів обираємо: Code Engineering -> Generate Source Code.

В полі Root Directory вказуємо місце збереження згенерованих файлів.

В згенерованих файлах створені всі оголошені на діаграмі атрибути та методи без реалізації. Приклад згенерованого файлу можна побачити на презентації.

На подальший стадіях розробки будуть використовуватись згенеровані файли для написання прототипу. Також буде проаналізовано додаткові прецеденти та додано до системи відповідний функціонал.