Список використаних джерел

1 Марчук Ю. Н. Компьютерная лингвистика / Ю. Н. Марчук. – М.: АСТ: Восток-Запад, 2007. – С. 60-70.

2 SyTech – разработка программного обеспечения: аналитические системы, электронный документооборот, корпоративные системы, информационные порталы [Электронный ресурс]. – Режим доступа: http://www.sytech.ru.

3 TextAnalyst 2.0 – персональная система автоматического анализа текста [Электронный ресурс]. – Режим доступа: http://www.analyst.ru.

4 Автоматическая обработка текста [Электронный ресурс]. – Режим доступа: http://www.aot.ru.

5 Landauer, T. K., Foltz, P., and Laham, D. 1998. An Introduction to Latent Semantic Analysis. DiscourseProcesses, 25: 259-284.

6 Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R.A. 1990. Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science, 41: 391-407.

7 Нгуен Ба Нгок. Обзор подходов семантического поиска [Электронный ресурс] / Нгуен Ба Нгок, А. Ф. Тузовский // Доклады Томского государственного университета систем управления и радиоэлектроники: периодический научный журнал. – 2010. – № 2 (22). Ч. 2. – С. 234-237. – Режим доступа: http://www.tusur.ru/filearchive/reports-magazine/2010-2-2/234.pdf.

8 Косинов Д. Локальные параметры текстов и проблема определения почти-дубликатов [Электронный ресурс] / Д. Косинов // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии. – 2008. – Т. 1. – С. 83-85. – Режим доступа: http://www.vestnik.vsu.ru/pdf/analiz/2008/01/kosinov.pdf.

УДК 004.4

А.М. Акименко, канд. фіз.-мат. наук
Чернігівський державний технологічний університет, м. Чернігів, Україна

ДЕЯКІ АСПЕКТИ ЗАСТОСУВАННЯ UML ПРИ РОЗРОБЛЕННІ СКЛАДНИХ ПРОГРАМНИХ СИСТЕМ

Розглянуто проблеми, пов'язані з формуванням завдання на розробку коду програми після завершення стадій аналізу та проектування складної програмної системи. Запропоновано використання специфікацій сумісного типу для організації роботи груп експертів та постановки завдань на виконання.

Ключові слова: *розробка програм, сумісний тип, специфікація, діаграма варіантів використання, технічне завдання.*

Рассмотрены проблемы, возникающие в процессе формирования задачи на разработку кода программы после завершения стадий анализа и проектирования сложной программной системы. Предложено использование спецификации совместного типа для организации работы групп экспертов и постановки задачи на исполнение.

Ключевые слова: *разработка программ, совместный тип, спецификация, диаграмма вариантов использования, техническое задание.*

The problems associated with the formation of a task to develop the application code after the stages of analysis and design of sophisticated software systems. Proposed the use of the specifications for the type of joint organization of the expert group and set goals for performance.

Key words: *development programs, joint type, specification, use cases diagram, technical task.*

Постановка проблеми. Традиційно розробка складної програмної системи складається з таких етапів [1]:

1. Постановка та аналіз завдання, визначення вимог до проекту.
2. Проектування.
3. Розробка, кодування.
4. Тестування та оцінювання якості.
5. Впровадження та супровід.

Як бачимо, розроблення програмного забезпечення – це доволі складний процес. Кожна зі стадій розроблення має визначені методики її виконання та перелік документів, які повинні бути сформовані для переходу на наступний етап. Однак існує кілька

факторів, які викликають проблеми під час розроблення складного програмного проекту та збільшують ризики його виконання. Серед таких факторів основну увагу розробники приділяють таким моментам:

- складність формування вимог до програмних систем;
- відсутність відповідних засобів, що описують поведінку систем з великою кількістю станів;
- комплексна розробка програмного проекту;
- необхідність збільшення ступеню повторюваності коду.

Ці фактори мають вплив на кожну стадію розробки, але найбільш вразливими є моменти переходу від одного етапу розроблення до іншого. **Метою** цієї статті є вивчення проблеми формування завдання на розроблення програмного коду після завершення стадій аналізу та проектування складної програмної системи.

Аналіз останніх досліджень і публікацій. Сьогодні під час розробки складного програмного забезпечення, зазвичай, використовують одну з двох методологій – структурну або об'єктно-орієнтовану.

Перша методологія для розробки складних програм рекомендує виділяти в програмі окремі підпрограми (процедури), які вирішують вузькоспеціалізовані завдання, тобто проводити процедурну декомпозицію.

Друга методологія використовує більш складний підхід, при якому в предметній області виділяють окремо функціонуючі елементи. Поведінка цих об'єктів моделюється з використанням спеціальних засобів, а потім вже з готових елементів збирається складна програмна система. Таким чином, в основу другої методології покладена об'єктна декомпозиція.

Одна з найпоширеніших (більше 60 % використання) методик об'єктно-орієнтованої методології розроблення програмного забезпечення має назву **Rational Unified Process (RUP),** яка в загальному вигляді має такі складові (фази):

1 фаза "Початок". Виконують такі дії:

- формуються бачення та границі проекту;
- створюється економічне обґрунтування;
- визначаються основні вимоги, обмеження та ключова функціональність продукту;
- створюється базова версія моделі прецедентів;
- оцінюються ризики.

Після завершення початкової фази повинні бути підготовлені такі документи:

- попередній план проекту з оцінкою вартості робіт;
- попередній план розроблення програмного забезпечення (ПЗ), до складу якого вміщено план управління вимогами, організацію проекту, план управління змінами, план управління ризиками та план контролю якості;
- затверджені документи по проекту.

На основі представленої документації оцінюються перспективи та приймається рішення про продовження роботи над проектом.

2 фаза "Уточнення". В цій фазі аналізується предметна область та будується архітектура майбутньої системи. Після завершення цієї фази план проекту повинен містити точні часові та вартісні оцінки розробки системи відповідно до вимог та обраної архітектури. Готуються такі документи:

- уточнений план проекту з графіком його виконання;
- повна специфікація вимог до програмного продукту;
- прототипи функціонального інтерфейсу та інтерфейсу користувача;
- план тестування;
- уточнений план розробки ПЗ;
- затверджені документи по проекту.

3 фаза "Побудова". У фазі «Побудова» відбувається реалізація більшої частини функціональності продукту. Ця фаза завершується першим зовнішнім релізом системи.

Кожна ітерація конструювання повинна супроводжуватися такими документами:

- план наступної ітерації;
- всі документи та моделі з попередніх фаз, у разі їх модифікації на цій ітерації.

4 фаза "Впровадження". В цій фазі створюється фінальна версія продукту, яка передається від розробника до замовника. У випадку, якщо якість продукту не відповідає очікуванням користувачів або критеріям, встановленим у фазі "Початок", фаза "Впровадження" повторюється знову.

Однією з переваг технології RUP є акцентування уваги розробника на веденні проектної документації. Але основна увага, в першу чергу, приділяється документації, яка має зовнішній характер, тобто орієнтована на організацію взаємодії з замовником програмного проекту. Вимог до ведення внутрішнього документообігу в методології RUP не визначено. Окрім однієї: для ведення документації потрібно використовувати мову моделювання програмних проектів UML. Регламент використання мови моделювання в методології RUP не описано, тому під час створення внутрішніх документів розробники програмного забезпечення дуже часто не звертають уваги на цей аспект управління процесом розробки.

Розглянувши процес розробки, перейдемо до питання керування цим процесом.

Сучасний бізнес вимагає спеціалізованого програмного забезпечення підвищеної складності, пристосованого під конкретні бізнес-процеси. Розробники програмного забезпечення, орієнтуючись на замовника, застосовують новітні технології та скорочують строки виробництва програмних продуктів. Враховуючи це, процес виробництва стає все більш непередбачуваним. Необхідність створення високоякісного програмного забезпечення в умовах обмежених строків та бюджету потребує підвищення рівня взаємодії груп розробників з іншими проектними підрозділами.

Початкову фазу та фазу "Уточнення" компанія-розробник виконує, задіявши для цього аналітиків, які займаються питаннями аналізу поставленого завдання та підготовки попередньої документації. Керованість процесом розроблення на цих етапах виробництва дуже висока, оскільки над завданням працює кількісно обмежений колектив, який включає до свого складу аналітиків та керівників груп розробки. Проблеми витримки строків та виконання бюджету на цих етапах також не мають гострого характеру.

При переході від другої фази до третьої над створенням складної програмної системи починає працювати значно більша кількість працівників. До процесу підключаються групи розробників (програмісти). Кількість персоналу, задіяного в процесі розроблення програми, збільшується і виникають проблеми, пов'язані не тільки з керованістю проектом, які вирішуються за допомогою застосування стандартних методик управління, але і специфічні, які характерні тільки для IT-індустрії. Одна з цих проблем – взаємодія між менеджерами проектів та групами розробки, які викликані різними підходами до оформлення документації, що використовується цими категоріями працівників.

Аналітики та менеджери проектів, готуючи документи, орієнтуються в першу чергу на замовника, не враховуючи специфічні потреби внутрішнього користувача, тобто безпосереднього розробника програмного коду.

Мета статті. Метою роботи є вивчення питання про застосування діаграм мови моделювання UML для керівництва роботою програмістів у фазі "Побудови" складного програмного продукту, зменшення ризиків його виконання та формалізації процесу підготовки документації по проекту.

Виклад основного матеріалу. Застосування мови UML у процесі розробки програмного проекту – необхідна умова використання методології **RUP**. Одне з завдань, яке вирішує UML, – це специфікація моделі системи.

У цьому контексті мається на увазі побудова повної моделі системи. UML дозволяє описати всі існуючі рішення, які стосуються аналізу, проектування і реалізації, які приймаються в процесі розроблення та розгортання програмного продукту.

Характерною особливістю мислення більшості програмістів є те, що роздуми про реалізацію проекту, для нього рівнозначні написанню коду програми. Іншими словами, якщо ми думаємо над проектом – ми пишемо код. Дійсно, деякі речі краще за все виражаються безпосередньо в коді мовою програмування. Такий підхід також можна вважати моделюванням, хоч і неформальним. За такого підходу виникає кілька проблем. По-перше, обмін думками стає можливим тільки у випадку, коли всі учасники розуміють одну базову мову. По-друге, не можна отримати уявлення про деякі аспекти програмної системи без моделі, межі якої виходять за рамки мови програмування. По-третє, якщо аналітик ніколи не втілював в явній формі розроблені моделі, інформація може бути втрачена, при зміні колективу розробників. У кращому випадку результати аналізу можна буде тільки частково відтворити, виходячи з реалізації проекту.

Для розуміння принципів побудови програмної системи та її роботи використовується опис функціональності через варіанти використання (діаграма Use Case) [2]. Варіанти використання це – опис послідовності дій, що виконує програма у відповідь на зовнішній вплив користувачів або інших програмних систем.

Варіанти використання призначені в першу чергу для визначення функціональних вимог до системи, що, відповідно, впливає на керованість процесом розробки ПЗ. Все основні види діяльності (аналіз, проектування, тестування) виконуються на основі варіантів використання. На стадіях аналізу і проектування Use Case дозволяють зрозуміти, по-перше, вплив результатів, які бажає отримати замовник, на архітектуру системи і, по-друге, поведінку компонентів системи, що реалізують її функціональність.

Отже, діаграма варіантів використання з супутньою специфікацією цілком і повністю задовольняє потреби аналітиків проекту в оформленні технічної документації проекту і як додаток до технічного завдання (ТЗ) може використовуватись для опису вимог замовника на абстрактному рівні.

На етапі оформлення ТЗ аналітик проекту стикається з дилемою. З одного боку, технічна документація готується для замовника, який не володіє IT-термінами. З другого боку, ТЗ буде використовуватися розробниками (програмістами), для яких важливими є саме технологічні особливості проекту.

Для вирішення цієї проблеми пропонується ввести такі типи специфікації діаграми варіантів використання:

1 ТИП. Бізнес-сценарій. Не враховує технологічні аспекти. Описує бізнес-процеси, які будуть реалізовані в рамках програмного проекту.

2 ТИП. Системний сценарій. Характеризує функції програмної системи та визначає сервіси, що надаються системою користувачам.

3 ТИП. Сумісний сценарій. Має ознаки як бізнес-сценарію, так і системного. Описує реалізацію бізнес-процесів з використанням технічної термінології.

Приклад діаграми варіантів використання, яка розроблялася під сумісний тип сценарію, наведено на рисунку 1.

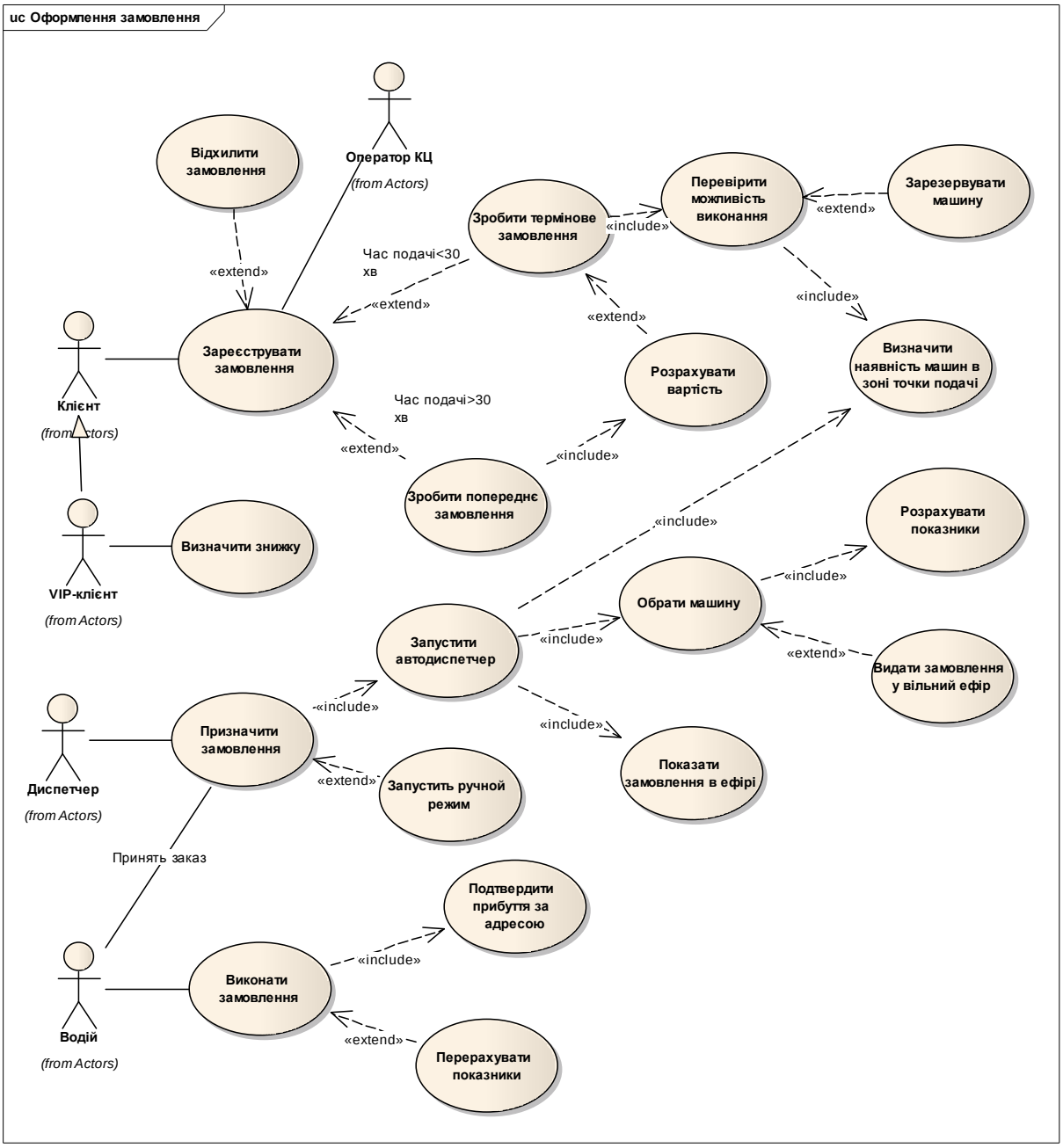

Рис. 1. Приклад діаграми варіантів використання для опису процесу оформлення замовлення в Диспетчерські службі таксі

Таблиця 1 та таблиця 2 містять зразок специфікації сумісного типу для діаграми варіантів використання, наведеної на рисунку 1.

Таблиця 1 описує акторів, що взаємодіють із системою, та їх функціональні обов'язки, враховуючи місце, яке актори займають у бізнес-процесах організації-замовника. В таблиці 2 наводиться приклад опису програмних сервісів, зроблений звичайною мовою, але з урахуванням алгоритміки їх реалізації.

Таблиця 1

Приклад специфікації акторів

Актор	**Опис**
Оператор КЦ	Працюють з клієнтською частиною Диспетчерської служби (ДС). Приймає замовлення по телефону від клієнта. Вносить інформацію про замовлення у форму реєстрації
Клієнт	Клієнт Диспетчерської служби. Є споживачем послуг, які надаються Диспетчерською службою
VIP-клієнт	Має всі права клієнта. Є або постійним клієнтом, або власником дисконтної картки, яка надає право на знижку під час оплати послуг
Диспетчер	Контролює роботу Диспетчерської служби. Керує розподілом замовлень, контролює виконання замовлень
Водії	Особи, які виконують замовлення, прийняті від клієнтів

Таблиця 2

Приклад специфікації варіантів використання

Варіант використання	**Опис**	**Примітка**
1	2	3
Зареєструвати замовлення	До сервісу мають доступ Клієнт та Оператору КЦ. Процес реєстрації починається з отримання вхідного виклику від клієнта. ДС визначає приналежність номера клієнта до списку VIP-клієнтів або до “Чорного” списку. Якщо номер у “Чорному” списку, виклик відхиляється до з’єднання з оператором	**1 Точка розширення** *(за умови, якщо час виклику менше 30 хв):* Зробити термінове замовлення. **2 Точка розширення** *(за умови, якщо час виклику більше 30 хв):* Зробити попереднє замовлення. **3 Точка розширення** *(умови див. в описі):* Відхилити замовлення
Зробити термінове замовлення	Сервіс запускається автоматично після введення часу подачі машини в разі, якщо час подачі менше 30 хв. Клієнт повідомляє адресу подачі, після чого запускається сервіс «Перевірити можливість виконання». За умови отримання позитивної відповіді, Оператор повідомляє клієнта про можливість виконання замовлення і вводить інші параметри замовлення: - (перелік параметрів). Після введення даних запускається сервіс «Розрахувати вартість». Оператор повідомляє вартість проїзду з урахуванням встановленої знижки. Якщо клієнт згоден – замовлення передається на оформлення. Якщо клієнт не згоден – запускається сервіс «Відхилити замовлення»	**Точка включення:** Перевірити можливість виконання. **Точка розширення** *(за умови введення всіх необхідних даних):* Розрахувати вартість
Зробити попереднє замовлення	Сервіс запускається автоматично після введення часу подачі машини у випадку, якщо час подачі більше 30 хвилин. Оператор вносить параметри замовлення: - (перелік параметрів). Після введення даних запускається сервіс «Розрахувати вартість». Оператор повідомляє вартість проїзду з урахуванням встановленої знижки	**Точка включення:** Розрахувати вартість

Продовження табл. 2

1	2	3
Відхилити замовлення	Скасування на етапі реєстрації допускається лише у випадках: 1) немає машин в зоні адреси подачі та клієнт не може довго чекати; 2) клієнта не влаштовує рівень послуг, вартість або час подачі і він сам скасовує замовлення. Встановлюється статус замовлення <CANCEL>, причина «Відмова клієнта»	
Перевірити можливість виконання	Визначаємо наявність машин у зоні подачі. Якщо в зоні є хоча б один позивний, оператор пропонує клієнтові час подачі, розрахований системою. Якщо клієнт згоден на запропонований час, резервуємо машину Якщо в зоні позивних немає, оператор отримає повідомлення «МАШИН У ЗОНІ НЕМАЄ» і повідомляє клієнтові про неможливість подати машину протягом найближчого часу	
Розрахувати вартість	Для розрахунку вартості замовлення визначається відстань між точкою місцезнаходження автомобіля, точкою подачі, проміжними пунктами, точкою призначення. Прокладається оптимальний маршрут руху машини по карті. Визначається його довжина в км. Після отримання довжини маршруту визначаємо його вартість за відповідним тарифом. Облік дисконтної картки: отриману суму зменшуємо на знижку. Оператор бачить розраховану вартість замовлення і повідомляє її клієнтові	
Підтвердити прибуття за адресою	...	
...	...	...

Відмінність наведеної специфікації сумісного типу від інших полягає в тому, що сценарій кожного варіанта використання являє собою запис алгоритму реалізації програмного сервісу системи, виконаний звичайною мовою. Такий підхід дозволяє підвищити ступінь розуміння технічної документації з боку замовника, що, відповідно, дозволяє залучити його безпосередньо до роботи над проектом навіть на стадії розробки програмного коду.

Тісна співпраця дає можливість оперативно реагувати на зауваження з боку замовника та вносити відповідні зміни до програмного коду.

Інший аспект, який є ще однією перевагою сумісного типу специфікації, полягає у використанні його для аналізу та ідентифікації ризиків, пов'язаних з вимогами до програмного проекту [3]. Запис сценаріїв звичайною мовою дозволяє залучити більш широке коло експертів, включаючи фахівців з інформаційних технологій та фахівців у конкретній предметній галузі. Це, по-перше, дозволить отримати більш точну оцінку можливих ризиків. По-друге, участь фахівців з інформаційних технологій дозволить оцінити складність розробки програмних сервісів та ще на ранніх стадіях розроблення внести необхідні зміни в проект.

Такий підхід позитивно впливає на загальні ризики виконання програмного проекту, зменшуючи їх.

Ще однією перевагою такого типу специфікації є пом'якшений перехід до формалізації проектної документації, яка оформлюється у вигляді діаграм UML. Приклад наведено на рисунку 2.

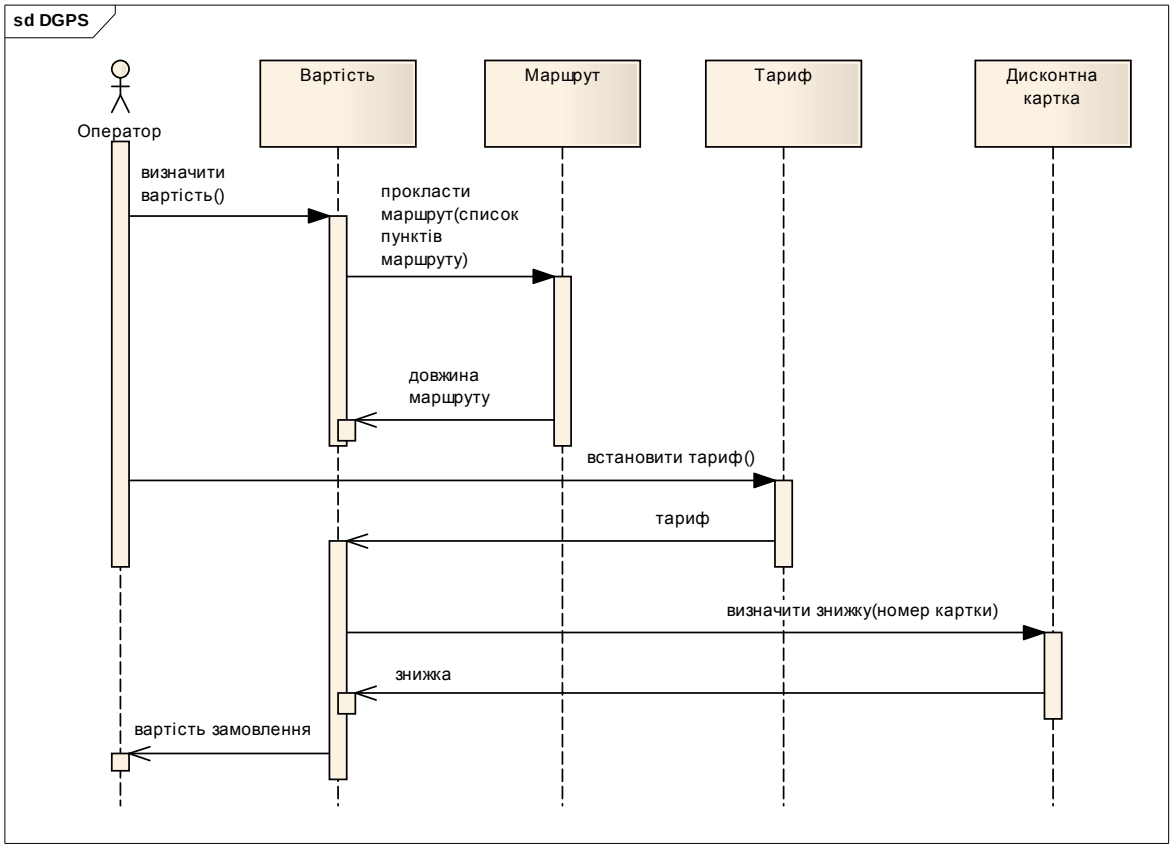

Рис. 2. Приклад діаграми діяльності, яка реалізує сценарій "Розрахувати вартість"

Діаграма діяльності в цьому випадку містить інформацію, яка використовується для розроблення діаграми класів програмної системи та може бути безпосередньо передана розробнику як завдання на виконання.

Висновки і пропозиції. На основі проведеного аналізу застосування методики управління розробкою складних програмних систем пропонується використовувати новий підхід до формалізації проектної документації, що базується на застосуванні суміс-ного типу оформлення специфікації діаграми варіантів використання, що дозволяє:

– підвищити керованість розробкою програмної системи;

– зменшити загальні ризики виконання програмного проекту.

Список використаних джерел

1. Project Management Body of Knowledge (PMBOK), PMI Standard Committee / William R. Duncan, Director of Standards. – USA, 1996.

2. Акименко А. М. Використання UML під час проектування складних програмних систем / А. М. Акименко // Вісник Чернігівського державного технологічного університету. Серія "Технічні науки": наук. зб. – Чернігів: ЧДТУ, 2011. – № 49. – С. 164-170.

3. Акименко А. М. Ідентифікація ризиків програмного проекту / А. М. Акименко // Вісник Чернігівського державного технологічного університету. Серія "Технічні науки": наук. зб. – Чернігів: ЧДТУ, 2011. – № 53. – С. 170-176.