УДК 004.415.2.045 (076.5)

ПІДХІД ДО УТОЧНЕННЯ ПОВЕДІНКОВИХ МОДЕЛЕЙ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ ПАТТЕРНІВ ПРОЕКТУВАННЯ

О.В. Чебанюк

Національний авіаційний університет,
03058, Київ, проспект космонавта Комарова 1,
Тел.: 044-406-76-41,
E-mail: chebanyuk.elena@gmail.com

У роботі представлено підхід до уточнення поведінкових моделей програмного забезпечення (ПЗ), які представляються діаграмами кооперацій. Уточнення поведінкових моделей ПЗ може бути як і окремою операцією Model-Driven Architecture (MDA) та Model-Driven Development (MDD) [1], так і складовою у вирішенні завдань трансформації та верифікації моделей ПЗ [2].

Запропонований підхід базується на співставленні формалізованого опису процесів застосування з шаблонами, що представляють поведінкові складові паттернів проектування, та подальшого уточнення діаграм кооперацій відповідно до цих шаблонів. Поведінкові складові паттернів проектування містять формалізоване представлення функціональних вимог до застосування, які відповідають визначенному паттерну проектування.

Сформовано поведінкові шаблони паттернів проектування «Міст» та «Стан», використовуючи які, спроектовано уточнені діаграми кооперацій.

Систематизовано результати досліджень проблемного домену «Проектування розкрійних схем рулонних матеріалів деталей взуття та шкіргалантереї». Представлено специфікацію вимог до бібліотеки класів, що вирішує такі завдання цього проблемного домену, як побудова еквідістанти (образу деталі), укладок, розкладок, секцій та розкрійних схем.

Продемонстровано приклад уточнення поведінкової моделі ПЗ для виконання завдання побудови щільних укладок розкрійних схем деталей взуття та шкіргалантереї із використанням формалізованого аналітичного представлення уточнених діаграм кооперацій.

An approach to behavioral software models refinement is proposed in this paper. Behavioral software models are represented as UML collaboration diagrams. The operation of behavioral software models refinement can be both executed as a separate operation of Model-Driven Architecture (MDA) and Model-Driven Development (MDD) approaches and as a constituent of other technics which require software models transformation or verification [2].

The proposed approach is based on matching of applications' formalized process description with behavioral constituents of design patterns. Behavioral constituents of design patterns contain formalized representation of application functional requirements that are corresponded with some design pattern.

The behavioral design patterns constituents for patterns "Bridge" and "State" are formed. Using this constituents refinement collaboration diagrams are designed.

The results of problem domain explorations "Designing of cutting schemas for shoe and leather good details" are systematized. The requirements specification to class library that solves such tasks of the problem domain as designing of convex hull (detail representation), layings, layouts, sections, and cutting schemas is presented.

The example of behavioral software models elicitation for solving task of designing laying for shoe and leather good with using formalized analytical representation is represented.

Актуальність

У життєвих циклів розробки ПЗ, які характеризуються послідовним перебігом процесів (RUP та водоспад) процедури виправлення помилок та зміни архітектурних рішень після остаточного визначення вимог до ПЗ є такими, що вимагають багато часу та коштів.

Тому більшого поширення набувають інкреметно-ітеративні підходи розробки ПЗ [3]. У таких підходах весь час розробки ділиться на ітерації (тривалість ітерації зазвичай два тижні) та функціональність ПЗ нарощується поступово при виконанні завдань ітерації. Прикладами інкрементно-ітеративних підходів розробки ПЗ є життєві цикли сімейства Agile [3].

Характерною рисою розробки ПЗ відповідно до методології Agile є можливість змінювати або уточнювати вимоги до ПЗ та артефакти програмної розробки на кожній ітерації. Під ***артефактами програмної розробки*** [4] розуміють будь-яку річ, яка має відношення до розробки ПЗ. Прикладами артефактів є: документація, архітектурні рішення, сценарії тестування, специфікація вимог тощо.

За умови частої зміни вимог до ПЗ артефакти програмної розробки зручно замінювати моделями програмного забезпечення.

Більшість моделей, які використовуються у процесі розробки програмного забезпечення, можуть бути поділені на статичні та динамічні (поведінкові). Класифікацію моделей ПЗ представлено у роботах [2, 5].

Статичні моделі ПЗ відображають його структуру, а саме: об'єкти, взаємозв'язки між ними, атрибути та операції. До таких моделей ПЗ належать UML діаграми класів, компонентів та пакетів [5].

Динамічні або поведінкові моделі ПЗ відображають поведінку програмних систем показуючи змін станів об'єктів, їх взаємодію, процеси та потоки даних. До таких моделей ПЗ належать UML діаграми кооперацій, послідовностей, діяльності та станів [5].

Зміна вимог до ПЗ спричиняє зміну поведінкових моделей ПЗ, які в свою чергу містять вихідну інформацію для проектування та уточнення багатьох артефактів програмної розробки [1].

Складність отримати поведінкову модель ПЗ, яка відповідає вимогам доменної інженерії, а саме повнота, достовірність та непротирічність, обумовлює використання інкрементно-ітеративного підходу для проектування таких моделей ПЗ [2]. Зазвичай процедура отримання поведінкової моделі ПЗ відбувається у два етапи. Перший – отримання приблизної моделі при аналізі функціональних вимог. Наступний - подальше уточнення моделей ПЗ.

Використання запропонованого підходу, який дозволяє отримати якісні поведінкові моделі ПЗ, дозволить більш ефективно вирішувати наступні завдання:

- синхронізувати зміни поведінкових моделей з іншими артефактами програмної розробки;
- уточнювати поведінкові моделі при застосуванні методів трансформації моделей ПЗ як з моделей верхніх рівнів (CIM моделі) [1] у моделі середнього рівня (PIM моделі) [1], так і навпаки;
- проектувати MDE артефакти, які будуть застосовуватися для повторного використання (наприклад пр. доменному аналізі);
- уточнювати вимоги до програмного забезпечення.

Постановка задачі. Розробити підхід до уточнення поведінкових моделей ПЗ при зміні артефактів програмної розробки.

Вимоги до результату поведінкові моделі, отримані відповідно до запропонованого підходу, повинні мати мінімальну кількість об'єктів та зв'язків між ними і водночас відображати всю актуальну інформацію про процеси ПЗ та задовольняти вимогам доменної інженерії.

Об'єкт дослідження – процес уточнення поведінкових моделей ПЗ.

Предмет дослідження – характеристики та властивості поведінкових моделей ПЗ.

Огляд робіт

Залучення технік та методів підходів MDA та MDE у процес розробки ПЗ є передумовою появи низки робіт, присвячених уточненню статичних та поведінкових моделей ПЗ. Операції уточнення моделей базуються на використані різного типу шаблонів, за якими проводиться модифікація UML діаграм. У роботі [7] представлено класифікацію паттернів проектування та запропоновано механізм їх використання для уточнення структури діаграм класів. Для цього сформовані правила, які дозволяють визначити підмножину зі всіх складових діаграм класів, які підлягають уточненню при виконанні певної операції уточнення. У роботі також запропоновано метрики, що дозволяють оцінити ефективність результатів уточнення моделей. У роботі [8] представлено метод, який дозволяє оцінити ефективність перетворень діаграм класів на різних ітераціях розробки ПЗ. Цей метод може використовуватися при веденні історії ітерацій розробки ПЗ та оцінки ефективності повторного використання архітектурних рішень. Запропонований підхід базується на використанні мови Notation Business Process (BPMN) для моделювання бізнес-процесів при уточненні вимог до ПЗ. Визначено характеристики якості вимог до ПЗ, здійснена їх порівняльна оцінка. Визначено оцінку впливу кожного критерію на якість вимоги у цілому. Для візуалізації вимог та процесів теж використовується BPMN.

Представлена робота є продовженням робіт [9–12]. У роботі [10] було представлено метамову опису процесів проблемного домену. За допомогою метамови представляється формалізований опис вимог до програмного забезпечення та поведінкових складових паттернів проектування. У роботі [11] запропоновано алгебру опису статичних моделей ПЗ. За допомогою алгебри опису статичних моделей ПЗ визначено функціональність об'єктів діаграм кооперацій. У роботах [9, 12] досліджено процеси проблемного домену «Проектування розкрійних схем рулонних матеріалів» та спроектовано поведінкові моделі процесів, що представляються діаграмами кооперацій.

Обґрунтування вибору засобів уточнення поведінкових моделей ПЗ

Паттерни проектування містять одночасно опис функціональних вимог до ПЗ та архітектурних рішень [7]. Визначимо складові паттернів проектування, що дозволяють описати архітектурні рішення як ***статичні***. Відповідно, складові паттернів проектування, що характеризують функціональні вимоги до програмного забезпечення та бізнес процеси, визначимо як ***динамічні***.

Відповідно, при використанні інкрементно-ітеративних життєвих циклів розробки ПЗ, залучення технік та методів співставлення функціональних вимог паттернам проектування у процес розробки ПЗ, дозволяє пов'язати функціональні вимоги з фрагментами поведінкових моделей ПЗ, уточнити відповідні поведінкові моделі ПЗ.

Проектування та уточнення поведінкової моделі ПЗ відбувається у два етапи. Перший етап – створення поведінкової моделі у першому наближенні із використанням сценаріїв виконання процесів відповідно до специфікації вимог. Поведінкова модель представляється діаграмою кооперацій. Далі виконується формальний опис процесів, які зазначені у специфікації. Для формального опису процесів ПЗ та збереження інформації про них використаємо метамову опису процесів проблемного домену, яка представлена у роботі [10].

Наступний етап – уточнення поведінкової моделі ПЗ відповідно до шаблонів діаграм кооперацій, що відповідають паттернам проектування. Формальний опис процесів ПЗ співставляється з поведінковими складовими паттернів проектування (рис. 1). Якщо формальний опис процесу ПЗ відповідає динамічній складовій певного паттерну проектування, то відповідна поведінкова модель уточнюється з використанням шаблону діаграми кооперацій, що відповідає цьому паттерну.

Схематично процес уточнення поведінкових моделей ПЗ показано на рис. 1.

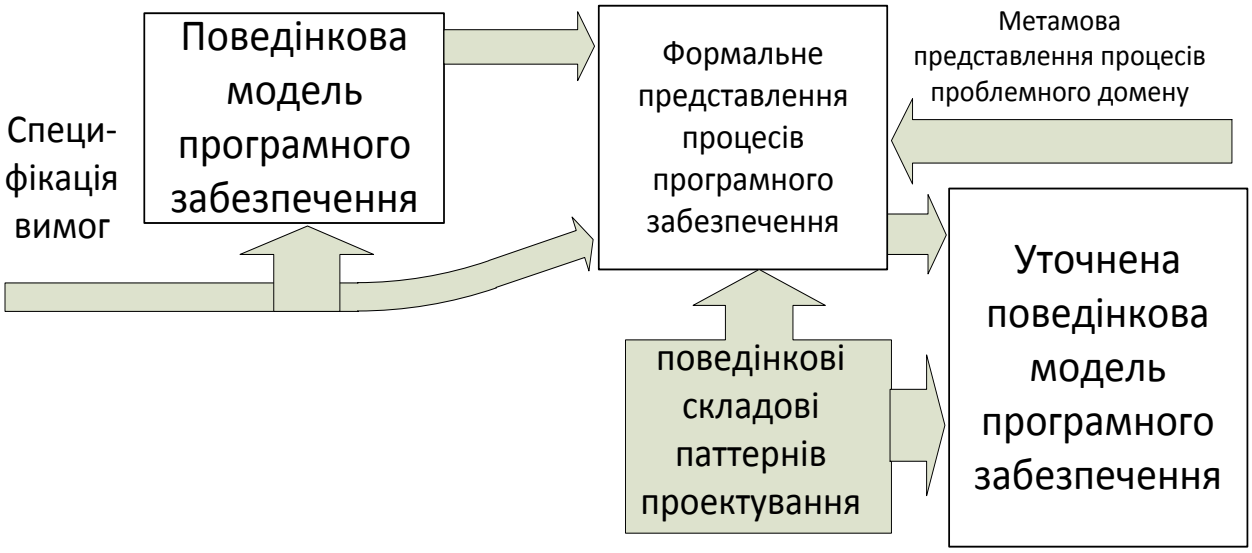

Рис. 1. Схема уточнення поведінкових моделей ПЗ із використання паттернів проектування.

Визначимо етапи уточнення поведінкових моделей ПЗ відповідно до запропонованого підходу.

1. Проектується поведінкова модель програмного забезпечення, яка представляється діаграмою кооперацій із використанням специфікації вимог.

2. Виконується опис процесів ПЗ, які представлено у цій поведінковій моделі у термінах метамови представлення процесів проблемного домену.

3. Співставляються динамічні складові паттернів проектування та формальний опис процесів ПЗ.

4. Коригується як формальний опис процесів ПЗ, так і поведінкова модель, що відображає ці процеси.

Для ефективної реалізації такого підходу уточнення моделей потрібно підготувати наступну вихідну інформацію:

- визначити ознаки, які дозволять співставити функціональні вимоги до ПЗ з певним паттерном проектування;
- сформувати динамічні компоненти паттернів проектування, представивши їх у термінах метамови опису процесів проблемного домену;
- визначити шаблони модифікованих фрагментів діаграм кооперацій для кожного паттерну проектування, які зберігають всю вихідну інформацію про процеси проблемного домену.

Формування опису поведінкових складових паттернів проектування та шаблонів уточнених діаграм кооперацій

Проілюструємо процес підготовки такої інформації для поведінкових паттернів проектування «Міст» та «Стан».

Розглянемо паттерн проектування «Міст». Визначимо ознаки того, що потрібно використовувати паттерн проектування «Міст» при описі вимог у специфікації.

Є декілька об'єктів (об'єкти o_1 та o_2 на рис. 1), які мають як спільну, так і відмінну функціональність. Ці об'єкти виконують певні завдання, використовуючи алгоритми, які мають як спільні так, і відмінні етапи. Для виконання цих завдань об'єкти o_1 та o_2 можуть використовувати як різні алгоритми (функціональність алгоритмів закладена у об'єктах $a\lg_1$ та $a\lg_2$ рис. 1), так і інші об'єкти діаграми кооперацій (об'єкти ob_1 та ob_2 на рис. 1). Порядок виконання цих завдань і їх кількість може змінюватися. Такі функціональні вимоги відповідають діаграмі кооперацій на рис. 2.

Діаграма кооперацій ілюструє три можливі сценарії виконання процесів.

Послідовність операцій 1, 2, 3 та 4 відповідає сценарію, коли певний об'єкт використовує певний алгоритм (об'єкт alg) та інший об'єкт, а саме ob_1 для виконання визначеної задачі.

Послідовність операцій 5, 6, 7 та 8 відповідає сценарію, коли певний об'єкт використовує властивості та методи різних об'єктів для виконання певної задачі.

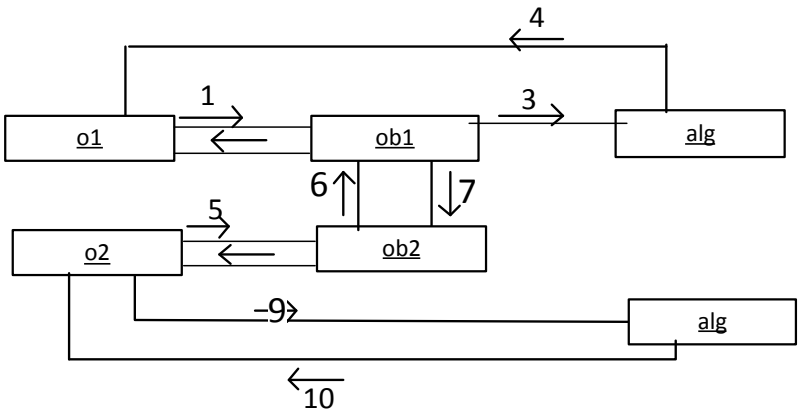

Рис. 2. Фрагмент діаграми кооперації, що відповідає паттерну проектування «Міст»

Послідовність операцій 9 та 10 відповідає сценарію, коли певний об'єкт використовує деякий алгоритм для виконання визначеної задачі.

Для формулювання динамічних компонентів паттерну проектування «Міст» визначимо наступні колекції об'єктів: O – колекція об'єктів, які виконують певні дії за допомогою як різних алгоритмів, так і інших об'єктів; $a\lg$ – колекція об'єктів, функціональність яких дозволяє реалізувати алгоритми виконання цих дій; Ob – колекція об'єктів, функціональність яких потрібна для вирішення задач об'єктів, що є елементами колекції O.

$$\begin{cases} con(O(O_1)) \to O(O_1) \cup ref(a\lg[j]) \to H_1 \\ F(O(O_1))^{ref} = F(O(O_1)) \cup F(a\lg[j]) \\ H_1 = \{m_0^{a\lg[j]} \to m_1^{a\lg[j]} \to \ldots \to m_n^{a\lg[j]}\} \\ \ldots \\ con(O(O_t)) \to O(O_t) \cup ref(Ob[k]) \to H_t \\ F(O(O_t))^{ref} = F(O(O_t)) \cup F(a\lg[i]) \\ H_t = \{m_0^{Ob[i]} \to m_1^{Ob[i]} \to,\ldots,\to m_n^{Ob[i]}\}, \end{cases} \quad (1)$$

Наведемо пояснення щодо динамічних складових паттерну проектування «Міст» (1). Розглянемо два елементи колекції O, які позначимо як O_i та O_j. Так як міст дозволяє змінювати як абстракцію, так і реалізацію, то відповідно до першого (другого) сценарію створюється об'єкт O_i (O_j) з посиланням на об'єкт, який інкапсулює собою певний алгоритм з колекції $a\lg$ або об'єкт з колекції Ob (1). Позначимо алгоритм з колекції $a\lg$, як $a\lg[j]$, а об'єкт з колекції Ob, як $Ob[k]$. Відповідно до нотації метамови процес створення об'єкта O_i з посиланням на об'єкт $a\lg[j]$ запишеться наступним чином:

$$con(O(O_i)) \to O(O_i) \cup ref(a\lg[j]).$$

Відповідно, функціональність об'єкта O_i розширюється на функціональність об'єкта $a\lg[j]$. У нотації алгебри опису статичних моделей ПЗ це запишеться наступним чином:

$$F(O(O_1))^{ref} = F(O(O_1)) \cup F(a\lg[j]).$$

Далі об'єкт O_i виконує набір операцій за допомогою викликів певних методів об'єкта $a\lg[j]$. Так, як послідовність операцій має значення, то використовується позначення метамови H. Приклад $H_1 = \{m_0^{a\lg[j]_0} \to m_1^{a\lg[j]_1} \to \ldots \to m_n^{a\lg[j]_t}\}$ ілюструє, що для виконання операцій з послідовності H_1 викликаються методи $m_i^{a\lg[j]}$, $i = 1, \ldots, n$, (n – кількість методів у ланцюжку H_1) об'єкта $a\lg[j]$ у певній послідовності.

Аналогічні міркування приводяться для пояснення другого сценарію паттерну проектування «Міст» з використанням об'єкта, що є елементом колекції Ob.

Підготуємо шаблон модифікованої діаграми кооперацій для паттерну проектування «Міст».

Після аналізу виразу (1) та діаграми на рис. 2 формуємо уточнену діаграму кооперацій, що відповідає функціональним вимогам паттерну «Міст».

Наведемо рекомендації за співставленням функціональних вимог, складовим діаграми кооперацій на рис. 3. $a[i]$ – один об'єкт із колекції об'єктів O. Він виконує певні операції за допомогою або різних алгоритмів ($a\lg$ – колекція таких алгоритмів), або використовуючи інші об'єкти (Ob – колекція таких об'єктів).

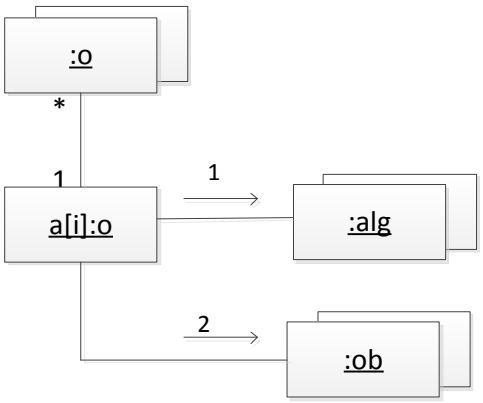

Рис. 3. Уточнена діаграма кооперацій, що відповідає паттерну проектування «Міст»

Для проектування такої діаграми кооперації під час аналізу функціональних вимог необхідно визначити які сутності проблемного домену, відповідають колекціям об'єктів O, Ob та $a\lg$.

Визначимо функціональні ознаки паттерну проектування «Стан». Об'єкт (Ob_1 на рис. 3) може виконувати визначені дії за умови знаходження його у певному стані (на рис. 4 перша умова – $cond_1$, друга умова – $cond_2$ і .т. д.). При переході об'єкта із стану у стан можуть виконуватися певні дії ($action_1$ – перша дія, $action_2$ – друга дія і т. п.), для виконання яких задіяна функціональність певних об'єктів. Фрагмент діаграми кооперацій, який представляє такі функціональні вимоги показано на рис. 4.

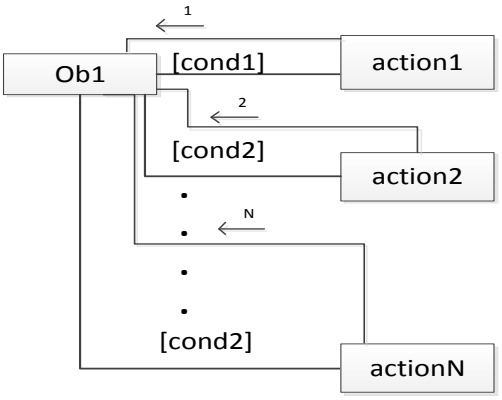

Рис. 4. Фрагмент діаграми кооперацій, що відповідає паттерну проектування «Стан»

Отже сформуємо формальний опис процесів, який відповідає паттерну проектування «Стан».

$$\begin{cases} con(Ob_1(O)) \to \Phi\ , \\ \Phi = \{m_0^{action_i}, m_1^{action_j}, \ldots, m_t^{action_k}\},\ \ i, j, k = 1, \ldots, n. \end{cases} \tag{2}$$

Наведемо пояснення виразу (2). Створюється об'єкт Ob_1, який може виконувати різні операції, перебуваючи у різних станах. Процес виконання різних операцій характеризується викликом різних методів з колекції action. Так як порядок виконання операцій довільний то використовується позначення Ф метамови. Після аналізу виразу (2) та діаграми на рис. 3. формуємо уточнену діаграму кооперацій, що відповідає функціональним вимогам паттерну проектування «Стан».

Пояснимо діаграму на рис. 5. Для кожного елемента $a[i]$, з колекції *action* визначимо, які дії буде виконувати об'єкт Ob_1, перебуваючи у певному стані, та умови *(cond*[*i*]*)*, при яких відбувається зміна стану об'єкта (повідомлення changeState() на рис. 4). Кількість станів об'єкта визначає кількість елементів у колекції *action.*

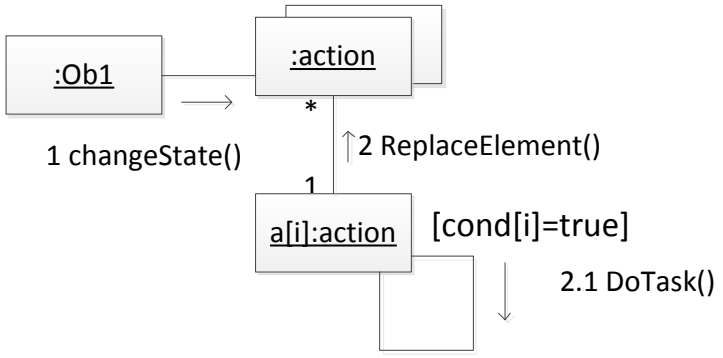

Рис. 5. Уточнена діаграма кооперацій, що відповідає паттерну проектування «Стан»

Використання запропонованого підходу для уточнення поведінкових моделей ПЗ

Представимо приклад використання запропонованого підходу для уточнення поведінкових моделей ПЗ проблемного домену «Проектування розкрійних схем для деталей взуття та шкіргалантереї», які представлені діаграмами кооперацій.

1. Проектування поведінкової моделі ПЗ та специфікації вимог.

Скористуємося поведінковими моделями ПЗ, що представлені у роботах [11, 12].

Представимо специфікацію вимог для проблемного домену «Побудова розкрійних схем рулонних матеріалів деталей взуття та шкіргалантереї» із зазначенням посилань на роботи, де проаналізовано процеси проблемного домену (таблиця).

Таблиця. Специфікація вимог для проектування бібліотеки класів проблемного домену «побудова розкрійних схем рулонних матеріалів»

Код функціональної вимоги	Опис функціональної вимоги
1	2
F1	Зчитування інформації про деталь із *.dgt формату та формування образу деталі [13]
F1.1	Формування інформації про групову деталь шляхом об'єднання контурів декількох деталей [14]
F2	Нанесення декоративних елементів на деталь [14]
F 2.1	Збереження інформації про ключові характеристики декоративних елементів, нанесених на деталь [14]
F 2.2	Розробка формату файлів для збереження інформації про розкрійні схеми з нанесенеми на деталі декоративними елементами [15]
F 2.3	Розробка формату файлу, що дозволяє збереження інформації про координати зовнішніх контурів декоративних елементів [15]
F 3	Проектування щільних укладок деталей взуття та шкіргалантереї [11, 16, 17, 18, 19, 20]
F 3.1	Проектування щільних укладок для однотипних деталей [17, 18]
F 3.1.1.	Визначення векторів одинарної решітки [17, 18]
F 3.2	Проектування щільних укладок для двох різних типів деталей [18, 19]
1	2
F 3.2.1	Визначення векторів подвійної решітки [18]
F 3.2.2	Побудова годографа вектор-функції щільного розміщення для подвійних укладок [18]
F 3.3	Проектування щільних укладок для групових деталей [21]
F 3.3.1	Об'єднання деталей у групову деталь [21]
F 3.3.2	Побудова годографа вектор-функції щільного розміщення для укладок групових деталей [21]
F 3.4	Визначення лінійних ефектів для укладок деталей шкіргалантереї [11, 13, 20]

Завершення таблиці

1	2
F4	Проектування розкладок [13, 22, 23, 24]
F 4.1	Задача „Розкладка А” Знайти таке системне розміщення двох многокутників, що мають довільну конфігурацію зовнішніх контурів на матеріалі прямокутної форми з їх змінною орієнтацією в заданих межах, яке забезпечує максимальне значення показника використання матеріалу із врахуванням інших технологічних умов та обмежень [23]
F 4.2.	Задача „Розкладка В” Знайти таке системне розміщення двох многокутників, що мають довільну конфігурацію зовнішніх контурів на матеріалі прямокутної форми без обмежень на їх орієнтацію на матеріалі, яке забезпечує максимальне значення показника використання матеріалу із врахуванням інших технологічних умов та обмежень [23]
F 4.3.	Поворот решіток при проектуванні розкладок [24]
F5	Проектування секцій [24]. Постановка задачі: знайти таке системне розміщення деталей взуття на матеріалі прямокутної форми із шириною *Sh* для фіксованої орієнтації деталей, яке забезпечує максимальне значення показника використання матеріалу із врахуванням потреби в деталях та технологічних умов та обмежень
F 5.1	Щільне суміщення секцій [23]
F 5.2	Пошук оптимальної перестановки секцій [23]
F6	Розрахунок траєкторії різального апарату при розкрої струменем води або лазера [25]

Для більш детального дослідження процесів використовуються математичні та технологічні постановки задач укладка, розкладка та секція, які представлені у роботах [20, 23, 24].

2. Використовуючи цю поведінкову модель ПЗ та специфікацію вимог виконується опис процесів ПЗ у термінах метамови представлення процесів проблемного домену.

Представимо опис процесів які виконуються при проектуванні щільних укладок, у термінах метамови опису процесів.

2.1. Формальний опис процесів побудови щільних укладок деталей взуття охоплює вимоги специфікації F 3.1, F3.2, F3.3 [13, 22–24].

Розглянемо класи, які використовуються для побудови щільних укладок, а саме: деталь (клас $C(Detail)$), об’єднана деталь або груповий об’єкт(клас $C(join\_Detail)$), ГВФЩР (клас $C(Godograph)_1$), решітка (клас $C(Grid)$) та укладка (клас $C(Laying)$). Функціональність всіх класів крім класу годограф розглянута у роботах [9, 12]. Клас $C(Godograph)_1$ відповідає за побудову годографа вектор функції щільного розміщення (ГВФЩР) для однотипних деталей, клас $C(Godograph)_2$ відповідно за побудову різнотипних, клас $C(join\_Detail)$ описує функціональність об’єкта, що представляє об’єднану деталь [21].

Використовуючи спефицікацію вимог розглянемо завдання, які потрібно виконати для побудови щільних укладок відповідно до специфікації вимог (таблиця).

2.1.1. Проектування щільних укладок для однотипних деталей [17, 18].

2.1.1.1. Визначення векторів одинарної та подвійної решіток.

Вектори решітки для укладок будуються за допомогою ГВФЩР шляхом аналізу взаємного розташування деталей [18]. У термінах метамови клас решітка, а саме: $C(Grid)$, використовує функціональність класу годограф. Одинарні решітки будуються для одного типу деталей, використовуючи об’єкти класу $C(Detail)_1$. У термінах нотації метамови опису процесів проблемного домену операції створення різних типів об’єктів ГВФЩР запишемо наступним чином:

$$con(C(Godograph_1)) \to Godograph_1(O) \cup ref\,(Detail_1(O))$$ для одинарних решіток

та

$$con(C(Godograph_2)) \to Godograph_2(O) \cup ref\,(Detail_1(O), Detail_2(O))$$ для подвійних.

Для побудови решітки об’єкт класу решітка створюється із посиланням на об’єкт класу годограф. Отже функціональність класу $C(Grid)$ розширюється на функціональність класу, що містить алгоритм побудови ГВФЩР різних типів. У нотації алгебри опису статичних моделей ПЗ це запишеться наступним чином:

$$F(C(Grid))^{ref} = F(Grid(O)) \cup F(Godograph_1(O))$$ або

$$F(C(Grid))^{ref} = F(Grid(O)) \cup F(Godograph_2(O)).$$

Укладки будуються на векторах решітки. Різні укладки використовують решітки та деталі різних типів. Відповідно при створенні класу укладка ($C(Laying)$) використовуються посилання на об’єкт класу решітка та деталі різних типів, які входять до цієї укладки. Так як посилання на об’єкти класів деталей вже є у класі решітка, то процедура створення об’єкта класу «Укладка» запишеться наступним чином:

$$con(C(Laying)) \to Laying(O) \cup ref\,(Grid(O)).$$

Відповідно для побудови різних типів укладок використовуються різні алгоритми, які мають як спільні так і відмінні етапи. Позначимо послідовність дій, які виконуються для побудови щільних укладок одного типу деталей як $H^{Grid}{}_1$, двох типів деталей $H^{Grid}{}_2$. Дії, що виконуються при побудові обох типів укладок представлено у роботі [18].

Операція генерації множини допустимих решіток H^{best_Grid} однакова для всіх типів укладок [18]. Вона описана у роботах [11, 18].

Розглянемо клас груповий об'єкт. Так як груповий об'єкт представляє собою об'єднання декількох деталей, то при створенні цього класу використовуються об'єкти класів $C(Detail)_1$, $C(Detail)_2$ та $C(Detail)_3$. Решта операцій, що дозволяють побудувати щільну укладку для групового об'єкта збігається з послідовністю операцій H_2.

Використовуючи (1) та нотацію метамови і алгебру опису статичних моделей ПЗ представимо формальний опис процесів ПЗ:

$$\begin{cases}
con(C(Detail_1)) \to Detail_1(O) \\
con(C(Godograph_1)) \to Godograph_1(O) \cup ref\,(Detail_1(O)) \\
con(C(Grid_1)) \to Grid(O) \cup ref\,(Godograph_1(O)) \\
F(C(Grid))^{ref} = F(Grid_1(O)) \cup F(Godograph_1(O)) \to H^{Grid}{}_1 \\
H^{Grid}{}_1 = \{\beta(a1)_0^{Grid} \mapsto \beta(alpha)_1^{Grid} \mapsto \beta(a2)_2^{Grid} \mapsto \beta(betta)_3^{Grid}\} \\
\\
con(C(Detail_2)) \to Detail_2(O) \\
con(C(Godograph_2)) \to Godograph_2(O) \cup ref\,(Detail_1(O), Detail_2(O)) \\
con(C(Grid_2)) \to Grid(O) \cup ref\,(Godograph_2(O)) \\
F(C(Grid))^{ref} = F(Grid_2(O)) \cup F(Godograph_2(O)) \\
con(C(Grid)) \to Grid(O) \cup ref\,(Godograph_2(O)) \to H^{Grid}{}_2 \\
H^{Grid}{}_2 = \{\beta(a1)_0^{Grid} \mapsto \beta(alpha)_1^{Grid} \mapsto \beta(a2)_2^{Grid} \mapsto \beta(betta)_3^{Grid} \mapsto \beta(q)_4^{Grid}\} \\
\\
con(C(Detail_3)) \to Detail_3(O) \\
con(C(join_Detail)) \to join_Detail(O) \cup ref\,(Detail_1(O), Detail_2(O), Detail_3(O)) \to H_2{}^{Grid} \\
\\
H^{Grid}{}_1(H^{Grid}{}_2) \to con(C(Laying)) \to Laying(O) \cup ref\,(Grid(O)) \to H^{best_Grid} \\
F(C(Laying))^{ref} = F(Laying(O)) \cup F(Grid(O)) \\
H^{best_Grid} = \{\beta(estimate)_0{}^{(Laying(O))} \mapsto \beta(add)_1{}^{(Laying(O))}\}
\end{cases} \quad (3)$$

3. Співставимо формальний опис процесів ПЗ з динамічними складовими паттернів проектування. Для побудови різних типів решіток використовуються різні алгоритми, які мають як відмінні та схожі ознаки. Відповідно укладки будуються із використанням різних типів об'єктів (решітка для однотипних деталей, для двох різних типів деталей та групових деталей), а алгоритми побудови ГВФЩР для різних типів укладок також мають як спільні, так і відмінні операції.

Отже, функціональні вимоги до застосування, що будує щільні укладки, збігаються з ознаками паттерну проектування «Міст», а саме: використання об'єктів, що мають як спільні, так і відмінні властивості алгоритмів, що виконують визначену задачу та мають певні спільні операції.

Аналізуючи вирази (1) та (3) визначаємо об'єкти уточненої діаграми кооперацій, що співставляються з сутностями проблемного домену.

O – об'єкт укладка. Для побудови укладок використовується як алгоритми, що мають, як спільні, так і відмінні етапи та інші об'єкти.

$a\lg$ – колекція алгоритмів, які використовуються при побудові щільних укладок різних типів, що містять як спільні, так і відмінні етапи. Відповідно $a\lg[0]$ – алгоритм побудови укладок для однотипних деталей, $a\lg[1]$ – алгоритм побудови укладок для двох типів деталей, $a\lg[2]$ – алгоритм побудови укладок для групових деталей.

Ob – колекція об'єктів, які використовуються при побудові укладок.

$Ob[0]$ – решітка для укладок, що складаються з деталей одного типу, $Ob[1]$ – відповідно двох та $Ob[2]$ – решітка для укладок, які містять групові об'єкти.

4. Уточнимо поведінкову модель ПЗ. Уточнена діаграма кооперації процесу побудови щільних укладок для двох типів деталей відповідає діаграмі, представленій на рис. 3. Кожен алгоритм з колекції alg цієї діаграми може бути представлений окремою поведінковою моделлю [24], яка ілюструє деталі реалізації.

Висновки

Огляд робіт, присвячених уточненню моделей, показує, що більшість методів уточнення моделей ПЗ спрямовано на дослідження та покращення характеристик статичних моделей ПЗ [1, 2, 7, 8]. Методи уточнення динамічних моделей ПЗ вимагають залучення таких засобів, як ВРМН або методів чи технік уточнення вимог і включають досить трудомісткі операції. Часто ефективне використання таких методів вимагає певної попередньої або супровідної роботи (формування артефактів доменної інженерії або доменний аналіз, можливе аналіз бізнес процесів тощо). Такий підхід є не дуже ефективним у інкрементно-ітеративних життєвих циклах розробки ПЗ, через те, що під час проведення трудомістких операцій вже можуть змінитися вимоги.

На відміну від методу уточнення поведінкових моделей, який представлено у [8] підхід, запропонований у цій роботі, дозволяє уточнювати поведінкові моделі, зменшуючи трудомісткість операції їх уточнення. Так як у інкрементно-ітеративних життєвих циклах розробки ПЗ таку операцію потрібно проводити доволі часто, то як вихідну інформацію для уточнення поведінкових моделей ПЗ можливо використати або формальний опис процесів, або одразу співставити функціональні вимоги поведінковим складовим паттернів. У порівнянні із методами, що запропоновані у [2, 7] трудомісткість уточнення діаграм зменшується, а наперед визначені готові шаблони поведінкових діаграм дозволяють спростити процедуру візуалізації уточнених поведінкових моделей ПЗ.

Також запропонований підхід може використовуватися, як етап методів MDA, які вирішують задачу перетворення моделей.

1. *Beatriz M., Pereira1 J., Giachetti G., Hermosilla F., Estefan S.* A General Framework for the Development of MDD Projects // Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development Spain, Barselona. – 2013. – P. 258–264.
2. *Stephan M., Cordy J.* A Survey of Model Comparison Approaches and Applications // Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development. Spain, Barselona. – 2013. – P. 265–277.
3. *Stavru S.,Krasteva I. And Ilieva S.* Challenges of Model-Driven Modernization-An Agile Perspective // In Proceedings of the 1st international conference on model-Driven Engineering and Software Development. Spain, Barselona. – 2013, P. 219–230.
4. *Tamlr Kllnger, Peri L.* System and method for publication classification automatically determining relationships between software sources Patent Aug. 22, 2013.
5. *Gupta S., Singla J.* S A component-based approach for test case generation // International Journal of Information Technology 5.2 2012. – P. 239–243.
6. *http:*//www.omg.org/mda/faq_mda.htm
7. *Lano K. and Kolahdouz-Rahimi S.* Optimising Model-transformations using Design Patterns. In Proceedings of the 1st International Conferenceon Model-Driven Engineering and Software Development. Spain, Barselona. – 2013. – P. 77–82.
8. *Störrle H.* Making Sense to Modelers-Presenting UML Class Model Differences in Prose // In Proceedings of the1st International conference on model-driven engineering and software development Spain, Barselona. – 2013. – P. 39–48.
9. *Chebanyuk E.* Algebra describing software static models // International journal Information technologies and knowledge. – 2013. – № 1. – Vol. 7. – P. 83–93.
10. *Chebanyuk E.* Metalanguage for description problem domain processes // Міжнародна конференція "Сучасна інформатика. Проблеми, досягнення та перспективи розвитку".,11-13 вересня 2013. – К. Інститут програмних систем. – 2013. – С. 63–64.
11. *Chebanyuk O.V., Chuprinka V.I.* One approach of constructing problem domain metamodel. Інженерія програмного забезпечення. – 2011. – № 4 (8). – С. 13–21.
12. *Chebanyuk E.* Method of domain models designing. International models and analysis. – 2014, N 1. –Vol. 3.
13. *Чупринка В.І., Хоменко О.О., Свістунова Л.Т.* Комплексний підхід до розв'язання задачі щільного розміщення об'єктів складної форми на площині // Проблеми програмування. – 2010. – № 2-3. – С. 621–628.
14. *Чупринка В.И., Чебанюк Е.В.* Алгоритм сохранения информации о декоративных элементах на деталях обуви // Техническое регулирование: базовая основа качества товаров и услуг: сб. науч. трудов. – Шахты (РФ): ЮРГУЄС, 2009. – С. 70–73.
15. *Чебанюк О.В., Чупринка В.І.* Математичне та програмне забезпечення побудови розкрійних схем з декоративними елементами // Вісник Східноукраїнського національного університету імені Володимира Даля. –2010. – № 9 (151). – С. 194–199.
16. *Чупринка В.И., Чебанюк Е.В.* Математическая модель оценки эффективности раскладок при построении раскройных схем рулонных материалов// Техническое регулирование: базовая основа качества товаров и услуг.: зб. науч. трудов – Шахти ЮРГУЄС, 2012. – С. 193–198.
17. *Чупринка В.І.; Пінчук А.В.; Чебанюк О.В.* Автоматизована підготовка інформації про схеми розкрою рулонних матеріалів на однакові плоскі геометричні об'єкти // Вісник Східноукраїнського національного університету імені Володимира Даля. –2008. – № 5(139). – С. 194–199.
18. *Чебанюк О.В., Чупринка В.І.* Методика автоматичної побу дови розкрійних схем для двох видів плоских геометричних об'єктів // Проблеми програмування. − 2008. – № 2–3. − С. 730−734.
19. *Чупринка В.І., Чебанюк О.В.* Метод програмного проектування найщільніших решітчастих укладок // Проблеми програмування. – 2010. – № 2–3. – С. 629–635.
20. *Чупринка В.И., Мурженко В.С., Омельченко П.В.* Автоматизированное проектирование схем раскроя при прямоугольно-гнездовом методе раскроя // Международный сборник научных трудов «Техническое регулирование: базовая основа качества товаров и услуг», – Шахты: ЮРГУЭС, 2013. – С. 70–72.
21. *Чупринка В.І.* Автоматизоване проектування раціональних схем розкрою рулонних матеріалів на деталі виробів шкіргалантереї// Інформаційна безпека. – 2011. – 7: 2. – С. 46–50.
22. *Чупринка В.І., Чебанюк О.В.* Алгоритм автоматичної підготовки вихідної інформації для побудови раціональних схем розкрою // Вісник Київського національного університету технологій та дизайну. – 2006. –№ 6(32). – С. 182–186.
23. *Чупринка В.І., Чебанюк О.В.* Метод автоматичного проектування раціональних схем розкрою на деталі взуття // Проблеми програмування. – 2012. – № 2–3. – С. 168–174.
24. *Чупринка В.І., Чебанюк О.В.* Доменний аналіз методів автоматичної побудови розкрійних схем зі змінним кутом повороту решіток // Вісник Східноукраїнського національного університету імені Володимира Даля. – 2012. – № 9(151). – С. 194–199.
25. *Чупринка В.И., Чебанюк Е.В., Мурженко В.С.* Оптимизация маршрута режущего інструмента при автоматическом раскрое материалов с помощью воды или луча лазера // Междунар. сб. науч. тр. Южно-Рос. гос. ун-та экономики и сервиса. – Шахты. Изд-во ЮРГУЄС, 2011. – С. 93−95.