https://doi.org/10.31891/2219-9365-2024-78-8
УДК 004.4'242.2:004.415.25004.4'242.2:004.415.25

СТИСЛО Тарас
ЗВО «Університет Короля Данила»
https://orcid.org/0000-0002-2377-7985
e-mail: taras.styslo@ukd.edu.ua
СТИСЛО Оксана
ЗВО «Університет Короля Данила»
https://orcid.org/0000-0002-7348-2501
e-mail: oksana.styslo@ukd.edu.ua
ДЕМЧИНА Микола
ЗВО «Університет Короля Данила»
e-mail mykola.demchyna@ukd.edu.ua
БІЛОУС Василь
ЗВО «Університет Короля Данила»
e-mail vasyl.v.bilous@ukd.edu.ua

АРХІТЕКТУРА ТА МЕТОДОЛОГІЯ ІНТЕГРАЦІЙНОГО ФРЕЙМВОРКУ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА ОСНОВІ КОНЦЕПЦІЇ VIEWPOINTS

В роботі представлено структуру ViewPoints для розподіленої багатонапрямленої розробки програмного забезпечення, наведено структурні елементи фреймворку, ViewPoints і шаблони ViewPoint, представлено можливості підтримки ViewPoint-Oriented Software Engineering (VOSE), а також представлено розробку методів у структурі ViewPoints і досліджено як метод розробляється та створюється розробниками і подається як інженерний процес, що впливає на спосіб використання методу.

Ключові слова: інтеграція процесів, об'єднання правил, діаграма даних, агрегація процесів, методологія, архітектура, класифікація, шаблон проєктування.

STYSLO Taras, STYSLO Oksana, DEMCHYNA Mykola, BILOUS Vasyl
King Danylo University

ARCHITECTURE AND METHODOLOGY OF AN INTEGRATION FRAMEWORK FOR SOFTWARE DEVELOPMENT BASED ON THE VIEWPOINTS CONCEPT

In this work, the issue of managing the complexity of the division of problems and integrated, systematic software development is considered. The paper describes the complexity of software development as a "many viewpoint problem" and sets the limits within which the problem can be solved. Attempts to reconcile the desired separation of concerns provided by the framework with the integration required for systematic software development are also considered. This alignment is achieved by using multiple perspectives during development, which ensures integration of the methods by which these perspectives are developed.

The paper presents the ViewPoints framework for distributed multi-directional software development, detailing the structural elements of the framework, ViewPoints, and ViewPoint templates. It demonstrates the capabilities of supporting ViewPoint-Oriented Software Engineering (VOSE) and introduces the development of methods within the ViewPoints structure. The study explores how a method is developed and constructed by developers, presenting it as an engineering process that influences how the method is utilized.

A ViewPoint-oriented approach to method development is also proposed. In the context of the ViewPoints framework, methods consist of ViewPoint templates connected by rules between ViewPoints. The role of the method developer involves selecting appropriate ViewPoint templates that constitute the method, and then describing their individual representation styles and working plans. This method development process can also involve the reuse (and typically adaptation) of existing ViewPoint templates.

Keywords: process integration, rule combination, data diagram, process aggregation, methodology, architecture, classification, design pattern.

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

В даній роботі розглядається питання управління складності поділу проблем та інтегрованій, систематичній розробці програмного забезпечення. Робота описує складність розробки програмного забезпечення у вигляді «проблеми багатьох точок зору» та встановлює межі, в яких цю проблему можна вирішити. Також розглядаються спроби узгодити бажаний поділ проблем, передбачений структурою з інтеграцією, необхідною для систематичної розробки програмного забезпечення. Таке узгодження досягається шляхом використання кількох точок зору під час розробки, що забезпечує інтеграцію методів, за допомогою яких ці точки зору розробляються.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

В роботі [1] описується система Pegasus, яка інтегрує різнорідні бази даних, дозволяючи ефективне управління та доступ до даних з різних джерел в єдиній системі. Цей підхід до інтеграції баз даних може бути співставлений з методами та процесами, що використовуються для індустріалізації розробки програмного забезпечення в Японії, про які йдеться в роботі [2]. Там акцентується увага на підвищенні ефективності та якості розробки.

Додатково, у роботі [3] присвяченій системі SREM, описується розподілена система проектування обчислень, її еволюція та вплив на галузь, що доповнює розуміння про індустріалізацію та ефективність розробки програмного забезпечення. Водночас, методи автоматичного програмування для доменно-специфічних завдань, які розглядаються в роботі [4], включають техніки та підходи для автоматизації процесу програмування, що можуть бути корисними для підвищення продуктивності у розробці програмного забезпечення.

Нарешті, робота [5] надає порівняльний аналіз різних методологій інтеграції схем баз даних, висвітлюючи їхні переваги та недоліки, та пропонує рекомендації щодо вибору методології в залежності від контексту застосування. Це дозволяє більш комплексно підійти до проблеми інтеграції та управління даними, зважаючи на різні підходи та їхні характеристики.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою роботи є: виконання порівняльного аналізу моделей та методів інтеграції та агрегації процесів розробки програмного забезпечення. Основною метою також є розробка нової концепції побудови програмних рішень на основі фреймворку ViewPoints. Суть цієї концепції полягає в тому, щоб відійти від централізованої архітектури до децентралізованої, з метою покращення ефективності та якості розробки програмного забезпечення.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Архітектурна мета фреймворку ViewPoints полягає в тому, щоб відійти від централізованої архітектури до децентралізованої (рис. 1). Термін «архітектура» використовується для опису структурних елементів і комунікаційних принципів фреймворку. Таким чином, можна описувати централізовані архітектури, розподілені архітектури, конвеєрні архітектури тощо.

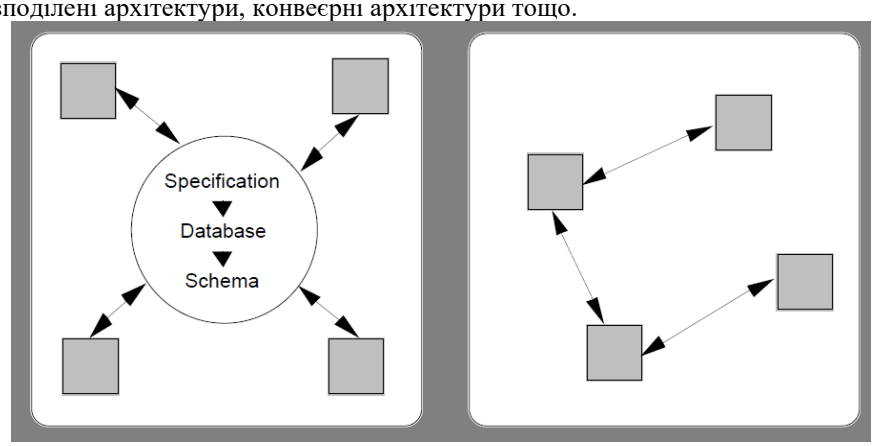

Рис.1. Централізована та децентралізована архітектури

Таким чином, ViewPoints є дистрибутивним хоча це не виключає наявність централізованих ViewPoints, що містять глобальні дані або керуючу інформацію. Однак існує кілька рівнів децентралізації з різними рівнями складності їх досягнення. Наприклад, децентралізація специфікації шляхом її розбиття на більш дрібні, більш керовані модулі є звичним явищем і точно вписується в модель розробки, орієнтовану на ViewPoint. З іншого боку, використання фізично або логічно децентралізованих баз даних для зберігання частин специфікації складніше. Більше того, якщо базові схеми специфікацій у різних базах даних також відрізняються, то керування процесом розробки специфікацій за сучасної технології є проблематичним.

Наш поточний підхід полягає в тому, щоб мати повністю децентралізовану архітектуру для фреймворку ViewPoints і працювати над невирішеними питаннями інтеграції та управління, а не приймати централізовану архітектуру, яку ми потім намагаємося децентралізувати. Ми також віддаємо перевагу використанню попарних перевірок зв'язків між точками огляду як засобу інтеграції.

Крім того, стверджується, що децентралізована архітектура фреймворку ViewPoints є кращим відображенням «реального» розвитку кількома учасниками розробки, які мають різні точки зору. Учасники розробки природно розподілені, і хоча вони можуть ділитися знаннями або навіть мати спільний погляд на світ, ці знання розподіляються між ними і не проводиться централізовано. Насправді, як правило, багато

надлишкової або дубльованої інформації зберігається різними учасниками розробки, і тому єдиний, централізований масив інформації є неточним представленням процесу.

Сфера застосування інструментальної підтримки VOSE зосереджена навколо двох видів діяльності: «розробка методів» і «використання методів».

Для підтримки розробки методів потрібні інструменти для опису та складання шаблонів, а також для створення інструментів CASE для підтримки розробки специфікації ViewPoint на основі цих шаблонів. Для опису шаблону потрібні такі інструменти, як текстові та графічні редактори для визначення стилів шаблону, а також інструменти моделювання процесу для визначення робочих планів шаблону. Бібліотеки шаблонів також корисні для повторного використання попередньо визначених шаблонів. Створення інструментів CASE вимагає або програмування або наборів мета CASE-інструментів для створення або генерації, відповідно, інструментів для підтримки різних шаблонів.

Для підтримки використання методу, тобто розробки програмного забезпечення, орієнтованого на ViewPoint, потрібна інфраструктура розподіленої системи для підтримки розподілених точок огляду та їх взаємодії, а також гнучкого середовища, у якому можна розробляти точки огляду та керувати ними. Останній включає середовища, які полегшують використання інструментів, створених або згенерованих процесом розробки методів.

Структура ViewPoints також є засобом інтеграції інструментів. Це досягається шляхом розгляду інтеграції інструментів як окремого випадку інтеграції методів.

Якщо визначення шаблонів ViewPoint зроблено достатньо точними та повними, щоб генерувати CASE-інструменти з цих визначень, тоді ті самі перевірки та пов’язана модель процесу, яка забезпечила інтеграцію методів, також можна використовувати для інтеграції інструментів (рис. 2).

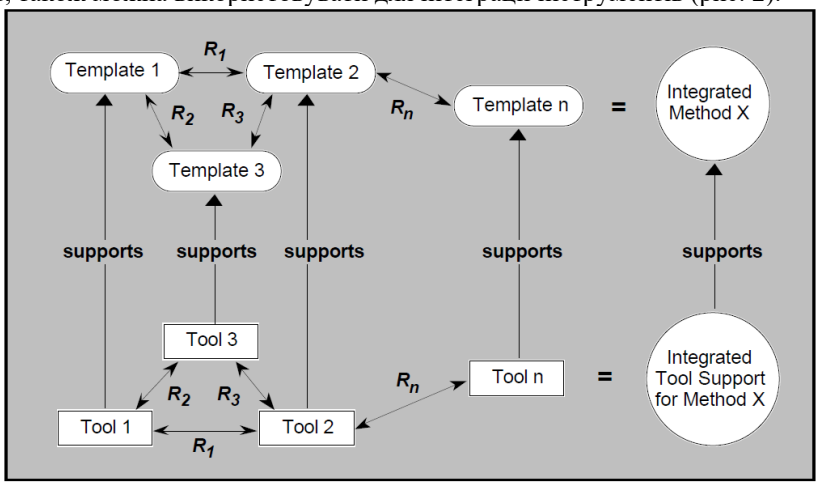

Рис. 2. Інструментальна інтеграція як випадок інтеграції методу

Модель об'єкта. Фреймворк ViewPoints — це об’єктний фреймворк, а View Points аналогічні «об’єктам» у традиційній об’єктній моделі. Вони мають «стан» (із слотами специфікації та робочих записів як «змінними екземплярів») і «поведінку» (з «методами», визначеними в робочому плані і слотами стилю). Вони мають унікальну ідентифікацію, позначену доменом і власником і також створені з «класів» (шаблонів), але наразі немає явної підтримки успадкування.

Основною метою є досягнення інтеграції методів у контексті багато напрямленої розробки програмного забезпечення, на прикладі VOSE. Фреймворк ViewPoints полегшує розробку програмного забезпечення шляхом розгортання кількох ViewPoints для представлення кількох точок зору та учасників розробки. Архітектура фреймворку за своєю суттю є децентралізованою, що дозволяє розподілену одночасну розробку. Шаблони ViewPoint полегшують визначення методів розробки, на основі яких створюються та розвиваються ViewPoints. Один шаблон ViewPoint представляє примітивний метод, тоді як набір шаблонів становить більш складний складений метод.

Таким чином, структура досягає поділу інтересів між ViewPoints з точки зору і учасників розробки і різних видів інженерних знань програмного забезпечення, які інкапсулює кожна ViewPoint. Розділення проблем також досягається на рівні розробки методу в термінах різних шаблонів, які становлять методи, і на рівні використання методу в термінах різних точок огляду, які утворюють специфікації системи.

Однак інтеграція цих проблем на різних етапах не є такою простою та вимагає зміни структури. Це, у свою чергу, впливає на характер процесу розробки програмного забезпечення, орієнтованого на ViewPoint, і спосіб, у який виконуються дії в межах структури.

На рисунку 3 показано можливості для інтеграції в структуру ViewPoints. Це також ілюструє необхідність вивчення інших видів діяльності для досягнення бажаної інтеграції методів.

ISSN 2219-9365

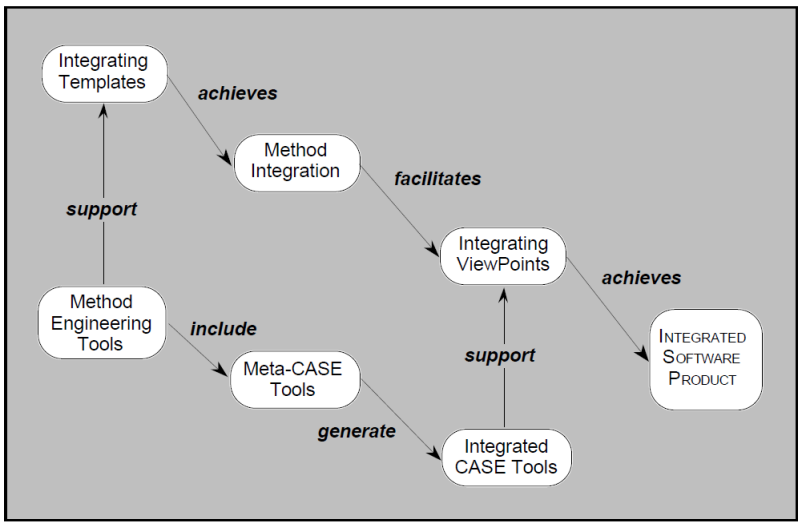

Рис. 3. Можливості інтеграції ViewPoints framework

Під час розробки методів інтеграція відбувається на рівні шаблону, тобто гарантується, що зв'язки між шаблонами визначені та виражені. Під час використання методу (розробка ViewPoint) інтеграція відбувається між ViewPoints, тобто гарантується, що правила між ViewPoint викликаються у відповідний час і застосовуються правильно.

Розробка методів у структурі ViewPoints спрощується тим фактом, що єдиний метод розробки програмного забезпечення може бути розроблений і створений модульним способом шляхом як визначення окремих шаблонів, так і повторного використання попередньо визначених шаблонів із бібліотеки повторного використання таких шаблонів. Підтримка інструментів для цього процесу також спрощена тим, що інструменти розробки методів у цьому контексті є або структурованими або мета-CASE-інструментами для генерації CASE-інструментів для підтримки, переважно простих, шаблонів.

Інтеграція методів у структуру ViewPoints зрештою пов'язана зі зв'язками між ViewPoint – як вони виражаються, коли викликаються та як застосовуються. Ці зв'язки спочатку проявляються як зв'язки всередині та між шаблонами, які називають зв'язками між ViewPoints. Викладено вимоги інтеграції методів під час розробки методів, коли визначено шаблони і «вимірюється» збереження зв'язків між точками огляду, створеними з цих шаблонів.

Багатонапрямлена розробка програмного забезпечення залучає кількох учасників, які незмінно дотримуються різних поглядів на проблему чи область вирішення. Ці учасники включають клієнтів, які мають різні суперечливі та взаємодоповнюючі вимоги до програмної системи, яку вони бажають придбати і розробники, які повинні виявити, визначити, проаналізувати та підтвердити ці вимоги, а потім спроектувати та створити програмну систему, яка задовольняє ці вимоги.

Іншим учасником процесу розробки програмного забезпечення є інженер з документування вимог (Method Engineer). Загалом, розробник методів відповідає за проектування та конструювання методу розробки, який, у свою чергу, надає розробникам систематичні процедури та вказівки для розгортання однієї або кількох нотацій для опису проблемних областей. Дедалі частіше роль інженера стає більш специфічною для сфери застосування, з «налаштованими» або «специфічними» методами, зібраними для конкретних організацій або проектів. Цьому сприяють інструменти комп'ютерної розробки методів (CAME), наприклад мета-CASE-інструменти.

Теоретично діяльність інженерів з документування вимог передує діяльності розробників програмного забезпечення. На практиці ця роль і ролі розробників програмного забезпечення та клієнтів нерозривно пов'язані та часто збігаються (рис. 4). І клієнти і розробники програмного забезпечення висуватимуть вимоги до розробника методів: клієнт може вимагати, щоб розробник програмного забезпечення використовував певний метод, а системні вимоги можуть вимагати налаштування цього методу.

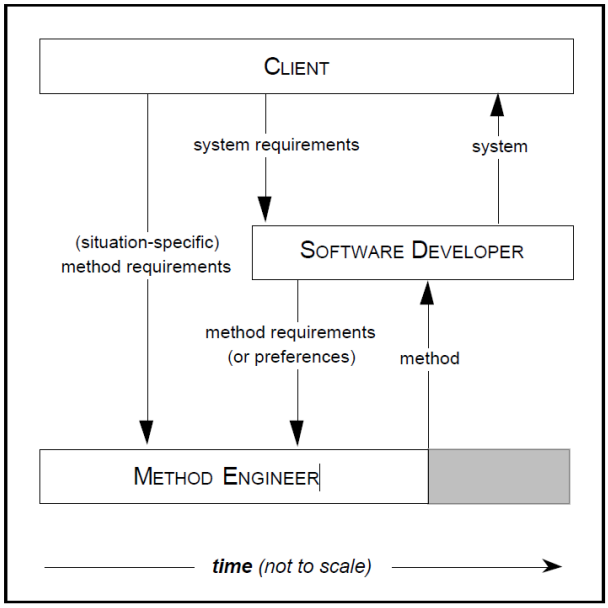

Рис. 4. Учасники розробки програмного забезпечення

На рисунку 4 заштрихована частина діяльності розробника методу вказує на те, що його участь у розробці може тривати навіть після надання методу розробки програмного забезпечення розробнику програмного забезпечення.

В структурі ViewPoints методом розробки програмного забезпечення є конфігурація (структурована колекція) шаблонів ViewPoint. Таким чином, ці шаблони є основними фрагментами (блоками) методів, з яких створюються специфікації системи (конфігурації ViewPoint). Структура методу в цьому контексті визначається внутрішніми та міжшаблонними зв'язками. Внутрішньошаблонні зв'язки — це зв'язки між ViewPoints, створеними з одного шаблону, тоді як зв'язки між шаблонами — це зв'язки між ViewPoints, створеними з різних шаблонів.

На рисунку 5 зображено ряд дій, які відбуваються як під час розробки методу, так і під час використання методу. Конкретний метод, показаний на рисунку, складається з шести шаблонів, позначених від T1 до T6. Стрілки між шаблонами позначають зв'язки між ними. Таким чином, T1 пов'язаний як з T2, так і з T3, тоді як T2 і T3 пов'язані з T4 і T5 відповідно. І T4, і T5 пов'язані з T6. Напрямки стрілок також вказують на «бажаний» або рекомендований порядок діяльності з розробки високого рівня, наприклад, порядок, у якому повинні створюватися екземпляри різних шаблонів. У цьому контексті роль інженера полягає в тому, щоб ідентифікувати, описати та зв'язати різні шаблони, які складають необхідний метод.

Товсті сірі стрілки на рисунку 5 представляють дії створення екземплярів у шаблонах («Дії тригера ViewPoint», описані в попередньому розділі). Результатом дії створення екземпляра на шаблоні Tx є ViewPoint VP Tx. Треба зауважити, що один шаблон, такий як T4, може бути створений багато разів, щоб отримати стільки ViewPoint (тому, наприклад, T4 створено двічі на рисунку, щоб отримати ViewPoint VP1 T4 і VP2 T4).

Нарешті, різні ViewPoint, створені в результаті цього процесу побудови екземплярів шаблонів, самі пов'язані екземплярами зв'язків, визначених між шаблонами, з яких вони були створені. Таким чином, зв'язок між шаблонами T2 і T4, стає зв'язком між ViewPoints між усіма ViewPoints, створеними з T2 і T4 відповідно. Іншими словами, VPT 2 пов'язаний як з VP1 T 4, так і з VP2 T4 через екземпляри зв'язку. Створення екземплярів шаблонів та їхніх зв'язків — це діяльність, яка відбувається під час так званої «розробки, орієнтованої на ViewPoint», тобто під час використання методу.

Рисунок 5 також ілюструє деякі складності методів розробки програмного забезпечення та різні способи, якими вони можуть бути розгорнуті. Загалом, методи не передбачають послідовний ряд процедур, а складаються з кількох слабко пов'язаних етапів, кожен з яких містить одну або більше методик або підтехнологій. Таким чином, розглядаючи шаблони ViewPoint як компоненти в конфігурації, специфікація системи потім розробляється шляхом створення екземплярів цих шаблонів, які створюються або «породжуються» динамічно в ході розробки.

На рисунку 5 не показано ітеративний процес уточнення, за допомогою якого окремі ViewPoints (частково) розробляються, а потім модифікуються та еволюціонують у результаті обходу стрілок взаємозв'язку між ними. Для того, щоб зрозуміти та слідувати такому процесу розробки, орієнтованому на ViewPoint, необхідно також зрозуміти процес розробки методів. Таким чином, метод у структурі ViewPoints

повинен бути розроблений і створений таким чином, щоб дозволяти створювати ViewPoints за потреби, частково розроблятися, якщо це буде необхідно, і періодично перевірятися в процесі розробки.

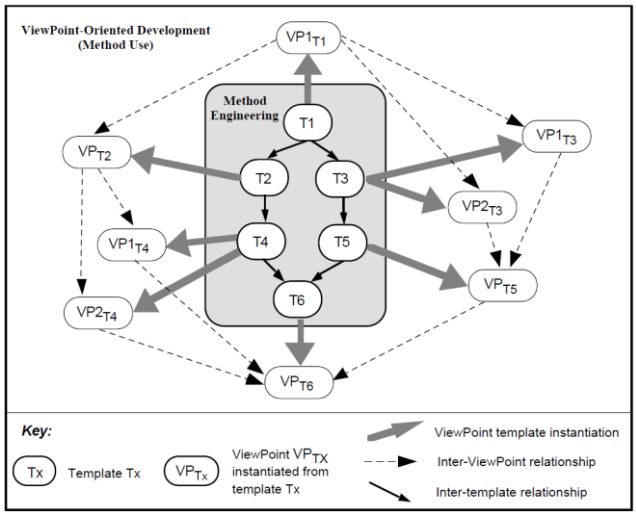

Рис. 5. Розробка методу з використанням концепції ViewPoint

Метою розробки методів у структурі ViewPoints є створення конфігурації шаблонів ViewPoint. Іншими словами, роль розробника методу полягає в тому, щоб визначити, спроектувати та створити шаблони, які складають метод. Як правило, розробники методів або «винаходять» нові методи з нуля, або збирають (повторно використовують і адаптують) методи з бібліотеки повторного використання фрагментів методів (шаблонів). Спосіб, у який методи «розрізаються» на їх складові шаблони є центральною діяльністю з розробки методів, яка спирається на досвід і застосування деяких простих евристик. На рисунку 6 показано процес розробки методу, орієнтованого на ViewPoint, який більш детально описано нижче.

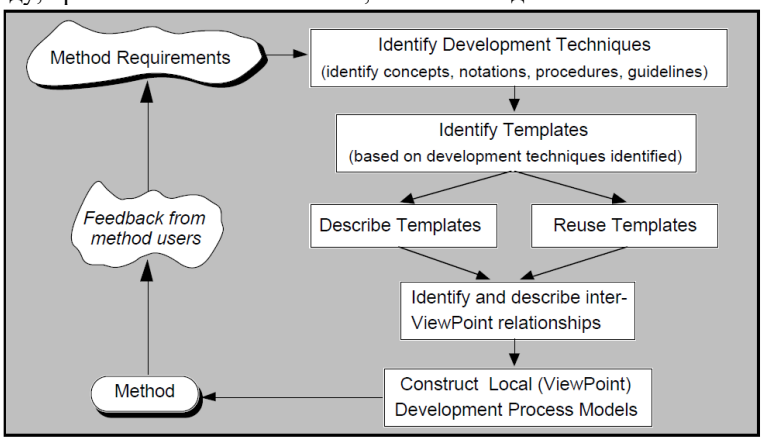

Рис. 6. Розробка та побудова методу на основі ViewPoints framework

Виходячи з невизначених вимог до методу розробки програмного забезпечення, інженер визначає методи розробки, які використовуватиме цільовий метод. Це включає в себе визначення різних нотацій і стратегій розвитку які будуть частиною цього методу.

Наприклад, згідно Object Modelling Technique (OMT) було визначено три типи технік моделювання, необхідних для створення об'єктно-орієнтованого дизайну, а саме об'єктне, динамічне та функціональне моделювання. З цією метою адаптовано три відомі методи розробки або побудови діаграм (зв'язок сутності, перехід стану та потік даних) і використано їх (відповідно) для представлення трьох моделей OMT.

Наступним кроком є визначення шаблонів ViewPoint, які необхідно створити, щоб описати методи, визначені вище. Гарною евристикою в таких ситуаціях є ідентифікація якомога більшої кількості простих технік розробки (наприклад, тих, які використовують прості нотації), а потім вибір єдиного шаблону для опису кожної техніки. Звичайно, інженер не буде вибирати які методи розробки і шаблони використовувати, а буде обмежений щодо видів технік або нотацій, умовами клієнтів.

Після того, як необхідні шаблони визначено, їх потрібно визначити більш точно. Іншими словами, стиль і слоти робочого плану для кожного шаблону повинні бути розроблені. Як альтернатива, шаблони, вже описані в інших методах, можуть бути адаптовані для задоволення вимог нового методу. Така адаптація зазвичай включає зміни в локальному стилі шаблону та діях робочого плану шаблону.

Дії перевірки Inter-ViewPoint застосовують правила між ViewPoint. Ці правила є специфічними для методу атрибутами шаблонів. Вони описують внутрішні та міжшаблонні зв'язки та є єдиними нелокальними властивостями шаблонів, якими розробникам методів потрібно займатися на цьому етапі. Якщо шаблон повторно використовувався з іншого контексту (методу), ці правила також потрібно змінити або повністю переписати. Якщо шаблон було визначено з нуля, ці правила потрібно писати з нуля. Правила Inter-ViewPoint відіграють низку важливих ролей у розробці та використанні методів, і вони будуть обговорюватися більш детально в розділі 6. Однак насамперед вони пов'язують між собою різні шаблони методів і, отже, є засобом інтеграції методів. Інтеграція методів сприяє багатомірній розробці програмного забезпечення інтегрованих програмних систем.

Останнім етапом розробки методу є визначення локальної моделі розробки (процесу) для кожного окремого шаблону. Кожна модель процесу визначає, коли мають відбуватися різні дії робочого плану; наприклад, коли потрібно викликати та перевірити правила між точками перегляду. Таким чином, локальні моделі процесів ViewPoints також служать для синхронізації зв'язку між ViewPoints шляхом координації виклику та застосування дій перевірки між ViewPoint. Координацію моделей процесу ViewPoint у цьому параметрі також можна назвати інтеграцією процесу.

Результат процесу розробки методу, описаного вище, є методом , який задовольняє початкові вимоги та може бути наданий розробникам програмного забезпечення та використаний ним для розробки відповідних систем програмного забезпечення. Хоча динамічна еволюція методів (під час використання методів) наразі не підтримується фреймворком (якщо створено екземпляри складових шаблонів цих методів), очевидно, що це проблема, яку потрібно вирішити, а потім ретельно підтримувати та контролювати.

Структура ViewPoints надає розробникам інфраструктуру для повторного використання. Найбільш очевидним багаторазовим компонентом у структурі є шаблон ViewPoint, який можна повторно використовувати принаймні трьома різними способами (схематично показано на рисунку 7):

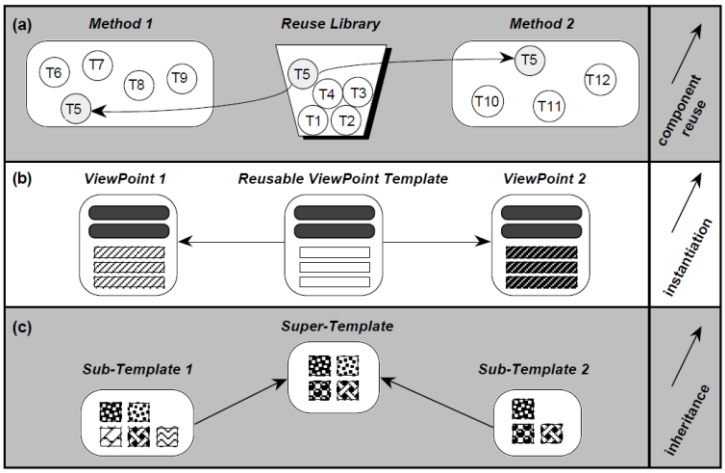

Рис. 7. Шаблони ViewPoint як засоби для повторного використання:
a) шаблони як багаторазові компоненти; b) Кілька ViewPoints, створених з одного (багаторазового) шаблону; c) Шаблони, що обмінюються (повторно використовують) знання через успадкування

1. Спосіб — розглядати шаблони як окремі компоненти, що належать до «бібліотеки повторного використання» (шаблонів для повторного використання). Їх можна окремо «підключити» до відповідного методу, зазвичай після певної адаптації. Повторне використання шаблонів у такий спосіб може заощадити значний час і зусилля розробників методів під час проектування та побудови методу, зменшивши необхідність визначати шаблони з нуля. Однак проблеми класифікації, підтримки та доступу до великих бібліотек багаторазових компонентів залишаються. Крім того, розробникам методів необхідно докласти зусиль у плануванні та проектуванні, щоб створити та заповнити такі багаторазові бібліотеки шаблонів.
2. Спосіб полягає в повторному використанні шаблонів ViewPoint за допомогою техніки інженірингу. Шаблони аналогічні «типам» або «класам», що містять загальну інформацію, яка передається екземплярам цих шаблонів, що дозволяє кільком ViewPoints використовувати однакові нотації або стратегії розробки, якщо вони створені з одного шаблону. Проте повторне використання шляхом інстанціювання зазвичай означає, що інформація «рівня типу» (рівня шаблону) дублюється в кожному екземплярі.
3. Спосіб, який не повністю підтримується, полягає у повторному використанні знань шаблонів через успадкування. Багато шаблонів можуть мати однакове представлення або знання процесу. Щоб уникнути дублювання таких знань у кожному новому створеному шаблоні, можна розробити структуру «супершаблон/підшаблон» (суперкласи / підкласи в структурах об'єктно-орієнтованого програмування). Наприклад, можна визначити «абстрактний шаблон» (абстрактний клас), який описує

загальну техніку діаграми потоку даних з двома додатковими під шаблонами для відповідного представлення двох нотаційних варіацій цієї техніки.

Повторне використання у структурі ViewPoints є допоміжним засобом для побудови методів. Можна заощадити значний час і зусилля, повторно використовуючи існуючі шаблони та адаптуючи їх до вимог нових методів. Шаблони є компонентами, які можна багаторазово використовувати, за винятком тієї частини їхніх робочих планів, яка визначає зв'язки між ViewPoint. Строго кажучи, дії перевірки між ViewPoint слід визначати окремо від окремих шаблонів, оскільки вони не є частиною жодної окремої методики розробки як такої, а радше описують зв'язки, які інженери методів хочуть виразити між цими техніками. Вони також більшою мірою залежать від методів і тому менш придатні для повторного використання, ніж решта елементів шаблонів ViewPoint.

Варто зауважити, що з точки зору користувача методу, структура ViewPoints також полегшує повторне використання. Самі ViewPoints є повторно використовуваними (спеціальними) компонентами, які можна класифікувати, абстрагувати та комбінувати.

Загалом використання методу – це розгортання його для визначення та розробки системи програмного забезпечення. У структурі ViewPoints це характеризується створенням і розвитком багатьох ViewPoints, кожна з яких стосується певного аспекту або частини системи, що розробляється. Таким чином, розробка програмного забезпечення, орієнтованого на ViewPoint, підтримує поширення кількох представлень під час вимог, кількох рішень під час проектування та мультипарадигмального програмування під час впровадження.

Правила Inter-ViewPoint розкривають різну структурну інформацію про методи, процеси розробки та специфікації системи залежно від етапу, на якому вони спостерігаються. Ця інформація може бути корисною при побудові інструментальної підтримки для фреймворку ViewPoint і може використовуватися для візуалізації структури методів, стану процесів розробки та структури систем, що розробляються.

Під час проектування методів, правила між точками описують зв'язки між шаблонами та між ними. Таким чином, у поєднанні з шаблонами, які вони стосуються, ці правила надають інформацію про структуру методів, які вони інтегрують. Метод у структурі ViewPoints — це конфігурація (структурована колекція) шаблонів ViewPoint. Ця структура визначається зв'язками (вираженими як правила) між складовими шаблонами методу. Ці відношення позначені стрілками на рисунку 8.

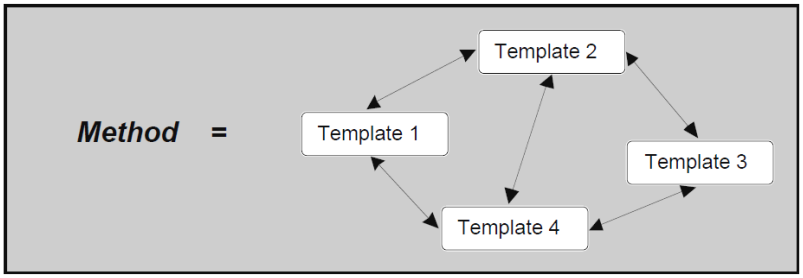

Рис. 8. Структура методу, що базується на ViewPoint і визначається правилами між шаблонами

В пропонованій моделі інтеграції ViewPoints визначення правил між ViewPoint також є кроком, на якому визначається структура методів.

У будь-який момент процесу розробки не обов'язково будуть створені всі необхідні ViewPoints. Крім того, у будь-який момент під час розробки кілька точок можна створити безпосередньо з поточної конфігурації. Які точки мають бути створені, визначається правилами між ними, які були визначені під час розробки методу. Ці правила представляють зв'язки між ViewPoints в конфігурації, і тому їх можна використовувати для визначення та «підказки» відсутніх точок. Рисунок 9 є «моментальним знімком» процесу розробки, орієнтованого на ViewPoint. Пунктирні лінії представляють ViewPoints, які ще не створені, але які «досяжні» з існуючих точок (вони досягаються шляхом виклику правил між точками показаних на рисунку).

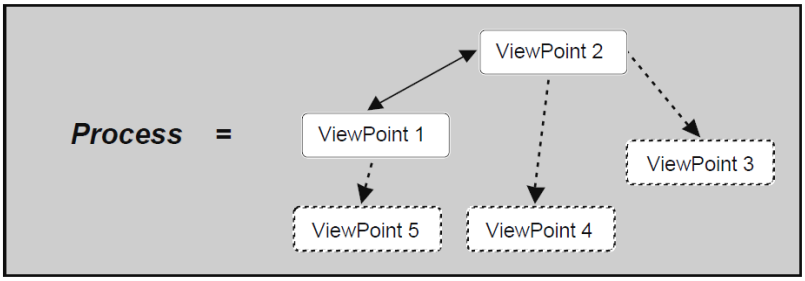

Рис. 9. Процес розробки визначається правилами між ViewPoint. Точки 1 і 2 були створені, а 3, 4 і 5 можуть бути побудовані безпосередньо з них

Метою процесу розробки, орієнтованого на ViewPoint, є створення системної специфікації (або системи програмного забезпечення), яка є узгодженою (відповідно до деяких правил між ViewPoint). У структурі ViewPoints метою є створення інтегрованої конфігурації ViewPoints, у якій зв'язки, які були визначені під час проектування методу, були перевірені та знайдені (або створені) для збереження. Рисунок 10 ілюструє, як структура специфікації системи визначається правилами між точками. Пунктирні лінії вказують на взаємозв'язки між ViewPoint, які ще потрібно перевірити та які не обов'язково дотримуються.

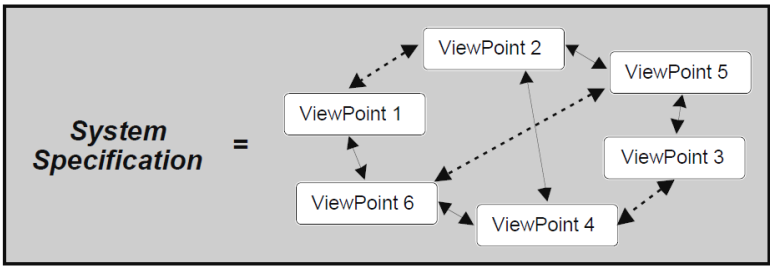

Рис. 10. Специфікація системи, що визначається правилами між ViewPoint

Рисунок 10 також підкреслює потенційну проблему масштабу, яку створює така конфігурація ViewPoints.

У моделі інтеграції ViewPoints застосування правил між точками є етапом на якому перевіряються зв'язки між точками огляду. Структурована колекція (конфігурація) точок, створена наприкінці цього кроку, є системною специфікацією або програмним забезпеченням.

Щоб ефективно підтримувати багатонапрямлену розробку програмного забезпечення, така розробка має базуватися на інтегрованих методах розробки програмного забезпечення. Таким чином, у контексті структури ViewPoints інтеграція методів є засобом для досягнення інтеграції ViewPoints. Інтеграція ViewPoints — це створення узгодженої колекції ViewPoints шляхом застосування правил узгодженості між ViewPoints. Ці правила визначають попарні зв'язки між точками огляду та є ключем до структурування та інтеграції методів, процесів і специфікацій у цьому параметрі.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У статті розглянуто та виконано порівняльний аналіз моделей інтеграції та агрегації процесів розробки програмного забезпечення та сервісів. Основна мета полягала в виділенні проблем, що виникають під час розробки програмного забезпечення, та розробці підходів для зменшення складності цього процесу і продукту, що ним створюється. Особливу увагу приділено можливості використання різних точок зору на програмну систему з метою підтримки роботи кількох учасників розробки в розподілених умовах.

У роботі використано концепцію інтеграції різних методів засобами ViewPoints через поєднання шаблонів. В роботі також запропоновано та обговорено роль правил між ViewPoints як інструменту для такої інтеграції. Модель інтеграції ViewPoints охарактеризована рядом дій, необхідних для досягнення інтеграції в цьому контексті. Модель базується на понятті правил між ViewPoints та їх наслідках. Маніпуляція правилами між точками є ключем до керування узгодженістю між точками огляду, а представлена модель є моделлю управління узгодженістю, що дозволяє ефективніше інтегрувати методи.

Представлені методи інтегровані шляхом ідентифікації та визначення предметних областей за допомогою правил між ViewPoints, а правила розподіляються між різними шаблонами. Це призводить до повністю розповсюджуваної конфігурації ViewPoints, яка краще підходить до розробки складних систем.

References

1. Ahmed, R., P. De Smedt, W. Du, W. Kent, M. Ketabchi, W. Litwin, A. Rafii and M. Shan (2018); The Pegasus Heterogeneous Multidatabase System; Computer, 24(12): 19-27, December 2018; IEEE Computer Society Press.
2. Akima, N. and F. Ooi (2019); Industrializing Software Development: A Japanese Approach; Software, 6(2): 13-21, IEEE Computer Society Press.
3. Alford, M. W. (2015); SREM at the Age of Eight: The Distributed Computing Design System; Computer, 18(4): 36-46, April 2015; IEEE Computer Society Press.
4. Barstow, D. R. (2015); Domain-Specific Automatic Programming; Transactions on Software Engineering, 11(11): 1321-1336, November 2015; IEEE Computer Society Press.
5. Batini, C., M. Lenzerini and S. B. Navathe (2016); A Comparative Analysis of Methodologies for Database Schema Integration; ACM Computing Surveys, 18(4): 323-364, December 2016; ACM Press.