МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 02.00.00.000 ПЗ

Група ІПм-22-1

Бучинський Олег

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Бучинський Олег Володимирович

(прізвище, ім'я, по батькові)

УДК 004.9

(індекс)

МАГІСТЕРСЬКА РОБОТА

Методи та алгоритми побудови стійкої ітеративної моделі процесу

розробки програмного забезпечення

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Бучинський О.В.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Шекета Василь Іванович, д.т.н., професор

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

В.о. завідувача кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

доц.

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
В.о. зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2023 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Бучинському Олегу Володимировичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи “ Методи та алгоритми побудови стійкої ітеративної моделі процесу розробки програмного забезпечення”

керівник проекту (роботи) Шекета Василь Іванович, д.т.н., професор

затверджені наказом закладу вищої освіти від “ 18 ” грудня 2023 р. № 738/7

2. Строк подання студентом проекту (роботи) 15 січня 2024 р.

3. Вихідні дані до проекту (роботи) Теоретичні концепції та формальні моделі побудови та функціонування програмних технологій розробки ПЗ

4. Зміст розрахунково - пояснювальної записки(перелік питань, які потрібно розробити)

1. Огляд рішень та засобів в області автоматизації процесу розробки програмного забезпечення
2. Структуризація раціональності процесу розробки програмних рішень
3. Розробка стійкої ітеративної моделі розробки програмного забезпечення
4. Концептуальне знання-орієнтоване перемоделювання методу розробки

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Сильні та слабкі сторони ISD та ME
2. Обґрунтування методу, що складається з цілей і цінностей
3. Оригінальна конструкція компонента методу
4. Структура прив’язки вартості для компонента методу моделі бізнес-випадку .
5. Діаграма, що описує компоненти методу та їх співвідношення цілей у проекті

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Нормоконтроль	доц., к.т.н. Вовк Р.Б.	
Перевірка на плагіат	доц., к.т.н. Вовк Р.Б.	

7. Дата видачі завдання 04 вересня 2023 р.

Керівник _____
(підпис)

Завдання прийняв до виконання_____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі магістерської роботи	01.10.2023	виконано
2	Аналіз концепцій та алгоритмів предметної області	25.10.2023	виконано
3	Огляд рішень та засобів в області автоматизації процесу розробки програмного забезпечення	10.11.2023	виконано
4	Структуризація раціональності процесу розробки програмних рішень	22.11.2023	виконано
5	Розробка стійкої ітеративної моделі розробки програмного забезпечення	01.12.2023	виконано
6	Реалізація функціональності запропонованої інформаційної технології	15.12.2023	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.01.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 85 с., 15 рис., 10 табл., 52 джерела.

Тема: Методи та алгоритми побудови стійкої ітеративної моделі процесу розробки програмного забезпечення

Об’єкт дослідження: задачі структуризації раціональності процесу розробки програмних рішень, визначення та обґрунтування методу розробки з описом концепції фрагменту методу розробки.

Мета роботи: концептуалізація методів розробки згідно з Method Engineering, виділення переваг і недоліків існуючих концептуалізацій методів, та синтез онтологічної основи для обґрунтування вибору методу розробки.

Предмет дослідження: ітеративні моделі компонентів методу концептуального знання-орієнтованого перемоделювання процесу розробки.

Результати дослідження:

В роботі побудовані рішення, щодо усунення проблеми невидимості в процесі розробки ПЗ та пов’язаних з нею складностей та інформаційних бар'єрів в яких використовується явне представлення знань і міркування щодо усунення обмежень такого типу.

Висновок

Досліджено проблеми правильного зберігання та пошуку знань різного роду при імплементації архітектури системи. Введено поняття, що описують дії та об’єкти в домені розробки які були створені шляхом читання та розуміння архітектурних документів та коментарів у вихідних файлах.

МЕТОД РОЗРОБКИ, КОНЦЕПТУАЛІЗАЦІЯ, МЕТОДИКА, ІТЕРАТИВНА МОДЕЛЬ, ГНУЧКА МОДЕЛЬ, МОДЕЛЬ RUP, КОМПОНЕНТ МЕТОДУ, ІНФОРМАЦІЙНА СИСТЕМА

ABSTRACT

Master Thesis: 85 pp., 15 fig., 10 tab., 52 sources.

Thesis Subject: Methods and algorithms for building a sustainable iterative model of the software development process

Object of research: tasks of structuring the rationality of the process of developing software solutions, defining and justifying the development method with a description of the concept of a fragment of the development method.

Research goal: conceptualization of development methods according to Method Engineering, selection of advantages and disadvantages of existing conceptualizations of methods, and synthesis of an ontological basis for justifying the choice of a development method.

Subject of research: iterative models of the components of the method of conceptual knowledge-oriented remodelling of the development process.

The results:

In the work, solutions are built to eliminate the problem of invisibility in the process of software development and related complications and information barriers, which use clear presentation of knowledge and reasoning to eliminate limitations of this type.

Conclusion

The problems of correct storage and retrieval of various types of knowledge during the implementation of the system architecture were studied. Concepts are introduced that describe actions and objects in the development domain that were created by reading and understanding architectural documents and comments in source files.

DEVELOPMENT METHOD, CONCEPTUALIZATION, METHODOLOGY, ITERATIVE MODEL, FLEXIBLE MODEL, RUP MODEL, METHOD COMPONENT, INFORMATION SYSTEM

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

RUP - Rational Unified Process

ME - Method Engineering

ISD - Information Systems Development

EME - Extended Method Engineering

MEL - Method Engineering Language

UML - Unified Modeling Language

ВСТУП

Актуальність дослідження.

Зростання вартості розробки програмного забезпечення, особливо у великих системах, добре задокументовано. Зусилля стримування цього зростання досягли обмеженого успіху. Існуючі розробки в інженерії програмного забезпечення, такі як мови високого рівня, розподіл часу та інтегровані середовища програмування подолали ряд труднощів в процесі розробки програмного забезпечення. Однак є інші проблеми, що визначають суть програмного процесу сьогодні. Виділяють чотири основні типи труднощей при створенні великих систем програмного забезпечення:

• Складність: складність процесу розробки, області застосування, внутрішньої частини системи тощо.

• Відповідність: програмне забезпечення не може мати регулярної формальної структури, тому що воно повинно адаптуватися до взаємодії з клієнтами відповідно до їхніх різноманітних потреб.

• Змінність: у будь-якій системі програмний компонент зазвичай найпростіший для модифікації.

• Невидимість: структура програмного забезпечення є прихованою сутністю. Єдиним зовнішнім доказом наявності програмного забезпечення є його поведінка та функціональність під час виконання.

Не існує єдиного, негайного вирішення цих проблем. Програмне забезпечення існує для заповнення потреб ринку, і не можна заперечувати, що ці потреби не змінюються. Таким чином, надзвичайна пластичність програмного забезпечення є вагомою причиною для зміни програмної частину системи, щоб задовольнити ці нові потреби.

Не маючи належного знання архітектури системи, розробники можуть робити помилки і виробляти некоректно працюючий код. Розробники можуть реалізовувати свої підсистеми таким чином, що порушує архітектурні принципи великої системи, яку вони модифікують.

Код може працювати правильно, і відповідно пройти тестування системи. Наявність таких порушень, в різні моменти, протягом певного періоду часу, призводить до втрати архітектури, що поступово руйнує первісну простоту системи.

Таким чином, цей брак знань серед розробників призводить до замкнутого кола, де система стає дедалі складнішою, а отже важче масштабованою відповідно.

Мета і задачі дослідження полягає в огляді рішень та засобів в області автоматизації процесу розробки програмного забезпечення, аналіз та дослідження предметної області методів та концепцій розробки програмного забезпечення.

Метою магістерської роботи є концептуалізація методів розробки згідно з Method Engineering, виділення переваг і недоліків існуючих концептуалізацій методів, та синтез онтологічної основи для обґрунтування вибору методу розробки.

Об'єктом дослідження – є задачі структуризації раціональності процесу розробки програмних рішень, визначення та обґрунтування методу розробки з описом концепції фрагменту методу розробки.

Предмет дослідження є вдосконалення ітеративної моделі за допомогою компонентів методу концептуального знання-орієнтоване перемоделювання процесу розробки.

Методи дослідження. В магістерській роботі використані методи інженерії знань застосовні до задач імплементації процесу розробки програмного забезпечення, методи моделювання бізнес-сутностей, методи ітеративних представлень, метододи побудови онтологічних представлень.

Для досягнення поставленої мети необхідно розв'язати такі задачі:

- виконати огляд рішень та засобів в області автоматизації процесу розробки програмного забезпечення;
- представити структуризацію раціональності процесу розробки програмних рішень;

- розробити стійку ітеративну модель розробки програмного забезпечення.

Наукова новизна роботи полягає в побудові рішення, щодо усунення проблеми невидимості в процесі розробки ПЗ та пов’язаних з нею складностей та інформаційних бар'єрів. В пропонованій стратегії використовується явне представлення знань і міркування щодо усунення обмежень такого типу. Створена система унікальна своєю здатністю забезпечувати семантичний пошук з інформаційної бази, яка об’єднує кілька видів представлень та забезпечує інтелектуальний індекс над бібліотекою багаторазового використання компонент, яка налаштована на велику програмну систему з функцією масштабованості. Цей підхід є перспективним, хоча показує деякі притаманні обмеження класифікаційного підходу.

Практичне значення одержаних результатів полягає в вирішенні проблеми правильного зберігання та пошуку знань різного роду при імплементації архітектури системи. Введено поняття, що описують дії та об’єкти в домені розробки. Вони були створені шляхом читання та розуміння архітектурних документів та коментарів у вихідних файлах. Це повністю ручний процес. Хоча кінцевий результат цих зусиль це КВ, до якої розробники можуть запитувати у спосіб, який має вплив на видимість системи, створення цієї бази знань є інтенсивним завдання, автоматизація якого є задачею високої комплексності, що вимагає вирішення завдань автоматичного сканування коду і виведення різного роду даних про функції коду.

Структура та обсяг роботи. Магістерська робота складається з вступу, трьох розділів, висновків та списку використаних джерел із 51 найменуваннями. Загальний обсяг роботи становить 85 сторінок, кількість рисунків 10, таблиць 10.

РОЗДІЛ 1. ОГЛЯД РІШЕНЬ ТА ЗАСОБІВ В ОБЛАСТІ АВТОМАТИЗАЦІЇ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Аналіз та дослідження предметної області методів та концепцій розробки програмного забезпечення

Необхідність підтримки методів для розробки інформаційних систем не є новою проблемою. У результаті розширення індустрії програмного забезпечення ці методи стають дедалі складнішими. Сьогодні існує безліч доступних методів і вони варіюються від високоструктурованих методів розробки, таких як Rational Unified Process (RUP) до менш формалізованих методів, зосереджених на швидкості та гнучкості, таких як екстремальне програмування (XP). Такий метод, як RUP, приділяє велику увагу допоміжним інструментам і створенню документів як засобу створення підтримки. Наприклад, RUP містить різноманітні інструменти для різних цілей і понад 80 різних основних артефактів. Гнучкі методи мають основу в практиці програмування та намагаються захопити найкращі практики щодо розробки систем в Agile методах. Мета полягає у створенні систем якості з меншими зусиллями та документацією. Точно невідомо, скільки методів доступно для використання, але відомо, що число перевищило 1000 доступних. Проте процес вибору конкретного методу та фактичної можливості його розгортання в організації-розробнику може бути громіздким.

Сьогодні дослідження показали, що розробники радше вчаться зазнавати невдач, дотримуючись методів, ніж покращувати якість своїх процесів. Використовувати метод не завжди легко, і важко зрозуміти зв’язок між результатами проекту та самим методом. У цьому контексті зрозуміло, що організації відчувають проблеми, коли справа доходить до питань методів. Деякі організації вирішують проблему, не використовуючи жодної

підтримки методів. У таких випадках деяке неявне знання зазвичай є основою для дій, які здійснюються під час розробки. Іншим способом сказати це було б сказати, що в таких ситуаціях, у кращому випадку, є неявна основа методу, або в гіршому випадку, чисті здогадки або спеціальні міркування. У таких випадках можна стверджувати, що існує явна відсутність можливостей контролювати події під час розробки через відсутність підтримки, яка зазвичай надається методами. Деякі організації стверджують, що використовують певний метод, щоб вони виглядали більш компетентними, навіть якщо вони все ще працюють так, як вони звикли. Проблеми, пов'язані з недостатньою продуктивністю, завжди можна в певному сенсі вирішити з методологічної точки зору. Під цим не мається на увазі, що новий, добре розгорнутий метод розробки систем завжди є срібною кулею для кожної організації. Швидше, цілком імовірно, що добре визначений і зрозумілий метод розробки інформаційних систем може допомогти групі розробників краще визначити можливі проблеми та сфери вдосконалення. Це може служити як спільна основа та відправна точка, коли справа доходить до спілкування щодо фактичного процесу розробки та того, як його можна покращити. Роль методу розробки систем у цих ситуаціях полягає в тому, щоб бути засобом для спілкування та засобом для створення синергії всередині команди.

Невід’ємною частиною будь-якого дослідження є саме ядро концепцій, які визначають, чого насправді стосується дослідження. Ці ключові концепції функціонують як парадигматичне визначення того, як проблеми та рішення сприймаються, вирішуються та передаються. Однак вони не лише дають вказівки щодо вирішення проблем. Як наслідок, вони також розмежовують можливості «мислити нестандартно», оскільки концепції спрямовують увагу на певні види явищ і, отже, також від інших типів явищ.

Точки зору ME (method engineering) та ISD (information systems development) мають відмінності в тому, як розглядаються та сприймаються проблеми, що стосуються методів розробки систем. Цілком ймовірно, що їхні

концептуалізації відрізняються, оскільки вони по-різному сприймають сферу використання методу.

Метод визначається як «кодифікована серія кроків, зроблених для виконання певного завдання або досягнення певної мети». Переходячи до сфери створення систем, то термін «методологія» використовується для опису «кодифікованого набору практик (іноді супроводжується навчальними матеріалами, формальними освітніми програмами, робочими аркушами та інструментами для створення діаграм), які можна багаторазово застосовувати для створення програмне забезпечення». Проте методологічний термін не позбавлений проблем, хоча більшість дослідників схильні віддавати йому перевагу.

1.1.1 Концептуалізація згідно з ISD

Якщо концептуалізації методів розробки систем у ME використовуються для формалізації, то в ISD вони використовуються для опису більш нематеріального явища. Більшість визначень того, що таке метод розробки систем у галузі ISD, більше зосереджені на словесному чи письмовому описі об’єкта дослідження. По суті, можна стверджувати, що дослідження ISD підкреслюють актуальність, а не суворість.

Різні визначення варіюються від дещо розпливчастих, які включають загальні твердження про те, що таке метод насправді; метод — це спосіб «застосувати конкретні концепції раціональності на практиці» або більш детальні, кажучи, що метод можна розглядати як; «повний набір концепцій і моделей, які є внутрішньо узгодженими» [2] . Це дійсно залежить від того, які аспекти управління в проектах розробки систем, які визначає бажає висвітлити.

Були зроблені спроби формально описати, що таке метод розробки систем, також у сфері ISD. В [3] представляють компонентний погляд на методи розробки систем, з наміром поєднати гнучкість зі стабільністю. Основна ідея компонентів методу полягає у використанні досить високого

рівня абстракції та структуруванні вмісту компонента на основі внутрішніх залежностей.

Компонент завжди складається з понять (про що говорити), процедури (які питання ставити) і нотацій (як виразити відповіді) [4]. Концепції є результатом основної точки зору розробника методу та спрямовують увагу на певні явища в контексті розвитку. Перспектива також впливає на форми співпраці, в яких реалізуються компоненти методу, і рамки, в які компоненти методу можуть бути інтегровані. Таким чином, метод розробки систем можна розглядати як контейнер кількох компонентів методу.

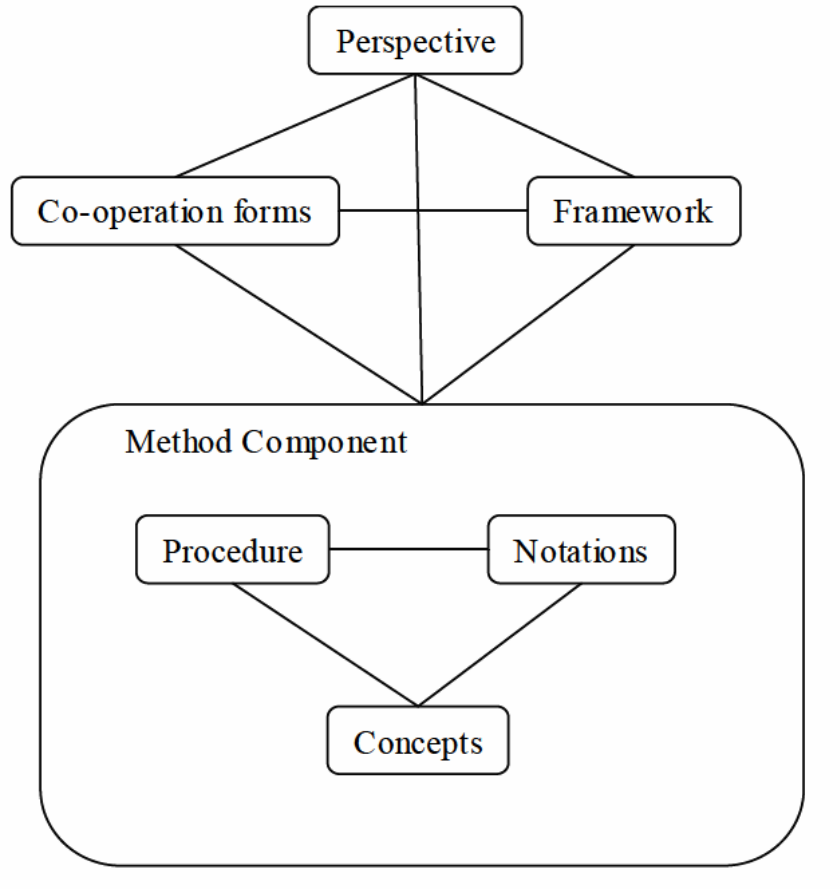

Рис. 1.1. Концепція компонента методу

Методи як такі задумані з причин. Цей нормативний вимір називається перспективою і є проявом теорії того, як повинні виконуватися проекти розробки систем. Перспектива методів пов'язана з принципами, цінностями, концепціями, досвідом, категоріями та визначеннями конструктора методу.

Як описано на рисунку 1.2, перспектива містить цінності та цілі, які впливають на те, як ми розуміємо явища в реальному світі. На основі наших цінностей ми класифікуємо та визначаємо світ, щоб інтерпретувати різні явища, з якими ми стикаємося. Ці елементи можна розглядати як цілі і, зрештою, реалізовувати в описі методу.

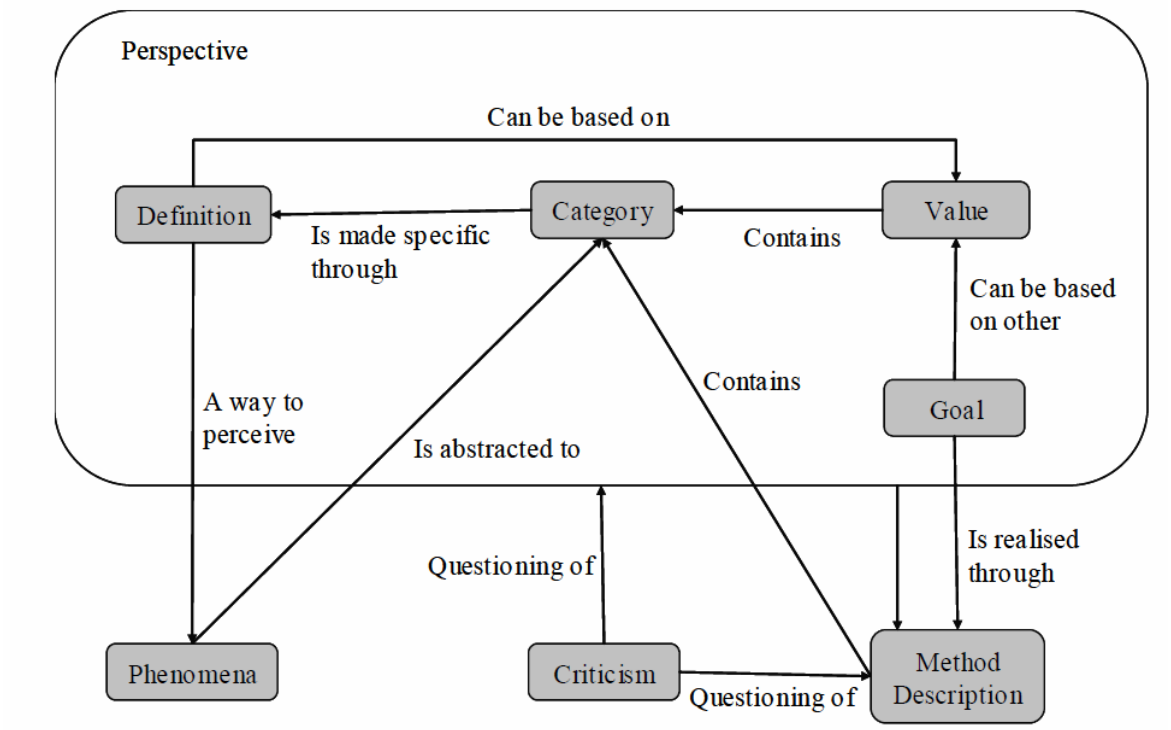

Рис. 1.2. Метод по відношенню до його перспективи

Таким чином, опис методу є наказовою концептуалізацією, яка зосереджує увагу на цілях, цінностях, категоріях, визначеннях і відволікає від інших явищ у реальному світі. Згодом усі методи слід розглядати з деякою ретельністю та критикувати. Найбільш фундаментальним аспектом цієї критики повинні бути питання про те, що включає перспектива, що вона виключає і як це проявляється в описі методу.

Очевидно, що будь-кому має бути зрозумілий сенс чіткого пояснення методів, що лежать в основі перспективи. Було б дуже корисно, якби

потенційний користувач методу міг визначити, чи є певний метод розробки систем хорошим вибором, просто дивлячись на перспективу методів.

Деякі перспективи в методах розробки систем були явно представлені назовні і тому вони більш відомі. Об'єктна орієнтація є яскравим прикладом. Ідеї, висунуті через перспективну концепцію в галузі ISD (Information Systems Development), не існують у сфері ME.

1.1.2. Фреймворки та моделі в ISD

Дослідження ME зазвичай присвячене структуруванню фрагментів методів або блоків методів у повні методи розробки систем. В ISD ці питання охоплюються використанням фреймворків або моделей для організації методів або частин методів, які будуть використовуватися в проекті розробки. Це схеми процесів розробки, які зазвичай поділені на певну форму фаз, кожна з яких зосереджена на певній проблемній області. Кожна фаза є результатом судження конструктора моделі, де фаза визначається та розмежовується як частина реального світу, яка заслуговує на дослідження. Прикладами моделей є відома модель водоспаду, яка виникла на початку 70-х років і спіральна модель.

У більш загальних рисах термін модель також використовується для позначення кінцевих продуктів у проекті розробки системи. наприклад, модель використання. Цей тип моделі також розмежовує конкретну частину реальності, яку варто дослідити, але вона не зосереджує контур проекту розвитку. Щоб розрізнити ці два типи моделей, я буду використовувати термін модель розвитку для позначення моделей із цим попереднім типом здібностей або фокусу.

Визначають два різні способи з'єднання методів або частин методу в моделі розробки або рамки. По-перше, існує потреба в інтеграції методів або частин методу таким чином, щоб гарантувати, що корисні результати можуть бути використані пізніше в процесі розробки.

По-друге, інтеграція також може бути виконана в межах певної фази, щоб збільшити фокус проблемної області фази. Цей тип інтеграції метафорично називається альянсом і ілюструє внутрішньопроцесну інтеграцію між методами або частинами методу в рамках конкретної фази моделі розробки.

На рисунку 1.3 проілюстровано в моделі фази розробки Rational Unified Process. Ланцюг або міжпроцесна інтеграція досягається через співпрацю між частинами методу 1 і 2 (MP1, MP2). Це означає, що MP2 спирається на результати MP1 . Прикладом цього може бути те, як модель бізнес-випадку використання використовується як вихідні дані для завдання виявлення моделі прецеденту використання.

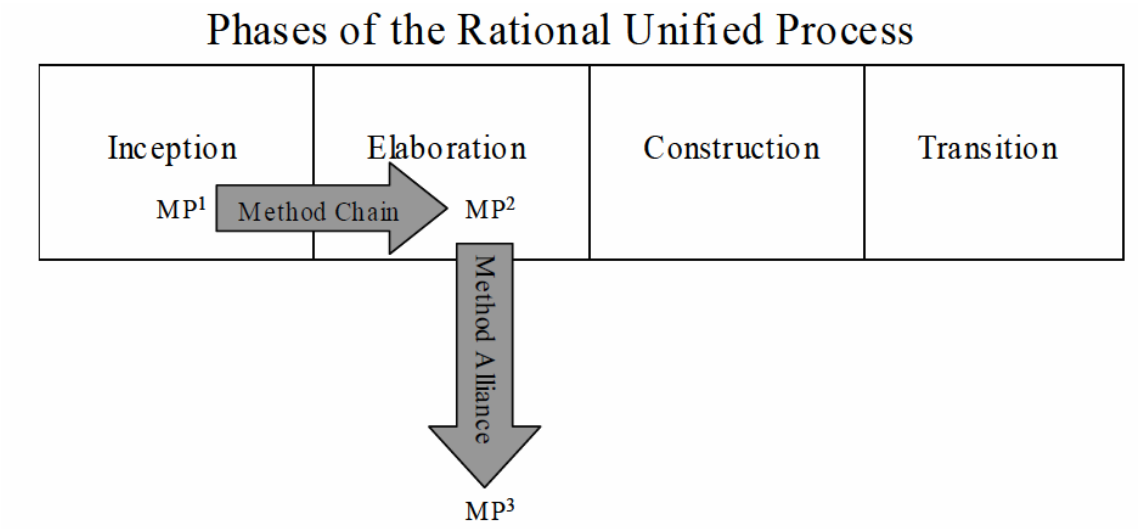

Рис. 1.3. Фазова модель методу

Альянс або внутрішньопроцесна інтеграція проілюстрована через співпрацю між частинами методу MP2 і MP3 . В альянсі дві частини методу використовуються для посилення конкретного фокусу в рамках даної фази. Прикладом альянсу може бути те, як модель випадку використання використовується для вдосконалення специфікації вимог до програмного забезпечення на етапі розробки.

1.2. Концептуалізація методів розробки згідно з Method Engineering

Використання поняття методів розробки систем у сфері ME в першу чергу орієнтоване на те, як описати методи та частини розробки систем. Якщо дослідження ISD підкреслюють релевантність, ME, як правило, зосереджується на створенні строгих рішень щодо того, як знайти відповідні методи розробки систем для унікальних проектів і організацій. Мета полягає в тому, щоб обробляти та адмініструвати планування та розробку нових або індивідуальних методів розробки систем для унікальних ситуаційних проектів. Таким чином, галузь ME здебільшого розглядає метод як щось, що можна описати та повідомити. Фундаментальним для цього завдання є питання про те, як метод має бути модульним, тобто розділеним на корисні частини.

У галузі ME ці модулі називаються фрагментами методу або фрагментами методу [5]. Пропонується класифікувати фрагменти методу за трьома вимірами: перспектива, рівень абстракції та рівень деталізації.

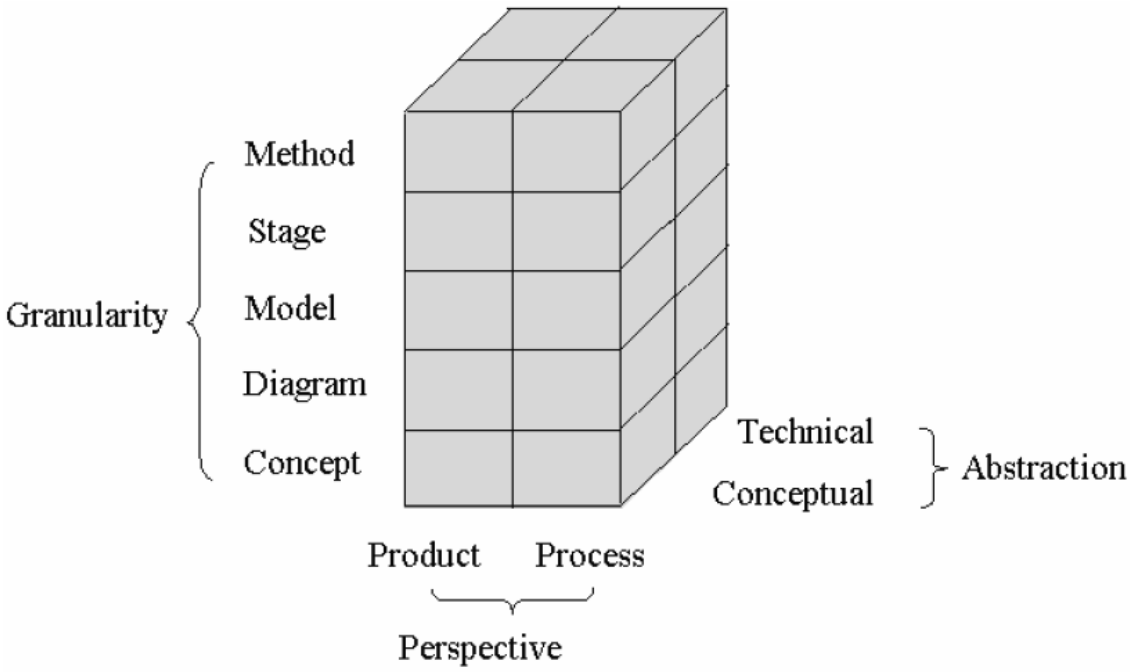

Рис. 1.4. Візуальне сприйняття концепції фрагмента методу

Перспективний вимір розглядає перспективу продукту та перспективу процесу. Фрагмент продукту — це результат, такий як документ або системний модуль. Фрагмент процесу представляє дії, які необхідно виконати для виробництва продукції.

Вимір абстракції складається з концептуального рівня та технічного рівня. Фрагменти методів на концептуальному рівні є описами методів розробки систем або їх частин. Фрагменти технічного методу — це виконувані інструменти, що реалізують методи (тобто CASE-інструменти). Розмір деталізації стосується того, на якому рівні деталізації (або агрегації) знаходиться конкретний фрагмент методу. Існує п’ять можливих рівнів: метод, етап, модель, діаграма та концепція.

У спробі посилити зв'язок між фрагментами процесу та фрагментами продукту було запропоновано поняття фрагментів методу. У концепції блоку ці два типи фрагментів об’єднані в один модуль. Таким чином, будь-який метод розробки системи можна розглядати як набір фрагментів методу на різних рівнях деталізації, таким чином цілий метод можна розглядати як фрагмент так само, як фрагмент методу може становити цілий метод розробки системи.

Блоки визначаються з точки зору процесу. Це означає, що, порівняно з концепцією фрагментів методу, ядром кожного блоку є керівництво для конкретного процесу та пов’язаних із ним фрагментів продукту [7]. Ця настанова втілює знання, необхідні для досягнення наміру в проекті розвитку. Отже, настанова має інтерфейс, який описує, коли її можна використовувати, і орган, який надає вказівки для досягнення наміру. Намір у цьому контексті означає побудувати щось корисне в проекті розробки. Прикладом наміру може бути виявлення випадку використання. Таким чином, намір можна розглядати як зосередження на завданнях, а не на фактичних цілях, які досягаються за допомогою цього завдання. Між блоками є інтерфейси, що описують, які ситуації потребують намірів і

вхідних даних [8]. Потім фрагменти використовуються для створення нових методів подібно до фрагментів методів.

1.3. Переваги і недоліки існуючих концептуалізацій методів

Порівняння концептуалізацій методів розробки систем у ME та ISD покаже як сильні, так і слабкі сторони в цих двох областях. Ці сильні та слабкі сторони створюють проблеми для того, як концепція методу розробки систем може сприйматися в EME. Проблеми передачі результатів дослідження в спосіб, який може сприйматися як значущий від однієї галузі дослідження до іншої, можуть бути пов’язані з тим, як ці дві галузі визначають і використовують концепцію методів розробки систем. Наприклад, твердження про перспективний вимір більш-менш неможливо використовувати в галузі ME, оскільки ці аспекти методів розробки систем не охоплюються їх визначеннями того, що таке метод розробки систем. Інший приклад комунікаційних проблем можна знайти в ситуації, коли прихильник ME намагається формально описати, як метод розробки систем повинен використовуватися для людини зі сфери ISD, не розрізняючи сам метод розробки систем і метод розробки систем в дії. Причини цього криються в тому, що ME не має методу використання або рівня дії. Вони просто розраховують, що розроблений метод буде використано за призначенням, хоча численні дослідження показують, що це рідко трапляється [8].

Сфера ME занадто формалізована. Принаймні, здається, є прагнення формалізувати передані знання про методи розробки систем. Проблеми, з якими стикається сфера ME, стосуються розмежування того, що насправді може бути формалізовано та передано. Однією з очевидних проблем із концепцією фрагментів методу є те, що формалізованим фрагментам бракує конгруентності, особливо у розмірі гранулярності. У цьому вимірі фрагмент може представляти цілий метод, а також лише одну окрему концепцію в

цьому методі. Зайве говорити, що це може спричинити проблеми, принаймні в ситуаціях, коли повідомляються методи. З іншого боку, відносно високий ступінь формалізації дає можливість розробити системи управління.

У порівнянні зі сферою ISD, перспективний вимір, здається, відсутній у ME. Це цікаво, оскільки ці питання частково висвітлюються у визначенні [9] того, що таке метод розробки систем: «Метод — це підхід до виконання проекту розробки систем, заснований на певному способі мислення, що складається з вказівок і правил, структурованих систематично в діяльності з розробки з відповідними продуктами розробки». Інтерпретація цього визначення могла б помістити перспективний вимір із поля ISD у фразу, що стосується «спосіб мислення». Коли справа доходить до формалізації методу розробки систем, будь-які спроби формалізувати цілі або цінності, пов'язані з перспективою методів розробки систем, відсутні. Поняття намірів у концепції фрагмента методу не перекладається на перспективний вимір ISD, оскільки намір є описом того, що потрібно зробити, а не фактичними причинами, чому щось потрібно зробити.

Зусилля щодо модуляції в галузі ME можуть бути корисними в таких ситуаціях, як комунікація та використання методів. У комунікаційних ситуаціях модулі методів служать інструментами для передачі знань методів у відповідних частинах. Модулярність у ситуаціях використання методу слугує контрольними точками та пояснює, як можна використовувати результати проекту. У цих випадках певний ступінь формалізації може бути корисним, оскільки вони можуть служити для прояснення питань щодо методів розробки систем. Ці аспекти формалізації методу були пропущені в галузі ISD. На рисунку 1.5 узагальнено показано, як обидві області бачать методи розробки систем, а також вказують на сильні та слабкі сторони в двох областях.

На рисунку 1.5 показано, що аспекти, які вважаються сильними в одній сфері, вважаються слабкими в іншій, і навпаки. Одним із способів вирішення

виявлених проблем було б визначення нової концептуалізації того, що таке метод розробки систем.

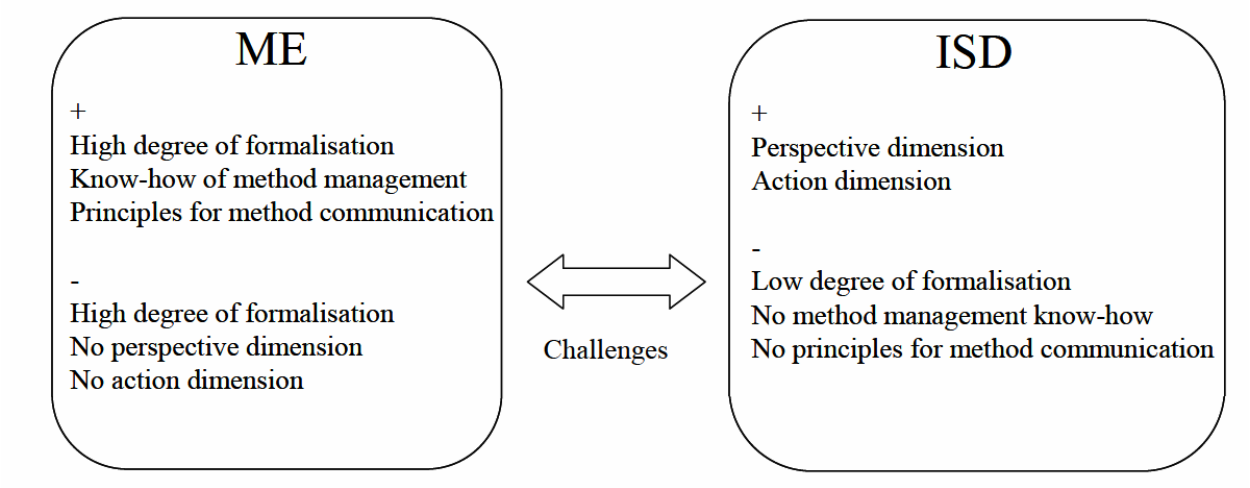

Рис. 1.5. Сильні та слабкі сторони ISD та ME

1.4. Синтез онтологічної основи для обґрунтування методу розробки

Сфера ME має незбалансований фокус і не має виміру дії. З іншого боку, ця галузь надає точні інструментальні підходи до адаптації методу та управління обґрунтуванням використання методу. Сфера ISD має однакову спрямованість, але має слабкий відгук. З іншого боку, поле забезпечує релевантну оцінку того, як метод може служити підтримкою в дії. Він також відображає філософський вимір методів і те, як учасники розвивають глибше розуміння практики розробки систем. Ця виявлена невідповідність між двома полями є відправною точкою для формулювання синтезу у формі EME.

У синтезованій моделі, що описує EME (extended method engineering), показаній на рисунку 1.6, три сфери є поєднанням двох областей дослідження. Оскільки поле ME природно найсильніше у сфері інженерії методів, верхня сфера збереже цю назву. Друга сфера повинна подолати проблему нездатності ME зосередити увагу на вимірі дій методів розробки систем. В ISD це охоплюється поняттям методу в дії. Згодом, оскільки ISD

дуже сильний у поясненні того, як насправді використовуються методи розробки систем, друга сфера буде називатися сферою методу в дії. Сфера відображення третього методу по суті однакова в обох вихідних полях, але з різницею у фокусі. У ME рефлексія в основному стосується обґрунтування використання методу, тоді як дослідження ISD стосується приватної раціональності акторів.

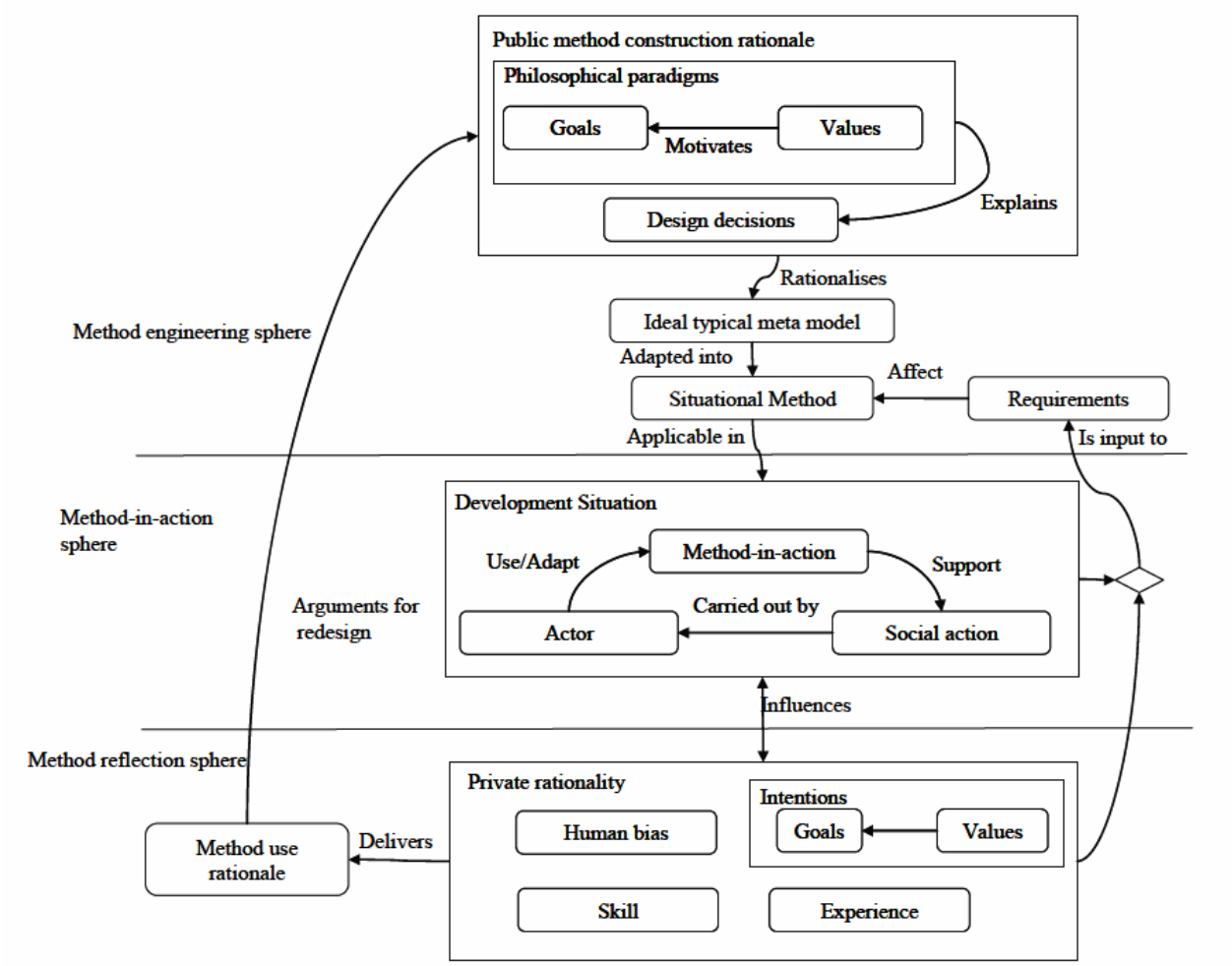

Рис. 1.6. Обґрунтування методу в EME, рівний фокус, подвійний зворотний зв’язок

У різних сферах оригінальні елементи були об’єднані, перейменовані та синтезовані в нові елементи, що описують, як концепція обґрунтування методу може бути зосереджена та використана в обох галузях дослідження. Синтез має на меті забезпечити можливості для подолання слабких сторін

двох сфер, надаючи строгій області ME можливість використати релевантність, яку забезпечує поле ISD. У той же час, синтез також спрямований на подолання слабких місць шляхом надання строгих інструментів і підходів до відповідної галузі ISD, надаючи їм кращі можливості для узагальнення своїх результатів і передачі неявного виміру знань методів розробки систем.

Сферу розробки методів було змінено, щоб відобразити обґрунтування побудови методу. Насправді поняття обґрунтування побудови методу дуже мало відрізняються від поняття публічного обґрунтування. Результатом є поєднання в обґрунтування побудови публічного методу. Обидва аспекти однаково важливі для точного визначення того, що являє собою метод, окремо від будь-яких ситуацій використання. Ця нова категорія містить філософські парадигми, описані на рисунку 1.6, а також основні цілі та цінності. Разом вони пояснюють, чому були прийняті певні дизайнерські рішення щодо ідеальної типової метамоделі.

Процес збору вимог для адаптації ідеальних типових метамоделей у ситуаційні методи також висвітлюється у сфері методотехніки. Важлива відмінність полягає в тому, що вхідні дані для вимог походять не лише від ситуації розвитку, але й від приватної раціональності акторів.

Друга сфера – це, звичайно, сфера методу в дії. Ця сфера в основному визначає нечітку ситуацію розвитку від ME за допомогою термінології ISD. Це також дозволяє адаптувати методи розробки систем через використання; те, що вважається випадковим у сфері ME, але все ще представляє широко поширену поведінку згідно з дослідженнями ISD. Використання методу у сфері методу в дії означає, що актори реалізують цілі та основні цінності шляхом застосування методу в різних ситуаціях розвитку. Це також означає, що на користувача методу впливає його/її особиста раціональність, яка знаходиться в сфері рефлексії методу, оскільки цілі та цінності використовуються в герменевтичному процесі взаємодії між сферою методу в дії та сферою рефлексії методу.

Третя сфера все ще вважається сферою рефлексії методу, але рефлексія поділяється на дві частини: одну частину, яку можна легко передати, і іншу, яку важче повністю висловити. Це робиться шляхом оцінки приватного виміру раціональності, вираженого полем ISD з одним важливим уточненням. По суті, наміри є причинами для дій і, таким чином, орієнтовані на ціль. Цілі не існують самі по собі. Ціль опису вказує на те, що хтось з певних причин вважав цінним. Таким чином, елемент наміру поділяється на та визначається як поєднання цілей і цінностей.

Висновки до розділу

В даному розділі проведено огляд літератури, який є основою для визначення обґрунтування методу та формулювання поля EME. Він також охоплює обговорення цілей для концепції методу та їх застовувань. Проведено пояснення підхід наукового дослідження дизайну, який застосовано у розвитку концепцій обґрунтування методу та компонентів методу.

РОЗДІЛ 2. СТРУКТУРИЗАЦІЯ РАЦІОНАЛЬНОСТІ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНИХ РІШЕНЬ

2.1. Визначення обґрунтування методу розробки

В попередньому розділі показано, як концепція обґрунтування методу розглядалася в двох областях ME та ISD. Залишається визначити концепцію таким чином, щоб вона могла сприйматися як значуща в обох галузях дослідження. Це абсолютно необхідно, якщо майбутні дослідження матимуть можливість описувати нові області та проводити дослідження більш загальної орієнтації методів розробки систем. Концептуалізація також сприяла б передачі знань між дослідниками, які зазвичай належать до будь-якої галузі, і могла б служити спільною основою для розробки нових типів досліджень методів розробки систем, досліджень, які зможуть подолати межі між областями, які раніше вважалися як несумірне. Поняття обґрунтування методу, що складається з цілей і цінностей, дає модель обґрунтування методу, як показано на рисунку 2.1.

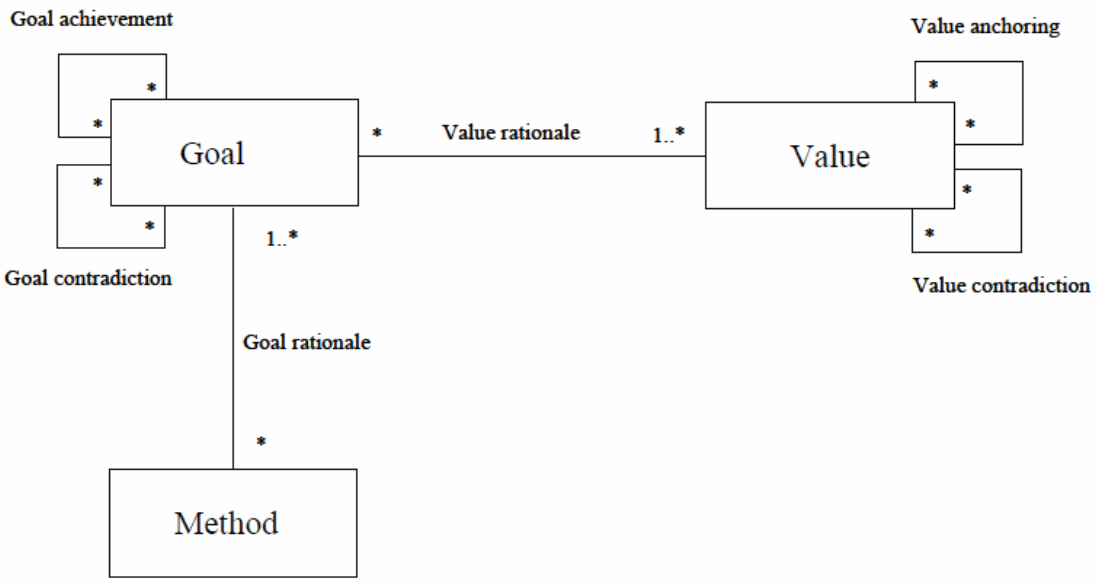

Рис. 2.1. Обґрунтування методу, що складається з цілей і цінностей

Згідно з цим міркуванням, пропонується два різних типи суб-обґрунтування, що формує обґрунтування методу:

1) метод, заснований на цілях, який називається обґрунтуванням цілі,

2) цілі, закріплені на цінностях, відповідно називаються обґрунтуванням цінності.

Асоціація обґрунтування мети виражає те, що метод має мету, пов'язану із запропонованою діяльністю. Все в методі розробки систем призначається з причини [10]. Співвідношення обґрунтування мети виражає те, що запропоновані дії призначені для досягнення цілей, досягнення яких розробник методу вважає важливим. Цілі на цьому рівні зазвичай дуже загальні. RUP стверджує, що метою є «…виробляти, у межах передбачуваного графіка та бюджету, високоякісне програмне забезпечення, яке відповідає потребам кінцевих користувачів» [11]. Цілі насправді не говорять про те, як ви повинні це зробити, але вони стверджують, що ви досягнете певного цільового стану, дотримуючись запропонованого методу.

Цілі можна розглядати як бажані цільові стани. Це означає, що певні цільові стани вважаються цінними для досягнення. Асоціація обґрунтування цінності представляє відношення цільових станів або цілей до їхніх основних цінностей. Приклади цінностей у RUP: «клієнти заслуговують на високоякісне програмне забезпечення» або «клієнти заслуговують на систему, яка відповідає їхнім потребам». Отже, за міркуваннями стоїть ціннісне судження, яке мотивує, чому бажано досягти кожного з цільових станів або цілей, яких намагається досягти метод.

Крім того, цілі можуть бути пов'язані одна з одною в ієрархії цілей. Прикладом цього є коли мета розглядається як засіб для досягнення іншої (вищої) мети. Подібним чином значення можуть бути прив'язані до інших значень. Я визначаю ці дві властивості обґрунтування як досягнення мети та закріплення цінності відповідно. Окрім досягнення мети, існує ймовірність того, що цілі суперечать, а не доповнюють одна одну – отже, існує додаткове відношення протиріччя цілей, визначене для набору цілей. Подібним чином

існує відношення ціннісної суперечності, визначене над набором значень. У добре продуманому методі розробки систем ви не обов'язково знайдете будь-які протиріччя такого типу. Однак ми повинні припустити, що існують можливості того, що протиріччя такого типу можуть існувати в методі розробки системи.

Як показано в нотації діаграми класів UML на рис. 2.1, метод пов’язаний принаймні з однією ціллю, а кожна ціль пов’язана принаймні з одним значенням. Це означає, що будь-який метод має прямо чи опосередковано сприяти досягненню загальної мети використання методу. Це також означає, що кожна мета має відповідник принаймні в одній цінності. З аналітичної точки зору це означає, що причиною використання конкретного методу може бути:

а) його внесок у досягнення інших цілей вищого рівня,

б) його реалізація (частин) основної філософії методу розробки систем, виражена ідентифікованими значеннями.

Цілі, що не належать до жодної категорії (називаються внутрішніми цілями), є цілями самі по собі і є сумнівними в процесі раціонального розвитку. Однак не можна очікувати знайти будь-які внутрішні цілі в доступному методі розробки систем, такому як, наприклад, RUP. Однак, щоб повністю зрозуміти концепцію обґрунтування методу, ми повинні припустити, що можуть існувати суперечності між цілями або цінностями. Тому ми включили ці асоціації в нашу модель замість обґрунтування методу на рис. 2.1. Якби ми цього не зробили, модель була б концептуально неправильною. Це визначення обґрунтування методу може бути використане в будь-якому напрямку (від сфери розробки методів і вниз, або від сфери відображення методу і вгору) у моїй синтезованій моделі на рисунку 1.6, уможливлюючи подвійний зворотний зв’язок через використання роздумів про метод стосовно його цілей. і цінності у сфері рефлексії методу та можливість виразити приватне обґрунтування використання методу через цілі, які він виконує, і цінності, яких він досягає, у спосіб, який може бути

корисним у сфері розробки методів. Це ключовий момент у моєму синтезі, оскільки він дає можливість передавати знання методів у всіх трьох сферах. Зі сфери розробки методу цілі та цінності передають знання методу через обґрунтування конструкції методу в сферу метод-бездіяльність. Цю концепцію методу можна використовувати в галузі ЕМЕ, оскільки вона створить типову інженерну концепцію методу для використання в областях, де вони раніше не застосовувалися.

Використання зовнішнього методу розробки систем означає, що користувач методу намагається досягти певних цілей і вважає, що метод надає належну підтримку його/її зусиллям. Цілі, яких він хоче досягти, вважаються корисними для створення системи, яку він/вона хоче створити. Оскільки метод розробки систем також має цілі, використання методу є спробою створити гармонію між цілями, які забезпечує цей метод, і цілями розробника. На дуже грубому рівні прикладом цього може бути те, що розробник намагається створити систему, яка відповідає потребам клієнтів. Якщо розробник потім вирішить використовувати метод розробки систем, який стверджує, що націлений на цю мету, існуватиме можливість гармонізації між цілями.

Такий стан називається резонансом раціональності [12] і описує стан, коли публічна раціональність методу збігається з приватною раціональністю актора в ситуації використання. Потім актор може використовувати цей метод добре поінформованим способом.

Рисунок 2.2 описує, як розробник також має цілі та цінності, які формують основу його намірів, так само, як метод розробки систем є проявом намірів творця методу. Оскільки велика частина концептуальних моделей відповідає, природно бачити, що можна досягти резонансу раціональності. По суті, це означає, що існує додатковий зв’язок між методом, розробником і метою. Це дасть концептуальну діаграму класів, як показано на рисунку 2.3.

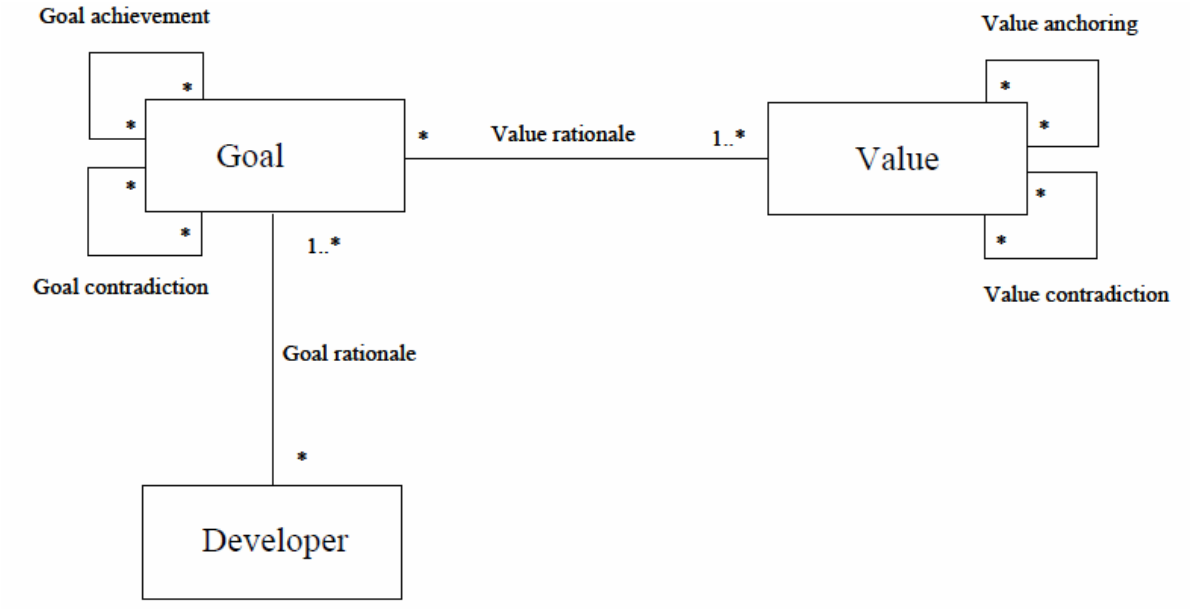

Рис. 2.2. Наміри розробника щодо цілей і цінностей

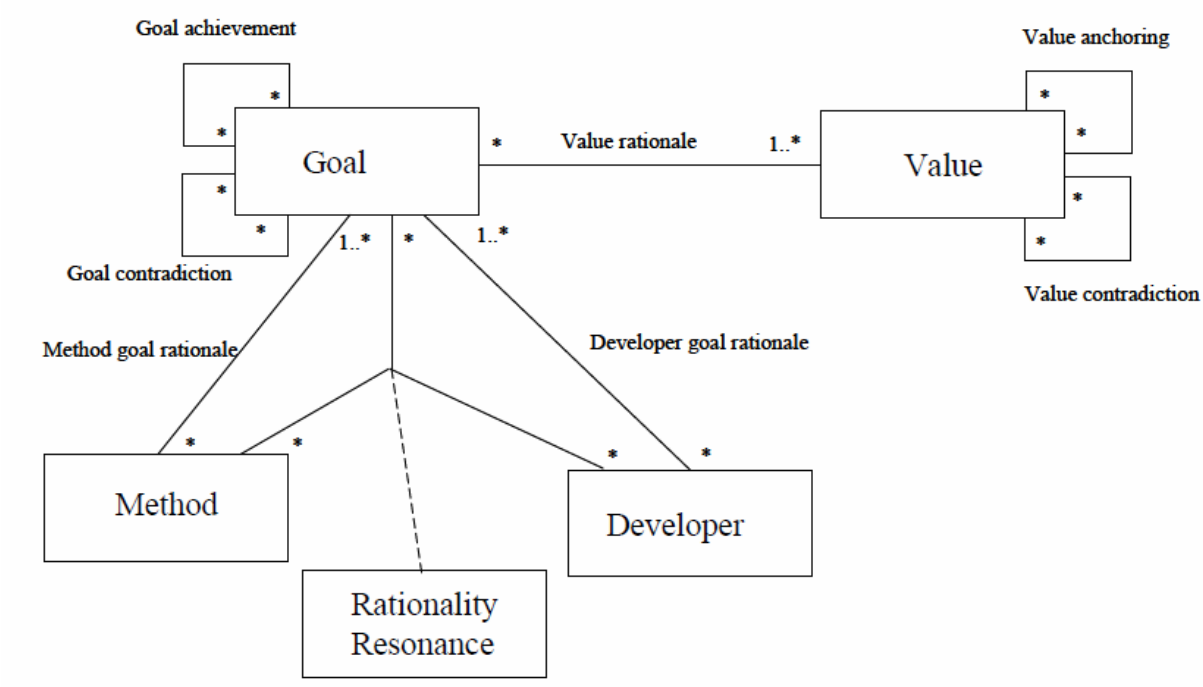

Рис. 2.3. Відповідність методу та намірів розробників, що створює резонанс раціональності

Як показано, резонанс раціональності виникає, коли існує відповідність між видами діяльності, запропонованими методом, намірами розробника та конкретною спільною метою, метою, яка базується на загальній основі

цінностей. Причини вибору множинності полягають у тому, що цілком можливо, що резонанс раціональності насправді може не відбутися, а також він може виникнути кілька разів під час дії методу.

Альтернативне розуміння того, що відбувається під час стану резонансу раціональності, можна пояснити за допомогою моделі, зображеної на рисунку 2.4. У моделі резонанс раціональності представлений як перетин двох видів раціональності. У реляційній алгебрі вираз публічна раціональність і приватна раціональність описує їх перетин.

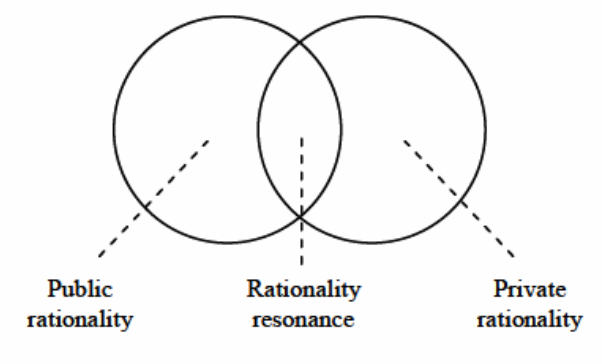

Рис. 2.4. Резонанс раціональності

2.2. Опис концепції фрагменту методу розробки

Концепція фрагменту методу, запропонована в [13]. По суті, фрагменти методу мають на меті функціонувати як концепція модуля для розробки методу. Таким чином, їх можна визначати та маніпулювати ними за допомогою мови розробки методів (MEL - method engineering language). Основна ідея полягає в тому, щоб побудувати новий метод розробки системи для конкретної ситуації для кожного проекту за допомогою фрагментів методу, що зберігаються в базі даних, яка називається базою методів.

Фрагменти методу в певному сенсі можна розглядати як самодостатні модулі, оскільки вони мають можливість класифікувати фрагменти як такі, що стосуються продукту або процесу. Фрагмент методу, який є достатньо великим, щоб охопити обидві ці перспективи, можна розглядати як

самодостатній. Однак немає правил, які б стверджували, що кожен фрагмент методу повинен мати відповідні перспективи в цьому сенсі, а також що кожен фрагмент методу повинен містити достатньо інформації для фактичного створення того, що можна вважати результатом. Пам'ятайте, що фрагмент методу може існувати на багатьох рівнях деталізації, і можна розглядати окрему концепцію чи символ як власний фрагмент методу. У цьому сенсі не можна сказати, що фрагменти методу завжди мають здатність функціонувати як самодостатні сутності.

Основна ідея фрагментів методів полягає у створенні нових методів розробки систем для окремих проектів. Це не завжди можливо, оскільки навчання методу часто потребує часу, а зусилля з розробки унікального методу розробки систем для кожного проекту можуть бути занадто дорогими. Щоб функціонувати як будівельні блоки для цього завдання, фрагменти методу мають властивість бути надзвичайно гнучкими. У той час як багато зусиль з адаптації методу мають тенденцію опускати або включати дії, щоб знайти добре функціонуючий процес розробки, прихильники фрагментів методу готові винаходити абсолютно нові дії методу та результати [14]. Це призводить до нестабільних модулів методу, які не відповідають вимогам, встановленим у цілях проектування, оскільки їх не можна вважати стабільними з часом. Їх не можна розглядати як такі, що мають вільні кінці, оскільки кожен фрагмент може мати різний розмір, починаючи від повного методу і закінчуючи окремою концепцією. Таким чином, можна мати фрагменти методу, які не мають жодного сенсу, якщо їх вирвати з контексту. Крім того, оскільки фрагменти можуть бути різного розміру, вони не відповідають вимогам конгруентності, що ускладнює створення однорідного методу розробки систем із блоків, які дуже відрізняються один від одного. У своєму аналізі вирішено визначити рівень деталізації для моєї концепції модуля методу, що відповідає рівню діаграми відповідно до фрагментів методу. Іншим способом інтерпретації цього було б

сказати, що аргументація поки що може визнати лише фрагменти методу, які знаходяться на цьому рівні, як придатні для розгляду.

2.2.1. Аналіз концепції послідовності методів

Концепція фрагмента методу — це ще одна концепція модуля методу, яка походить із галузі розробки методів. Частка — це спроба створити концепцію модуля з тіснішим зв’язком між процесом і продуктом. У концепції послідовності ці два типи фрагментів методу об’єднані в один модуль. Таким чином, будь-який метод розробки системи можна розглядати як набір фрагментів методу на різних рівнях деталізації, таким чином цілий метод можна розглядати як фрагмент так само, як фрагмент методу може становити цілий метод розробки системи.

Оскільки фрагменти методу можуть бути визначені на різних рівнях деталізації, вони мають ту саму помилку порівняно з цілями дизайну, що й фрагменти методу. Послідовність методу завжди має відповідний фрагмент процесу для кожного фрагмента продукту, і в цьому сенсі їх можна назвати самодостатніми, оскільки вони забезпечують і продукт, і процес виробництва цього продукту. Однак, оскільки фрагмент методу може мати будь-який рівень деталізації, він може давати модулі методу, які не здаються самодостатніми, оскільки їх може бути важко інтерпретувати поза контекстом.

Основна мета блоків методів полягає у створенні нових методів розробки систем для кожного окремого проекту, так само, як фрагменти методів, і з тими ж труднощами, що й цей підхід з точки зору вартості. Невизначений рівень деталізації також призводить до ситуації, коли фрагмент методу не завжди можна вважати послідовним і узгодженим, оскільки він може бути занадто малим для перенесення будь-яких глибших знань про розробку систем. Подібно до фрагментів методу, фрагмент методу можна моделювати, доки він не матиме жодного значення, оскільки він поза контекстом.

Блоки визначаються з точки зору процесу. Це означає, що, порівняно з концепцією фрагментів методу, ядром кожного блоку є керівництво для конкретного процесу та пов’язаних із ним фрагментів продукту [15]. Потім фрагменти використовуються для створення нових методів подібно до фрагментів методів. У цьому сенсі можна сказати, що блоки методів досягли мети розробки підключення. Однак їм не вистачає цільової спрямованості, визначеної метою дизайну. Порівняно з фрагментами методу, фрагменти методу мають більше спільного з вимогами, встановленими проектом, оскільки вони включають вказівки з відповідними намірами. Але, оскільки фактичні цілі намірів приховані, їх не можна розглядати як повне дотримання мети дизайну підключення.

2.2.2. Аналіз концепції компонента методу

У сфері ISD також існують концептуалізації методичних модулів. Вони не обов’язково визначені для розробки методів, оскільки їх також можна використовувати для розуміння того, що таке метод розробки систем і що він містить. Часто виступають за використання компонентів методу як концепції модуля для методів розробки систем. Порівняно з концепцією фрагмента методу та концепцією блоку методу, компонент методу займає певну позицію, коли йдеться про питання деталізації модулів методу. Компонент методу приймає початкову точку на рівні деталізації, що відповідає рівню діаграми в концепції фрагмента методу. Таким чином, вони відповідають тому, що ми шукаємо.

Компоненти методу мають на меті поєднати стабільність із гнучкістю. Основна ідея компонентів методу полягає у використанні досить високого рівня абстракції та структуруванні вмісту компонента на основі внутрішніх залежностей. Іншими словами, компонент методу має бути самодостатнім. Крім того, компонент завжди складається з понять (про що говорити), процедури (які запитання поставити) і нотацій (як виразити відповіді). Це дає

компонентам методу можливість розглядатися як самодостатні, оскільки вони містять достатньо в собі, щоб сприйматися як значущі.

Компоненти методу взяли відправну точку на визначеному рівні деталізації. У результаті цього вони представляють конгруентну картину концепції методичного модуля. Вони є внутрішньо послідовними та послідовними, оскільки всі вони містять аспекти, які пояснюють, з чим працювати, як працювати та як записувати результати. Оскільки всі компоненти методу мають таку структуру, вони можуть представити послідовну картину того, як слід виконувати певні дії з розробки. Отже, компонент методу може бути змодельований із більшого методу розробки систем і все ще сприйматися як значимий. Вимоги до класифікації модуля методу як компонента методу полягають у тому, чи представляє він узгоджену картину для обробки конкретної та обмеженої проблеми в проекті розробки системи.

Компоненти методу — це конструкція, яка має на меті поєднати стабільність із гнучкістю, тобто кожен компонент можна розглядати як окрему одиницю або будівельний блок для побудови нових методів (стабільність), але в той же час їх має бути легко моделювати на основі вже існуючих. існуючий метод і використовується в інших контекстах. Приклад такого компонента методу можна знайти в компоненті методу діаграми проблем [16]. Цей компонент можна використовувати в кількох контекстах, оскільки це спосіб структурувати проблеми, щоб знайти корінь проблеми. Проблеми можуть існувати в бізнесі, системі чи, можливо, в чиємусь особистому житті. Компонент методу діаграми проблеми може бути використаний у всіх цих контекстах, і тому вони є гнучкими. Повний метод розробки системи можна розглядати як контейнер або сукупність кількох компонентів методу, а метод можна вважати складеним із компонентів методу, які з’єднані один з одним. Мета розробки підключення стверджує, що модуль методу повинен мати можливість висловити, як він сприяє досягненню загальної мети для зусиль з розробки системи. Як уже було

сказано, перспектива компонента методу дійсно містить цілі та цінності, однак недостатньо чітко, щоб використовувати їх як інтерфейс між компонентами таким чином, щоб виконати мету розробки підключення.

2.2.3. Аналіз вимог до концепції стартового модуля

Аналіз фрагментів методу, фрагментів методу та компонентів методу призначений для того, щоб служити основою для прийняття рішень для вибору початкової концепції модуля. Фрагменти методу та блоки методу походять із області розробки методів, що зробило б їх ідеальними для спроби переробити концепцію модуля методу для EME. Компоненти методу скоріше походять із сфери ISD і тому повинні бути піддані вищому ступеню формалізації, щоб бути корисними та прийнятими.

Аналізуючи кожну проектну мету окремо, стає очевидним, що існують певні фактори, які роблять одну концепцію модуля більш прийнятною, ніж іншу. Мета проектування — наявність автономної концепції модуля методу — підкреслила проблему різного рівня деталізації фрагментів методу та компонента методу. Ця варіація може призвести до модулів, які не здаються самодостатніми, оскільки вони містять недостатньо для того, щоб сприймати їх як значущі самі по собі. Компоненти методу мають мінімальний набір складових, які відповідають рівню деталізації діаграми [17] відповідно до фрагментів методу. Ця властивість робить компоненти методу більш придатними як початкову концепцію модуля, оскільки він уже має «правильний» розмір. З іншого боку, ми могли б просто вирішити використовувати фрагмент методу або блок методу як початкову концепцію модуля і просто вказати, що він повинен мати розмір, який відповідає рівню деталізації діаграми.

Мета проектування внутрішньої послідовності та узгодженості тісно пов’язана з рівнем деталізації, оскільки модуль методу повинен містити достатньо вмісту, щоб він сам по собі сприймався як значимий. Фрагменти методу та фрагменти методу не відповідають цій вимозі з тієї самої причини,

через яку вони не відповідають меті розробки бути самодостатніми. Їх можна визначити в настільки малому вигляді, що самі по собі вони не мають жодного значення. Дотримуючись пропозицій щодо визначення фрагмента методу відповідно до [18] фрагмент методу можна визначити як такий, що знаходиться на концептуальному рівні деталізації, має перспективу продукту відповідно до перспективного виміру та має концептуальний рівень абстракції. Прикладом цього може бути такий фрагмент методу:

Рис. 2.5. Приклад фрагмента методу

Що насправді представляє чи зображує рисунок 2.5. Більшість людей скажуть, що це зображення людини. Але що означає людина ? Хто це намалював і чому? Ці запитання виникають через те, що фрагмент методу був змодельований у надто малий розмір, щоб мати будь-яке значення, оскільки він поза контекстом. Семантику символу людини необхідно інтерпретувати в контексті, який би надавав йому значення. Однак, якби символ з’явився на діаграмі під час проекту розробки системи, більшість людей, які беруть участь у проекті та знайомі з UML розпізнали б і інтерпретували символ як актора, деперсоналізоване представлення когось які можуть взаємодіяти з системою. Це показує, наскільки важливо мати контекст для явищ, які мають бути зрозумілими та несуть значення.

Фрагменти методу завжди мають відповідний фрагмент процесу для кожного фрагмента продукту, і отже, можна вважати такими, що досягли мети розробки внутрішньої узгодженості та узгодженості більшою мірою, ніж фрагменти методу. Поєднання фрагментів продукту та процесу завжди дає більш послідовне та узгоджене зображення модуля методу. Однак через невизначений рівень деталізації завжди існує ризик того, що метод розробки системи може сприйматися як надто фрагментований, якщо він визначений у

фрагментах методу, які є надто малими, щоб самі по собі сприйматися як значущі.

У цих термінах концепція компонента методу відповідає меті проектування, оскільки вона не представляє фрагментованого погляду на метод розробки систем. Компонент методу сам по собі є послідовним за своїм змістом і представляє цілісну картину методичної діяльності.

Можна вважати, що мета проектування зв'язності частково досягнута всіма трьома концепціями модуля методу. Фрагменти методу визначаються з точки зору процесу, і тому вони більше підходять для вирішення проблем підключення, ніж фрагменти методу. Проте мета розробки стверджує, що з'єднання модулів методу має бути засноване на інтерфейсах, які дозволяють виразити, як модуль методу може сприяти ланцюжку досягнення мети.

Усі три концепції модуля методу досягли мети розробки застосовності. Їх можна відобразити на будь-якому існуючому методі розробки систем. Якщо в концепції модуля методу є аспекти, які не можуть бути охоплені самим методом, завжди є можливість реконструювати ці аспекти.

2.2.4. Висновок щодо концепції модуля методу запуску

Ґрунтуючись на наведеному вище аналізі, ми виявили, що ми можемо використовувати будь-яку з проаналізованих концепцій модуля методу, оскільки як фрагменти методу, так і фрагменти методу можна перевизначати, щоб завжди мати необхідний рівень деталізації, який ми вирішили вважати. З іншого боку, компоненти методу вже мають таку якість, що призводить до висновку, що було б легше вибрати компоненти методу як початкову концепцію модуля, оскільки вони завжди розглядаються як послідовні та узгоджені модулі методу. Вибраний рівень деталізації сильно вплинув на наш процес прийняття рішень, оскільки він впливає на численні цілі дизайну. Якщо концепція модуля методу не має рівня деталізації діаграми, це може призвести до того, що модуль може сприйматися як безглуздий поза контекстом. Це впливає на можливості досягнення проектних цілей

автономності, внутрішньої послідовності та узгодженості, раціональності та зв’язності. Іншими словами, всі цілі дизайну, крім застосовності. Ця непряма проблема знову вказує на напрямок компонентів методу.

Фундаментальним моментом для оновленої концепції модуля методу є його можливості нести та виражати обґрунтування методу. Навіть якщо фрагмент методу має рівень деталізації діаграми, необхідно додати цей вимір раціональності. Компонент методу вже має цю властивість, хоча вона потребує більшої формалізації та чіткості. Замість того, щоб розглядати цілі та цінності в перспективі як щось, що лежить за кожним компонентом методу, нам потрібно розглядати їх як інтегровані частини в модулі методу та пояснювати, як ці цілі та цінності пов’язані та мотивують структуру та вигляд кожного окремого методу. модуль. Таким чином, ми повинні мати можливість створити концепцію модуля, яка є конгруентною з точки зору структурних питань у порівнянні з іншими модулями методів подібного роду. Для кожного з окремих модулів методу ми також повинні мати можливість визначити, які цілі та цінності пов’язані із запропонованою діяльністю та її результатами. Іншими словами, цільові зв’язки між цілями та ще невизначеними частинами модуля методу повинні бути визначені та зрозумілі, як показано на рисунку 2.6.

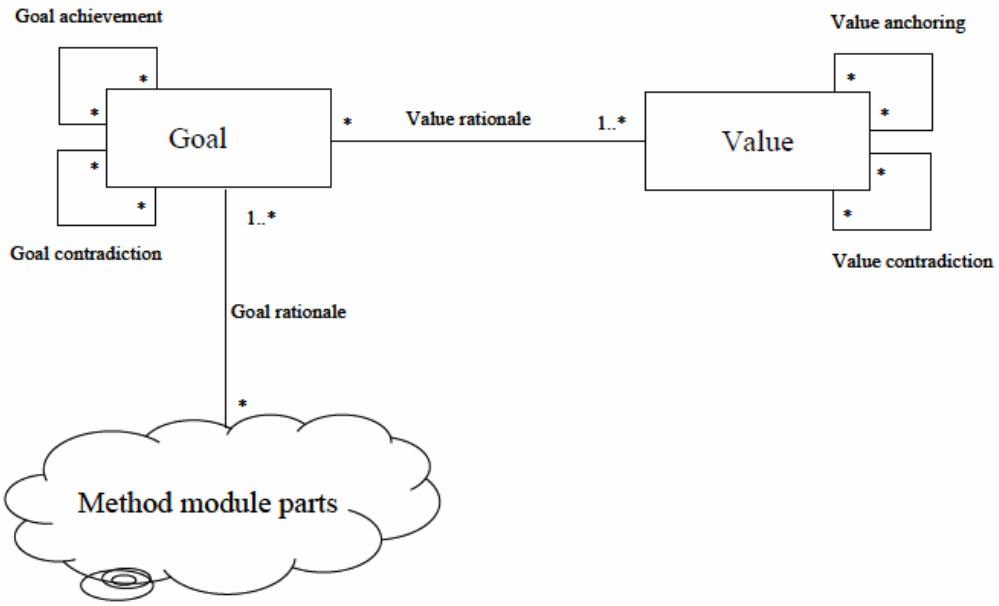

Рис. 2.6. Обґрунтування методу щодо невідомих частин модуля методу

У задачі перепроектування існуючої концепції методу в концепцію модуля методу, придатну для ЕМЕ і можливу для дослідження в усіх трьох сферах застосування методу, а саме сфері розробки методу, сфері методу в дії, і метод відображення сфери. Фрагменти методу та фрагменти методу вже широко відомі в області розробки методів, тоді як компоненти методу зазвичай не використовуються в діяльності з розробки методів і дослідженнях. Шанси на переробку модуля такі концепції, як фрагменти методів або фрагменти методів для ЕМЕ та отримання визнання та підтримки, будуть невеликими, оскільки ці концепції вже широко використовувалися в різних дослідницьких проектах з розробки методів. У нас буде більше шансів на успіх, якщо ми вирішимо запровадити нову концепцію, або, принаймні, концепцію, яка є новою в області розробки методів, таку як концепція компонента методу. Окрім цієї очевидної перешкоди, наш аналіз трьох концепцій модуля методу вказує на те, що компоненти методу ближчі до наших цілей проектування, ніж інші концепції модуля методу. Вони мають «правильний» рівень деталізації з самого початку, що надає їм самодостатній, внутрішньо послідовний і конгруентний вигляд, вони мають вимір раціональності, подібний до моєї інтерпретації та визначення обґрунтування методу, і їх можна зв’язати один з одним, коли їхні інтерфейси були визначені за допомогою явно створеного розміру обґрунтування. Вони також відповідають цілям розробки застосовності, оскільки їх можна відобразити на будь-якому існуючому методі розробки систем.

2.3. Концептуальне перемоделювання компонента методу

У цьому розділі ми проведемо концептуальне перемоделювання існуючої концепції компонента методу, щоб отримати вдосконалену концепцію компонента методу, придатну для ЕМЕ. Одним з важливих аспектів цього завдання була формалізація концепції компонента методу,

щоб дозволити маніпуляції. Ми вирішили формалізувати оновлений компонент методу за допомогою UML, щоб компоненти можна було зберігати, отримувати та маніпулювати ними. Компонент методу можна розуміти так, як це зображено нижче на рисунку 2.7. Він описує структуру та складові компонента методу з точки зору до окремих концепцій, які застосовуються в компоненті методу.

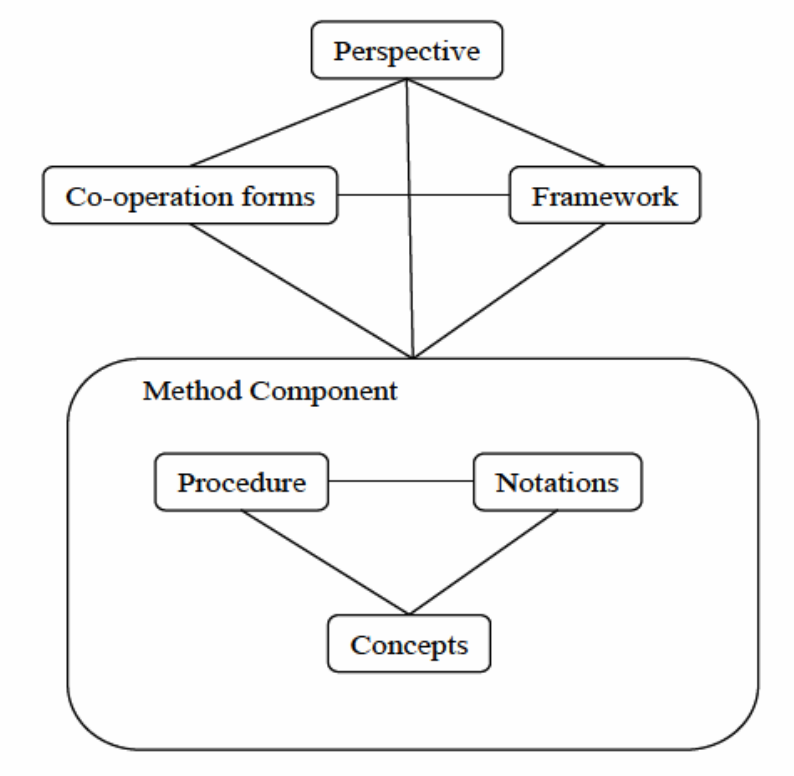

Рис. 2.7. Оригінальна конструкція компонента методу

Перше, що можна помітити, аналізуючи компонент методу, це те, що він має ядро, що складається з трьох елементів; процедура, позначення та поняття. Вони розглядаються як істотна частина компонента методу та частини, які фактично несуть знання про те, які дії мають відбуватися, коли використовується компонент методу. Поза компонентом методу ми знаходимо форми співпраці, рамки та перспективу. В основному це те, про що ми маємо на увазі, коли виявляємо, що вимір раціональності компонентів методу надто відокремлений від його наказового змісту. Невідомо, як

перспектива пов’язана зі складовими основних елементів компонентів методу.

Але почнемо наш аналіз із суті. Концепції є, мабуть, найважливішими елементами концепції компонента методу і вони повинні бути безпосередньо пов'язані з перспективним виміром, оскільки концепція є результатом концептуалізації. Визначення перспективи [19] містить визначення, категорії, цінності та цілі. Класифікувати різні явища та давати їм визначення означає концептуалізувати. На процес концептуалізації впливають цілі, яких ви намагаєтеся досягти, роблячи це, і людина зазвичай має явну чи приховану базу цінностей, яка змушує її цінувати одну мету над іншою. Таким чином, цілі та цінності впливають на процес концептуалізації. Результати, звичайно, є набором понять, які вважаються корисними в певному компоненті методу. Таким чином, концепції мають чітке відношення до перспективи і можуть бути описані як «те, про що говорять» у компоненті методу. Процедури — це директиви дій, які повідомляють користувачеві компонента методу, що робити. Між поняттями та процедурами існує зв’язок, оскільки питання, визначені в процедурах, стосуються понять. Навпаки, знання визначення поняття дало б змогу людині зрозуміти, які питання слід поставити, щоб знайти явища розвитку систем, які відповідають визначенню поняття. Прикладом цього є концепція Entity з моделювання ER [20]. Сутність у вільних термінах розуміється як повторне явище, про яке було б корисно знати та фіксувати в системі. Сутність також можна однозначно ідентифікувати порівняно з іншими сутностями такого ж типу. Суб’єкт також має певні властивості, які важливо знати та записувати. Прикладом сутності може бути Клієнт із відповідними властивостями, у якому записується ім’я, номер клієнта, адреса, номер телефону тощо. Знання визначення полегшує розуміння запитань, які потрібно поставити, щоб визначити, чи можна розглядати певне явище як клієнт чи ні. По суті, вам просто потрібно поставити запитання, які можуть відповісти, чи має явище риси, які

відповідають визначенню поняття, наприклад «чи було б корисно знати про це явище, що знову з’являється, і записати його в систему» тощо.

Нотація — це набір правил, символів і синтаксису для того, як результати мають бути зафіксовані та передані. Відношення до понять полягає в семантичному представленні понять у нотації.

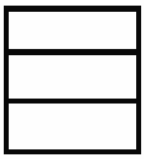

Рис. 2.8. Клас аналізу з RUP

Символ вище представляє клас аналізу з методу розробки систем RUP (Rational Unified Process). Символ являє собою квадрат із трьома полями. Сам по собі символ не є класом як таким, а просто представленням чогось. Значення символу контекстуально визначається семантикою, наданою символу творцем символу, у цьому випадку особами, відповідальними за створення RUP. Семантика відповідає визначенню, яке має представляти символ. У цьому випадку визначення таке: «Клас — це опис набору об’єктів, які мають однакові обов’язки, відносини, операції, атрибути та семантику». Визначення – це те, що мається на увазі, коли символ використовується в контексті розробки систем за допомогою RUP. Відношення між позначенням і поняттями можна порівняти з трикутником елементів значення. Таким чином, зв'язок між позначенням і поняттями є сильним, очевидним і ясним. Відношення між нотацією та процедурами залежить від знання понять. Звичайно, знання того, які типи запитань задавати, означає, що користувач методу знає, які поняття він/вона намагається знайти, і він/вона також, ймовірно, знає, які символи він повинен використовувати, щоб представити свої висновки за допомогою певних систем. контекст розвитку. Однак у своїй оновленій концепції компонента методу я вирішив називати процедури діями, оскільки ми вважаємо цей термін більш придатним.

Цей концептуальний аналіз оригінальної концепції компонента методу показує, що існує тісний зв'язок між процесом і елементами продукту. Оскільки ми намагаємося створити концепцію компонента методу, яку можна розглядати як самодостатню, внутрішньо узгоджену та узгоджену, щоб мати компоненти методу, які мають здатність нести значення самі по собі, важливо зберегти цей міцний зв'язок у ядрі нашої оновленої концепції компонента методу. Однак, оскільки ми маємо певний рівень деталізації, який потрібно враховувати, цей аспект потрібно додати. Професіонали методів інтуїтивно посилаються на артефакти, коли говорять про модулі методів. Крім того, оскільки артефакти самі по собі є ідентифікованими об'єктами, а також контейнером результатів у нотації, що використовується для створення артефакту, артефакт додається до нашої першої формалізованої версії переробленої концепції компонента методу. Формалізація виконується за допомогою діаграми класів з UML.

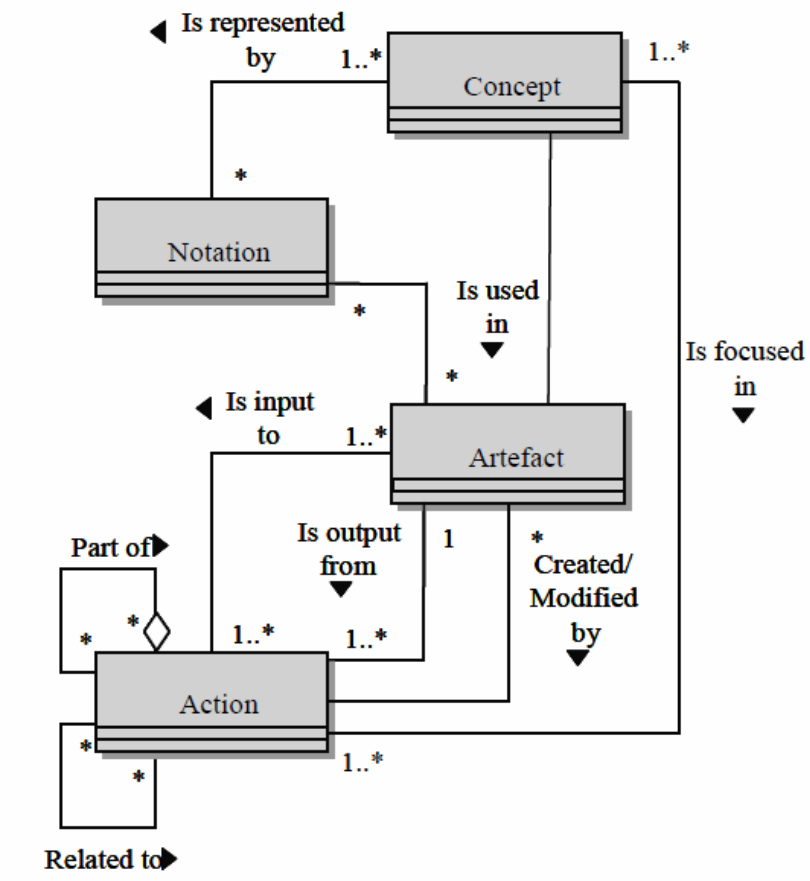

Рис. 2.9. Перероблена концепція компонента методу v.0.1

Оскільки артефакти часто використовуються для створення інших артефактів, ми додаємо дві асоціації, які мають можливість виражати, як, наприклад, діаграму варіантів використання можна розглядати як вхідні дані для діяльності зі створення моделі варіантів використання в RUP.

Наразі ми формалізували основну концепцію компонента методу з додаванням елемента артефакту та асоціацій для повторного використання артефакту. Результат можна побачити на рисунку 2.9.

Інша проблема з оригінальною концепцією компонента методу полягає в тому, що навіть якщо дії вказані, немає можливості вказати будь-яких акторів. Це викликає проблеми, оскільки багато сучасних методів розробки систем мають конкретні ролі акторів, тобто ролі, які люди, залучені до проекту розробки систем, беруть під час виконання певних дій. Прикладом цього є RUP, де є кілька ролей акторів, таких як системний аналітик, менеджер тестування або архітектор програмного забезпечення. Навіть гнучкі методи вказали такі ролі, як scrum master або власник продукту [22] або тренер екстремального програмування, програміст або трекер [23]. Це вказує на те, що було б доцільно включити ролі акторів у ядро оновленої концепції компонента методу.

Іншим аспектом, який стає очевидним, є необхідність формалізувати внутрішню структуру на окремі елементи методу. Роблячи це, ми можемо визначити перероблену концепцію компонента методу, щоб мати визначений набір складових. Це робиться за допомогою узагальнення асоціації між кожним окремим елементом методу (концепція, нотація, артефакт, дія та роль актора) та елементом методу класу контейнера. Контейнерний клас — це клас, який сам по собі не має екземплярів, лише екземпляри у формі інших класів, з якими він пов’язаний. Результатом цього є те, що ми можемо дотримуватися мети дизайну внутрішньої послідовності та узгодженості. Вважається, що компонент методу складається з елементів методу, показаних на рисунку 2.10, які разом залишають компонент методу з однорідним зовнішнім виглядом, що забезпечить конгруентність компонентів методу в

майбутньому. Результатом є концепція компонента методу, яка здатна функціонувати як самодостатня одиниця з можливістю бути значущою сама по собі і, таким чином, частково виконувати мету розробки самостійності. Внутрішню структуру компонента методу можна розуміти як сукупність визначених елементів методу, які разом стають ядром внутрішньої структури компонента методу. Це представлено через використання символу агрегації у формалізованому описі нашої переробленої концепції компонента методу на рисунку 2.10.

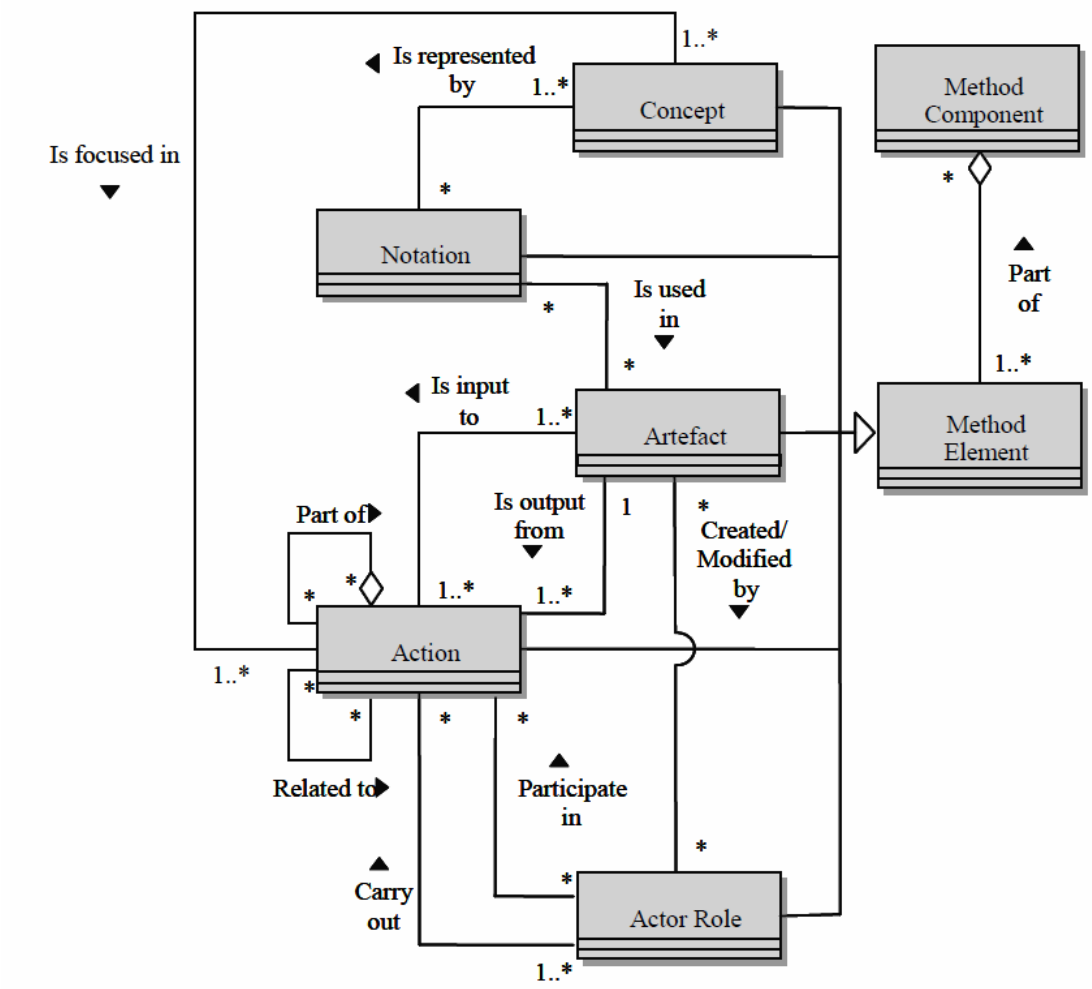

Рис. 2.10. Перероблена концепція компонента методу v 0.2.

Інший аспект, який необхідно розглянути, полягає в тому, щоб зробити зв’язок між компонентами методу, що лежать в основі перспективи, явним щодо його окремих елементів методу. В [23] описують перспективу, що

складається з категорій, визначень, цілей і цінностей. Однак, оскільки категорії та визначення є способами інтерпретації та позначення конкретних понять, немає очевидної потреби у формалізації цих частин у нашому описі UML. Концепції вже є, а категорії та визначення є частинами концептуалізації. Ці дві складові перспективи існують, але їх можна вважати вже присутніми в основі компонента методу, хоча й опосередковано. Крім того, поки що ми не маємо вказівок на те, що додавання категорій і визначень було б корисним для EME.

У перспективі залишаються лише цілі та цінності. Це фундаментальні частини обґрунтування методу і необхідні для того, щоб перероблені концепції компонентів методу могли розглядатися як справді самодостатні. Обґрунтування методу визначається як цільовий зв'язок між описом методу та цілями (обґрунтування цілі) і зв'язок між цілями та цінностями, на яких ґрунтуються цілі (обґрунтування цінності). Модулярність методу розробки систем на компоненти методу дозволить чітко сформулювати ці відносини по відношенню до кожного з елементів методу, які складають ядро компонента методу. Я вже стверджував, як первинні частини ядра (концепції, дії та позначення) пов'язані з перспективою, а доповнення, які я зробив після цього, пов'язані з перспективою подібним чином. Роль актора також є результатом цілеспрямованої та ціннісної категоризації, результатом якої є визначена концептуалізація.

Як наслідок, елемент методу актора також має чітке відношення до цілей і цінностей перспективи. Елемент методу артефакту вважається записом результатів, виражених за допомогою нотації, і містить різні символи або текстові описи, які відповідають концепціям (про що йдеться в документі) і діям, які виконуються під час створення документа (запитання, що ставляться). Крім того, артефакт можна розглядати як вмістилище інших артефактів. Під цим я маю на увазі документи в дуже вільному сенсі. Прикладом документа в цьому сенсі є те, що записується під час проектів розробки систем, щоб якось повідомити про це. Таким чином, пошук одного

класу аналізу слід вважати компонентом методу, оскільки пошук класів вважається розділеною діяльністю, оскільки вони призводять до окремих артефактів, які використовуються для створення нових артефактів. У RUP окремі класи записуються в діаграми, які пізніше використовуються для аналізу діаграм класів. Оскільки ніщо не обов'язково говорить про те, що всі класи повинні використовуватися в кожній діаграмі класів аналізу, вони повинні бути записані окремо та розглядатися як окремі компоненти методу, що знаходяться на рівні деталізації діаграми, оскільки навіть один клас аналізу може вважатися бути індивідуальним артефактом. Артефакти, які можна використовувати миттєво або пізніше в загальному процесі розробки систем для створення інших артефактів, таких як діаграми класів аналізу або дизайн бази даних. Це означатиме, що символ на рисунку 2.8 можна розглядати як артефакт із компонента методу класу аналізу. Таким чином, кожен артефакт має визначену мету свого існування та обґрунтування основного методу, який пояснює, чому артефакт створений.

Таким чином, можна розглядати всі елементи методу однаково щодо перспективних цілей і цінностей. У результаті ми додаємо асоціацію, що описує раціональне відношення цілі до елементів методу, і асоціацію обґрунтування цінності для опису відношення між цілями та цінностями. Ми також додаємо асоціації досягнення мети та цінності. Ми вирішили опустити асоціації конфлікту цілей і цінностей у формалізованій перепроектованій концепції компонента методу. Це не означає, що ми заперечуємо їх існування; це лише означає, що не обов'язково такі типи конфліктів будуть знайдені та їх необхідно зрозуміти. По суті, нам ще належить знайти такий метод розробки сучасних систем. Однак слід враховувати можливість того, що конфлікт такого типу цілком може існувати.

Нарешті, ми додаємо зв'язок між ціллю та агрегованим компонентом методу, що описує загальну мету компонента методу. Загальна ціль — це головна мета, яку досягне користувач методу, використовуючи даний компонент методу. Прикладом цього є Business Use Case Model від RUP.

Його загальну мету можна визначити як «узгодити бізнес-контекст для планування та подальшої розробки програмного забезпечення». На рисунку 2.11 зображено оновлену концепцію компонента методу. Таким чином, він представляє внутрішній погляд на те, як можна визначити та зрозуміти компонент методу.

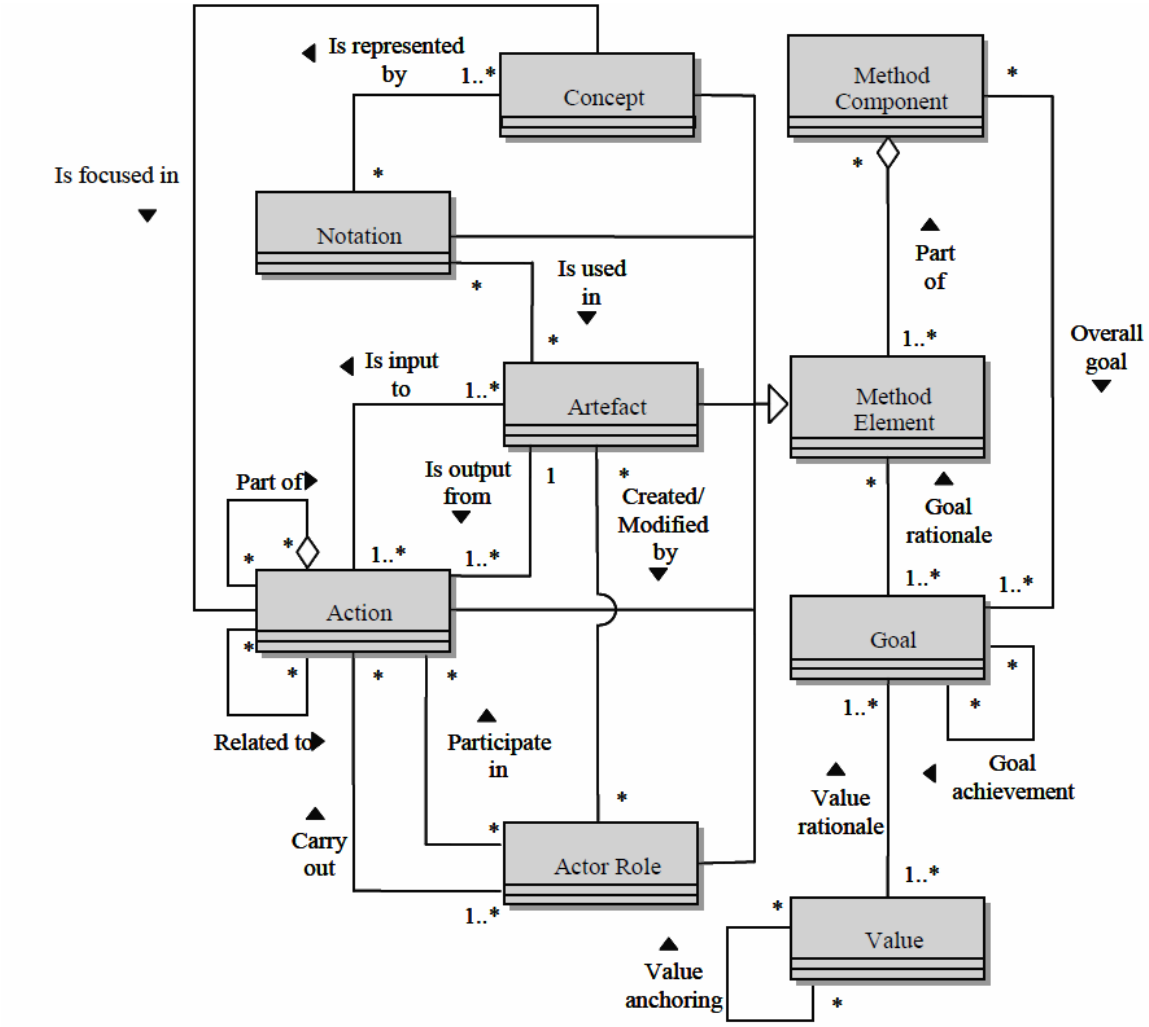

Рис. 2.11. Концепція компонента методу v 1.0 (внутрішнє представлення)

2.4. Зовнішня структура компоненти методу розробки

Поки що ми визначили внутрішню структуру компонента методу та зробили чітким зв'язок між цими структурами та базовою перспективою. Зробивши це, ми виконали цілі дизайну, самодостатність, внутрішню послідовність і узгодженість, а також раціональність.

Щоб досягти цієї мети дизайну, ми дозволяємо загальній меті кожного компонента методу сприяти досягненню цілей. Це пов’язує компоненти методу один з одним за допомогою обґрунтування методу. Рисунок 2.12. ілюструє, як слід сприймати перепроектовану конструкцію компонента методу, якщо розглядати їх як частини повного методу розробки систем і, отже, відповідати критерію зв’язності.

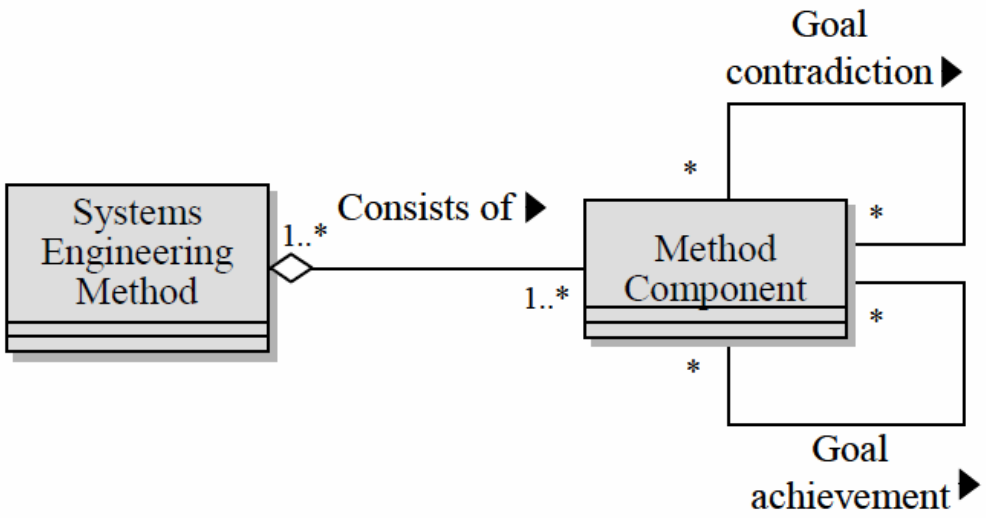

Рис. 2.12. Концепція компонента методу (зовнішнє представлення)

Завдяки зовнішньому погляду ми зосереджуємося на тому, як конкретний компонент методу сприяє досягненню цілей, поділяючи його обґрунтування через інтерфейс і додаючи до загальної мети проекту розробки системи. Це зображення знову формалізовано за допомогою UML. Відмінність у порівнянні з внутрішнім поглядом компонента методу полягає в тому, що на цьому рівні більша ймовірність виявлення протиріч цілей. Може бути так, що інженер намагається реконструювати неформальний або власний метод розробки, який, як виявилося, має такі типи протиріч. Можна також розглядати два компоненти методу, які виконують однакове завдання, як суперечливі, оскільки вони можуть змусити користувача методу виконати по суті те саме завдання двічі. Виконання однієї мети двічі навряд чи можна вважати позитивним по відношенню до загальної мети проекту розвитку в більшості випадків. Приклад цього можна знайти в UML, де діаграма

послідовності та діаграма співпраці мають однаковий кінцевий результат, а саме пояснюють потік взаємодії об’єктів порядку в системі.

Для вирішення відповідних аспектів цілі проектування зв’язності, коли компоненти методу мають бути з’єднані один з одним за допомогою обґрунтування методу, ми повинні визначити, як це робиться більш детально. Просто слідування UML не підійде, оскільки асоціація може бути довільно визначена між будь-якими двома класами. Отже, ми повинні визначити інтерфейс для концепції компонента методу, який може описати ланцюжок досягнення мети. Коли ми аналізуємо, що пропонує компонент методу з точки зору цілей, які вони досягають, ми визнаємо, що обґрунтування запропонованого методу виражається в результатах, записаних у вихідному артефакті компонента методу.

Крім того, вимоги компонента методу можуть бути виражені через його вхідні артефакти. Таким чином, ми вирішили визначити інтерфейс компонента методу як загальну мету та артефакти компонента методу. Останні класифікуються як передумови або результати. Використання нами терміна передумови не слід тлумачити як суворі передумови, які є обов’язковими. Ми розглядаємо методи як евристичні процедури або евритми, а отже, передумови слід розглядати як рекомендовані вхідні дані.

Висновки до розділу

В даному розділі досліджується роль, яку відіграє обґрунтування методу в ситуації, коли метод розробки системи приймається великою організацією розробки. Виконано концептуальний аналіз концепції компонента методу в спробі перепроектувати її для досягнення обраних цілей розробки. Під час цієї діяльності ми також формалізували оновлену концепцію компонента методу за допомогою UML.

РОЗДІЛ 3. РОЗРОБКА СТІЙКОЇ ІТЕРАТИВНОЇ МОДЕЛІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Дослідження компонента моделі бізнес-випадку

Опис моделі бізнес-випадків представлено з акцентом на дії, які прописані в контенті. Вміст – це те, що побачив би користувач методу, якби він застосував внутрішній вигляд до компонента методу. Фундаментальними елементами методу є позначення, концепції та дії, необхідні для виконання артефакту , який є результатом компонента методу. Крім того, у цій частині опису компонента методу ми знаходимо ролі акторів , які виконують ці дії.

Усі проблеми нотації вирішуються за допомогою UML [22], а концепції присутні через семантику символів і синтаксичні правила, які визначає UML, майже так само, як описано під час нашого концептуального перепроектування оригінальної концепції компонента методу . Таким чином, опис цих елементів методу буде опущено.

Інші аспекти, які не будуть розглянуті в компоненті прикладу методу, це індивідуальні цілі та цінності, які мотивують кожен окремий елемент методу. Незважаючи на те, що ці цілі та цінності повинні існувати з концептуальних міркувань, мало що вказує на те, що опис кожного компонента методу справді потрібно описувати так детально. Це можна вважати компромісом між точністю та вартістю. Рішення про впровадження конкретного компонента методу в організації, що займається розробкою, швидше базується на цілях і цінностях, які мотивують основний результат кожного компонента методу, первинний артефакт. Оскільки артефакт містить інші атомарні елементи методу, такі як індивідуальні концепції, можна сказати, що ці концепції та їхні цілі та цінності присутні в первинному артефакті. Ніщо не вказує на те, що було б корисно моделювати досягнення мети та структуру прив’язки цінності для кожного елемента методу в

ситуаціях практичного використання методу. Тому аналіз цих аспектів також буде опущено.

У нашому компоненті моделі бізнес-випадку ми знаходимо дві дії в RUP, які мають артефакт моделі бізнес-випадку як результат. Ці дві дії:

- Знайти бізнес-учасників і варіанти використання,
- Структурувати модель бізнес-випадків.

Під час роботи із строгим методом розробки систем, таким як RUP, автономний критерій є нетривіальним для застосування. Дві дії, які ми обговорюємо, мають кілька артефактів як результати, і це не відповідає нашій специфікації компонента методу, який має лише один. Цю проблему можна вирішити, дозволивши крокам і діям повторюватися в інших компонентах методу. Опис того, як це робиться, буде слідувати за основною презентацією артефакту моделі бізнес-випадку.

До того часу ми будемо дотримуватися опису дій, які пов’язані з основним результатом компонента методу, а саме, згаданим вище артефактом моделі бізнес-випадку використання.

Таблиця 3.1 ілюструє результати нашого аналізу. Таблиця розбита на дві колонки; ліворуч містяться дії, зібрані з RUP, а праворуч — вибрані кроки, пов’язані з результатом поставки компонента методу. Дія «Знайти бізнес-учасників і варіанти використання» містить дев’ять кроків у оригінальному описі методу. З цього набору ми вибрали менший набір із п’яти кроків, які пропонуються як необхідні в RUP для побудови моделі бізнес-випадку і шостий крок, на якому оцінюється модель бізнес-випадку. Таким чином, ця підмножина базується на цільовому обґрунтуванні цих дій і на тому, чи сприяють цілі цих дій цілям кінцевого артефакту, який є загальною метою компонента методу.

Друга дія, структурування бізнес-моделі використання, містить п’ять кроків, що дорівнює кількості кроків, передбачених у RUP. Ці кроки не є атомарними з огляду на їхні результати, оскільки їхні результати є

переробленими артефактами моделі бізнес-випадків, бізнес-учасників і бізнес-варіантів.

Таблиця 3.1.

Дії компонентів методу

Action	Step
Find Business Actors and Use Cases	Develop an Outline of the Workflow of Business Use Cases Describe How Business Actors and Use Cases Interact Package Business Use-Cases and Actors Present the Business Use-Case Model in Use-Case Diagrams Develop a Survey of the Business Use-Case Model Evaluate Your Results
Structure the Business Use-Case Model	Establish Include-Relationships Between Business Use Cases Establish Extend-Relationships Between Business Use Cases Establish Generalisations Between Business Use Cases Establish Generalisations Between Business Actors Evaluate Your Results

Рішення полягає в тому, щоб дозволити цим крокам повторюватися в послідовному компоненті бізнес-актора та компоненті бізнес-випадку використання. Відповідно, ці кроки можна розглядати як частини всіх трьох компонентів, інакше неможливо вважати їх незалежними. Це компроміс між проблемою надмірності та проблемою автономного компонента методу. Це також вирішує проблему наявності кількох результатів, як описано на попередній сторінці. Дозволяючи кожному компоненту методу мати лише один артефакт, визначений як його результат, ми можемо гарантувати, що обґрунтування запропонованого методу кожного компонента методу можна буде ідентифікувати. У той же час, дозволяючи повторним введенням необхідних вхідних даних, таких як бізнес-учасники та бізнес-випадки, ми

можемо гарантувати, що модель бізнес-випадків може вважатися такою, що відповідає самодостатній меті проектування.

Нарешті, ми повинні розглянути роль актора, пов’язану з цим компонентом методу. Відповідно до RUP, аналітик бізнес-процесів відповідає за модель бізнес-випадків використання, тому його слід включити до компонента методу. Крім того, ми можемо ідентифікувати дві зацікавлені сторони, клієнта та кінцевого користувача, які важливі під час роботи з результатом компонента методу. Отже, вони включені до частини вмісту компонента бізнес-випадку використання як актори.

3.1.1. Інтерфейс компонента моделі бізнес-випадку

Поки що ми розглянули лише внутрішній вигляд компонента Business method. Однак, щоб з’єднати компоненти методу один з одним, нам потрібно вказати інтерфейс, який має можливість виражати обґрунтування методу, запропоноване компонентом методу. Основним результатом кожного компонента методу є артефакт і цілі, які він виконує, а також цінності, які мотивують, чому ці цілі можна вважати важливими. Як саме виконується завдання, нецікаво для зовнішнього переглядача компонента, коли він виконує завдання. У таких ситуаціях користувача компонента методу цікавить насамперед у результатах, запропонованих компонентом (результат) і необхідних вхідних даних (передумовах), необхідних для досягнення цих результатів. Таким чином, практичні завдання, що стосуються кількох компонентів методу, вимагають лише від користувача методу застосування зовнішнього вигляду компонентів методу, з якими він працює.

Як відомо, інтерфейс компонента методу складається із загальної мети компонента методу та пов’язаних артефактів, які мають тип передумови або результату. Загальна мета передається суміжним компонентам методу через результат, оскільки можна вважати, що результат виражає мету, яку він досяг, через нотацію. Прикладом цього може бути мета моделі дизайну в RUP, яку можна визначити як «Створення об’єктної моделі, здатної

описувати реалізацію варіантів використання та слугувати абстракцією моделі реалізації та її вихідного коду» [23]. Виконуючи компонент методу Design Model, користувач методу матиме артефакт, який відповідає цій конкретній меті.

Таким чином, відправною точкою для нашого прикладу є результат компонента методу. Лише один артефакт визначається як готовий для того, щоб узгодити з вибраним рівнем деталізації діаграми. Це також робить компонент методу здатним виражати свою мету через один артефакт. Щоб створити цей результат, компонент моделі бізнес-випадку має шість артефактів, визначених як передумови: інструкції з бізнес-моделювання, запити зацікавлених сторін, глосарій, бачення, бізнес-учасник і бізнес-випадок використання.

Список у таблиці 3.2 описує артефакти, які використовуються у зв'язку з бізнес-моделлю варіантів використання. Він структурований у два стовпці, де перший стовпець містить назву артефактів, а другий стовпець містить різні ролі артефактів щодо компонента методу. Ролі артефактів описують, чи кожен із пов'язаних артефактів слід вважати результатом чи передумовою.

Таблиця 3.2.

Інтерфейс компонента методу: список артефактів

Artefact	Artefact Role
Business Use Case Model	Deliverable
Business Modelling Guidelines	Prerequisite
Stakeholder Requests	Prerequisite
Glossary	Prerequisite
Vision	Prerequisite
Business Actor	Prerequisite
Business Use Case	Prerequisite

Загальне обґрунтування методу компонента методу можна виразити через діапазон цілей, визначених у компоненті методу. Цілі пов'язані одна з одною в складних структурах і можуть бути ідентифіковані або

реконструйовані для кожного елемента методу, якщо необхідно. Як уже було сказано, ми не будемо проводити аналіз повного змісту елементів методу складової методу та їх обґрунтування в цілях. У цьому прикладі інтерфейсу компонента методу ми вирішили зосередитися на цілях, пов'язаних із результатом, оскільки їх можна вважати найбільш релевантними, оскільки вони чітко представляють обґрунтування запропонованого методу. Таблиця 3.3, наведена нижче, також містить два стовпці; лівий стовпець містить ідентифікаційні номери цілей, а правий – самі описи цілей.

Таблиця 3.3.

Інтерфейс компонента методу: список цілей

Goal Number	Goal Description
(0) Overall	To agree upon the business context in order to identify architecturally significant behaviour that should be automated and plan and follow up subsequent systems development
1	To agree that you understand the business context before you continue developing the system
2	To identify architecturally significant behaviour that should be automated
3	To plan and follow up on the business modelling effort and subsequent systems development

Усі описи цілей описані звичайним текстом, щоб допомогти користувачеві методу зрозуміти цілі та мати легше завдання оцінити, чи можна вважати обґрунтування певного запропонованого методу важливим чи ні. Часто цілі та значення артефакту компонента методу не вказуються явно в описі методу. У таких ситуаціях модульна поділ методу розробки системи на компоненти методу повинна включати аналіз, принаймні, обґрунтування методу, запропонованого первинним результатом. Інакше було б неможливо вказати інтерфейс для компонентів методу, і було б неможливо дотримуватися вимог підключення, встановлених у наших цілях проектування.

Ми визначили три цілі для результуючого артефакту компонента моделі бізнес-випадків, простеживши обґрунтування мети для елемента методу артефакту. Крім того, ми узагальнили ці цілі в загальну ціль, яка використовується для компонента методу , щоб мати можливість висловити загальну мету для компонента бізнес-випадку використання. Модель бізнес-випадків використовується для розуміння бізнес-контексту для майбутньої інформаційної системи, щоб визначити, які завдання в бізнес-контексті доцільно автоматизувати та необхідні для планування та подальших дій у проекті. Усі ці аспекти переформульовано в загальну мету, яку можна визначити в інтерфейсі моделей бізнес-випадків використання.

Аналіз структури прив’язки вартості компонента методу моделі бізнес-випадку представлено на рисунку 3.1. Рисунок найлегше зрозуміти, якщо розглядати його знизу вгору, оскільки там розташована мета.

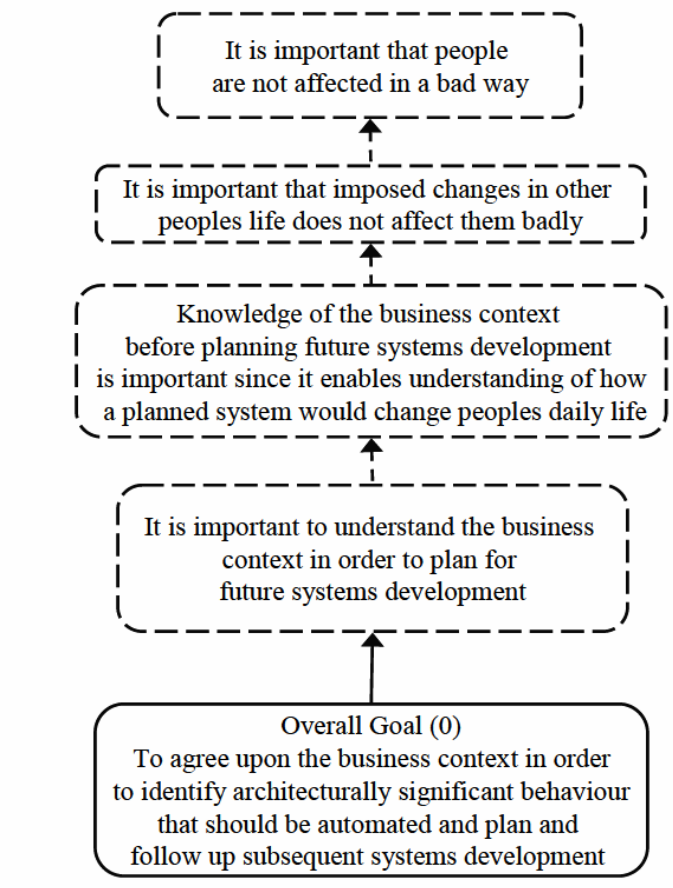

Рис. 3.1. Структура прив’язки вартості для компонента методу моделі бізнес-випадку

Над ціллю цінності описані та представлені ієрархічно, тобто представлені цінності вважаються прив’язаними до значення безпосередньо вище. Структура прив’язки значення обмежена загальною ціллю (0) компонента методу, представленою в таблиці 3.3, оскільки це єдина мета, видима в інтерфейсі компонентів методу. Аналізована мета має відношення до цінності відповідно до визначення обґрунтування методу. Найвищою цінністю можна вважати фундаментальні цінності, які відповідають етичним і моральним поглядам. Звичайно, є можливість знайти інші структури цінностей, які можуть бути зовсім іншими. На перший погляд загальна мета чогось може бути відокремлена від моральних аспектів. Часто буває так, що ситуація використання вирішує, які моральні та етичні точки зору опосередковано введені в дію. Ніж, головною метою якого є функціонування як ефективний інструмент для різання, можна використовувати різними способами. Проте структура ціннісного закріплення, яка мотивує, чому ця мета є бажаною, може мати багато форм. Однак слід усвідомлювати той факт, що в основі лежить структура закріплення цінностей, яка мотивує кожен компонент існуючого методу. Інакше компонент методу не був би визначений спочатку. Це також говорить нам про те, що існують можливості для пошуку нових областей застосування для компонентів методу, якщо користувач методу вирішить визначити відношення обґрунтування значення по-іншому порівняно з вихідним конструктором методу.

3.1.2. Підключення компонентів методу

Визначивши інтерфейс для нашого компонента методу Business case model, ми тепер маємо можливість підключити його до іншого компонента методу. Однією з передумов, визначених в інтерфейсі для нашого прикладу компонента методу, є бізнес - випадок використання. Отже, зв’язки цих двох компонентів один з одним повинні бути визначені в нашому прикладі. Ми обмежили приклад, щоб охопити лише один випадок досягнення цілі, щоб показати, як досягнуто цілі проекту підключення.

Коли користувач методу підключає компоненти методу, він застосовує зовнішній вигляд концепції компонента методу. Застосовуючи зовнішню перспективу до компонентів методу, користувач методу звертає увагу лише на інтерфейси компонентів методу, які він/вона вивчає. Вміст компонента методу вважається прихованим у внутрішньому поданні.

На рисунку 3.2 показано два компоненти методу, кожен з яких розкриває свої інтерфейси. Ліворуч ми знаходимо компонент бізнес-випадку використання, а праворуч ми знаходимо наш зразковий компонент бізнес-випадку використання.

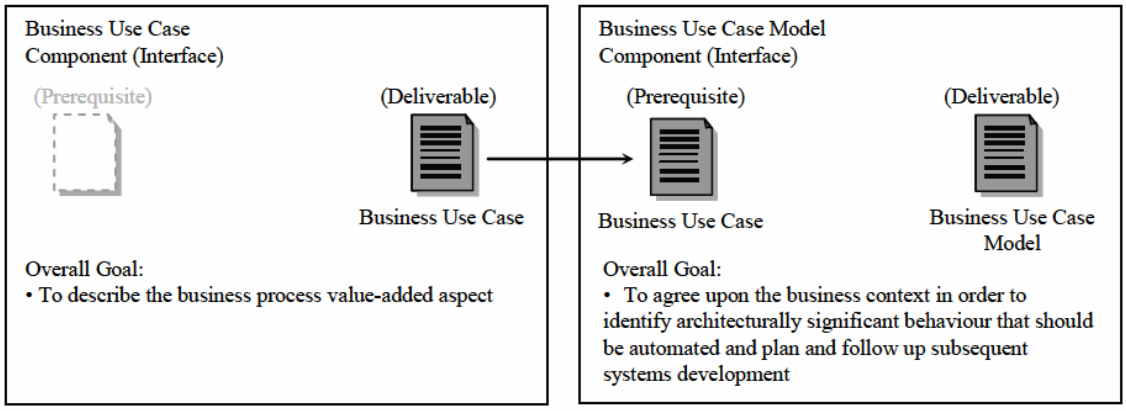

Рис. 3.2. З’єднання двох компонентів методу

Наш приклад компонента методу має результат із компонента методу Business use case як визначену передумову. Цей приклад підключення компонента методу обмежений лише однією передумовою. Фактично, компонент методу бізнес-випадку використання має 5 додаткових артефактів, визначених як передумови, як показано в таблиці 3.2. Крім того, для завершення інтерфейсу ми представляємо загальну мету кожного компонента. Загальну мету компонента бізнес-випадків можна визначити як «Описати аспект доданої вартості бізнес-процесу» . Загальна мета нашого прикладу компонента методу така ж, як визначено в таблиці 3.3 «Узгодити бізнес-контекст, щоб визначити архітектурно значущу поведінку, яка повинна бути автоматизована, а також планувати та стежити за подальшим

розвитком систем». Однак для завдання з’єднання лише цих двох компонентів необхідна лише ціль компонента Business case method.

На рисунку 3.2 показано зв'язок між нашими двома компонентами. З’єднання представлено у неформальному вигляді для підтримки читабельності. Однак, оскільки компоненти методу мають формалізоване представлення, також легко формалізувати зв’язки між компонентами методу, якщо це необхідно. Цього можна досягти шляхом застосування визначень UML відповідно до того, як вони застосовані на рисунку 2.12.

3.2. Вдосконалення ітеративної моделі за допомогою компонентів методу

Потреба в явному описі методу дала нам можливість застосувати життєздатність використання концепції компонента методу в задачі реконструкції методу. Нашою метою було дослідити компонент методу як комунікаційну концепцію в такому завданні. Як показано в попередніх розділах, фаза оцінки проекту під час впровадження RUP охоплює безліч аспектів, починаючи від загальних даних про проект і закінчуючи описом того, як проект на даний момент проводить розробку систем. Цей останній аспект оцінки зазвичай охоплюється та досліджується шляхом опитування осіб, які беруть участь у проекті. Вони описують, як вони працюють, а консультанти визначають проблеми. Тоді проблеми розглядаються як щось, що має бути вирішене за допомогою суспільної раціональності, знайденої в RUP. Нарешті, відповідні вибірки з RUP вибираються та впроваджуються в проект для вирішення проблем.

Процес реконструкції розробки неявних систем у проекті мав таку загальну структуру:

1. Застосуйте перспективу компонента методу до неявного процесу.

2. Визначте, яке обґрунтування методу пояснює, чому вони роблять речі так, як вони є.

3. Порівняйте процес з обґрунтуванням методу, запропонованим RUP.

4. Якщо потрібно, внесіть коригування, додавши або видаливши компоненти методу.

Eman 1

Членів команди запитали, що вони створили з точки зору артефактів під час розробки. Це було важливо, оскільки артефакт вважається основним результатом для кожного методу. Наш аналіз із Кроку 1 показав, що вони справді прийняли багато компонентів методу, які можна знайти в RUP. Більшість розробників мали досвід роботи з UML, тому використання різних компонентів методу, таких як діаграми варіантів використання та класи дизайну, було для них природним. Як наслідок, більша частина нашого початкового аналізу з кроку 1 показала, що проект вже прийняв RUP як метод розробки систем. Проте були деякі відмінності між тим, що пропонується в RUP, і тим, що відбувалося в процесі розробки.

Таблиця 3.4.

Список артефактів проекту при використанні методу реконструкції

	Artefact List
1	Process Description
2	Glossary
3	Supplementary Specifications
3	Use Case Specification
4	Business Rules
5	Activity Diagram
6	Use Case Model
7	Actor
8	Use Case Diagram
9	User Interface Prototype
10	Analysis Class
11	Analysis Class Diagram
12	Design Class
13	Database
14	Build
15	Component
16	Test Plan
17	Test Case Template
18	Test Case
19	Test Data
20	List of Measures
21	Test Log

Також стало очевидним, що частини їхніх систем розробляються інакше, ніж у RUP. Це було зрозуміло, коли ми зафіксували використання компонентів методу Шаблон тестового випадку та Перелік заходів. Ці компоненти методу не існують у RUP. Це скоріше адаптовані варіанти компонентів методу, задумані командою і засновані на їхньому попередньому досвіді тестування програмного забезпечення. У той час під час аналізу ми не приділяли особливої уваги змісту та загальним цілям цих компонентів методу. Ми просто записали їх, щоб отримати чіткий огляд суми вироблених артефактів і, отже, компоненти методу, які можна вважати частиною методу розробки. Результати етапу 1 наведені в таблиці 3.4.

Eman 2.

Після того, як ми отримали чітке уявлення про те, які артефакти вони створили, ми зосередилися на тому, чому вони створили їх на етапі 2. Наприклад, запитання щодо артефакту Актор зосереджується на тому, де ще в процесі розробки артефакт Актора потрібен як вхідні дані. Вони дали нам відповідь, що Актор потрібен для створення діаграм варіантів використання. Отже, додається відношення, яке виражає, як обґрунтування методу, запропоноване компонентом методу Актор, вплинуло на обґрунтування методу, запропоноване компонентом методу діаграми варіантів використання. Для кожного з визначених компонентів методу ми повинні були запитати команду, де в розробці цей артефакт був знову використаний і навіщо це було необхідно. Це було необхідно для того, щоб з'ясувати, що містять інтерфейси компонента методу з точки зору їхніх передумов артефактів та їхніх загальних цілей. У більшості випадків загальна мета раціональної структури відповідала RUP. Наприклад, діаграма класів Analysis досягла мети дати огляд ранньої концептуальної моделі для аспектів у системі з відповідальністю та поведінкою. Цей компонент методу вважався необхідною умовою для методу клас дизайну компонента. Клас Design надає визначення набору об'єктів, які мають однакові обов'язки, зв'язки, операції,

атрибути та семантику, і зазвичай використовується в RUP для прийняття рішення про те, що реалізувати як функції або інформацію, які потрібно зберігати в базі даних. Отримання класів Design у RUP передбачає аналіз поточної діаграми класів Analysis.

Іншим аспектом, який став очевидним, було те, що вони пропустили певні кроки порівняно з RUP. Як приклад, RUP передбачає виконання компонента методу під назвою Use case realization, щоб полегшити виявлення потенційних класів Design. RUP також передбачає, що класи дизайну збираються в компоненті моделі дизайну, який не мав відповідника в методі розробки PSU. Ці пропущені компоненти методу зазвичай функціонують як спосіб збору результатів із ряду артефактів, щоб створити огляд і мінімізувати ризик забуття результатів на цьому шляху. Обґрунтування методу, яке вони надають, також зазвичай вважається непотрібним у проектах меншого розміру, оскільки вони функціонують як збірники суми обґрунтувань методу, наданих іншими компонентами методу. Пропуск цих компонентів методу не обов'язково призведе до неправильного результату, навіть якщо ризик отримання неправильних результатів підвищиться. Ми розглядали PSU як великий проект і, таким чином, можливо, опинилися в ситуації, коли деякі аспекти можуть, але не обов'язково, будуть забуті.

Цей процес реконструкції продовжувався, доки не було зрозуміло обґрунтування кожного компонента методу та не було визначено їхні зв'язки з іншими компонентами методу з точки зору обґрунтування мети методу. Команда PSU змогла пояснити свої ідеї, і ми могли легко зрозуміти, як вони могли досягти резонансу раціональності за допомогою свого власноруч розробленого методу розробки систем.

Компоненти методу та їхні зв'язки були задокументовані на діаграмі, показаній на рисунку 3.3. Це огляд компонентів методу, які використовуються в проекті PSU. Кожен компонент методу названо на честь артефакту, який він доставляє, і показує, як артефакти сприяють реалізації цілей інших компонентів методу. Стрілка від компонента методу до іншого

означає, що перший компонент методу вважається таким, що пропонує свій артефакт, що доставляється, як передумову для останнього.

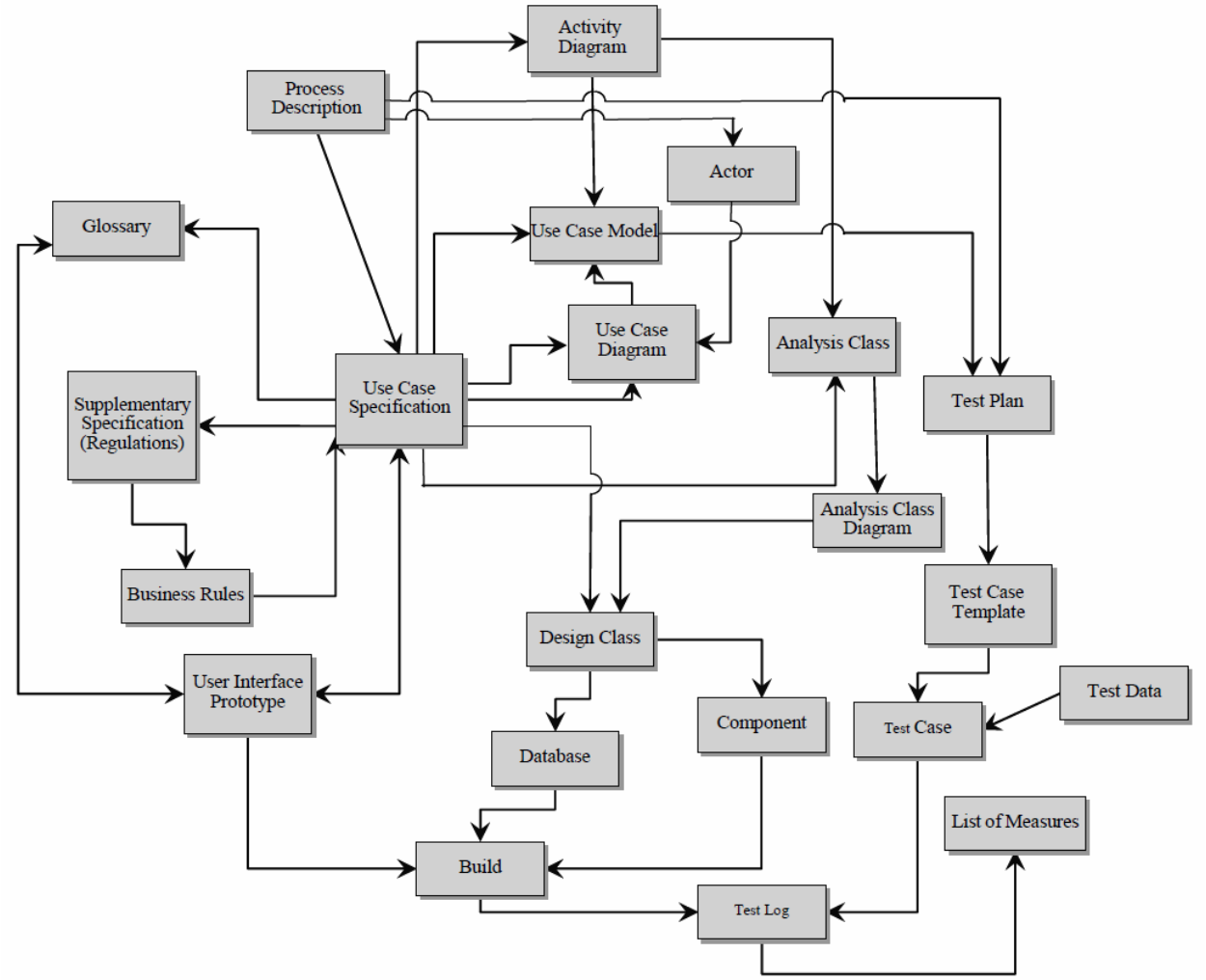

Рис. 3.3. Діаграма, що описує компоненти методу та їх співвідношення цілей у проекті

Етап 3

На етапі 3 ми порівняли отримані результати з раціональністю, наданою RUP. Порівняння між результатами реконструкції та існуючими методами розробки систем також здійснюється інтуїтивно під час етапу 2, оскільки цього майже неможливо уникнути. Часто ми стикалися з тим, що наше розуміння ролі певного компонента методу ґрунтувалося на зв'язку його з тим, як він використовується в RUP. Як уже згадувалося, ролі, які відігравали компоненти методу, зазвичай збігалися один з одним. Вивчаючи

діаграму, можна порівняти процес розробки з RUP. Незважаючи на те, що команда проекту вже самостійно перейняла більшу частину діяльності RUP і мала метод розробки, дуже схожий на RUP, існують певні інформаційні відмінності.

У RUP пропонуються артефакти під назвою «Підсумок оцінки тесту», «Список ідей для тестування», «Результат тесту» та «Запит на зміну» . Ці артефакти не можна знайти на діаграмі артефактів. Натомість вони замінили їх власноруч створеними артефактами, які покривають більшу частину обґрунтування методу, запропонованого артефактами RUP.

Шаблон тестового випадку, створений у процесі розробки проекту, пропонує обґрунтування методу у формі можливості вказувати багаторазові тестові сценарії. Цей артефакт замінює список ідей тестування з RUP, призначений для визначення потенційних тестів. У проекті вони вважали, що компонент методу шаблону тестового випадку має більшу цінність як внесок у їхні тестові випадки, і знайшли резонанс раціональності в цьому рішенні. У них був проект, який, на їхню думку, вони могли зрозуміти досить добре, щоб уявити, що можна було б планувати свою тестову діяльність на основі задуманого набору шаблонів, а не на основі окремих списків, як пропонував RUP. Якщо проект у майбутньому вирішить стати більш дотримуваним RUP, вони зможуть легко обміняти свої шаблони тестових випадків на списки тестів RUP.

Подібним чином, перелік заходів вважався достатнім для того, щоб команда могла записати результати тестів. Цей артефакт також відстежував можливі запити на зміни, які вимагалися в результаті проведених тестів, і описував зміни, які будуть потрібні, а також зміни, які вже були зроблені. Як наслідок, список перевірок замінив компонент методу запиту на зміну з RUP. Проект по-іншому розглядав результат тесту, оскільки вони підготували журнал тестування для відстеження результатів тестів і визнали його достатнім з точки зору обґрунтування запропонованого методу для створення

списку заходів. Підсумок оцінки тесту, запропонований у RUP, взагалі не мав відповідника в проекті.

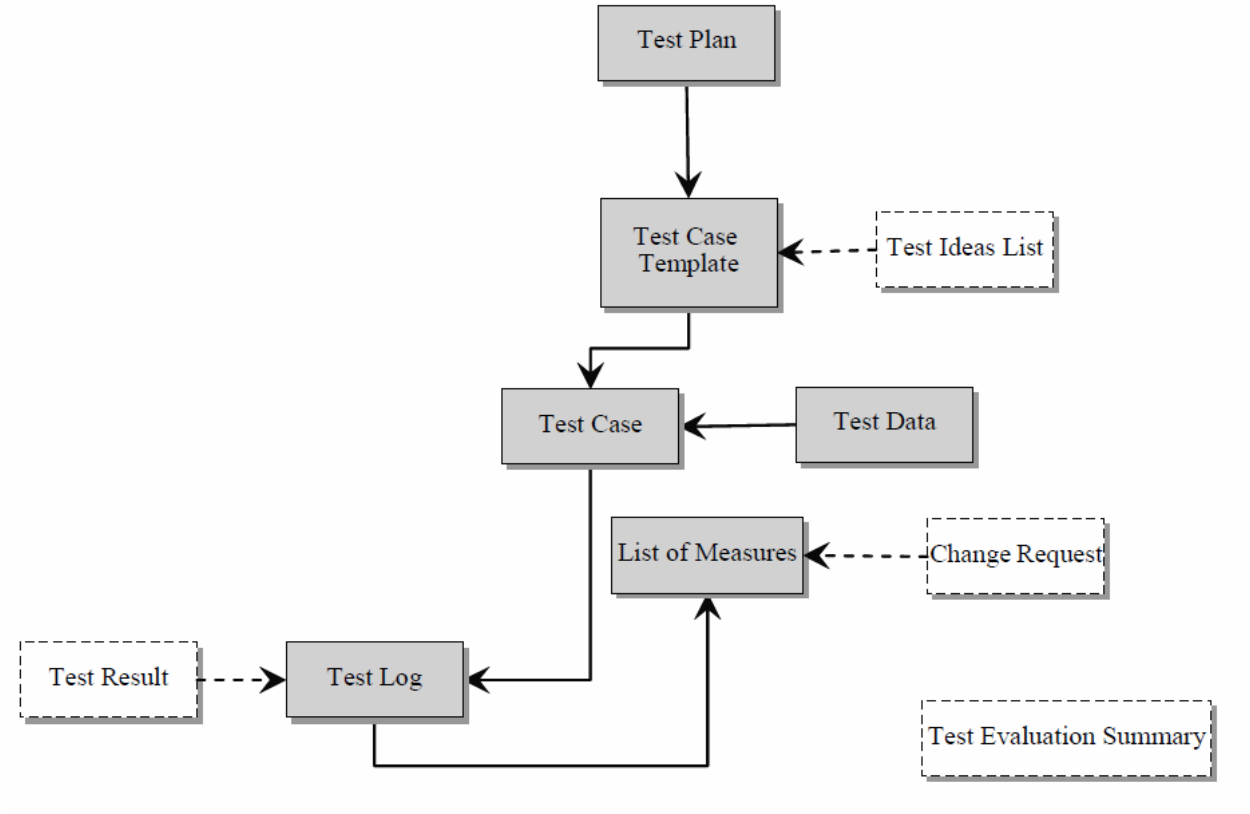

Рис. 3.4. Деталь реконструйованого методу з пропущеними компонентами методу RUP

Рисунок 3.4 описує відмінності між методами розробки реконструйованих систем RUP і класичного проекту. Пунктирні рамки — компоненти методу RUP, які були замінені в проекті. Пунктирні стрілки вказують на позицію, де команда замінила компонент методу RUP на компонент методу, розроблений самостійно.

Остаточний результат етапу 3 показав, що проект має метод розробки системи, який відповідає RUP у багатьох аспектах. Насправді, відмінності настільки незначні, що можна сказати, що проект вже перейняв RUP з деякими адаптаціями.

Як уже зазначалося, проект ще не впровадив зміни, але стверджував, що планує зробити це найближчим часом. Іншим аспектом, який дав групі проекту певний важіль, був той факт, що вони ефективно досягали

результатів і змогли підтримувати роботу команди без змін. Таким чином, впровадження спочатку буде розглянуто, коли настане час для проекту повністю прийняти RUP.

Аналітична робота такого роду може бути виконана для порівняння реконструйованого неформального та/або неявного методу розробки систем з обґрунтуванням загальнодоступного методу, запропонованого методом зовнішньої розробки систем. Під час виконання завдання з оцінки проекту реконструкція існуючого процесу розробки сприяла б створенню чіткої картини того, що насправді відбувається під час складного бізнесу розробки систем. Дуже часто, пояснюють консультанти, неявне знання процесів розробки забувалося, а впровадження проводилося з початковою точкою попереднього досвіду інших проектів впровадження. Знання такого типу дадуть особам, відповідальним за процес впровадження методу, більшу глибину в оцінці проектів і краще розуміння того, як проект розробляє системи сьогодні і чим цей процес відрізняється від методу розробки систем, який розглядається для впровадження. Що стосується досліджень EME, реконструкція, що містить етапи 1 і 2, може допомогти зрозуміти, як методи розробки систем передають обґрунтування свого методу в практиках розробки систем. Оскільки наш підхід до реконструкції методу з точки зору компонента методу має на меті прояснити структуру досягнення мети в проекті розробки та зробити її більш явною, можна буде повідомити іншим неявний метод розробки систем. Крім того, можна буде, додавши крок 3, порівняти метод розробки реконструйованих систем із суспільною раціональністю, яку пропонує інший метод розробки систем.

3.3. Концептуальне знання-орієнтоване перемоделювання методу розробки

У цьому розділі ми повернемося до концепції компонента методу. Випадок, що описано в розділі 3.2. дав можливість запустити компоненти

методу в дію та застосувати їх для практичного завдання розробки методу. У світлі результатів реконструкції неявного та неформального методу розробки систем проекту можна оцінити концепцію компонента методу та внести корективи відповідно до наукового підходу проектування.

Концепція компонента методу була задумана через цілі проектування і згодом реалізовані в оновленому дизайні. Цілі дизайну — це сукупність властивостей, якими повинна володіти концепція компонента методу. Наступний підрозділ присвячений порівнянню початкових проектних цілей та досвіду оцінки проекту. Порівняння проводитиметься із застосуванням перспективи концептуального моделювання, подібно до того, що мало місце, коли концепція компонента методу була формалізована та перепроектована.

Мета проектування 1. Автономність: має бути можливим розглядати компонент методу як автономну частину методу розробки системи з огляду на керівні принципи, які описують результат і процес виробництва такого результату. Таким чином компонент методу може бути змодельований з методу розробки систем або виступати в якості будівельного блоку під час побудови нових методів або під час інтеграції методів.

Eman 1 у нашому дослідницькому підході до реконструкції методу складався із застосування кожного артефакту, задіяного в процесі розробки. Це означало, що шляхом переліку артефактів ми негайно визначимо потенційні компоненти методу, оскільки кожен артефакт слід розглядати як результат, що пропонує загальну мету компонента методу, необхідну десь ще в процесі розробки системи. Це також означало, що перспектива артефакту миттєво гарантувала, що отримані компоненти методу відповідатимуть вибраному рівню деталізації діаграми. Отже, нам не потрібно було б хвилюватися, що результат реконструкції дасть компоненти методу, які будуть занадто малими або занадто великими. Реконструкція процесу розробки ПГУ показала, що розглядати його як складений із самодостатніх компонентів методу не становить жодних проблем. Рівень деталізації артефакту дуже добре відповідав тому, як команда проекту сприймала процес

розробки. У цей час ми не приділяли уваги тому, як саме було виявлено кожен артефакт. Тому ми не аналізували зміст кожного компонента методу. Однак такий аналіз можна було б виконати з початковою точкою в діаграмі компонентів методу на рисунку 3.3. Реконструкція також показала, що компоненти методу, розроблені самою командою, можна легко замінити компонентами з RUP, якщо це необхідно, що вказує на те, що ідея компонент методу як будівельного блоку є життєздатним.

Ціль дизайну 2. Внутрішня послідовність і узгодженість: компонент методу повинен сприйматися як внутрішньо узгоджена і узгоджена сутність. Іншим способом висловити це є те, що конструкт має бути стабільним у часі та без втрат. У результаті компонент методу буде сприйматися як значимий. Принцип полягає в прагненні створити конгруентність між компонентами методу та однорідність методів розробки систем.

Обробка кожного артефакту в процесі розробки однаково під час реконструкції дозволить нам дотримуватися аспекту конгруентності компонентів методу. За властивостями вони по суті подібні один до одного, але зміст і обґрунтування методу відрізняються і надають їм індивідуальних особливостей. Однак, оскільки ми не проводили аналіз змісту кожного компонента методу, важко повністю оцінити цю мету дизайну.

Наш процес реконструкції був більше зосереджений на зовнішньому вигляді компонентів методу, який спрямовував нас на інтерфейси компонентів методу, а не на їх повний вміст, який можна знайти у їх внутрішньому вигляді. Тим не менш, цілі, які можна знайти в інтерфейсах разом з артефактами, що поставляються, вважаються внутрішніми частинами компонентів методу. Ці аспекти були зосереджені в кожному компоненті методу. Додатковий аналіз інших внутрішніх аспектів можливий, зосередившись на артефакті, оскільки артефакт, що поставляється, є контейнером для результатів, досягнутих під час використання компонента методу, а «використання» означає, що в дію вводиться ряд концепцій і використовується визначена нотація для вести облік результатів. Це можна

вважати процесом реверсивного проектування. Оскільки доставлений артефакт опосередковано містить більшу частину вмісту компонента методу, можна застосувати наш підхід до реконструкції методу і все одно стверджувати, що перспектива компонента методу підтримується.

Ціль дизайну 3. Раціональність: Дії виконуються та призначаються з причин. Таким чином, компонент методу повинен мати визначений цільовий стан, мету свого існування. Аргументи вмотивовані компонентами, властивими обґрунтуванням методу, і стосуються цілей і цінностей, які реалізує компонент методу.

Eman 2 під час реконструкції нашого методу – це те, де ця мета проекту була в основному задоволена. Під час цього кроку ми проаналізували кожен артефакт у нашому списку артефактів і повторно визначили їх як компоненти методу. Перехід між суворим фокусуванням на артефакті до компонентів методу є простим, оскільки він більш-менш передбачає лише визначення того, що повинен існувати компонент методу із вмістом, який є результатом кожного артефакту. Якби не було контенту, просто не було б артефакту. Артефакт також містить запропоноване загальне обґрунтування методу компонента методу відповідно до нашої мети дизайну раціональності, оскільки він виражає мету свого існування разом із своїми цілями в інтерфейсі.

Застосування перспективи компонентів методу дозволило нам створити уявлення про процес розробки з обґрунтуванням методу на передньому плані, оскільки етап 2 вимагав аналізу цільової раціональної структури, яка стоїть за процесом розробки та мотивує її. Кожен компонент методу реєструвався відповідно до того, як він сприяв реалізації цілей в інших компонентах методу через зв’язки з іншими компонентами методу, що виражає обґрунтування методу. Діаграма, що описує ідентифіковані компоненти методу та їхні цільові співвідношення на рисунку 3.3, показує, як ми структурували початковий список артефактів на компоненти методу.

Реконструкція неявного методу розробки також показала, що можна класифікувати та записувати самостійно розроблені компоненти методу після того, як їх загальні цілі визначені та зрозумілі у зв’язку з їхніми артефактами. Члени команди мали метод розробки з обґрунтуванням методу, який був дуже схожий на запропонований RUP, але вони також внесли деякі коригування та замінили деякі компоненти в тесті з компонентами самостійного методу. Ці останні компоненти методу необхідно проаналізувати, а їхнє методологічне обґрунтування має бути зрозуміле під час етапу 2. Інакше процес реконструкції може дати картину, яка не відповідає тому, що насправді відбувається під час розробки. Процес реконструкції показав, що можна зосередити та записати обґрунтування методу самостійно розроблених, а також стандартних компонентів методу.

Ціль дизайну 4. Зв'язок: компоненти методу повинні бути можливими для з'єднання один з одним. Кожен компонент методу, який розглядається як частина методу розробки системи, повинен сприяти досягненню ланцюжка цілей, додаючи загальну мету конкретного проекту. Таким чином, компоненти методу повинні мати можливість виражати, як вони пов’язані один з одним і як ці зв’язки можуть сприяти досягненню загальних цілей, пов’язаних із конкретним проектом.

Зосередження спільної раціональності через інтерфейси компонентів методу під час етапу 2 безпосередньо призвело до проблем, що стосуються цілі дизайну підключення. Процес пошуку зв’язків між компонентами методу за раціональністю, яку вони пропонують як результати, чітко показує нам, що їх можна зв’язати таким чином і таким чином досягти мети дизайну. Етап 2 у нашому підході та перенесення погляду з артефактів на інтерфейси компонентів методу сприймалися всіма залученими особами (командою проекту, консультантами та дослідниками) як природний спосіб обробки колекції артефактів після того, як вони були перевизначені як окремі компоненти методу. Це показало, що мета дизайну підключення виявилася успішно реалізованою в проекті концепції компонента методу.

Ціль проектування 5. Застосовність: повинна бути можливість відобразити конструкцію компонента методу на будь-який існуючий метод розробки систем у будь-якій ситуації.

У процесі реконструкції ми успішно застосували концепцію компонента методу на методі неявної розробки систем. Ми також розглядали модулі з RUP як компоненти методу під час етапу 3, де ми порівнювали методологічне обґрунтування тестових компонентів з тестовою дисципліною в RUP. Це також передбачало застосування компонента методу в частині RUP. Крім того, ми змогли зберегти перспективу компонентів методу під час аналізу компонентів методу, які були самостійно задумані. Ці результати свідчать про те, що у нас немає нових причин сумніватися у можливості застосування компонентів методу .

Наведений вище концептуальний аналіз не означає, що ми взагалі не вносили жодних змін у концепцію компонента методу. Ми фактично зрозуміли один аспект, який раніше не розглядали. Як уже було сказано вище, ми не проводили повного аналізу змісту кожного компонента методу. Однак, коли ми обговорювали результати аналізу з членами команди, ми завершилися обговоренням ролей акторів для деяких компонентів. У деяких випадках ролі Актора можна вважати виконанням або участю в певній дії. Завжди існуватиме екземпляр асоціації <Carry out> між ролями актора та дією, оскільки дія потребує актора. Це не стосується асоціації < Participate in>. Це вже визначено на діаграмі UML через вибір множинності для двох асоціацій. Що потрібно додати, так це визначення, яке стверджує, що екземпляр класу Action, тобто об’єкт Action, не може мати двох посилань на той самий об’єкт ролі Actor. Наприклад, рольовий клас актора можна створити в двох об’єктах ролі актора, можливо, системного аналітика та кінцевого користувача. Екземпляр класу Action може бути створений у діаграмі активності Draw. У цьому випадку необхідно вказати, що об’єкт діаграми діяльності Draw може мати лише одне з будь-яких посилань на кожен об’єкт ролі актора. Якщо екземпляр об’єкта Action має два або більше

посилань на об’єкти ролі Актора, посилання не можуть вести до того самого інстанційованого об’єкта ролі Актора. Отже, об’єкт системного аналітика (класу ролі актора), який має посилання <Carry out> на діаграму діяльності Draw (класу Action), не може мати посилання <Participate in> одночасно. Це обмеження визначається за допомогою обмеження Xor, яке згодом додається до нашої діаграми UML.

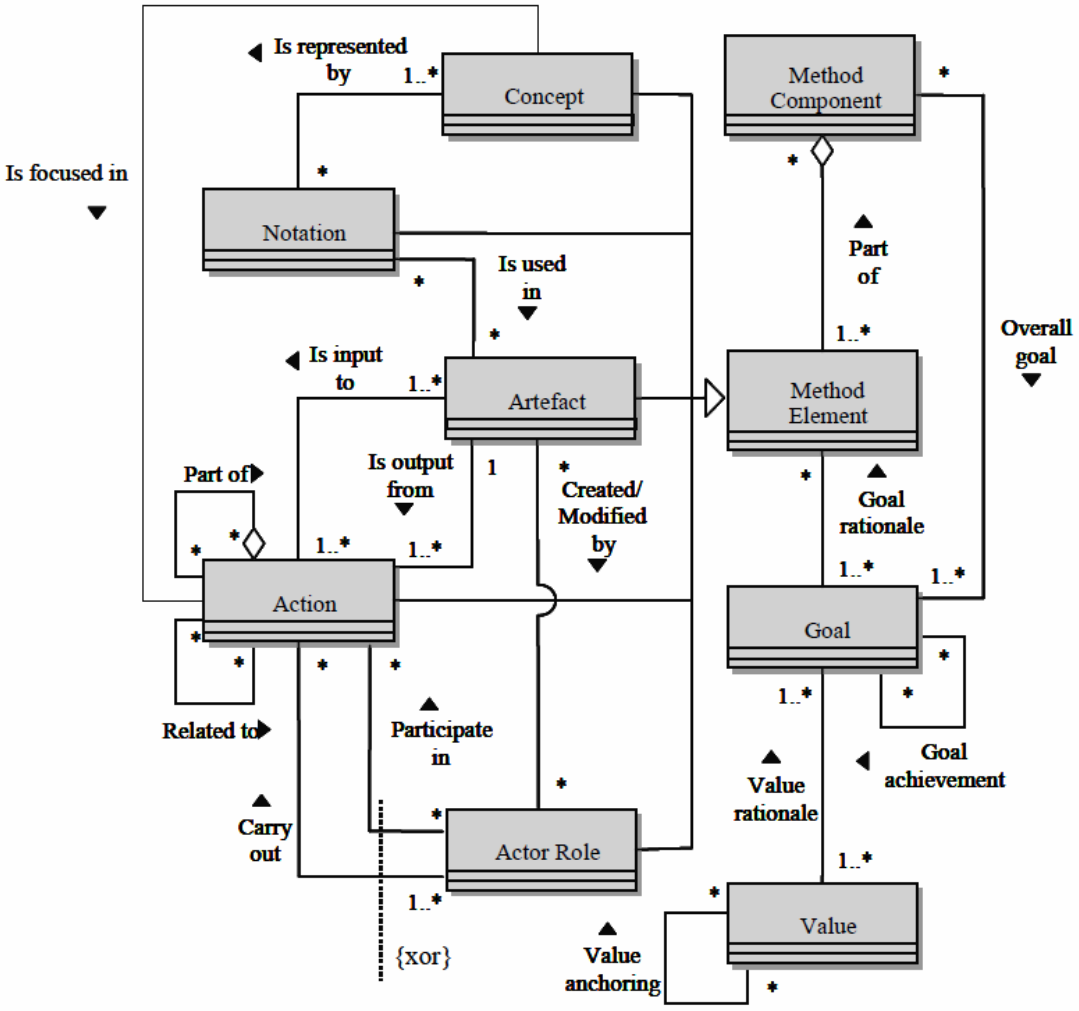

Рис. 3.5. Концепція компонента методу v1.1

3.4. Обґрунтування пропонованого методу стійкої ітеративної моделі розробки

Обґрунтування методу, визначене як цільова і ціннісна структура, яка мотивує метод розробки систем, є тим, що вже включено в концепцію

компонента методу через його мету розробки раціональності. Ніщо в цьому прикладі не вказувало на те, що нам потрібно внести будь-які зміни у визначення обґрунтування методу. Однак концепція обґрунтування методу відіграє додаткові ролі під час завдання реконструкції методу та досвіду навчання від використання обґрунтування методу у цьому контексті варто відзначити.

У випадку класичного проекту, що розглядався в підрозділі 3.2 ми часто використовували концепцію резонансу раціональності, щоб визначити та зрозуміти, як члени команди PSU знайшли підтримку методу в своєму власноруч розробленому методі. Приклад цього можна знайти в дискусії про відмінності між реконструйованим методом та RUP. Щоб зрозуміти, як шаблон тестового випадку та список заходів надали підтримку в процесі, ми, перш за все, мали зрозуміти, як члени команди стверджували, що вони отримали підтримку від цих компонентів методу. Це означає, що ми мали зіставити їх особисту раціональність із тим, що насправді робилося в цих компонентах методу. Це означає, що ми як дослідники в EME використовуємо концепцію резонансу раціональності, щоб інтерпретувати міркування, які мотивують неявні методи розробки систем. Можливість зосередитися на резонансі раціональності під час дослідження EME, здається, має великі переваги, оскільки дає дослідникам можливість оцінити аспекти зрілості та обізнаності щодо розробки систем.

Реконструкція неявного методу розробки систем показала, що обґрунтування методу можна включити в концепцію компонента методу. Ми не зазнали жодних недоліків у тому, як обґрунтування методу було пов’язано з окремими елементами методу в частині змісту компонентів методу. Однак стало ще більш очевидним, що проблеми, пов'язані з моделюванням структури прив'язки цінності методу розробки систем можуть бути правильно розглянуті. Аналіз структури прив’язки значення складний, оскільки структура залежить від користувача. У звичайній практиці розробки систем це може бути не дуже поширеною ситуацією, коли базові значення

компонента методу можуть призвести до проблем. Був такий компонент методу; цілі, яких ця базова цінність мала на меті досягти, ймовірно, виглядатимуть сумнівними. Хто колись чув про аморальний компонент методу? Тим не менш, ми ще раз повинні підкреслити той факт, що кожен компонент методу має базові цінності, які роблять цілі вартими прагнути. Проте проведення аналізу основної структури закріплення цінностей під час більшості досліджень ЕМЕ, здається, мало практичної користі, хоча можна уявити ситуації, коли такий аналіз може бути корисним. Одним із прикладів є ситуація, коли новий метод розробки систем створюється з фактичним наміром базуватися на вартості. Можна подумати про цінність програмістіві звільнення від нудної документації, яку легко визначити в екстремальному програмуванні або ідею про те, що покращення якості робочого життя та підвищення задоволеності роботою користувачів мають бути основною метою процес розробки систем, як запропоновано в [31].

Висновки до розділу

Отже, в цьому розділі представлено приклад, у якому ми представляємо компонент методу відповідно до нашої переробленої версії, щоб показати, як слід розуміти компонент методу. Ми вирішили проілюструвати оновлену концепцію компонента методу, показавши внутрішнє подання компонента бізнес-моделі. Після цього ми представили зовнішній вигляд нашого компонента методу та показали, як його можна підключити до іншого компонента. У нашому прикладі ми підключили наш компонент методу до компонента Business use case.

В розділі також доведено можливе використання методичного компонента в методі реконструкції методу розробки неявних систем. На додаток до простого застосування точки зору компонентів до методу неявної розробки систем, ми також досліджували можливості формулювання етапів для виконання завдання реконструкції методу. Цей дуже простий підхід

складається з двох основних кроків, де неявний метод спочатку аналізується артефактами, створеними на цьому шляху. Цей фокус дає артефакту правильний рівень деталізації для компонентів методу, коли кожен артефакт згодом перевизначається в компонент методу. На другому етапі звертається увага на інтерфейси компонентів методу та визначається, як компоненти методу пов’язані між собою обґрунтуванням мети, запропонованим їхніми артефактами, що поставляються. Запропоновані наступні кроки стосуються лише ситуацій, коли метод розробки реконструйованих систем потрібно порівняти з іншим явним методом або якщо потрібно додати або видалити певні компоненти методу.

ВИСНОВКИ

В магістерській роботі розглянуто моделі, методи та засоби знання-орієнтованої комунікації в процесі розробки програмних систем для імплементації в стійкій ітеративній моделі розробки.

Основним поняттям у цій роботі є поняття обґрунтування методу. Це концепція, яку було вирішено визначити за допомогою UML. Концепція функціонувала як вхідний матеріал для інтерпретаційного аналізу під час тематичних досліджень. Концепція визначає обґрунтування методу як відношення між компонентом методу, метою, яку він реалізує і цінностями, які мотивують, чому цілі можна вважати бажаними. Таким же чином розробник системи може мати цілі, які він хоче реалізувати і структуру обґрунтування цінностей, яка мотивує ці цілі. Коли мета компонента методу збігається з метою розробника системи в ситуації використання, що називається методом у дії, ми можемо говорити про стан резонансу раціональності.

Магістерська робота також охоплює дизайн концепції методу розробки систем, яка має основу в обґрунтуванні методу. Концепція бере свій початок від концепції компонента методу і може розглядатися як еволюція оригінальної концепції. Мотивацією для створення концепції модуля методу розробки систем було створення можливостей зосередитися на частинах методів розробки систем, не втрачаючи з уваги обґрунтування методу. Еволюція була досягнута завдяки процесу проектування, де можна використовували концептуальне моделювання, щоб отримати концепцію компонента методу, яка б відповідала вимогам, встановленим цілями проектування. Цілі дизайну були результатом аналізу, а саме це: самодостатність, внутрішня послідовність і узгодженість, раціональність, зв’язаність та застосовність. Таким чином, метод розробки систем можна розуміти як сукупність пов’язаних компонентів методу, структурованих в ієрархії досягнення цілей.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Andersen E S Systemutveckling : Principer, Metoder Och Tekniker, 2., rev. och utök. uppl. /, Studentlitteratur, Lund. 1994.
2. Argyris C and Schön D A Organizational Learning Ii : Theory, Method and Practice, Addison-Wesley, Reading, Mass. 1995.
3. Avison D and Fitzgerald G Where Now for Development Methodologies?, Communications of the ACM, 46, 2003.
4. Avison D, Wood-Harper A, Vidgen R T and Wood J R G A Further Exploration into Information Systems Development: The Evolution of Multiview2, IT and People, 11, 1998.
5. Avison D E and Fitzgerald G Information Systems Development : Methodologies, Techniques and Tools, 2., McGraw-Hill, London. 1995.
6. Baskerville R and Wood-Harper A , A Critical Perspective on Action Research as a Method for Information Systems Research, Journal of Information Technology, vol. 2001.
7. Beck K Extreme Programming Explained: Embrace Change, Addison-Wesley, Reading, MA. 2000.
8. Benbasat I, Goldstein D K and Mead M The Case Research Strategy in Studies of Information Systems, MIS Quarterly, 11, (3), 369–388. 1987.
9. Berger P L and Luckmann T The Social Construction of Reality: A Treatise in the Sociology of Knowledge, Anchor books, New York. 1967.
10. Bergström S and Råberg L Adopting the Rational Unified Process : Success with the Rup, Addison-Wesley, Boston, MA. 2004.
11. Beynon-Davies P and Williams M D The Diffusion of Information Systems Development Methods, Journal of Strategic Information Systems, 12, (1), 29-46. 2003.
12. Boehm B W A Spiral Model of Software Development and Enhancement, IEEE Computer, 21, (5), 61–72. 1988.

13. Booch G, Rumbaugh J and Jacobson I The Unified Modeling Language User Guide, Addison-Wesley, Harlow, UK. 1999.
14. Brinkkemper S Method Engineering: Engineering of Information Systems Development Methods and Tools, Information and Software Technology, 38, (4), 275– 280. 1996.
15. Brinkkemper S, Saeki M and Harmsen F Meta-Modelling Based Assembly Techniques for Situational Method Engineering, Information Systems, 24, (3), 209– 228. 1999.
16. Chen P The Entity-Relationship Model: Toward a Unified View of Data, ACM Transactions on Database Systems, 1, (1), 9–36. 1976.
17. Conklin J and Begeman M L Gibis: A Hypertext Tool for Exploratory Policy Discussion, ACM Transactions on Office Information Systems, 6, (4), 303-331. 1988.
18. Cronholm S and Ågerfalk P J On the Concept of Method in Information Systems Development, In Proceedings of the 22nd Information Systems Research Seminar in Scandinavia (IRIS 22), Vol. 1 (Ed, Käkölä T), pp. 229–236. 1999.
19. Davies M F Irrational Beliefs and Unconditional Self-Acceptance. Iii. The Relative Importance of Different Types of Irrational Belief, Journal of Rational-Emotive and Cognitive-Behaviour Therapy, 26, (2), 102-118. 2008.
20. Dix A, Finlay J, Abowd G and Beale R Human-Computer Interaction (Second Edition), Prentice Hall Europe. 1998.
21. Dyer Jr W G and Wilkins A L Better Stories, Not Better Constructs, to Generate Better Theory: A Rejoinder to Eisenhardt, Academy of Management Review, Vol. 16, (No.3), 613-619. 1991.
22. Eisenhardt K Building Theories from Case Study Research, Academy of Management Review, Vol 14, (4), 532-550. 1989.
23. Firesmith D G, Henderson-Sellers B and Graham I The Open Modeling Language (Oml) Reference Manual, Cambridge University Press : Sigs Books, New York. 1998.

24. Fiske J Kommunikationsteorier - En Introduktion (in English: Introduction to Communication Studies), Wahlström & Widstrand Förlag, Borås. 1994.
25. Fitzgerald B Formalised Systems Development Methodologies: A Critical Perspective, The Informations Systems Journal, 6, (1), pp 3-23. 1996.
26. Fitzgerald B The Use of Systems Development Methodologies in Practice: A Field Study, The Information Systems Journal, 7, (3). 1997.
27. Fitzgerald B An Empirical Investigation into the Adoption of Systems Development Methodologies, Information & Management, 34, (6), 317-328. 1998.
28. Fitzgerald B An Empirically-Grounded Framework for the Information Systems Development Process, In Proceedings from the 19th International Conference on Information Systems, Helsinki, Finland. 1998.
29. Fitzgerald B, Russo N L and O'Kane T Software Development Method Tailoring at Motorola, Communications of the ACM, 46, (4). 2003.
30. Fitzgerald B, Russo N L and Stolterman E Information Systems Development: Methods in Action, McGraw-Hill, Berkshire, UK. 2003.
31. Fugate M, Kinicki A J and Prussia G, E Employee Coping with Organizational Change: An Examination of Alternative Theoretical Perspectives an Models, Personnel Psychology, 61, 1-36. 2008.
32. Geertz C The Interpretation of Cultures : Selected Essays, Basic Books, New York. 2000.
33. Goldkuhl G Stöd Och Struktur I Systemutvecklingsprocessen, Research report in Swedish, Deptartment of Computer and Information Science, Linköping University. 1991.
34. Goldkuhl G Välgrundad Metodutveckling (in Swedish, Well-Grounded Method Development), Linköping university, Linköping, Sweden. 1994.
35. Goldkuhl G The Grounding of Usable Knowledge: An Inquiry in the Epistemology of Action Knowledge, Linköping University, CMTO Research Papers 1999:03, Linköping, Sweden. 1999.

36. Goldkuhl G, Lind M and Seigerroth U Method Integration: The Need for a Learning Perspective, IEE Proceedings - Software, 145, (4), 113–118. 1998.
37. Goldkuhl G and Röstlinger A Förändringsanalys: Arbetsmetodik Och Förhållningssätt För Goda Förändringsbeslut, Studentlitteratur, Lund. 1988.
38. Goldkuhl G and Röstlinger A Joint Elicitation of Problems: An Important Aspect of Change Analysis, In Human, Organizational, and Social Dimensions of Information Systems Development, (Eds, Avison D E, et al.), North-Holland, pp. 107–125. 1993.
39. Goldkuhl G and Ågerfalk P J Actability: A Way to Understand Information Systems Pragmatics, In Coordination and Communication Using Signs: Studies in Organisational Semiotics 2, (Eds, Liu K, et al.) Kluwer Academic Publishers, Boston, pp. 85–113. 2002.
40. Habermas J, Carleheden M and Molander A Kommunikativt Handlande : Texter Om Språk, Rationalitet Och Samhälle, 2. uppl., Daidalos, Göteborg. 1996.
41. Harmsen A Situational Method Engineering, Doctoral Dissertation, Moret Ernst & Young Management Consultants, Utrecht, The Netherlands. F 1997.
42. Hegel G W F, Manning Delaney B and Wallenstein S-O Andens Fenomenologi, Thales, Stockholm. 2008.
43. Henderson-Sellers B Who Needs an Object-Oriented Methodology Anyway?, Journal of OOP, 8, (6). 1995.
44. Henderson-Sellers B, France R, Georg G and Reddy R A Method Engineering Approach to Developing Aspect-Oriented Modelling Processes Based on the Open Process Framework, Information and Software Technology, 49, 761-733. 2007.
45. Hevner A R, March S T and Jinsoo P Design Science in Information Systems Research, MIS Quarterly, 28, (1), 75-105. 2004.
46. Hirschheim R and Klein H Rationality Concepts in Information Systems Development Methodologies, Accounting, Management and Information Technologies, 1, (2). 1991.

47. Hirschheim R, Klein H and Iivari J A Comparison of Five Alternative Approaches to Information Systems Development, Australian Journal of Information Systems, 5, (1). 1997.
48. Hirschheim R and Klein H K Four Paradigms of Information Systems Development, Communications of the ACM, 32, (10), 1199–1216. 1989.
49. Hirschheim R, Klein H K and Lyytinen K Exploring the Intellectual Structures of Information Systems Development: A Social Action Theoretic Analysis, Accounting, Management and Information Technologies, 6, (1–2), 1–64. 1996.
50. Hirschheim R and Newman M Symbolism and Information Systems Development: Myth, Metaphor and Magic, Information Systems Research, vol. 2, (1). 1996
51. Iivari J, Hirschheim R and Klein H A Dynamic Framework for Classifying Information Systems Development Methodologies and Approaches, Journal of Management Information Systems, 17, (3), 179-219. 2000.
52. Iivari J and Maansaari J The Usage of Systems Development Methods: Are We Stuck to Old Practice?, Information and Software Technology, 40, 501–510. 1996