# РОЗДІЛ 8

# АНАЛІЗ, МЕТОДОЛОГІЇ ПРОЕКТУВАННЯ ТА МОДЕЛІ АРХІТЕКТУР ІНФОРМАЦІЙНО-ІННОВАЦІЙНИХ ЕКОЛОГО-ЕКОНОМІЧНИХ СИСТЕМ

## *8.1. Аналіз методологій проектування інформаційних систем*

**Інформаційні системи та класифікація методологій проектування ІС.** На сучасному етапі розвитку суспільства інформаційні системи дедалі ширше використовуються в управлінні виробничими процесами, галузями та економікою в цілому.

Економічна інформаційна система (ІС) — це програмно-апаратна система, призначена для автоматизації діяльності кінцевих користувачів з управління економічним об'єктом, що забезпечує відповідно до закладених логікою обробки можливості збирання, нагромадження, зберігання, обробки і видачі за запитом інформації.

Сучасні економічні інформаційні системи створюються для обробки великих обсягів інформації при жорстких обмеженнях на час видачі результатів. Вони мають складну формалізацію процедур прийняття рішень для більшості задач, високий рівень інтеграції елементів, які входять до складу системи, значну кількість зв'язків між елементами, характеризуються гнучкістю і можливістю модифікації. ІС відрізняються за типами об'єктів управління, характером та обсягом розв'язуваних задач та ін.

Коли говорять про методологію проектування, то мають на увазі, що методологія реалізується через конкретні технології та стандарти (зокрема графічного моделювання), що їх підтримують, методи та інструментальні засоби, які забезпечують виконання процесів життєвого циклу [1].

У роботі [2], яка описує галузь знань «Проектування програмного забезпечення», виділяють дві основні групи нотацій у проектуванні: структурні описи (статичні); поведінкові описи (динамічні представлення), а також низка основних груп методів проектування: функціонально-орієнтоване (структурне проектування); проектування на основі структур даних; об'єктне проектування; компонентне проектування та ін.

Таким чином, з позицій використовуваних стандартів графічного моделювання та методів проектування можна виділити три основі групи методологій проектування інформаційних систем:

1) методології структурного проектування і моделювання (в основі яких лежить побудова графічних моделей функцій і процесів, що відбуваються в інформаційних системах);

2) методології об'єктного проектування і моделювання, в основі яких — виділення певних сутностей-об'єктів майбутньої системи;

3) методології компонентного проектування.

До кожної з груп належить велика кількість методологій, в основі яких лежать одні й ті самі стандарти графічного моделювання та розуміння того, що вважати основою для проектування інформаційних систем. Кожна з груп методологій має ряд інструментальних засобів, що її підтримують. Ми спробуємо проаналізувати можливості цих методологій щодо охоплення процесів життєвого циклу інформаційної системи, основуючись здебільшого на нотаціях, що використовуються різних групах і методах (оскільки вважаємо їх первинними), деякі приклади інструментальних засобів будемо наводити лише для того, щоб продемонструвати наявні реалізації.

**Характеристика методології структурного аналізу і проектування.** На даний час існують ряд методології структурного аналізу і проектування, в яких визначаються основні роботи з проектування, їх послідовність, правила використання операцій та методів [3], серед яких найвідоміші такі:

- SADT (Structured Analysis and Design Technigue) [4]; методології, розроблені на його основі, наприклад, IDEF0 (www.idef.com), які використовують функціональні діаграми;
- група методологій структурного аналізу і проектування SA / SD, в основі яких лежить використання діаграм потоків даних (DFD):

SA (Structured Analysis) структурного аналізу Де Марко [5];

SD (Structured Design) структурного проектування Стівенса, Маєра, Константайна [6];

SSA (Structured Systems Analysis) структурного системного аналізу Гейна — Сарсона [7];

SA/SD структурного системного аналізу і проектування Йордана [8];

SRD (Structured Requirements Definition), розроблена Ken Orr в середині 1970-х років [9];

SSADM (Structured Systems Analysis and Design Method) [10] створена на початку 1980-х років і прийнята в 1993 р. як національний стандарт Великобританії та ін.;

- методології структурного проектування систем реального часу, що включають контроль потоків робіт і діаграми станів:

SDRTS (Structured Design of Real Time Systems) структурного проектування систем реального часу Уорда-Меллора [11], розширення SA/SD;

SA/RT (Structured Analysis with Real-time-Extensions) структурного аналізу з розширенням для систем реального часу Хартлі [12], аналог SDRTS, та ін.;

IE (Information Engineering) Інформаційного моделювання Мартіна [13] та ін.

Існують різноманітні класифікації структурних методологій:

1) стосовно шкіл: Software Engineering (SE) і Information Engineering (IE);

2) за порядком побудови моделі: процедурно-орієнтовані, орієнтовані на дані; інформаційно-орієнтовані;

3) за типом цільових систем: для систем реального часу та інформаційних систем [14].

Так, школа SE передбачає низхідний поетапний підхід до розробки програмних систем, у результаті якого здійснюється поетапна декомпозиція функцій системи до рівня, достатнього для кодування. Такий підхід може використовуватися як у розробці систем реального часу, так і в розробці інформаційних систем. IE являє собою дисципліну побудови систем взагалі, а не лише розробки програмних систем і включає етапи вищого рівня (наприклад, стратегічне планування), однак на етапі проектування програмних систем ці дисципліни аналогічні. Школа SE використовується лише для побудови інформаційних систем.

Процедурно-орієнтований підхід регламентує первинність проектування функціональних компонентів стосовно проектування структур даних: вимоги до даних розкриваються через функціональні вимоги. При підході, орієнтованому на дані, вхід і вихід є найважливішими — структури даних визначаються першими, а процедурні компоненти є похідними від даних. Інформаційно-орієнтований підхід, як частина IE-дисципліни, відрізняється від підходу, орієнтованого на дані, тим, що дозволяє працювати з неієрархічними структурами даних.

Основна особливість систем реального часу полягає в тому, що вони контролюються зовнішніми подіями, на які повинна бути відповідна реакція. Такі структурні методології підтримують цю особливість.

Різні методології в тій чи тій мірі описують різні структурні аспекти ІС за допомогою різних типів діаграм. Так, у межах одні-

єї методології можуть використовуватися і діаграми потоків даних (DFD) — для відображення функціонування системи, сумісно зі словниками даних і специфікаціями процесів; і діаграми сутність-зв'язок (ERD) — для відображення даних; і навіть діаграми переходів станів (STD) — для відображення поведінки системи і часі.

Співвідношення застосування в існуючих CASE-засобах методів структурного аналізу становить, за матеріалами CASE Consulting Group, 90 % для DFD і 10 % — для SADT. За іншими даними, це співвідношення виглядає як 94 до 3 % (ще 3 % CASE-засобів використовують інші структурні методи).

Сучасні інструментальні засоби можуть підтримувати відразу кілька методології за рахунок реалізації різноманітних діаграмних технік. Для прикладу можна назвати:

- AllFusion Process modeler 4.1 (Bpwin 4.1) (www.ca.com) — підтримує IDEF0, IDEF3 та DFD-нотації;
- Aonix (www.aonix.com) – підтримує як методи SA/SD, так і SA-RT;
- System Architect (www.popkin.com) — підтримує методи Гейна — Сарсона, Йордона-Де-Марко, Йорда-Меллора, SSADM;
- WinA&D 5.1 (http://www.excelsoftware.com) — дозволяє зображувати DFD, ERD та інші моделі.

Залежно від використовуваних методіві засобів, методології структурного аналізу і проектування дозволяють підтримувати певні стадії життєвого циклу ІС. Більшість структурних методології підтримують у термінах ДСТУ 34.601-90 1-5 стадії. У термінах спіральної моделі життєвого циклу це фази аналізу вимог і проектування. Тобто надають можливості формалізації знань про вимоги до майбутньої системи та специфікації компонентів системи. Детально структурні методології описані в [15].

Узагальнимо, що всі методології структурного проектування ІС передбачають низхідний підхід, за якого:

— здійснюється поетапне розбиття системи на функціональні підсистеми (функції — підфункції — задачі, аж до рівня процедур);

— формується ієрархічна структура функцій системи з обмеженою кількістю елементів на одному рівні (3–7);

— функції системи можуть ув'язуватися з даними та об'єктами, що їх виконують.

При цьому система, що автоматизується, зберігає цілісне представлення, у якому всі частини взаємопов'язані.

Важливим принципом цього підходу є саме низхідне проектування, оскільки у разі розробки системи «знизу вверх», тобто від

окремих задач до всієї системи, — вважається, що цілісність губиться, виникають проблеми при інформаційному стикуванні окремих компонентів.

Поетапне виділення підфункцій спрощує аналіз, оскільки дозволяє на початкових етапах не заглиблюватися в деталі виконання різних функцій, а розглядати їх в цілому, поступово деталізуючи на наступному кроці. Як результат, складові частини задачі організовуються в ієрархічні деревоподібні структури.

Іншим важливим принципом є широке використання графічних нотацій, оскільки візуальне сприйняття інформації полегшує процеси проектування.

Утім найважливішим здобутком структурного проектування, на нашу думку, стало створення моделі даних ІС, що пов'язана із функціями.

Як свідчить практика, у відомих CASE-засобах ця модель даних відривається на певному етапі від функцій, і до кодогенерації доходить окремо. Однак структурні методології можна розглядати як перші кроки до пов'язування процесів проектування з процесами розробки, що вказали на перспективи і дозволяють автоматично генерувати системи на основі побудованих моделей.

Також у різних методологіях структурного проектування ми можемо спостерігати, як поступово додаються нові описові виміри інформаційних систем. Так, SADT чітко регламентувала опис системи з позицій її функцій і використовуваних для них даних, і лише вказувала на можливість використання інших описів. SA/SD вже додатково передбачає ідентифікацію об'єктів, що надають необхідну для системи інформацію; деталізацію опису процесів та окремі діаграми для опису логіки взаємодії програмних модулів.

Проте ані деталізовані описи процесів, ані широкі можливості моделювання реалізацій за допомогою структурних карт на сьогодні не змогли стати більше, ніж моделями.

**Можливості методології об'єктного аналізу і проектування**. Об'єктний підхід до проектування з'явився як логічна потреба в результаті еволюції технологій програмування у 1990-х роках [16]. Так, об'єктний підхід до програмування передбачає:

— опис об'єктів, як моделей реального світу;

— сполучення структур даних з методами їхньої обробки в абстрактних типах даних — класах об'єктів;

— наслідування властивостей класів підкласами та утворення ієрархій наслідування та інші.

Для проектування інформаційних систем, в яких передбачалось використовувати об'єктні мови програмування, логічною була ду-

мка використовувати середовища, в яких би система відразу описувалась з позицій об'єктного підходу.

Різними авторами було створено ряд об'єктно-орієнтованих методів візуального моделювання і проектування інформаційних систем. Однак жоден з цих методів не дістав загального визнання поки Буч, Рембо і Якобсон не поставили перед собою завдання створити уніфіковану мову моделювання (Unified Modeling Language, UML) [17].

Головними у розробці UML були такі цілі:

- надати користувачам готову до використання виразну мову візуального моделювання, що дозволяє їм розробляти осмислені моделі й обмінюватися ними;
- передбачити механізми розширюваності та спеціалізації для розширення базових концепцій;
- забезпечити незалежність від конкретних мов програмування і процесів розробки;
- забезпечити формальну основу для розуміння цієї мови моделювання;
- стимулювати зростання ринку об'єктно-орієнтованих інструментальних засобів;
- передбачити підтримку таких високорівневих концепцій розробки, як співробітництво, середовища, зразки і компоненти;
- інтегрувати кращий практичний досвід.

Значимість розробки уніфікованої мови моделювання відмітили цілий ряд компаній — розробників програмного забезпечення, які взяли активну участь у розробці версії 1.0 мови UML. В даний час стандартом є мова UML 2.0. і на ринку програмного забезпечення CASE-засобів представлено цілий ряд програмних продуктів, що підтримують цю версію мови, серед яких Rational Rose, Paradigm Plus, Select Enterprise, Microsoft Visual Modeler for Visual Basic та ін.

Повні специфікації стандарту OMG на мову UML надані для вільного доступу на http://www.omg.org. UML 2.4.1 прийнято як міжнародний стандарт ISO / IEC 19505-1, 19505-2. Остання версія UML 2.5 опублікована в червні 2015 р. (рис. 8.1). Детально мова UML описана у [17—19].

Діаграми мови надають досить широкі можливості для опису інформаційних систем як з позицій моделей даних, так і з позицій опису функцій систем та об'єктів, задіяних у них. При цьому моделі даних описують на якісно новому (об'єктному) рівні із зазначенням методів обробки.

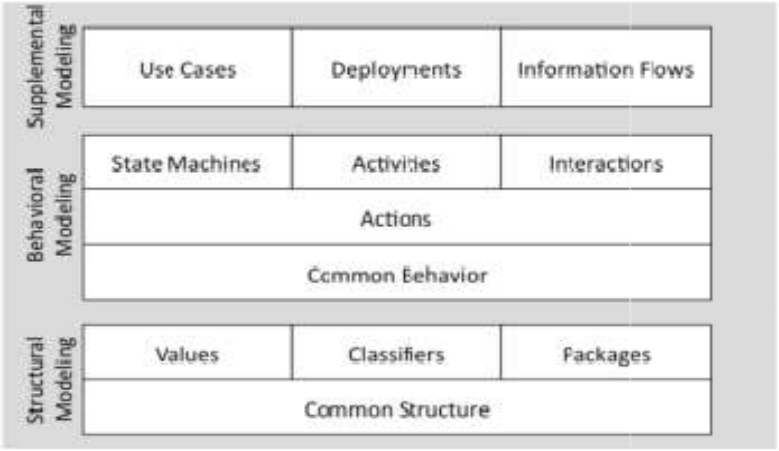


Рис. 8.1. Структура мови UML 2.5

Так, Г. Буч [16] наголошує, що у процесі об'єктно-орієнтованого аналізу: здійснюється ідентифікація об'єктів і їхніх властивостей; установлюється перелік операцій (методів обробки), виконуваних над кожним об'єктом залежно від його стану (подій); визначаються зв'язки між об'єктами для утворення класів; встановлюються вимоги до інтерфейсу з об'єктами.

Опис функцій носить дещо інший характер, ніж при структурному моделюванні — немає вже поступової деталізації. Натомість передбачається і трирівнева ієрархія опису. Найзагальніше представлення — за допомогою випадків використання і деталізація на діаграмах взаємодії (між об'єктами) і діаграмах станів (конкретизується функціонування окремих об'єктів).

Проблема полягає у тому, що основним застосуванням UML на сьогодні все ще залишається наочне відображення моделі проектованої системи для обговорення і спілкування розробників між собою та із замовником. Складність, на нашу думку, полягає в тому, що не деталізовані специфікації, за допомогою яких відбувався б перехід від проектних моделей до реалізацій систем.

У результаті розробники інструментальних засобів на власний розсуд реалізують ці можливості. Так, IBM Rational Rose, орієнтований на інтеграцію з іншими програмними засобами для забезпечення всього життєвого циклу створення інформаційних систем, декларуючи підтримку UML різних версій дозволяє будувати діаграми всіх типів. Однак максимум, що далі на основі цих діаграм може бути здійснено — генерація класів. Усі описи

не дають результатів на етапі розробки (генерації програмних кодів) — стани та дії над об'єктами, за допомогою яких можна описувати алгоритми описуються впусту.

Наприклад, у такому засобі, як Gentleware Poseidon for UML (www.gentleware.com), можливості роботи з кодом явно ширші: передбачається редагування коду для методів діаграм класів і навіть (у варіанті Embedded Edition 5.0) генерація коду програм на C++, ANSI C, та Java за діаграмами станів.

Популярність легендарного Telelogic TAU (тепер Rational TAU http://www-01.ibm.com/software/awdtools/tau/) також була пов'язана з можливостями генерації кодів програм за діаграмами станів.

У [20] серед проблем використання UML для опису програм вказуються:

- громіздкість відображення відповідностей між елементами на різних рівнях представлення;
- відсутність наочності взаємодії інтерфейсів і компонентів;
- неможливість опису динаміки поводження системи в цілому.

Вказують також на непридатність UML у використанні на великих складних спеціалізованих проектах, що мають модульну структуру і для яких розробляються нові мови представлення архітектури [21]. Саме до таких проектів можна віднести проекти з розробки віртуальних організацій.

Для розширення можливостей мови OMG розробляли ряд профайлів, перелік яких подано на сайті http://www.omg.org/technology/documents/profile_catalog.htm. Зокрема, для інтеграції додатків підприємства (Enterprise Application Integration, EAI), роботи з розподіленими об'єктами -Enterprise Distributed Object Computing (EDOC) та ін. Однак, використання профайлів не знайшло широкої підтримки інструментальними засобами, за винятком одного з останніх — SysML (http://www.omgsysml.org/), що став самостійною мовою моделювання, яка використовується в рамках концепції модель-орієнтованої розробки інформаційних систем.

Підставами, що забезпечують усебічний опис систем SysML, виділяють такі:

1. Моделювання структури системи (ієрархії та взаємодії між частинами) за рахунок модифікації діаграм класів у два типи діаграм блоків.

2. Моделювання поведінки системи за рахунок модифікації діаграм дійі використання діаграм станів і послідованостей.

3. Моделювання вимог до системи за рахунок введення нового типу діаграм для специфікації вимог — Requirement Diagrams.

4. Моделювання властивостей об'єктів за допомогою нового типу діаграм — Parametric Diagrams.

В описі структури системи в SysML замість класичних UML-діаграм (класів, об'єктів і ін.) використовують діаграми внутрішніх і зовнішніх блоків — модульних одиниць, інкапсулюючих атрибути, операції й обмеження, а також розподіл в інші модельні елементи та вимоги.

Загалом, діаграми UML набули в SysML конкретного змісту, а також зв'язності через механізми розподілу (allocation), зв'язку значень та верифікації. Мова дістала можливості для підтримки розробки систем на різних стадіях — від специфікації вимог, аналізу і проектування до тестування працездатності.

Як і UML, SysML є мовою, а не методологією. Процеси ж розробки інформаційних систем з використанням SysML описані в [22—24]. Варіант використання SysML на різних стадіях створення ІС від компанії Telelogic подано на рис. 8.2.

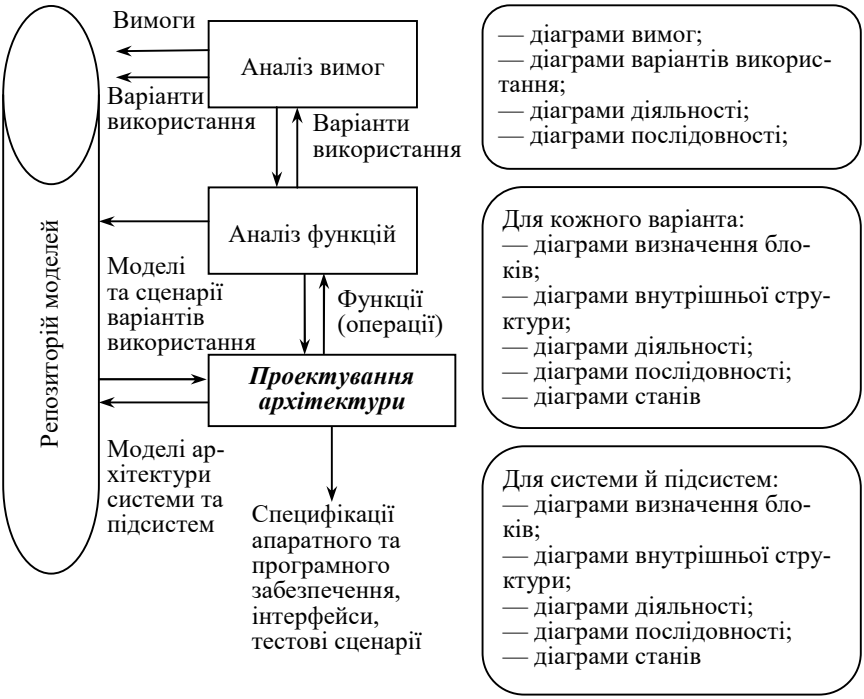


Рис. 8.2. Процес розробки систем, оснований на використанні SysML

У SysML підтримується обмін моделями і даними в форматі XML Metadata Interchange (XMI) і, як витікає з рис. 8.2, отримані моделі передбачається зберігати в репозиторії з метою їх повторного використання.

На сьогодні SysML уособлює найсистемніший підхід до графічного моделювання складних систем, орієнтована саме на розробку на основі моделей. Існує ряд засобів, що її підтримують: Artisan Studio, SysML Toolkit (EmbeddedPlus), Magic Draw, Sparx Systems Enterprise Architect, IBM / Telelogic Tau and Rhapsody, TopCased, Visio SysML template та ін.

Крім UML серед графічних мов, на основі яких здійснюється об'єктне проектування інформаційних систем, знайшла поширення, хоч і значно менше за UML, мова SDL (Specification and Description Language).

SDL є об'єктно-орієнтованою формальною мовою, розробленою Міжнародним телекомунікаційним союзом (ITU-T) як рекомендації Z.100. Можливості мови передбачають опис структури, поведінки і даних системи [25]. На відміну від UML, SDL — формальна мова, оскільки передбачає точну визначеність символів і понять. Типовими сферами використання стандарту вважаються телекомунікаційні, аерокосмічні та ін. складні розподілені системи, тобто ті системи, в яких моделювання функціональності не менш важливе за моделювання об'єктів.

Розроблені специфікації інтерфейсів SDL з іншими мовами моделювання подані на сайті http://www.itu.int/rec/T-REC-Z.100/en. У [25] відношення між іншими мовами моделювання та SDL подано у вигляді схеми (рис. 8.3).

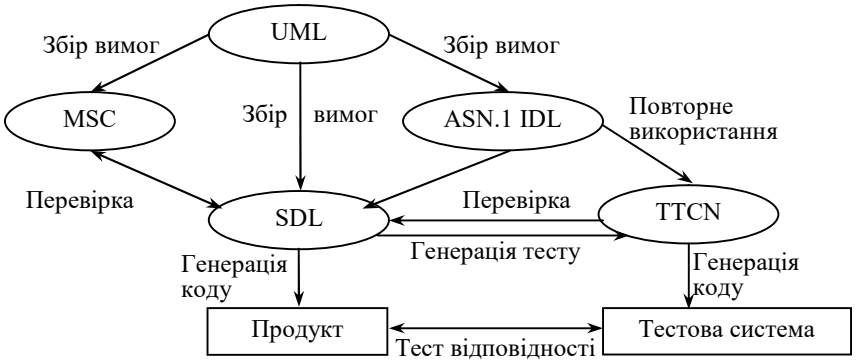


Рис. 8.3. Відношення інших мов з SDL

Під MSC розуміють Message Sequence Chart [26] діаграми послідовності. ASN.1 — abstract system notation. IDL — interface description language для CORBA-архітектури [27]. TTCN — Testing and Test Control Notation version [28]. Зазначені мови графічного опису нами не розглядаються через їх спеціалізованість.

UML показана як мова, що призначена для використання на стадії аналізу вимог до інформаційної системи та їх опису. Отримані в результаті класи можуть бути використані в діаграмах SDL (так само, як і дані, описані в ASN.1 та IDL). З специфікації системи на SDL можуть бути генеровані тестові модулі в TTCN або ж коди системи.

Одна з найважливіших переваг SDL у зіставленні з UML полягає у тому, що SDL може виконувати ієрархічну декомпозицію внутрішньої структури системи. У такий спосіб можливе моделювання структур довільної складності та опис динаміки системи в цілому.

Реалізована мова в IBM Rational / Telelogic SDL Suite (www.ibm.com), SDL Trados Studio 2009 (www.sdl.com), Cinderella SDL (http://www.cinderella.dk/csdl.html) та ін.

Як зазначається у [29], мови описів і специфікацій SDL-2000 і MSC суттєво підвищують рівень автоматизації процесів проектування алгоритмів апаратно-програмних систем. Однак вони не забезпечують коректної специфікації вимог до систем. Серед недоліків називають такі:

- відсутня регламентація взаємодії блоків і підблоків у SDL (напрями зв'язків і часовий порядок) — опису властивостей;
- складність операційної семантики;
- необхідність використання ряду погано інкапсульованих мов описів і специфікацій;
- відсутність підтримки програмними засобами проектування специфікацій у повному обсязі.

Через наявність таких недоліків і проводяться дослідження розвитку SDL. Зокрема, можна назвати роботи зі створення мови SDL/PLUS [30], системи програмування алгоритмів і моделей (СПАМ) [31] та поведінкової моделі Real, що об'єднує STD і SDL нотації [32].

Оскільки як мова SDL більш погоджена за UML і має чіткішу семантичну основу, а UML — виразніша і широко використовувана. Свого часу проводились дослідження з об'єднання двох мов. Так, серед цих досліджень, крім [33], висвітлювались пропозиції зі створення мови SMDL, що поєднувала б можливості UML і SDL [29].

Крім того, для ефективного аналізу та верифікації систем, описаних за допомогою SDL, розроблялися методи автоматичного відображення текстових моделей SDL у мережі Петрі [34].

Однак опис мов — UML чи SDL, не містить відомостей про те, яким чином і в якій послідовності варто розробляти діаграми під час виконання конкретних проектів. Відповідна інформація задається об'єктними методологіями проектування інформаційних систем.

Існують методології об'єктно-орієнтованого проектування, в основі яких лежить використання мови UML. Ці методології різняться процесом проектування, і використання тієї чи тієї методології залежить від типу розроблюваного програмного забезпечення.

Серед відомих методологій об'єктно-орієнтованого проектування можна назвати RUP (підтримується засобами Rational Software), MSF (Microsoft), Oracle PJM (Oracle) та ін., описані у [15, 35].

Однією з найвідоміших методологій об'єктного проектування на сьогодні можна назвати RUP (Rational Unified Process — раціональний уніфікований процес) [36], що розроблена Якобсоном, Бучем і Рембо у 1999 р. Вона передбачає чітко визначений процес, що охоплює увесь життєвий цикл проекту, ролі та відповідальність окремих виконавців, виконувані ними задачі, використовувані в процесі розробки моделі, звіти та ін.

Отже, виділяють статичну і динамічну структури RUP.

Динамічна структура RUP складається з чотирьох фаз, які також можуть розподілятись на ітерації:

1) *дослідження* — Inception (визначення меж системи, моделювання бізнес-процесів та робота з вимогами, видалення економічних ризиків): результат — перший прототип системи;

2) *уточнення плану* — Elaboration (опрацювання вимог і вибір основних проектних рішень): концептуальний прототип перетворюється на реальну систему, яку можна протестувати та оцінити через обрані архітектурні рішення;

3) *побудова* — Construction (швидка та економічна розробка коду системи): система готова до передачі замовнику для бета-тестування і прийом-здавальних випробувань;

4) *розгортання* — Transition (підготовка розробленого продукту до передачі замовнику або тиражування і розповсюдження).

Перехід з фази на фазу можливий лише після виконання задач фази та являє собою контрольну точку процесу.

Статична структура RUP складається з дисциплін, які розподіляють на процеси, задачі, артефакти, ролі.

Уся розробка системи розглядається в RUP як процес створення артефактів. Будь-який результат роботи проекту, будь-які вихідні тексти, об'єктні модулі, документи, передані користувачеві, моделі — це підкласи артефактів проекту. Кожен член проектної групи створює свої артефакти і несе за них відповідальність. Програміст створює програму, керівник — проектний план, а аналітик — моделі системи.

Практично RUP — не набір жорстких правил, а рекомендації з використання кращих практичних методів розробки ПЗ, таких як:

- ітеративна розробка;
- керування вимогами;
- використання модульних архітектур;
- візуальне моделювання;
- перевірка якості;
- відстеження змін.

У разі необхідності виконання формальних вимог вітчизняних або закордонних стандартів можна розробити шаблони документів, які будуть створюватися автоматично на основі існуючих моделей [37].

Моделювання здійснюється за допомогою Software Process Engineering Metamodel (SPEM) — стандарта моделювання процесів, основаного на Unified Modeling Language (UML). Для забезпечення інструментальної підтримки всіх процесів життєвого циклу RUP рекомендує використання спеціалізованих інструменттальних засобів IBM Rational.

Для підтримки моделювання на основі методології RUP у Rational Rose Enterprise використовують Model Framework, що забезпечує загальну структуру моделі; керування стилями; визначення мінімальної кількості діаграм для реалізації проекту; зв'язок дій у RUP із діаграмами; забезпечує основу для генерації звітів. Цей каркас моделювання передбачає чотири варіанти подання моделі. Для кожного варіанта представлення передбачається використання певних типів діаграм UML (табл. 8.1).

Таким чином, можливості моделювання інформаційних систем у RUP повністю визначаються можливостями UML.

Однак більшість сучасних методологій проектування носять процесний характер і не орієнтовані на використання якоїсь встановленої мови моделювання. Це стосується в першу чергу гнучких (Agile) методологій — SCRUM, eXtreme Programming (XP), Crystal, Adaptive Software Development (ASD), Feature Driven Development (FDD), Dynamic System Development Method (DSDM) та ін. Тобто в них специфікують процеси, що відбува-

ються під час розробки, вимоги до команди розробників, однак не вказують мови і засоби моделювання, як це робиться, скажімо, у RUP. Одна і та сама методологія може використовувати різні інструменти залежно від досвіду та переваг команди розробників.

*Таблиця 8.1*

**ВИКОРИСТАННЯ ЕЛЕМЕНТІВ UML У RUP**

| Представлення | Елементи |
|---|---|
| Варантів використання | Актори<br>Варіанти використання<br>Асоціації<br>Діаграми варіантів використання<br>Діаграми послідовності<br>Діаграми кооперацій<br>Пакети |
| Логічне представлення | Класи<br>Діаграми класів<br>Асоціації<br>Діаграми дій<br>Діаграми станів<br>Пакети |
| Представлення компонентів | Компоненти<br>Діаграми компонентів<br>Пакети |
| Представлення розміщення | Процеси<br>Процесори<br>Пристрої<br>Діаграми розміщення |

**Компонентний підхід до проектування.** Появу компонентного підходу до проектування розглядають як розвиток об'єктного підходу для проектування великих розподілених систем.

Компонентна розробка програмного забезпечення (Component Based Development — CBD) — це спосіб розробки, при якому можливе повторне використання раніше створених компонентів, за умови, що вони розроблялися з цією можливістю повторного використання.

До плюсів повторного використання програмного забезпечення у роботі [38] відноситься:

- підвищення надійності;
- зменшення проектних ризиків;
- ефективне використання фахівців;
- дотримання стандартів;
- прискорення розробки.

У той же час виникають проблеми підвищення вартості супроводу системи, пошуку і адаптації компонентів та ін. Під компонентом розуміють незалежний модуль програмного коду, призначений для повторного використання і розгортання [39].

Компоненти відрізняються також від класів об'єктно-орієнтованих мов за рядом характеристик:

- компонент є більшою структурною одиницею, ніж клас. Реалізація компонента часто складається з кількох тісно зв'язаних один з одним класів;
- компонент, як правило, не прив'язаний до певної мови програмування.

У той же час поняття компонента відмінне від традиційного поняття програмного модуля, оскільки компонент — самостійна атомарна для розгортування програмна одиниця, яка може постачатися чи видалятися окремо від усієї іншої системи, тоді як програмний модуль має чітко описаний інтерфейс з оточенням.

Набір правил визначення інтерфейсів компонентів та їхніх реалізацій, а також правил, за якими компоненти працюють у системі і взаємодіють один з одним, прийнято поєднувати під назвою компонентної моделі (component model).

Існує кілька компонентних моделей від різних розробників, серед найвідоміших:

— COM (Component Object Model), COM+ від Microsoft [40];

— EJB (Enterprise Java-Beans) від Sun Microsystems [41];

— CCM (CORBA Component Model) від OMG (http://www.omg.org/spec/CORBA/3.1/).

Компонентна модель COM визначає протокол для створення і використання компонентів як усередині одного процесу, так і між різними процесами або комп'ютерами. Додатки для COM-моделі можуть створюватися засобами таких мов і середовищ розробки, як Visual Basic, C++, NET та ін., однак орієнтовані на операційні системи від Microsoft, тоді як JavaBeans не має властивості незалежності від мови програмування, утім підтримує різні платформи. CORBA відрізняється досить громіздким IDL-інтерфейсом і як результат — складністю відображення однієї мови в іншу.

Застосування компонентного програмування покликане забезпечити простішу і швидшу розробку прикладного програмного забезпечення на основі використання готових модулів-компонент. Повторне використання компонентів дозволяє істотно скоротити витрати і терміни розробки програмного забезпечення. У найбільшій мірі орієнтованими на компонентну розробку є гнучкі методології [15] та методології Microsoft Solutions

Framework (MSF) — MSF for Agile Software Development і MSF for CMMI Process Improvement.

MSF рекомендує якнайчастіше збирати поточні версії усіх компонентів рішення для проведення тестування й аналізу. Цей підхід застосовується як до розробки програмного коду, так і до створення компонентів апаратного і програмного забезпечення [42].

У середовище Visual Studio Team System інтегруються шаблони процесів залежно від того, яка процесна методологія розробки обрана для проекту: якийсь з варіантів гнучкої (Agile) розробки, SCRUM, EUP, FDD чи CMMI (можуть бути завантажені з http://msdn.microsoft.com/ru-ru/vsts2008/aa718795.aspx). У Visual Studio Team System можуть використовуватися також візуальні конструктори з багатими можливостями для моделювання.

Загалом, на рівні платформ програмування компонентність підтримується досить широко. Однак на рівні засобів візуального моделювання, призначених для аналізу і проектування таких систем, компонентність підтримується слабко і неповно.

У процесі компонентної розробки низка авторів пропонує опиратися на моделювання на основі UML [43]. При цьому в UML компонента ототожнюється з класом, або елементом, фізичної структури ПЗ. У роботі [44] досліджено можливості використання інших методів графічного моделювання при компонентній розробці програмного забезпечення, зокрема, мови SDL (Specification and Description Language), моделей методології ROOM та ін. Автор згадує загальність існуючих методологій моделювання, їх недостатню практичну орієнтованість і робить спробу об'єднати об'єктно-орієнтовані засоби аналізу систем (UML) із засобами детального проектування (SDL, ROOM) у рамках єдиної мови візуального проектування компонентного програмного забезпечення.

Слід наголосити, що досить часто у проектуванні систем з використанням компонентних технологій взагалі не передбачено графічне моделювання і розробники опираються лише на документування коду.

Так, компонентна розробка в рамках COM [45] передбачає використання ієрархії класів MFC — узгодження префіксів імен класів, інтерфейсів тощо під час кодування.

### *Висновки*

Таким чином, на прикладі компонентної розробки можна побачити, як графічне моделювання може бути витіснене за рахунок подання у доступній формі семантики взаємозв'язків між елементами системи.

Узагальнюючи, можна ствердити, що поява компонентної розробки програмного забезпечення висунула нові вимоги до моделювання систем і використання існуючих мов візуального моделювання виявилось занадто громіздким і таким, що сповільнює процеси. Як структурне програмування обумовило появу структурних методів моделювання систем, об'єктно-орієнтоване програмування — появу об'єктно-орієнтованого підходу до проектування та об'єктно-орієнтованих мов візуального моделювання, так і компонентна розробка програмного забезпечення вимагає іншого погляду на процеси моделювання систем. Тому ряд колективів розробляють мови моделювання, орієнтовані на моделювання саме компонентних архітектур, серед яких Platform-Independent Component Modeling Language (PICML), розроблена у [46], а також Cadena [47], розроблена Kansas State University (KSU) та ін.

Слід також наголосити, що на сьогодні ідеї компонентного підходу до поширення бібліотек компонентів і формування на їх основі програм набувають глобального розмаху, й зокрема, через широке використання таких засобів розробки двох- і тривимірних додатків, як Unity (https://unity3d.com) і та ін.

### *Список використаних джерел до п. 8.1*


1. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. Москва: Финансы и статистика, 1998. 98 с.
2. IEEE Guide to the Software Engineering Body of Knowledge (1). SWEBOK®, 2004. 335 с.
3. Кальянов Г.Н. Консалтинг при автоматизации предприятий. Подходы, методы, средства. Москва: СИНТЕГ, 1997. 316 с.
4. Марка Д.А., МакГоуэн К. Методология структурного анализа и проектирования SADT. СПб.: Невский диалект, 2003. 560 с.
5. DeMarco T. Structured Analysis and System Specification, Prentice-Hall, Englewood Cliffs, N.J. 1979. 33 с.
6. Stevens W.P., Myers G.J., Constantine L.L. Structured design. *IBM Systems Journal*. 13 (2). P. 115-139, 1974.
7. Structured Systems Analysis: Tools and Techniques, by C. Gane and T. Sarson. New York: IST, Inc., 1977. 373 p.
8. Yourdon, E. and Edward, J. Modern Structured Analysis. Prentice Hall, Englewood Cliffs, N.J. 1989. 661 с.
9. Orr K. Structured requirements definition. Ken Orr. Topeka, KS, 1981. 235 с.
10. Downs E., Clare P., Coe I. Structured Systems Analysis and Design Method, Application and Context. Second Edition. Prentice Hall, Englwood Cliffs, NJ, 1992. 407 с.

11. Ward P.T., Mellor S.J. Structured Development For Real-Time Systems Vol. II: Essential Modeling Techniques. Yourdon Press (Prentice Hall). Englewood Cliffs, NJ, 1985.

12. Hatley D. J., Pirbhai I. A. Strategies for real-time system specification, Dorset House Publishing Co., Inc., New York, NY, 1987. 43 p.

13. Martin, James and Clive Finkelstein. Information Engineering. Technical Report (2 volumes), Savant Institute, Carnforth, Lancs, UK. Nov 1981.

14. Кальянов Г.Н. Консалтинг при автоматизации предприятий. Подходы, методы, средства. Москва: СИНТЕГ, 1997. 316 с.

15. Береза А.М., Козак І.А. Проектування систем оброблення інформації: навч. посібник. Київ: КНЕУ, 2008. 448 с.

16. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++; пер. с англ. 2-е изд. Москва: Издательство Бином, СПб.: Невский диалект, 2000. 560 с.

17. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя: ; пер. с англ. Москва: ДМК, 2000. 432 с.

18. Мацяшек Л.А. Анализ и проектирование информационных систем с помощью UML 2.0. Третье издание. Москва: Вильямс, 2008. 816 с.

19. Фаулер М., Скотт К. UML в кратком изложении. Применение языка объектного моделирования; пер. с англ. Москва: Мир, 1999.

20. Hofmeister C., Nord R.L., Snoi D. Describing Software Architecture with UML. In Proceed of the First Working IFIP Conference on Software Architecture, San Antonia, TX, February 1999. IEEE Computer Society Press, p. 145-160.

21. Abdurazik A. Suitability of the UML as an Architecture Description Language with Application to Testing ISE-TR-00-01, February, 2000. Information and Software Engineering George Mason University, Fairfax, Verginia 22030. URL: http://www.isse.gmu.edu/techrep/2000/00_01_abdurazik.pdf

22. Weilkiens T. Systems Engineering with SysML UML Modeling, Analysis, Design. Denise E. M. Penrose 2007. 320 p.

23. Hoffman H.P., SysML-Based Systems Engineering Using a Model-Driven Development Approach. *Telelogic whitepaper*. 2008. 1 July URL: http://modeling.swd.ru

24. Рыжов Д., Иванов Д. Процесс разработки программно-аппаратных систем на основе визуального моделирования с использованием SysML/UML: материалы конф. Software Engineering Conference (Russia) SEC(R). 2008. URL: www.secr.ru

25. ITU-T, Rec. Z.100, Specification and Description Language (SDL), Geneva, 2000. URL:http://www.iec.org/online/tutorials/acrobat/sdl.pdf

26. ITU-T. Rec. Z.120 Message Sequence Charts (MSC). URL: http://www.itu.int/rec/T-REC-Z.120-200404-I/en.

27. Object Management Group — Common Object Request Broker Arhitecture: Core Specification. (CORBA), v3.0. July 2002. URL: http://www.omg.org/technology/documents/formal/corba_iiop.htm

28. ITU-T. Rec. Z.161: Testing and Test Control Notation version 3: TTCN-3 core language. URL: http://www.itu.int/rec/T-REC-Z.161-200711-I/en

29. Ластовченко М.М., Макаренко Н.Н., Марущак В.И. Интеллектуализация программных средств описа¬ния и спецификации телекоммуникационных сис¬тем и процессов их функционирования. *Проблеми програмування*. 2006. № 1. с. 62-69.

30. Барздинь Я.М., Кальниньт А.А., Строде Ю.Ф., Сыцко В.А. Язык спецификаций SLD/Plus и методы использования. Рига: ЛГУ им. П. Стучки, 1986. 296 с.

31. Ионин Г.Л., Седов Я.Я, Супе В.В. Язык моделирования ПАЛМ. Рига: ЛГУ им. П. Стучки, 1982. 107 с.

32. Кознов Д.В. Поведенческая модель Real — основа для визуальных представлений поведения объектов. Объектно-ориентированное визуальное моделирование; под ред. проф. Терехова А.Н. СПб: Изд-во С.-Петербургского университета, 1999. С. 101-122.

33. ITU-T, ITU Recommendation Z.109: «SDL Combined with UML». URL: http://www.itu.int/itu-doc/itu-t/rec/z/index.html

34. Churina T. G. Colored Petri net approach to modeling of SDL specifications. Joint Bulletin of NCC & IIS. Ser.: Comput. Sci. 2000. N 13. P. 18-39.

35. Крачтен Ф. Введение в Rational Unified Process. Изд. 2-е. Москва: Вильямс, 2002. 240 с.

36. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. Спб.: Питер, 2002. 496 с.,

37. Галахов И.В., Лапыгин Д.В., Новичков А.Н., Подоляк О.Р., Позин Б.А. Автоматизированное создание документов серии ГОСТ 34 и 19 с помощью инструментальных средств фирмы IBM Rational: материалы III Всерос. практ. конф.: «Стандарты в проектах современных информационных систем», 2003.

38. Соммервил И. Инжннерия программного обеспечения; пер. с англ. 6-е изд. Москва: Вильямс, 2002. 624 с.

39. Грищенко В.Н., Лаврищева Е.М. Методы и средства компонентного программирования. *Кибернетика и системный анализ*. 2003. № 1. С. 39-55.

40. Роджерсон Д. Основы COM. Microsoft Corp., 1997. 228 с.

41. Монсон-Хейфел Р. Enterprise JavaBeans. Москва: Символ-Плюс, 2002. 671 с.

42. Microsoft Solutions Framework Модель процессов MSF. вер. 3.1 «Белая книга». *White Paper*. 2002. июнь.

43. Путилин А.Б., Юрагов Е.А. Компонентное моделирование и программирование на языке UML. Практическое руководство по проектированию информационных систем. Москва: Пресс, 2005. 664 с.

44. Кознов Д.В. Визуальное моделирование компонентного программмного обеспечения: дисс. на соискание уч. ст. канд. физ.-мат. наук. СПб, 2000. 346 с.

45. Платт Д.С. Знакомство с Microsoft. NET. Рус. ред. Москва, 2001. 244 с.

46. Balasubramanian K., Balasubramanian J., Parsons J., Gokhale A., Schmidt D.C. A Platform-Independent Component Modeling Language for Distributed Real-Time and Embedded Systems. Proceedings of the 11th IEEE Real Time on Embedded Technology and Applications Symposium. 2005. p. 190-199.

47. Hatcliff J., Deng W., Dwyer M., Jung G., Prasad V. Cadena: An Integrated Development, Analysis, and Verification Environment for Component-based Systems. in Proceedings of the 25th International Conference on Software Engineering, Portland, OR, May 2003.

## *8.2. Моделі архітектури інноваційних підприємств*

### *Вступ*

Наприкінці XX ст. остаточно набула сталості тенденція до неперервного розвитку підприємств. В її основі лежать зміни потреб ринку, бажання економічного зростання, нові можливості і загрози, пов'язані, зокрема з впровадженням нових технологій, перерозподілом ринків, реструктуризацією трудових ресурсів, усвідомленням нових реалій ведення бізнесу. Найчастіше підприємства реалізують невеликі коригування стратегії, організації, процесів або інфраструктури. Водночас тиск зовнішнього середовища робить питанням їхнього виживання інноваційність — необхідність далекосяжного передбачення майбутнього і кардинальних, безпрецедентно швидких трансформацій на основі всіх доступних знань і творчих ідей. Їх підтримку забезпечує дисципліна архітектури підприємства, яка, крім організаційних процесів, передбачає опис усіх ключових елементів підприємства і взаємозв'язків між ними як сукупності моделей. Численність і розмаїття моделей ускладнюють вирішення поставлених завдань, що зумовлює потребу в аналізі підходів до їх створення та вироблення керівних рекомендацій.

***Результати дослідження.*** Питанням трансформації економіки присвячено праці П. Друкера, Дж. Гелбрейта, Р. Тібольда, К. Кларка, Ж. Фурастьє та інших економістів. Зокрема, П. Друкер у теорії інноваційної економіки й підприємницького суспільства [1] стверджує, що нині:

— основною продукцією та начинням усіх товарів і послуг є нові рішення;

— провідну роль в економіці відіграють мільйони малих і середніх підприємств, очолюваних підприємцями, які діють на свій страх і ризик;

— динаміка економіки й суспільства визначається не стільки наукою і вченими, скільки мільйонами людей, які самостійно приймають часто інтуїтивні, творчі рішення;

— метою діяльності корпорацій знову стала максимізація доходу акціонерів на основі іншого менеджменту, з іншими принципами та іншою практикою;

— поряд з трьома попередніми секторами економіки виник і випереджає інших четвертий сектор: галузь безприбуткових суспільних підприємств;

— знання залишаються основним, провідним фактором продуктивності, водночас відбувається реорганізація галузей навколо виробництва знань і реструктуризація всієї економіки навколо сфери виробництва інформації;

— інтелектуалізація праці — основний напрям її розвитку, а витрати на виробництво й поширення знань — основна форма інвестицій;

для розкриття найважливіших економічних процесів крім мікро- і макроекономіки необхідна метаекономіка, яка враховувала б вплив таких потужних неекономічних факторів, як демографія, освіта, нові технології, екологія, тип психології людей, рівень культури та ін.

Ці постулати на рівні підприємства зумовлюють необхідність перетворення процесів керування трансформацією на систематичний, цілісний, всебічний, глибоко обґрунтований підхід, що підтверджується, наприклад, дослідженнями У. Роуза [10]. Основою цього підходу є керування архітектурою — неперервна циклічна діяльність з генерування варіантів побудови підприємства у майбутньому, їх оцінювання, вибору, планування реалізації з моніторингом і внесенням коректив. Велика кількість відповідних рішень має величезну вагу, оскільки часто передбачає докорінні зміни підприємства, носить стратегічний характер, приймається в умовах неповноти інформації і невизначеності наслідків, є ризикованими, вимагають різноаспектних досліджень з обов'язковим використанням великої кількості різнопланових даних і знань з внутрішніх і зовнішніх джерел за умов обов'язкового залучення експертів і працівників різної спеціалізації та рівнів управління підприємства. Отже, нині керування архітектурою само по собі є інноваційною діяльністю.

Архітектура вже стала галуззю теоретичних досліджень і практичних застосувань з великим обсягом напрацювань для всіх типів організаційно-технічних і технологічних систем, охоплюючи такі їх аспекти, як структура, поведінка, артефакти, метрики,

часові характеристики. Водночас описані вище фактори вимагають удосконалення існуючих концепцій архітектури підприємства у трьох головних напрямках:

- розгляд підприємства у контексті його залежності від ближнього та дальнього економічного, соціального, культурного та екологічного оточення;
- динамізм — всю архітектуру, склад і параметри об'єктів, їх структур, бізнес- і виробничих процесів, кінцевих продуктів, а також процесів взаємодії з зовнішніми контрагентами і споживачами слід розглядати як об'єкт для змін за структурою та іншими параметрами;
- груповий характер прийняття рішень зі змінюваним складом групи.

У цьому контексті розглянуто існуючі рамкові моделі архітектури підприємства (методики, фреймворки), що визначають, методи проектування через елементи, з яких складається архітектура, та їх взаємозв'язки (рис. 8.2).

Слід зазначити, що як і стандарти, розроблені такими організаціями, як IEEE, ISO, The Open Group, жодна з яких не є всеохоплюючою, не здобула на практиці привілейованого значення і не стала єдиним стандартом де-факто [4, 8, 9, 11, 13]. Незважаючи на певні тенденції уніфікації, методики різняться за повнотою, спрямованістю, орієнтацією на певне коло користувачів, іншими особливостями. Водночас застосування жодної з методик не визначає повністю результат проектування. Отже, для архітекторів залишається актуальним складне завдання вибору методики, стандартів, інструментальних засобів.

Каноном і відправною точкою для визначення терміна «архітектура» стала запропонована Дж. А. Захманом структура архітектури підприємства [3, 11, 14, 15] — набір узгоджених описів, які співвідносяться з комірками матриці з відображенням істотних аспектів розгляду підприємства (стовпці «Що» — об'єкти, дані, «Як» — дії, функції, «Де» — місця, мережа, «Хто» — люди, ролі, «Коли» — час, «Чому» — мотиви) та послідовності розробки його архітектури.

Рядки матриці подають рівні розгляду архітектури (контекстний, концептуальний, логічний, фізичний, деталізований) відповідно до задіяних / зацікавлених осіб (планувальник, власник, проектувальник, розробник, субпідрядник). Рядки є результатом трансформації під час руху згори донизу за стовпчиком, а не декомпозиції чи деталізації, що підкреслюється застосуванням кольорів, починаючи з версії, виданої у 2011 р. (рис. 8.3).

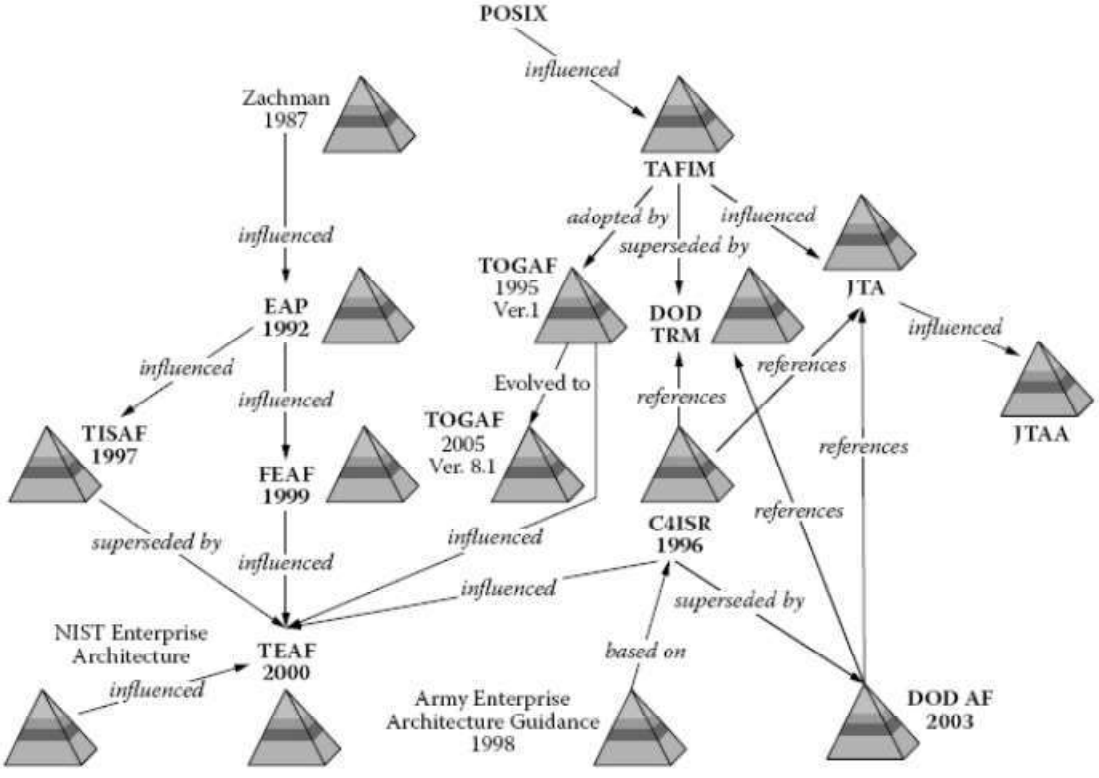


Рис. 8.2. Взаємозв’язки між фреймворками архітектури підприємства

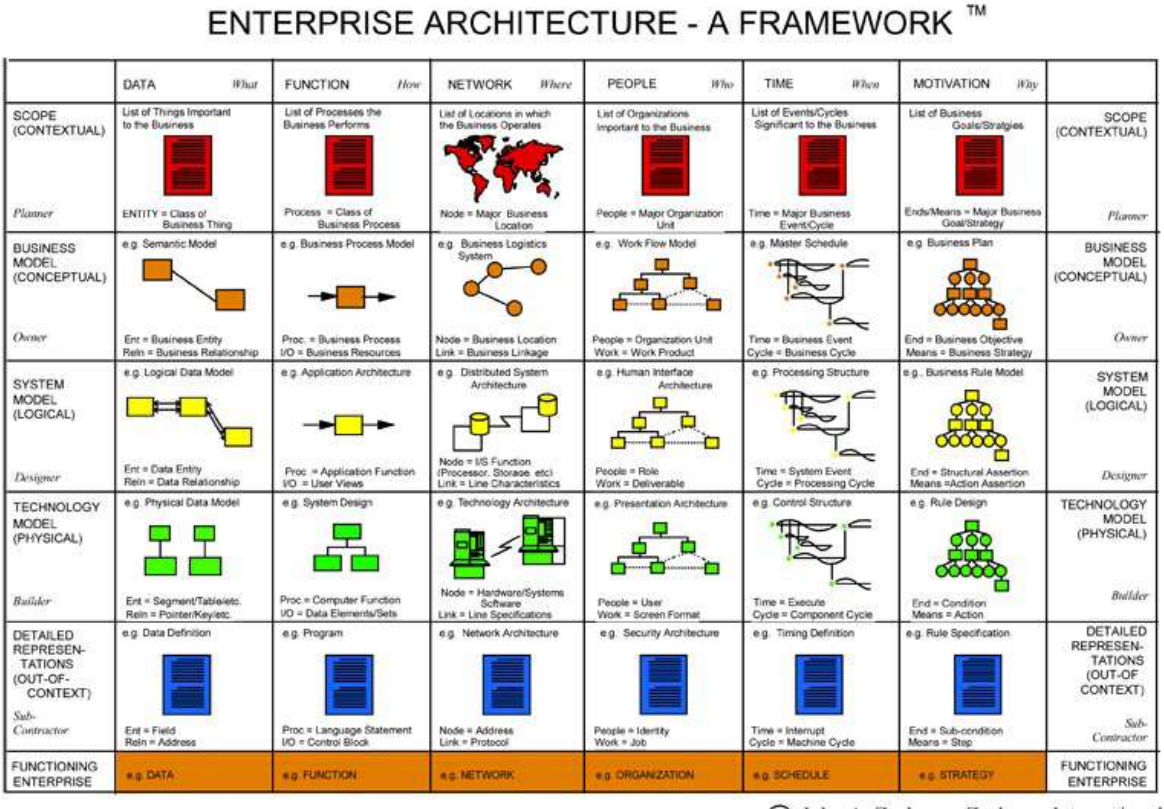


Рис. 8.3. Модель Захмана (2001)

Процес деталізації відбувається в межах однієї комірки, таким чином, кожна комірка подає не просто окрему модель як складову архітектури підприємства, а й увесь процес її розробки. Іншою проблемою матричного подання архітектури була відсутність будь-якого відображення ідеї інтеграції моделей, що має відбуватись за стовпчиками. Принципові зміни було внесено до моделі версії 3.0 (рис. 8.4), опублікованої у 2011 р. [15].

Додатково до чергового уточнення назв стовпців і комірок матриці, неясний термін «фреймворк» (від англ. framework), яким до цього часу позначали модель, було змінено на термін «онтологія» для охоплення повної множини елементів, з яких складається підприємство, і підкреслення того факту, що інші методики у більшості своїй походять від запропонованої Дж. Захманом схеми, яка залишається базисом сучасних архітектур. Додаткові стрілки вздовж рядків і стовпців зазначають необхідність вирівнювання (alignment) і, відповідно, трансформації (transformations) чи складової інтеграції (composite integrations). Процедури трансформації додатково подано ламаними стрілками між комірками одного стовпця.

Дещо змінено інтерпретацію рядків — кожен з них відповідає окремій моделі, яка, в свою чергу, розробляється для певної аудиторії:

— обсяг (Scope Contexts), списки визначення сфер (Scope Identification Lists) — для керівництва, планувальників бізнес-контексту (Executive Perspective, Business Context Planners);

— бізнес-концепції (Business Concepts), моделі визначення бізнесу (Business Definition Models) — для менеджменту, власників бізнес-концепції (Business Management Perspective, Business Concept Owners);

— системна логіка (System Logic), моделі подання систем (System Representation Models) — для архітекторів, проектувальників бізнес-логіки (Architect Perspective, Business Logic Designers);

— технологічна фізика (Technology Physics), моделі специфікації технологій (Technology Specification Models) — для інженерів, розробників фізики бізнесу (Engineer Perspective, Business Physics Builders);

— компоненти засобів (Tool Components), моделі конфігурації засобів (Tool Configuration Models) — для техніків, виконавців бізнес-компонентів (Technician Perspective, Business Component Implementors).


Рис. 8.4. Модель Захмана, версія 3.0 (2011)

Було значно розширено зміст шостого рядка, який відповідає рівню працюючого підприємства. Рядки матриці, за задумом Дж. Захмана, втілюють шість фаз уречевлення абстрактної ідеї: ідентифікація, визначення, подання, специфікація, конфігурація, конкретизація до екземплярів. Як екземпляри тут розглядають операції (Operation Instances, Implementations). Інакше кажучи, власне архітектура підприємства подається рядками з першого по п'ятий, а шостий рядок містить її втілення — конкретні екземпляри об'єктів (Inventory), процесів (Process), місць розташування (Distribution), відповідальності (Responsibility), термінів (Timing), мотивації (Motivation).

Модель Захмана не містить методів її створення або використання, тому вживання терміна «онтологія» цілком доречно. Водночас це є головним обмеженням — усі елементи подають знімок підприємства на певний момент часу, хоча моделі можуть змінюватись ще під час проектування архітектури. Механізм поширення змін від окремої комірки до інших також відсутній. Інше обмеження полягає у відокремленні підприємства від навколишнього середовища — усі передбачені моделі подають підприємство зсередини, включно зі стовпчиком «Де» («Where»), який призначений для відображення каналів дистриб'юції. Цільова аудиторія охоплює користувачів всередині підприємства, починаючи з власників і керівництва до працівників нижньої ланки управління, не передбачаючи взаємодію з зацікавленими особами (стейкхолдерами).

Обмеження онтології Захмана частково подолані в інших методиках, що їх можна назвати похідними [2, 4, 6, 9, 11, 12]. Зокрема, модель «3D-підприємства», запропонована Є. Б. Зіндером, додає до матриці Захмана вимір часу, розташовуючи у ньому інтервали виконання проектів і стадії розвитку підприємства з усіма його компонентами. Модель E2AF (Extended Enterprise Architecture Framework, Фреймворк архітектури розширеного підприємства) містить чотири аспекти (бізнес, інформація, інформаційна система, технологічна інфраструктура) і шість рівнів абстракції (контекстуальний, взаємодії, концептуальний, логічний, фізичний і трансформаційний). Модель FDA (Four Domains architecture, чотиридоменна архітектура) поділяє комірки матриці Захмана на дві частини — архітектуру проекту (Architecture-in-Design), що розробляється під час проектування, та архітектуру виконання (Architecture-in-Operation), що відображає реальні бізнес-процеси та інформаційні системи. Модель SAM (Strategic Architecture Model, Стратегічна модель архітектури) доповнює

модель Захмана ітеративним підходом з поєднанням напрямків «зверху вниз» і «знизу вверх» і «сферами інтересів» (цілі і завдання, організація, бізнес-процеси, прикладні системи, технології, проекти, бізнес-компоненти, дані, бізнес-функції, інфраструктура), що пов'язують факти стосовно підприємства і відношення між ними у групи для поліпшення організації та аналізу всіє зібраної інформації.

Методологія Gartner спрямована на визначення процесу розробки архітектури підприємства як послідовності кроків і завдань учасників процесу, містить практичні рекомендації, але не деталізує їх до моделей процесу. У власне моделі архітектури розрізняють чотири взаємопов'язані рівні: середовище бізнес-взаємодії (Business Relationship Grid), бізнес-процеси і стилі бізнес-процесів, шаблони, технологічні будівельні блоки.

Концептуальний характер має і GERAM (Generalised Enterprise Reference Architecture and Methodology, Довідкова архітектура та методологія для узагальненого підприємства) [5, 7], розроблена як доповнення стандарту ISO 15704 з метою визначення комплексу концепцій, методів і моделей проектування і супроводження підприємства протягом усього часу його існування. Вона складається з таких компонентів:

— загальна корпоративна довідкова архітектура (Generic Enterprise Reference Architecture, GERA) визначає загальні концепції, такі як життєвий цикл корпоративних систем, моделювання бізнес-процесів, мови моделювання для різних користувачів архітектури, інтегроване подання моделі з різних точок зору (views);

— загальні методології інжинірингу підприємства (Generic Enterprise Engineering Methodologies, GEEM) — моделі процесів з інструкціями для кожного етапу;

— загальні мови моделювання підприємства (Generic Enterprise Modeling Languages, GEML) — загальні конструкції (будівельні блоки) для моделювання підприємства;

— загальні інструменти моделювання підприємства (Generic Enterprise Modeling Tools, GEMT) — загальна реалізація методологій інтеграції підприємств, мов моделювання та іншої підтримки створення та використання корпоративних моделей;

— моделі підприємства (Enterprise Models, EM) — операції підприємства, подані загальними конструктами мови моделювання;

— онтологічні теорії (Ontological Theories, OT) — найзагальніші аспекти концепцій, пов'язані з підприємством, у термінах суттєвих характеристик та аксіом;

— загальні моделі підприємства (Generic Enterprise Models, GEMs) — еталонні моделі, що відображають спільні характеристики багатьох підприємств;

— загальні модулі (Generic Modules, GMs) — придатні до застосування продукти.

Поряд з онтологією Захмана провідну роль серед методологій побудови архітектури підприємства сьогодні відіграє TOGAF (The Open Group Architecture Framework), розроблена консорціумом The Open Group з кардинальними відмінностями від інших методологій [13].

TOGAF визначає і підтримує чотири підмножини загальної архітектури підприємства:

- архітектура бізнесу встановлює бізнес-стратегію, управління, організацію та ключові бізнес-процеси;
- архітектура даних описує структуру логічних і фізичних даних організації та ресурси управління даними;
- архітектура додатків забезпечує розробку окремих програмних додатків, що використовуються, їх взаємодію та взаємозв'язок з основними бізнес-процесами організації;
- архітектура технології описує логічні програмні та апаратні можливості, необхідні для підтримки роботи бізнес-служб, даних і додатків включно з IT-інфраструктурою, проміжним програмним забезпеченням, мережами, засобами комунікацій, стандартами тощо.

Водночас чи не найважливішою складовою TOGAF є ADM (Architecture Development Method, метод розробки архітектури), що регламентує архітектурний процес, доповнюючи таким чином таксономію Захмана.

Перевірений і повторюваний ітеративний цикл визначення та реалізації архітектури за ADM включає такі фази (рис. 8.5):

— попередня (Preliminary) — ініціювальні та підготовчі заходи, необхідні для створення архітектурної спроможності (Architecture Capability), зокрема, налаштування TOGAF та визначення архітектурних принципів;

— A (Architecture Vision) — бачення архітектури (визначення масштабу ініціативи з розробки архітектури та стейкхолдерів, створення бачення архітектури та отримання дозволу на продовження розробки);

— B (Business Architecture) — архітектура бізнесу;

— C (Information Systems Architectures, архітектура інформаційних систем).

— D (Technology Architecture, архітектура технології) — розроблення відповідних архітектур для підтримки узгодженого бачення архітектури;

— E (Opportunities & Solutions, можливості та рішення) — початкове планування реалізації та визначення засобів доставки архітектури, визначеної на попередніх фазах;

— F (Migration Planning, планування міграції) — визначення шляхів переходів від базової до цільової архітектури;

— G (Implementation Governance, керування реалізацією) — архітектурний нагляд за реалізацією;

— H (Architecture Change Management, керування змінами архітектури) — процедури керування змінами нової архітектури;

— керування вимогами (Requirements Management) — процес управління вимогами до архітектури протягом ADM.


Рис. 8.5. Цикл розробки архітектури за ADM

ADM надає архітекторам послідовності етапів і процесів, вимоги до архітектури, плани проектів, оцінки відповідності проекту і та ін., підкріплені найкращими практиками та багаторазовим набором наявних архітектурних ресурсів. Входи і виходи для ADM у контексті цілісної архітектури підприємства деталізує

фреймворк змісту архітектури (Architecture Content Framework), який визначає робочі продукти трьох категорій: доставка (Deliverable) — результати проекту, офіційно задокументовані, узгоджені і підписані зацікавленими сторонами, або посилкові моделі, стандарти чи стани архітектури на певний момент часу, нагромаджені в архітектурному репозиторії (Architecture Repository); артефакт (Artifact) — робочий продукт, який описує аспект архітектури у формі каталогу (списку речей), матриці або діаграми; будівельний блок (Building Block) — потенційно придатний для багаторазового використання компонент бізнесу, IT або архітектурної спроможності, який можна поєднати з іншими будівельними блоками для створення архітектури (Architecture Building Blocks, ABBs), та відповідних рішень (Solution Building Blocks, SBBs).

На рис. 8.6 подано план мета-моделі змісту, яка визначає всі типи будівельних блоків архітектури, їхні можливі описи, а також припустимі взаємозв'язки між ними.

Принциповою відмінністю і перевагою TOGAF є концепція континуума підприємства (Enterprise Continuum), за допомогою якої встановлюється контекст архітектури підприємства і пояснюється, як універсальні рішення можуть бути використані та пристосовані для підтримки конкретних потреб. Континуум підприємства можна розглядати як «фреймворк у фреймворку», віртуальне сховище всіх ресурсів — описів, моделей, конструктивних блоків, шаблонів, точок зору та інших артефактів, доступних для розробки архітектури підприємства.

Внутрішніми ресурсами, зокрема, є попередні результати роботи над архітектурою, придатні для повторного використання. Зовнішні відносно підприємства ресурси є як загальносистемними (наприклад, TRM, розроблена у рамках TOGAF), так і спеціалізованими для певних аспектів IT (наприклад, архітектура мережі, архітектура інформаційної безпеки), видів додатків (електронна комерція, управління ланцюгами поставок тощо), окремих галузей (TMF для телекомунікацій, ARTS для роздрібної торгівлі, POSC для піротехніки тощо).

Для впорядкування переходу «фундаментальна архітектура — загальносистемна архітектура — галузева архітектура — архітектура підприємства» використовується континуум архітектури (The Architecture Continuum). Просування між рівнями, з одного боку, відображає задоволення потреб підприємства та його бізнес-вимог, а з другого — добір архітектурних компонентів і будівельних блоків. Водночас відбувається прогрес у таких напрямах,

як «логічний — фізичний», «горизонтальний — вертикальний», «загальний — спеціалізований», «таксономія — специфічна архітектура».

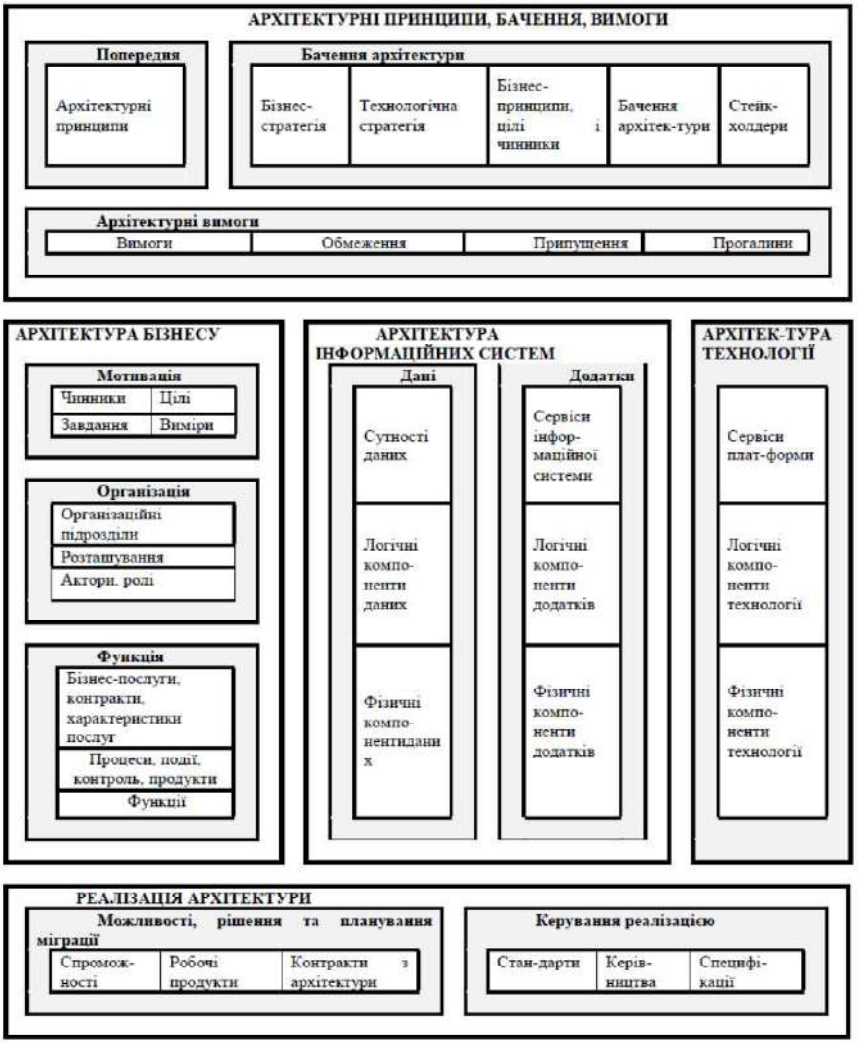


Рис. 8.6. Мета-модель змісту архітектури

TOGAF пропонує дві довідкові моделі у складі континуума підприємства:

— фундаментальна архітектура The TOGAF Foundation Architecture. Її першою складовою є технічна еталонна модель (Technical Reference Model, TRM), яка містить таксономію, що визначає термінологію та опис компонентів і концептуальної структури інформаційної системи, і граф візуального подання таксономії. TRM поділяє всі компоненти на три частини — прикладне програмне забезпечення, платформа додатків та інфраструктура комунікацій, а також описує інтерфейси між ними. Друга складова — інформаційна база стандартів (Standards Information Base, SIB), що їх можна використати для визначення архітектури підприємства на основі фундаментальної архітектури;

— загальносистемна еталонна модель інтегрованої інформаційної інфраструктури (The Integrated Information Infrastructure Reference Model, III-RM), зорієнтована на підтримку необмежених інформаційних потоків (Boundaryless Information Flow). III-RM є підмножиною фундаментальної архітектурі TOGAF і, відповідно, має аналогічну структуру. Водночас у III-RM розширений опис у частині бізнес- та інфраструктурних додатків щодо підтримки інтеграції.

Континуум архітектури прямо підтримується континуумом рішень (The Solutions Continuum). Якщо перший пропонує загальні правила, представлення та взаємозв'язки, включаючи зв'язки трасування та виведення, та структурування будівельних блоків архітектури (ABBs), то другий подає шляхи реалізації континууму архітектури, визначаючи доступність рішень (SBBs). Добір і керування будівельними блоками рішень здійснюються з використанням активів континуума архітектури.

Будівельні блоки як архітектури, так і рішень мають розглядатись у контексті активів підприємства, таких як політики, стандарти, стратегічні ініціативи, організаційні структури, можливості рівня всього підприємства тощо. Лише такий контекст надає можливість класифікувати рішення і скласти потрібну підприємству архітектуру. Для ефективного оперування всіма архітектурними активами TOGAF рекомендує використовувати репозиторій масштабу підприємства з такими вмістом (рис. 8.7): мета-модель архітектури (Architecture Metamodel), архітектурна спроможність (Architecture Capability), архітектурний ландшафт (Architecture Landscape), інформаційна база даних стандартів (Standards Information Base), довідкова бібліотека (Reference Library), журнал управління (Governance Log).

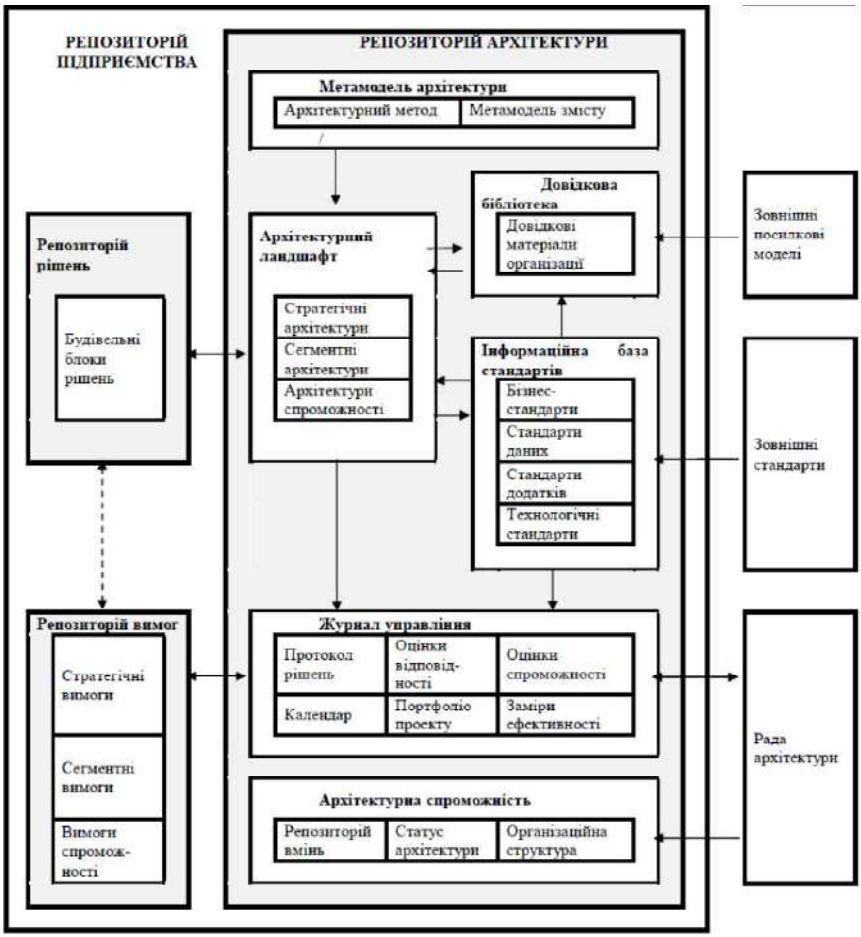


Рис. 8.7. Структурна схема архітектурного репозиторію

Зовнішні еталонні моделі і стандарти приймаються підприємством і нагромаджуються у відповідних довідкових базах. Зі стандартами узгоджуються як еталонні моделі, так і практики управління. Адаптована еталонна модель стає основою формування архітектурного ландшафту, артефакти якого структуруються згідно з обраним фреймворком. Після послідовного уточнення моделі архітектури з урахуванням попередньо сформованих вимог обираються будівельні блоки рішень. Найкращі практики також нагромаджуються у базі даних для використання як стандарти та посилкові моделі у майбутньому. Прийняті

управлінські рішення про архітектурний ландшафт і відповідності стандартам скеровані на підсилення архітектурної спроможності. Рішення також фіксуються у репозиторії та виступають чинниками для формування архітектурних вимог.

Таким чином, TOGAF детально регламентує процес створення архітектури підприємства. Методологія є дуже гнучкою і носить рекомендаційний характер, її застосування та кінцевий результат великою мірою залежить не лише від ситуації та вимог підприємства, а й від особи архітектура. При цьому оптимальність, як і підхожість та якість одержаної архітектури, не гарантується. Серед інших недоліків TOGAF зазначимо складність практичного застосування методології у повному обсязі й відсутність відповідних прикладів, брак конкретних рекомендацій щодо адаптації методології до потреб підприємства.

### *Висновки*

Можна зробити висновок, що існуючі фреймворки, стандарти і методології містять повний набір моделей як процесів розробки архітектури, так і її складових. Рекомендації TOGAF доцільно взяти за основу проектних рішень щодо архітектурного репозиторію. З метою підтримки процесів аналізу контексту, вибору моделі, її сполучення з іншими моделями або фреймворками, адаптації одержаних моделей, розробки власних моделей та їх інтеграції в існуючі процеси та організаційні структури згідно існуючих рекомендацій мають бути розроблені моделі прийняття архітектурних рішень і системи їх підтримки.

### *Список використаних джерел до п. 8.2*

1. Дракер П. Классические работы по менеджменту = Classic Drucker. Москва: Альпина Бизнес Букс, 2008. 220 с.

2. Зиндер Е.З. Архитектура предприятия в контексте бизнес-реинжиниринга — Intelligent Enterprise. *Корпоративные системы*. 2008. №4 (180).

3. Карпенко С.В. Применение модели Захмана для проектирования ИТ-архитектуры предприятия. *Центр Бизнес-Знаний*. 2010.

4. Кондратьев В.В. Управление архитектурой предприятия. Москва: Инфра-М, 2015. 358 с.

5. Bernus P. and oth. Enterprise Architecture: Twenty Years of the GERAM Framework. Proceedings of the 19th World Congress The

International Federation of Automatic Control Cape Town, South Africa. August 24-29, 2014. URL: https://ac.els-cdn.com/S1474667016421154/1-s2.0-S1474667016421154-main.pdf?_tid=1a8f3932-1159-11e8-af6a-00000aacb35f&acdnat=1518593683_bd684a660a4aba76ff1a85d3c1628919

6. Federal Enterprise Architecture Framework. Dev. by: The Chief Information Officers Council (USA). URL: http://www.cio.gov/documents/bpeaquide.pdf

7. GERAM: Generalised Enterprise Reference Architecture and Methodology. URL: http://www2.mitre.org/public/eabok/pdf/geram.pdf

8. ISO 15704:2000 «Industrial Automation Systems Requirements for Enterprise-Reference Architectures and Methodologies. 1999». URL:http://www.iso.org/iso/cataloque detail.htm?csnumber=28777

9. Minoli D. Enterprise Architecture A to Z: Frameworks, Business Process Modeling, SOA, and Infrastructure Technology. Auerbach Publications, 2008. 512 p.

10. Rouse W.B., ed., Enterprise Transformation: Understanding and Enabling Fundamental Change. Hoboken, NJ: Wiley, 2006.

11. Sowa J. F., Zachman J. A. Extending and Formalizing the Framework for Information System Architecture. *IBM System Journal*. Vol. 31. № 3. 1992.

12. Spewak S.H. Enterprise Architecture Planning. N.Y. *John Wiley&Sons Inc*. 2003.

13. The Open Group Architecture Framework TOGAF® Version 9.1. an Open Group Standard. URL: http://pubs.opengroup.org/ architecture/togaf9-doc/arch/index.html

14. The Zachman Framework™: The Official Concise Definition, John A. Zachman. URL: http://www.zachmaninternational.eom/ index.php/home-article/13#maincol

15. The Zachman Framework Evolution: overview of the evolution of the Zachman Framework by John P. Zachman, April 2009. URL: https://www.zachman.com/ea-articles-reference/54-the-zachman-framework-evolution