

---

# Зміст

---

Передмова .....	13
Передмова автора .....	17
Подяки.....	21
Про автора .....	23

## Частина I. ВСТУП

<b>Розділ 1. Що таке дизайн та архітектура? .....</b>	<b>27</b>
Мета .....	28
Приклад із практики .....	29
Причини неприємностей .....	31
Точка зору керівництва .....	32
Що не так? .....	32
Висновки .....	35
<b>Розділ 2. Історія про дві цінності .....</b>	<b>37</b>
Поведінка .....	38
Архітектура .....	38
Найбільша цінність .....	39
Матриця Ейзенгауера .....	40
Битва за архітектуру .....	41

## Частина II. ПОЧАТКОВІ ОСНОВИ: ПАРАДИГМИ ПРОГРАМУВАННЯ

<b>Розділ 3. Огляд парадигм.....</b>	<b>45</b>
Структурне програмування .....	46
Об'єктно-орієнтоване програмування.....	46
Функціональне програмування.....	46
Пожива для розуму.....	47
Висновки .....	47
<b>Розділ 4. Структурне програмування.....</b>	<b>49</b>
Доказ .....	50
Оголошення шкідливим .....	52
Функціональна декомпозиція .....	52
Формальні докази відсутні .....	53

	Наука приходить на порятунок . . . . .	53
	Тестування . . . . .	54
	Висновки . . . . .	54
<b>Розділ 5.</b>	<b>Об'єктно-орієнтоване програмування . . . . .</b>	<b>57</b>
	Інкапсуляція . . . . .	58
	Спадкування . . . . .	61
	Поліморфізм . . . . .	63
	Сильні сторони поліморфізму . . . . .	65
	Інверсія залежності . . . . .	66
	Висновки . . . . .	68
<b>Розділ 6.</b>	<b>Функціональне програмування . . . . .</b>	<b>71</b>
	Квадрати цілих чисел . . . . .	72
	Незмінюваність і архітектура . . . . .	73
	Обмеження змінності . . . . .	74
	Реєстрація подій . . . . .	76
	Висновки . . . . .	77
<b>Частина III. ПРИНЦИПИ ДИЗАЙНУ</b>		
<b>Розділ 7.</b>	<b>Принцип єдиної відповідальності . . . . .</b>	<b>83</b>
	Ознака 1: ненавмисне дублювання . . . . .	85
	Ознака 2: злиття . . . . .	86
	Рішення . . . . .	87
	Висновки . . . . .	89
<b>Розділ 8.</b>	<b>Принцип відкритості/закритості . . . . .</b>	<b>91</b>
	Уявний експеримент . . . . .	92
	Керування напрямком . . . . .	96
	Приховування інформації . . . . .	96
	Висновки . . . . .	96
<b>Розділ 9.</b>	<b>Принцип підстановки Барбери Лісков . . . . .</b>	<b>97</b>
	Інструкція з використання спадкування . . . . .	98
	Проблема квадрат/прямокутник . . . . .	98
	LSP і архітектура . . . . .	99
	Приклад порушення LSP . . . . .	100
	Висновки . . . . .	101
<b>Розділ 10.</b>	<b>Принцип розподілу інтерфейсів . . . . .</b>	<b>103</b>
	Принцип розподілу інтерфейсів і мова . . . . .	105
	Принцип розподілу інтерфейсів та архітектура . . . . .	105
	Висновки . . . . .	106
<b>Розділ 11.</b>	<b>Принцип інверсії залежності . . . . .</b>	<b>107</b>
	Стабільні абстракції . . . . .	108
	Фабрики . . . . .	109

Конкретні компоненти .....	110
Висновки .....	111

#### Частина IV. ПРИНЦИПИ ОРГАНІЗАЦІЇ КОМПОНЕНТІВ

<b>Розділ 12. Компоненти .....</b>	<b>115</b>
Коротка історія компонентів .....	116
Переміщення .....	119
Компонувальники .....	119
Висновки .....	121
<b>Розділ 13. Зв'язність компонентів .....</b>	<b>123</b>
Принцип еквівалентності повторного використання і випусків .....	124
Принцип узгодженої зміни .....	125
Схожість із принципом єдиної відповідальності .....	126
Принцип спільного повторного використання .....	126
Зв'язок із принципом поділу інтерфейсів .....	127
Діаграма протиріч для визначення зв'язності компонентів ...	128
Висновки .....	129
<b>Розділ 14. Сполучуваність компонентів .....</b>	<b>131</b>
Принцип ациклічності залежностей .....	132
Щотижневі збірки .....	132
Усунення циклічних залежностей .....	133
Вплив циклів у графі залежностей компонентів .....	135
Розрив циклу .....	136
«Мінливість» .....	137
Проектування згори вниз .....	138
Принцип стабільних залежностей .....	139
Стабільність .....	139
Метрики стабільності .....	141
Не всі компоненти мають бути стабільними .....	143
Абстрактні компоненти .....	145
Принцип стабільності абстракцій .....	145
Куди помістити високорівневі правила? .....	145
Введення в принцип стабільності абстракцій .....	145
Міра абстрактності .....	146
Головна послідовність .....	146
Зона болю .....	148
Зона марності .....	148
Як не потрапити в зони виключення .....	149
Відстань до головної послідовності .....	149
Висновки .....	151

## Частина V. АРХІТЕКТУРА

<b>Розділ 15.</b>	<b>Що таке архітектура</b> . . . . .	155
	Розроблення . . . . .	157
	Розгортання . . . . .	157
	Робота системи . . . . .	158
	Супровід . . . . .	159
	Збереження різноманітності варіантів . . . . .	159
	Незалежність від пристрою . . . . .	161
	Спам . . . . .	162
	Фізична адресація . . . . .	164
	Висновки . . . . .	165
<b>Розділ 16.</b>	<b>Незалежність</b> . . . . .	167
	Варіанти використання . . . . .	168
	Ефективність роботи . . . . .	168
	Розроблення . . . . .	169
	Розгортання . . . . .	169
	Збереження різноманітності варіантів . . . . .	170
	Розділення рівнів . . . . .	170
	Розділення варіантів використання . . . . .	171
	Режим розділення . . . . .	172
	Можливість незалежного розроблення . . . . .	173
	Можливість незалежного розгортання . . . . .	173
	Дублювання . . . . .	173
	Режими розподілу (ще раз) . . . . .	174
	Висновки . . . . .	176
<b>Розділ 17.</b>	<b>Кордони: проведення ліній розподілу</b> . . . . .	177
	Кілька сумних історій . . . . .	178
	FitNesse . . . . .	181
	Які кордони проводити і коли? . . . . .	183
	Про введення і виведення . . . . .	185
	Архітектура з плагінами . . . . .	186
	Аргумент на користь плагінів . . . . .	187
	Висновки . . . . .	189
<b>Розділ 18.</b>	<b>Анатомія кордонів</b> . . . . .	191
	Перетин кордонів . . . . .	192
	Жахливий моноліт . . . . .	192
	Компоненти розгортання . . . . .	194
	Потоки виконання . . . . .	195
	Локальні процеси . . . . .	195
	Служби . . . . .	196
	Висновки . . . . .	196

<b>Розділ 19. Політика і рівень</b> . . . . .	197
Рівень . . . . .	198
Висновки . . . . .	201
<b>Розділ 20. Бізнес-правила</b> . . . . .	203
Сутності . . . . .	204
Варіанти використання . . . . .	205
Моделі запитів і відповідей . . . . .	207
Висновки . . . . .	208
<b>Розділ 21. Кричуща архітектура</b> . . . . .	209
Тема архітектури . . . . .	210
Мета архітектури . . . . .	210
А як щодо веб? . . . . .	211
Фреймворки — це інструменти, а не спосіб життя . . . . .	211
Тестовані архітектури . . . . .	212
Висновки . . . . .	212
<b>Розділ 22. Чиста архітектура</b> . . . . .	213
Правило залежностей . . . . .	215
Сутності . . . . .	216
Варіанти використання . . . . .	216
Адаптери інтерфейсів . . . . .	216
Фреймворки і драйвери . . . . .	217
Лише чотири кола? . . . . .	217
Перетин кордонів . . . . .	217
Які дані перетинають кордони . . . . .	218
Типовий сценарій . . . . .	219
Висновки . . . . .	220
<b>Розділ 23. Презентатори і скромні об'єкти</b> . . . . .	221
Шаблон «Скромний об'єкт» . . . . .	222
Презентатори та представлення . . . . .	222
Тестування і архітектура . . . . .	223
Шлюзи до баз даних . . . . .	223
Перетворювачі даних . . . . .	224
Слухачі служб . . . . .	224
Висновки . . . . .	225
<b>Розділ 24. Неповні кордони</b> . . . . .	227
Пропустити останній крок . . . . .	228
Одномірні кордони . . . . .	229
Фасади . . . . .	229
Висновки . . . . .	230
<b>Розділ 25. Рівні та кордони</b> . . . . .	231
Полювання на Вампуса . . . . .	232

	Чиста архітектура . . . . .	233
	Перетин потоків . . . . .	236
	Розбиття потоків . . . . .	236
	Висновки . . . . .	238
<b>Розділ 26.</b>	<b>Головний компонент . . . . .</b>	<b>239</b>
	Кінцева деталь . . . . .	240
	Висновки . . . . .	244
<b>Розділ 27.</b>	<b>Служби: великі й малі . . . . .</b>	<b>245</b>
	Сервісна архітектура . . . . .	246
	Переваги служб? . . . . .	246
	Упередження щодо незалежності . . . . .	246
	Упередження щодо можливості незалежного розроблення і розгортання . . . . .	247
	Проблема з кошенятами . . . . .	247
	Порятунок в об'єктах . . . . .	249
	Служби на основі компонентів . . . . .	250
	Наскрізні завдання . . . . .	251
	Висновки . . . . .	252
<b>Розділ 28.</b>	<b>Кордони тестів . . . . .</b>	<b>253</b>
	Тести як компоненти системи . . . . .	254
	Проектування для простоти тестування . . . . .	254
	Програмний інтерфейс для тестування . . . . .	255
	Структурна залежність . . . . .	256
	Безпека . . . . .	256
	Висновки . . . . .	256
<b>Розділ 29.</b>	<b>Чиста вбудована архітектура . . . . .</b>	<b>257</b>
	Тест на прог-придатність . . . . .	260
	Вузьке місце цільового обладнання . . . . .	263
	Чиста вбудована архітектура — архітектура, що підтримує тестування . . . . .	263
	Рівні . . . . .	263
	Обладнання — це деталь . . . . .	265
	Не розкривайте деталей про обладнання користувачам HAL . . . . .	266
	Процесор — це деталь . . . . .	266
	Операційна система — це деталь . . . . .	270
	Програмування із застосуванням інтерфейсів і можливість підстановки . . . . .	272
	Принцип DRY і директиви умовної компіляції . . . . .	272
	Висновки . . . . .	273

## Частина VI. ДЕТАЛІ

<b>Розділ 30. База даних — це деталь</b> . . . . .	277
Реляційні бази даних . . . . .	278
Чому системи баз даних настільки поширені? . . . . .	279
Чи збережуться диски? . . . . .	280
Деталі . . . . .	280
А продуктивність? . . . . .	281
Повчальна історія . . . . .	281
Висновки . . . . .	282
<b>Розділ 31. Веб — це деталь</b> . . . . .	283
Нескінченний маятник . . . . .	284
Мораль . . . . .	285
Висновки . . . . .	286
<b>Розділ 32. Фреймворки — це деталь.</b> . . . . .	287
Автори фреймворків . . . . .	288
Нерівноправний шлюб . . . . .	288
Ризики . . . . .	289
Рішення . . . . .	289
Попереджаю вас . . . . .	290
Висновки . . . . .	290
<b>Розділ 33. Практичний приклад: продаж відео</b> . . . . .	291
Продукт . . . . .	292
Аналіз варіантів використання . . . . .	292
Компонентна архітектура . . . . .	294
Керування залежностями . . . . .	295
Висновки . . . . .	296
<b>Розділ 34. Загублений розділ</b> . . . . .	297
Упакування за рівнями . . . . .	298
Упакування за особливостями . . . . .	300
Порти і адаптери . . . . .	301
Упакування за компонентами . . . . .	303
Диявол у деталях реалізації . . . . .	308
Організація та інкапсуляція . . . . .	309
Інші режими розподілення . . . . .	312
Висновки: загублена порада . . . . .	314

## Частина VII. ДОДАТОК

<b>Архітектурна археологія</b> . . . . .	317
Профспілкова система обліку . . . . .	318
Laser Trim . . . . .	325
Контроль алюмінієвого лиття під тиском . . . . .	328

4-TEL .....	329
Комп'ютер зони обслуговування .....	334
Вибір ремонтників для відправки .....	334
Архітектура .....	335
Велика модернізація .....	336
Європа .....	337
Наостанок про SAC .....	337
Мова С .....	338
С .....	339
BOSS .....	339
pCCU .....	340
Пастка планування .....	341
DLU/DRU .....	342
Архітектура .....	343
VRS .....	344
Назва .....	345
Архітектура .....	345
Висновки про VRS .....	346
Електронний секретар .....	346
Кінець електронного секретаря .....	348
Система відрядження ремонтників .....	348
Clear Communications .....	350
Обставини .....	351
Дядечко Боб .....	352
Телефонний дзвінок .....	353
ROSE .....	353
Продовження дискусій .....	354
...Під будь-яким іншим ім'ям .....	354
Реєстраційні іспити для архітекторів .....	355
Висновки .....	358
<b>Предметний покажчик .....</b>	<b>359</b>



---

# Що таке дизайн та архітектура? 1

---



За довгі роки навколо понять «дизайн» і «архітектура» накопичилося багато плутанини. Що таке дизайн? Що таке архітектура? Чим вони відрізняються?

Одна з цілей цієї книжки — усунути весь цей безлад і визначити раз і назавжди, що таке дизайн та архітектура. Перш за все я стверджую, що між цими поняттями немає жодної різниці. *Узагалі жодної.*

Слово «архітектура» часто використовують у контексті загальних міркувань, коли не йдеться про низькорівневі деталі, стосовно яких зазвичай вживають слово «дизайн». Але такий розподіл безглуздий, коли мова про те, що робить справжній архітектор.

Візьмемо для прикладу архітектора, що спроектував мій новий будинок. Цей будинок має архітектуру? Звичайно! А в чому вона виражається? Ну... Це форма будинку, зовнішній вигляд, а також розташування кімнат і організація простору всередині. Але коли я розглядав креслення, створені архітектором, я побачив на них купу деталей. Я побачив розташування всіх розеток, вимикачів і світильників. Я побачив, які вимикачі будуть керувати тими чи іншими світильниками. Я побачив, де буде розміщений вузол опалення, а також місце розташування і розміри водогрійного котла і насоса. Я побачив докладний опис, як потрібно конструювати стіни, дах і фундамент.

Простіше кажучи, я побачив усі дрібні деталі, які підтримують усі високорівневі рішення. Я також побачив, що низькорівневі деталі і високорівневі рішення разом складають дизайн будинку.

Те саме стосується архітектури програмного забезпечення. Низькорівневі деталі і високорівнева структура є частинами одного цілого. Вони утворюють суцільне полотно, яке визначає форму системи. Одне без іншого неможливе; немає жодної чіткої лінії, яка розмежовувала б їх. Є просто сукупність рішень різного рівня деталізації.

## Мета

У чому полягає мета таких рішень, мета гарного дизайну програмного забезпечення? Головна мета — не що інше, як моє утопічне визначення:

*Мета архітектури програмного забезпечення — зменшити трудовитрати на створення і супровід системи.*

Мірою якості дизайну може слугувати проста міра трудовитрат, необхідних для задоволення потреб клієнта. Якщо трудовитрати невеликі

і залишаються невеликими протягом експлуатації системи, система має гарний дизайн. Якщо трудовитрати збільшуються з виходом кожної нової версії, система має поганий дизайн. Ось так все просто.

## Приклад із практики

Для прикладу розглянемо результати досліджень із практики. Вони засновані на реальних даних, наданих реальною компанією, котра побажала не розголошувати своєї назви.

Спочатку розглянемо графік зростання чисельності інженерно-технічного персоналу. Ви, напевно, погодитесь, що тенденція вражає. Зростання чисельності, що показане на *рис. 1.1*, мало б слугувати ознакою успішного розвитку компанії!



**Рис. 1.1.** Зростання чисельності інженерно-технічного персоналу  
*Відтворюється з дозволу автора презентації Джейсона Гормана (Jason Gorman)*

Тепер погляньте на графік продуктивності компанії за той самий період, що вимірюється в кількості рядків коду (див. *рис. 1.2*).



Рис. 1.2. Продуктивність за той самий період

Очевидно, тут щось не так. Навіть із урахуванням того, що випуск кожної версії підтримується зростаючою кількістю розробників, схоже, що кількість рядків коду наближається до своєї межі.

А тепер погляньте на по-справжньому гнітючий графік: на рис. 1.3 показане зростання вартості рядка коду з плином часу.



Рис. 1.3. Зміна вартості рядка коду з плином часу