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 спроєктував 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, погодитеся, що тенденція вражає. Зростання чисельності, що показане на *рис. 1.1*, мало б слугувати ознакою успішного розвитку компанії!

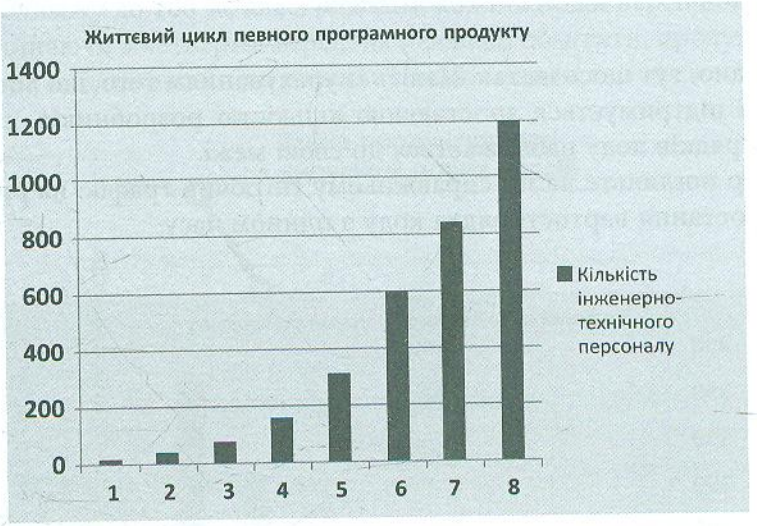

Рис. 1.1. Зростання чисельності інженерно-технічного персоналу

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*).

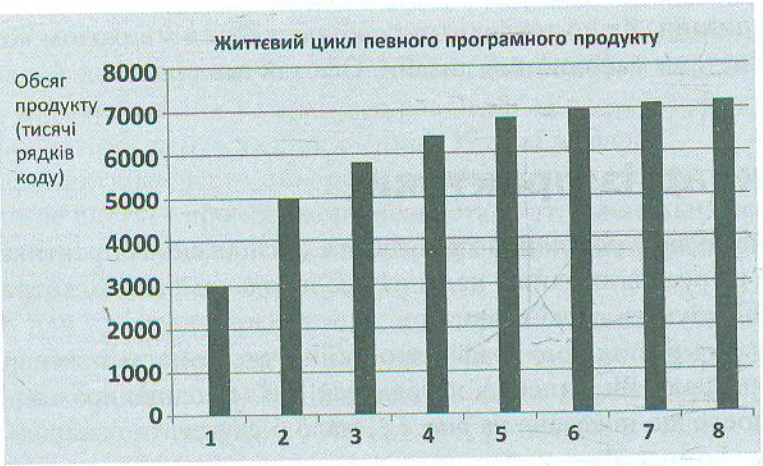

Рис. 1.2. Продуктивність за той самий період

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.

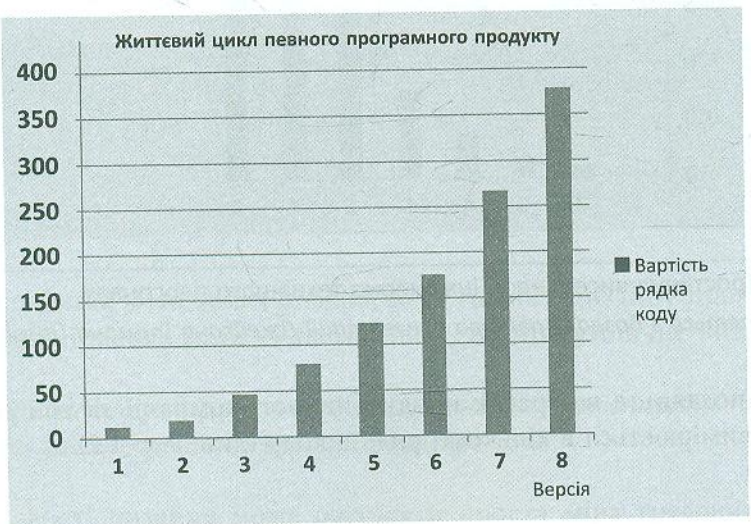

Рис. 1.3. Зміна вартості рядка коду з плином часу

Ця тенденція свідчить про нежиттєздатність. Хоч якою б рентабельною наразі не була компанія, накладні витрати, які постійно зростають, поглинуть прибуток і призведуть до застою, а то й до краху.

Чим зумовлене таке жахливе зниження продуктивності? Чому рядок коду у версії 8 продукту коштує в 40 разів дорожче, ніж у версії 1?

Причини неприємностей

Причини неприємностей ви бачите й самі. Коли системи створюються поспіхом, коли збільшення штату програмістів — єдиний спосіб продовжувати випускати нові версії, коли чистоті коду або дизайну приділяють мінімум уваги або не приділяють уваги взагалі, можна навіть не сумніватися, що така тенденція рано чи пізно призведе до краху.

На *рис. 1.4* показано, який вигляд має ця тенденція у перекладі на мову продуктивності розробників. Спочатку розробники показують продуктивність, близьку до 100 %, але з виходом кожної нової версії вона знижується. Починаючи з четвертої версії, як неважко помітити, продуктивність розробників наближається до нижньої межі — до нуля.

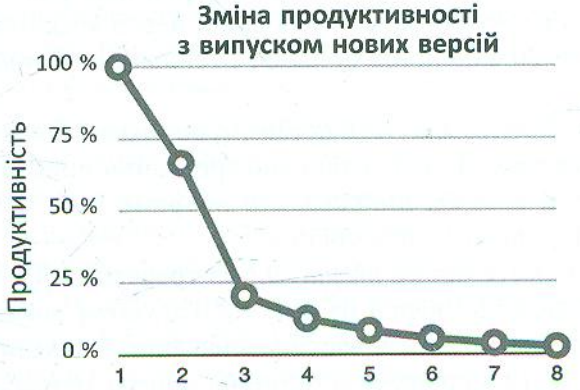

Рис. 1.4. Зміна продуктивності з випуском нових версій

З точки зору розробників така ситуація справляє неймовірно гнітюче враження, тому всі вони продовжують працювати *з повною віддачею*. Ніхто не ухиляється від роботи.

І все ж таки, незважаючи на понаднормову працю і самовідданість, вони просто не можуть зробити більше. Усі їхні зусилля тепер спрямовані не на реалізацію нових функцій, а на боротьбу з безладом. Більшу частину часу вони зайняті тим, що раз у раз переносять безлад із одного місця в інше, щоб отримати можливість додати ще якусь дрібничку.