Відгуки на «Чистий *Agile*»

У подорожі до вершин, яку *Agile* здійснив разом із Дядьком Бобом, траплялися й успіхи, й невдачі. Ця чудова книжка — почасти поєднання історії та особистих мемуарів, сповнене мудрості. Якщо ви хочете зрозуміти, що таке *Agile* і як відбулося його становлення, ця книжка — для вас.

Граді Буч

Розчарування Боба читається в кожному реченні книжки «Чистий *Agile*», проте воно виправдане. Усе, що *існує* у світі *Agile*,— ніщо на тлі того, що *можна* було б із нього створити. Ця книжка — погляд Боба на те, на чому слід зосередити увагу заради досягнення «можливого». Він зробив свій внесок, і до нього варто дослу́хатися.

Кент Бек

Корисно почитати про погляди Дядька Боба на *Agile*. У цій книжці знайдеться щось і для новачка, і для *знавця Agile*. Я згоден майже з усім у ній. Деякі її розділи змушують мене усвідомити власні недоліки, хай йому грець. Це змусило мене здійснити подвійну перевірку коду (85,09 %).

Джон Керн

Ця книжка ніби історична лінза, крізь яку можна повніше й точніше роздивитися розвиток *Agile*. Дядько Боб — один із найрозумніших відомих мені людей, і він безмежно захоплюється програмуванням. Якщо хтось і може демістифікувати еволюцію *Agile*, то це він.

З передмови Джеррі Фіцпатріка

Для ознайомлення з повним списком доступних публікацій відвідайте *informit.com/martinseries*.

Серія Роберта С. Мартіна призначена для розробників програмного забезпечення, керівників команд, бізнес-аналітиків та менеджерів, які хочуть підвищити свої вміння та знання до найвищих рівнів кваліфікації. Вона містить книжки, які орієнтують професіоналів програмного забезпечення на принципи, закономірності та практики програмування, управління проєктами програмного забезпечення, збирання вимог, проєктування, аналіз, тестування тощо.

Кожному програмісту,
котрий хоч колись боровся
з вітряками та водоспадами

Зміст

Передмова

У чому саме полягає методологія розробки *Agile*? Як виникла? Як еволюціонувала?

У цій книжці Дядько Боб дає вдумливі відповіді на ці запитання. Він також визначає багато способів, за допомогою яких суть *Agile* було неправильно або викривлено потрактовано. Його погляд є актуальним, оскільки він — авторитет у цьому питанні, бо доклав руку до появи й розвитку *Agile*.

Із Бобом ми товаришували багато років. Познайомилися, коли я влаштувався до відділу телекомунікацій у компанію *Teradyne* 1979 року. Працюючи електротехніком, я допомагав установлювати та підтримувати продукцію; пізніше я став розробником апаратного обладнання.

Приблизно за рік після мого влаштування на роботу компанія почала шукати нові ідеї для своєї продукції. У 1981 році ми з Бобом запропонували ідею електронного телефонного реєстратора — по суті, систему голосової пошти з функціями переадресації дзвінків. Компанії сподобалася концепція, і невдовзі ми почали розробку «*E. R.* — Електронного реєстратора». Наш прототип було зроблено на найвищому технічному рівні. Ним керувала операційна система *MP/M* на процесорі *Intel 8086*. Голосові повідомлення зберігалися на п'ятимегабайтному жорсткому диску *Seagate ST-506*. Я працював над розробкою обладнання для голосового порту, а Боб писав прикладну програму. Коли я закінчив свою роботу, то також написав частину коду програми. Тож відтоді мене можна називати ще й розробником.

Десь у 1985 чи 1986 році *Teradyne* різко зупинила розробку *E. R.* і, не доводячи до нашого відома, відкликала заявку на патент. Це

було бізнес-рішення, про яке компанія невдовзі пошкодувала, а ми з Бобом про нього шкодуємо й дотепер.

Зрештою кожен із нас пішов із *Teradyne* у пошуках інших можливостей. Боб започаткував консалтинговий бізнес десь у Чикаго. Я став розробником програмного забезпечення й наставником. Нам удалося тримати зв'язок навіть тоді, коли я переїхав до іншого штату.

До 2000 року я викладав об'єктно-орієнтований аналіз та проєктування в *Learning Tree International*. Курс містив *UML* та Уніфікований процес розробки програмного забезпечення (*USDP*). Я добре знався на цих технологіях, але не на *Scrum*, екстремальному програмуванні чи подібних методологіях.

У лютому 2001 року було опубліковано Маніфест *Agile* (*Agile Manifesto*). Спершу я зреагував так, як і багато інших розробників: «Що за *Agile*?» Єдиний Маніфест, відомий мені тоді, належав перу Карла Маркса, ярого комуніста. Чи закликав цей *Agile* стати до зброї? Бісові радикали із когорти розробників програмного забезпечення!

Маніфест справді став своєрідним заколотом. Він мав на меті надихнути на створення лаконічного чистого коду за допомогою спільного адаптивного підходу зі зворотним зв'язком і запропонував альтернативу процесам-«важковаговикам», як-от каскадна модель та *USDP*.

Минуло 18 років із моменту опублікування Маніфесту. Проте для більшості розробників нині це як історія Стародавнього світу. Із цієї причини ваше розуміння *Agile* може відрізнятися від намірів, які було покладено в основу його творцями.

Праця Боба має на меті поставити все на свої місця. Ця книжка ніби історична лінза, крізь яку можна повніше й точніше роздивитися розвиток *Agile*. Дядько Боб — один із найрозумніших відомих мені людей, і він безмежно захоплюється програмуванням. Якщо хтось і може демістифікувати еволюцію *Agile*, то це він.

Джеррі Фіцпатрік,
Software Renovation Corporation,
березень 2019 року

Вступ

Ця книжка не є науковим дослідженням. Я не робив ретельного огляду літератури. Те, що ви збираєтеся прочитати,— лишень мої особисті спогади, спостереження та думки, навіяні моєю 20-річною причетністю до *Agile*.

Книжку написано в розмовній манері. Іноді я добирав грубувату форму висловлювання. І хоча я не любитель лихословити, усе ж одне [трішки змінене] лайливе слово потрапило на ці сторінки, бо я не зміг придумати інакшого способу передати потрібний смисл.

Насправді ця книжка не пересичена гнівом. Коли я вважав за необхідне, то наводив кілька посилань заради пояснень написаного. Я звірявся з поглядами тих, хто був у спільноті *Agile* так само довго, як і я. Я навіть попросив декількох людей додати власні думки та висловити власну незгоду, що знайшло своє місце в окремих розділах. Проте не слід сприймати цю книжку як наукову працю. Можливо, краще сприймати її як мемуари — бубніння старигана, який розганяє всіх цих новоспечених «дітей» *Agile* зі свого «газону».

Ця книжка для програмістів і просто зацікавлених темою осіб. Це не технічне видання. Тут немає коду. Вона призначена для того, щоб зробити огляд первинної мети розробки *Agile* без заглиблення в технічні деталі програмування, тестування й управління.

Ця книжка невеличка. Це тому, що тема не дуже розлога. *Agile* — це невелика ідея про невеликі проблеми, які вирішують маленькі команди програмістів, котрі виконують негроміздкі завдання. Отже, *Agile не* для великих команд програмістів, які отримують масштабні проєкти. Дещо іронічно, що це негроміздке рішення для дрібних проблем узагалі має назву. Зрештою, маленьку проблему, про яку йдеться, було вирішено у 1950–1960-ті, майже тоді, коли було винайдено програмне забезпечення. Ще тоді невеликі команди розробників програмного забезпечення навчилися добре виконувати дрібні завдання. Однак цей «потяг» зійшов з рейок у 1970-х, коли невеликі команди розробників негроміздких проєктів заплуталися в ідеях, які пропагували виконання великих завдань великими командами.

А хіба ми не повинні робити саме це? О боже, ні! Великі завдання не виконують великими командами; їх виконують завдяки співпраці багатьох невеликих команд, які роблять багато дрібних речей. Це інстинктивно розуміли програмісти 1950-х та 1960-х, і саме про це забули в 1970-х.

Чому забули? Підозрюю, що це сталося через різкий стрибок. Кількість програмістів почала значно зростати в 1970-х. До цього у світі було лише кілька тисяч програмістів. А після — уже сотні тисяч. Зараз це число наближається до ста мільйонів.

Ті перші програмісти в 1950-х і 1960-х не були молодими. Вони почали програмувати в 1930-х, 1940-х та 1950-х. До 1970-х, коли відбувся «стрибок», ветерани програмування почали виходити на пенсію. Тож нове покоління так і не отримало необхідного навчання. Неймовірно молода група 20-річних уливалася в лави робітників відповідно до того, як із них вибували досвідчені люди, чиї знання не було ефективно передано.

Дехто скаже, що ця подія стала своєрідним початком для темних часів у програмуванні. Протягом 30 років ми боролися з ідеєю, що нам слід робити великі справи з великими командами, ніяк не усвідомлюючи, що секрет полягає у виконанні багатьох дрібних речей багатьма маленькими командами.

Потім, у середині 1990-х, ми стали усвідомлювати, що щось було втрачено. Почала зароджуватися й розвиватися ідея роботи невеликими командами. Вона поширювалася спільнотою розробників програмного забезпечення, набираючи оберти. У 2000 році ми зрозуміли, що вся галузь потребує перезавантаження. Нам потрібно було пригадати все те, про що наші попередники інстинктивно знали. Необхідно було ще раз усвідомити, що великі справи робляться спільно великою кількістю невеликих команд, які вирішують невеликі завдання.

Для популяризації ідеї ми дали їй назву «*Agile*».

Я написав цю передмову на початку 2019 року. Минуло майже два десятиліття з моменту перезавантаження 2000 року, і мені здається, що надійшов час іще одного. Чому? Тому що просте й невелике послання *Agile* протягом останніх років стало заплутаним. Його змішали з поняттями *Lean*, *Kanban*, *LeSS*, *SAFe*, *Modern*, *Skilled* та багатьма іншими. Згадані ідеї непогані, але вони не відображають оригінальну ідею *Agile*.

Отже, час нам знову пригадати те, що знали наші предки в 1950–1960-ті і що ми повторно з'ясували в 2000 році. Час згадати, чим насправді є *Agile*.

У цій книжці ви не знайдете нічого надто нового, захопливого чи неймовірного, революційного чи такого, що ламає стереотипи. А дізнаєтеся ви про *Agile* так, як це було сказано 2000 року. Хоча загалом тут викладатимуться думки під іншим кутом. Я включатиму сюди те, про що ми дізналися за останні 20 років. Але загалом меседж цієї книжки такий же, як і у 2001 році та 1950-му.

Це давній меседж. Але це правдивий меседж. Він пропонує нам негроміздке рішення маленького завдання, поставленого перед невеликими командами програмістів, які виконують дрібні завдання.

Слова подяки

Мої перші слова подяки адресовано двом відважним програмістам, які радо відкрили (або перевідкрили) методи, викладені в цій книжці,— Ворду Каннінгему та Кенту Беку.

Наступний на черзі — Мартін Фаулер, без чиєї твердої руки в ті старі часи революція *Agile*, імовірно, від початку була би приречена на провал.

Кен Швабер заслуговує на особливу згадку за невгамовну енергію в просуванні та введенні в обіг *Agile*.

Мері Поппендік також заслуговує на особливу згадку за безкорисливу та невичерпну енергію, яку вона вклала в рух *Agile*, та за її піклування про *Agile Alliance*.

На мій погляд, Рон Джеффріз завдяки своїм виступам, статтям, блогам та теплоті свого характеру виступав як совість раннього руху *Agile*.

Майк Бідл самовіддано боровся за справу *Agile*, але його спіткала безглузда смерть від рук безхатька на вулицях Чикаґо.

Інші автори-творці Маніфесту *Agile* також посідають у цій книжці особливе місце. З-поміж них: Арі ван Беннекум, Алістер Кокберн, Джеймс Ґренінґ, Джим Гайсміт, Енді Гант, Джон Керн, Браян Марік, Стів Меллор, Джефф Сазерленд та Дейв Томас.

Джим Ньюкірк, мій друг і діловий партнер, невтомно працював на підтримку *Agile*, долаючи особисті труднощі, які більшість із нас (і я, без сумніву) не можуть собі навіть уявити.

Далі я хотів би згадати людей, які працювали в *Object Mentor Inc*. Усі вони прийняли на себе початковий ризик упровадження й просування *Agile*. Багатьох із них можна побачити на фотографії, зробленій під час першого курсу *XP Immersion*.

Задній ряд: Рон Джеффріз, автор, Браян Баттон, Лоуел Ліндстром, Кент Бек, Міка Мартін, Анжеліка Мартін, С'юзан Россо, Джеймс Ґренінг.
Передній ряд: Девід Фарбер, Ерік Мід, Майк Гілл, Кріс Бігей, Алан Френсіс, Дженніфер Конке, Таліша Джефферсон, Паскаль Рой.
Відсутні на фото: Тім Оттінґер, Джефф Ланґр, Боб Косс, Джим Ньюкірк, Майкл Фезерс, Дін Вемплер і Девід Хелімскі.

Я також хотів би висловити вдячність людям, які згуртувалися й створили *Agile Alliance*. Дехто з них є на фото, зробленому на цьогорічному серпневому засіданні альянсу.

Зліва направо: Мері Поппендік, Кен Швабер, автор, Майк Бідл, Джим Гайсміт. (Немає на фото: Рона Крокера.)

І насамкінець, дякую всім співробітникам *Pearson*, особливо моєму видавцю Джулі Файфер.

1 Вступ до Agile

У лютому 2001 року група із 17 експертів у галузі програмування зібралася в Сноуберді, містечку в штаті Юта, щоб поговорити про невтішний стан своєї сфери. Тоді більшість програм створювали за допомогою неефективних, важких процесів із великою кількістю ритуалів, як-от каскадна модель та перевантажені екземпляри *Rational Unified Process* (*RUP*; уніфікованого процесу розробки). Метою цих експертів було створення маніфесту, який запровадив би ефективніший і легший підхід.

То було нелегке завдання. Усі 17 експертів були людьми з різним досвідом та мали розбіжності в поглядах. Очікування, що така група прийде до консенсусу, мали невеликі шанси справдитися. Проте попри все це, консенсусу досягнули, було написано *Agile Manifesto*, і народився один із найпотужніших рухів, чиє функціонування є найдовшим у галузі програмного забезпечення.

Усі рухи в галузі програмного забезпечення йдуть передбачуваним шляхом. Спочатку захоплені прихильники в меншості, ще одна меншина — захоплені недоброзичливці, а переважна більшість належить до тих, кому байдуже. Багато рухів помирають або, принаймні, ніколи не переходять до наступної фази. Пригадайте аспект-орієнтоване програмування, логічне програмування чи *CRC*-карти. Однак деякі перетинають прірву і стають надзвичайно популярними та дискусійними. Деяким навіть удається залишити протиріччя позаду і просто стати частиною сучасної думки. Прикладом цього можна вважати об'єктно-орієнтоване (ОО) мислення. Це стосується й *Agile*.

На жаль, щойно рух отримує популярність, він стає розмитим через нерозуміння та привласнення. Продукція й методики, котрі не

мають нічого спільного з оригінальним рухом, запозичують його ім'я, щоб нажитися на чужій популярності та значущості. Так відбулося і з *Agile*.

Метою цієї книжки, написаної майже за два десятиліття після подій у Сноуберді, є встановлення прямого посилу. Вона є спробою бути максимально прагматичною, описуючи *Agile* без дурниць і натяків.

У книжці представлено основи *Agile*. Багато хто прикрасив і розширив ці ідеї, і в цьому немає нічого поганого. Однак такі розширення та прикраси не тотожні підходу під назвою *Agile*. То *Agile* з іншими складниками. А в цій книжці ви прочитаєте саме про те, що ж таке *Agile* зараз, чим *Agile* був і чим неминуче буде завжди.

Історія *Agile*

Коли розпочався *Agile*? Імовірно, понад 50 000 років тому, коли люди вперше вирішили співпрацювати заради досягнення спільної мети. Ідея постановки невеликих проміжних цілей і вимірювання просування після досягнення кожної з них є достатньо інтуїтивно зрозумілою й настільки притаманною людині, що варто вважати її чимось революційним.

Коли ж *Agile* з'явився в сучасній промисловості? Важко сказати. Я уявляю, що перша парова машина, перший млин, двигун внутрішнього згоряння та літак було створено завдяки методам, які ми б зараз назвали *Agile*. Здійснення невеликих вимірюваних кроків є надто природною та притаманною людині якістю, тому навряд чи це могло реалізуватися в інакший спосіб.

А коли ж почалися часи *Agile* у розробці програмного забезпечення? Хотів би я бути мухою на стіні, коли 1936 року Алан Тюрінг писав свою статтю[1]. Припускаю, що багато «програм» було розроб-

[1] Turing A. M. 1936. On computable numbers, with an application to the Entscheidungsproblem [proof]. *Proceedings of the London Mathematical Society, 2* (видана в 1937), 42(1):230–65.— Кращий спосіб ознайомитися з цією працею — прочитати видання Чарлза Петцольда: Petzold C. *The Annotated Turing: A Guided Tour through Alan Turing's Historic Paper on Computability and the Turing Machine*. Indianapolis, IN: Wiley, 2008.

лено ним із застосуванням невеликих кроків та великої кількості перевірок. Я також уявляю, що перший код, який він написав для автоматичної обчислювальної машини (*Automatic Computing Engine*) у 1946 році, було створено поетапно, маленькими кроками, з великою кількістю перевірок і навіть із застосуванням реального тестування.

Перші дні існування програмного забезпечення сповнено прикладами вирішення завдань, які ми зараз би охарактеризували як *Agile*. Наприклад, програмісти, які писали програмне забезпечення для керування космічною капсулою *Mercury*, працювали з кроком у пів дня навперемінно з проведенням модульного тестування.

Про ці часи було багато написано. Крейґ Лармен та Вік Базілі задокументували історію, яку узагальнено у «Wiki» Ворда Каннінгема[1], а також у книжці Лармена *Agile & Iterative Development: Guide of Manager*[2].

Але *Agile* був не єдиним вибором. Насправді існувала конкурентна методологія, яка мала значний успіх у виробництві та промисловості в цілому: наукова організація управління.

Наукова організація праці (*Scientific Management*) — це адміністративно-управлінський підхід згори донизу. Керівники застосовують наукову організацію праці для визначення найкращої процедури для досягнення мети, а потім спрямовують її усім підлеглим для неухильного виконання. Інакше кажучи, існує велике попереднє планування з подальшим ретельним детальним виконанням.

Наукова організація праці, мабуть, того ж віку, що й піраміди, Стоунгендж чи якісь інші великі витвори родом із давнини, бо неможливо повірити, що такі споруди було створено без неї. Знову ж таки, ідея повторення успішного процесу є занадто інтуїтивно зрозумілою та притаманною людині, щоб вважати її якоюсь революцією.

Наукова організація праці отримала свою назву з праць Фредерік Вінслоу Тейлора в 1880-х. Тейлор формалізував і комерціалізував підхід та заробив капітал як консультант з управління. Ця технологія була надзвичайно успішною й призвела до масового підвищення ефективності та продуктивності впродовж наступних десятиліть.

[1] «Wiki» Ворда, c2.com є оригінальною Вікіпедією — першою, яка з'явилася в мережі Інтернет. Нехай її підтримка триває якнайдовше.

[2] Larman C. *Agile & Iterative Development: A Manager's Guide.* Boston, MA: Addison-Wesley, 2004.

Так трапилося, що в 1970-х світ програмного забезпечення опинився на роздоріжжі між цими двома протилежними методиками. Пре-*Agile* (*Agile* до отримання власне назви «*Agile*») робив короткі реактивні вимірювані та схильні до вдосконалення кроки для того, щоб просуватися в правильному напрямку до хорошого результату. Наукова організація праці відтерміновувала дії до створення ґрунтовного аналізу та детального плану. Пре-*Agile* добре спрацьовував на проєктах, які мали низьку вартість змін та вирішували частково визначені завдання з довільно визначеними цілями. Наукова організація праці найкраще підходила проєктам, які зазнавали великих витрат від змін та розв'язували дуже чітко визначені проблеми з надзвичайно конкретними цілями.

Головне питання полягало в тому, які проєкти були в галузі розробки програмного забезпечення? Чи була вартість змін високою і чи були чітко визначені конкретні цілі? Або: чи вартість змін у них була низькою і частково визначалися довільними цілями?

Не шукайте особливого смислу в останньому абзаці. Ніхто, наскільки мені відомо, насправді не ставив собі ці запитання. За іронією долі, шлях, яким ми пішли в 1970-х, здається, було обрано, імовірно, випадково, аніж умисно.

У 1970 році Вінстон Ройс написав документ[1], у якому описав свої ідеї щодо управління масштабними програмними проєктами. Документ містив схему (рис. 1.1, с. 24), яка зображувала його план. Ройс не був автором цієї схеми й не виступав за неї як план. Насправді схема відігравала роль «опудала» — логічної хиби, яку треба буде зруйнувати на наступних сторінках його праці.

Хай там як, розміщення схеми на видному місці та схильність людей робити висновки про зміст статті на основі схеми, розміщеної на першій чи другій сторінці, призвели до кардинального зрушення в галузі програмного забезпечення.

[1] Royce, W. W. 1970. Managing the development of large software systems. *Proceedings, IEEE WESCON,* August: 1–9. Accessed at http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf.

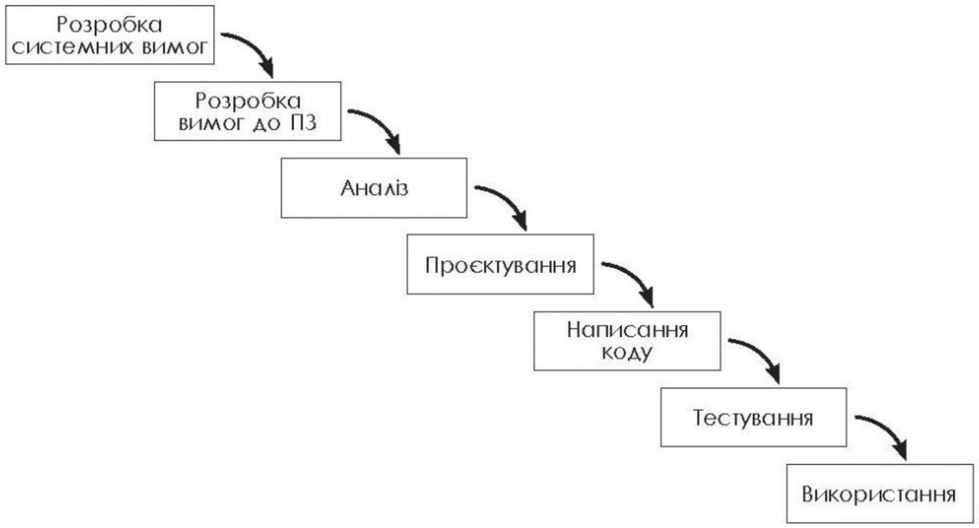

Рис. 1.1. Схема Вінстона Ройса, яка надихнула на розробку каскадної моделі

Початкова схема Ройса була більше схожа на воду, що стікала скелястим гірським хребтом, аніж на техніку, яка дістала назву «каскадна модель».

Каскадна модель стала логічним нащадком наукової організації праці. Її суть полягає в розробці ретельного аналізу, укладанні детального плану, а вже потім — у його виконанні аж до завершення.

Хоча Ройс і не рекомендував цю концепцію, але саме її було взято з його праці, і вона домінувала наступні три десятиліття[1].

Десь звідтоді починається мій шлях. У 1970 році мені було 18, я працював програмістом у компанії *A. S. C. Tabulating*, що в Лейк Блафф, штат Іллінойс. У компанії був комп'ютер IBM 360/30 із пам'яттю ядра 16 кБ, *IBM* 360/40 із пам'яттю 64 кБ і мінікомп'ютер *Varian 620/f* із пам'яттю 64 кБ. Я програмував для сімейства 360 на мовах *COBOL*, *PL/1*, *Fortran* і ассемблер. На ассемблері я писав для мінікомп'ютера *620/f*.

Важливо пам'ятати, як велося програмістам у ті часи. Ми записували наш код на програмних формулярах за допомогою олівців,

[1] Варто зауважити, що моє тлумачення цієї часової шкали взяли під сумнів у Розділі 7 праці Л. Боссавіта: Bossavit L. *The Leprechauns of Software Engineering: How Folklore Turns into Fact and What to Do About It*. Leanpub, 2012.

і в нас були оператори клавіш, які наносили програми на картки. Ми передавали наші ретельно перевірені перфокарти операторам ЕОМ, які виконували компіляції та проводили тестування під час третьої зміни, оскільки впродовж дня комп'ютери були надто зайняті роботою. Часто від початку написання коду до першої компіляції минало багато днів, і кожен цикл розробки зазвичай тривав добу. Дещо по-іншому відбувалася робота на *620/f*. Цю машину було виділено нашій команді, тому до неї ми мали цілодобовий доступ. Ми проводили два, три, часом навіть чотири цикли розробки й тестування на добу. Команда, до якої я належав, складалася з людей, котрі, на відміну від більшості програмістів того часу, уміли друкувати. Тому ми могли набивати власні перфокарти, не потрапляючи в залежність від примх операторів перфоратора.

Яку методологію розробки ми тоді використовували? Це, звісно, була не каскадна модель. Ми не мали вивершеної концепції чи детальних планів. Просто працювали над нашим кодом, компілювали, тестували його та виправляли помилки. Це був нескінченний цикл без структури. Це був зовсім не *Agile* чи навіть не пре-*Agile*. У тому, як ми працювали, не було дисципліни. Не було наборів програм для тестування і вимірюваних часових інтервалів. Ми кодили й фіксили баги, кодили й фіксили, день у день, місяць у місяць.

Я вперше прочитав про каскадну модель у галузевому журналі десь у 1972 році. Мені це здалося дарунком небес. Чи могли б ми заздалегідь проаналізувати проблему, потім запропонувати її рішення, а далі реалізувати задумане? Чи могли б ми насправді розробити графік на основі цих трьох етапів? Невже після виконання аналізу ми справді б на третину просунулися у виконанні проєкту? Я відчув силу концепції. Я хотів у це повірити. Якби ідея спрацювала, то мрію вдалося б утілити.

Очевидно, я був не один, бо багато інших програмістів і центрів програмування також підхопили цю ідею. І, як я вже казав, каскадна модель почала домінувати в нашому мисленні.

Вона домінувала, але не спрацьовувала. Протягом наступних тридцяти років я, мої однодумці, а також мої побратими й посестри