Зміст

Передмова. «Ага!» 5
Вступне слово 12
Уявіть собі світ, де розробники (Dev) та операційні системи (Ops) об'єднаються в DevOps. Вступ до «Посібника» 13

ЧАСТИНА I. «ТРИ ШЛЯХИ»

Вступ 30
1 Agile, безперервне постачання і «три шляхи» 34
2 Перший шлях: принципи потоку 41
3 Другий шлях: принципи зворотного зв'язку 52
4 Третій шлях: принципи безперервного навчання та експериментування 60

ЧАСТИНА II. ІЗ ЧОГО РОЗПОЧАТИ

Вступ 70
5 Вибір стартового потоку створення цінності 71
6 Розуміння роботи в нашому потоці цінності, забезпечення її прозорості та поширення на всю організацію 81
7 Як проектувати організацію та архітектуру, не забуваючи про закон Конвея 95
8 Як отримати чудові результати, інтегруючи операції у щоденну роботу розробників 113

ЧАСТИНА III. ПЕРШИЙ ШЛЯХ: ТЕХНІЧНІ ПРАКТИКИ ПОТОКУ

Вступ 126
9 Створіть основи конвеєра розгортання 127
10 Забезпечте швидке та надійне автоматизоване тестування 137
11 Започаткуйте і практикуйте безперервну інтеграцію 155

12 Автоматизуйте та запустіть релізи з низьким рівнем ризику 164
13 Архітектура релізів із низьким ризиком 189

ЧАСТИНА IV. ДРУГИЙ ШЛЯХ: МЕТОДИКИ ЗВОРОТНИХ ЗВ'ЯЗКІВ

Вступ 202
14 Створіть телеметрію, що дозволяє помічати і розв'язувати проблеми 204
15 Аналізуйте телеметрію, щоби краще передбачати проблеми і досягати цілей 223
16 Налаштуйте зворотний зв'язок так, аби розробники та інженери з експлуатації могли безпечно розгортати код 235
17 Інтегруйте розробку, що базується на гіпотезах, та A/B тестування у вашу щоденну практику 248
18 Створення процесів перевірки і координації для поліпшення якості поточної роботи 255

ЧАСТИНА V. ТРЕТІЙ ШЛЯХ: МЕТОДИКИ БЕЗПЕРЕРВНОГО НАВЧАННЯ ТА ЕКСПЕРИМЕНТУВАННЯ

Вступ 272
19 Започаткуйте та запровадьте навчання у щоденну роботу 273
20 Перетворіть локальні відкриття на глобальні поліпшення 286
21 Зарезервуйте час для організації навчання та вдосконалення 297

ЧАСТИНА VI. МЕТОДИКИ ІНТЕГРУВАННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ, УПРАВЛІННЯ ЗМІНАМИ І ДОТРИМАННЯ НОРМ

Вступ 308
22 Захист інформації як частина повсякденної роботи усіх інженерів компанії 310
23 Захист конвеєра розгортання 329

Заклик до дії. Висновок до «Посібника з DevOps» 342

Додатки 345
Додаткові джерела 359
Покажчик 362
Подяки 380
Про авторів 383

1 Agile, безперервне постачання і «три шляхи»

Цей розділ присвячений введенню в основи теорії ощадливого виробництва і «трьох шляхів» — саме на ці підвалини спирається сучасний стан DevOps. Головна увага приділяється теорії і принципам, виробленим за кілька десятиліть дослідження робочих середовищ, організації високої надійності виробництва, створенню моделей управління з високим рівнем довіри, покладених в основу методів DevOps. Отримані зрештою принципи і схеми дій, а також їхнє практичне застосування у створенні потоку технологічної цінності описані в інших розділах посібника.

ВИРОБНИЧИЙ ПОТІК ЦІННОСТІ

Одна із фундаментальних концепцій ощадливого виробництва — *потік створення цінності*. Спочатку визначимо це поняття в межах промисловості, а потім екстраполюємо на DevOps і технологічний потік створення цінності. Карен Мартін і Майк Остерлінг у книжці *Value Stream Mapping: How to Visualize Work and Align Leadership for Organizational Transformation* визначають потік створення цінності як «послідовність дій, що вживаються організацією з метою виконати запит клієнта», або як «послідовність дій, необхідних для розробки, випуску і постачання товарів клієнту, включно з обома потоками — як інформаційним, так і матеріальним».

У матеріальному виробництві потік створення цінності легко визначити і так само легко спостерігати за ним: він починається із отримання замовлення від клієнта і надходження сировини на склад. Аби забезпечити стислий і передбачуваний час виконання замовлення у будь-якому потоці створення цінності, зусилля організації зазвичай фокусуються на створенні плавного й рівномірного потоку роботи з використанням таких прийомів, як зменшення розміру партії, зниження обсягу незавершеного виробництва (WIP, Work in Process), недопущення переробок, аби виключити використання дефектних деталей на кінцевих етапах, і постійна оптимізація системи заради стійкого руху до глобальних цілей.

ТЕХНОЛОГІЧНИЙ ПОТІК ЦІННОСТІ

Ті самі принципи і методи, що забезпечують швидкість виконання роботи в процесах матеріального виробництва, можуть застосовуватися в галузі технологій (і взагалі будь-якої діяльності, що створює знання). У DevOps технологічний потік створення цінності зазвичай визначають як процес, необхідний для перетворення бізнес-гіпотез на технологічний сервіс, що постачає цінність замовнику.

Вихідними даними для процесу є формулювання бізнес-цілі, концепції, ідеї або гіпотези. Усе починається, коли ми приймаємо завдання в галузі розробки, додаючи їх до вже наявних у роботі беклогів (списків завдань до виконання).

Отже команда розробників, що дотримується типового Agile або ітеративного процесу, ймовірно, зможе адаптувати певну ідею відповідно до побажань користувачів, створивши різновид специфікації функціональних можливостей, що згодом будуть реалізовані у вигляді коду програми або у створюваному сервісі. За цим слідує розміщення програми або сервісу в репозиторії системи контролю версій, де кожна зміна інтегрується в основний код і тестується разом з усією системою ПЗ.

Оскільки цінність створюється, коли наші сервіси працюють у виробничому середовищі, слід забезпечити не лише швидкий потік постачання, а й виконання розгортання без хаосу і перебоїв — затримок в обслуговуванні та порушень функціонування сервісів, вимог безпеки або сумісності.

ФОКУС НА ЧАСІ РОЗГОРТАННЯ

У подальших частинах книжки ми зосередимося на одній зі складових частин потоку створення цінності — часі розгортання. Починається він тоді, коли якийсь інженер[1] із команди потоку створення цінностей (включно з розробниками, тестувальниками, відділами експлуатації та інформаційної безпеки) отримує зміну від системи контролю версій, а закінчується, коли зміна починає успішно працювати у виробничому середовищі, створюючи продукт для клієнтів і генеруючи зворотний зв'язок і телеметрію.

До першого етапу роботи входять проєктування і розробка, що певним чином схоже на ощадливу розробку продуктів, і характеризується високою мінливістю і невизначеністю, часто вимагає творчого підходу до виконання роботи, що може ніколи не знадобитися. Це призводить до різної тривалості даного етапу. На відміну від нього, другий етап, що включає тестування і виробництво, цілком відповідає принципам ощадливого виробництва. Він вимагає творчого підходу й досвіду і має тенденцію до передбачуваності та автоматизації, його мета — забезпечити корисний результат із мінімальною

[1] Тут і далі слово «інженер» означає будь-яку людину, яка працює в команді, що виробляє потік цінності, а не тільки розробників.

мінливістю (наприклад, невеликий і передбачуваний час виконання, близьку до нуля кількість дефектів).

Замість того щоби пропускати великі обсяги роботи через такі частини потоку створення цінності, як «проєктування і розробка», а потім через потік «тестування і виробництво» (наприклад, коли використовується великий каскадний процес або довготривалі функціональні гілки коду) наша мета полягає в тому, щоби тестування і виробництво відбувалися одночасно із проєктуванням і розробкою, що забезпечує швидкий рух потоку та високу якість. Цей метод стає успішним, коли виконання здійснюється невеликими частинами, а якість забезпечується на кожному етапі потоку створення цінності[1].

Визначення часу виконання і часу обробки

У Lean-спільноті час виконання вважається однією із двох характеристик, що зазвичай використовуються для вимірювання продуктивності потоків цінності, інша характеристика — час обробки (іноді його також називають «часом контактування» або «часом виконання завдання»[2]).

Якщо відлік часу виконання замовлення починається з його оформлення і закінчується з виконанням, то час виробництва відраховується з моменту, коли ми починаємо роботу над замовленням, тобто не враховується той період часу, доки замовлення перебувало у черзі на обробку (рис. 2).

Рис. 2. Час виконання і час обробки

Оскільки час виконання замовлення — найважливіша для клієнта характеристика, то звичайною метою є вдосконалення процесів заради скорочення цього параметра, а не часу обробки. Однак співвідношення часу обробки

[1] Відверто кажучи, у випадку використання методик на кшталт розробки через тестування, тестування може починатися ще до того, як буде написаний перший рядок коду.

[2] У цій книжці термін «час виконання» буде використовуватися з тієї ж причини, яку відзначили Карен Мартін і Майк Остерлінг: «Для того щоби мінімізувати можливу плутанину, ми уникаємо використання терміну "час циклу", оскільки він має кілька значень, у тому числі синонімічні для часу виробництва й темпу або частоти видачі результатів».

і часу виконання замовлення є важливим критерієм оцінки ефективності: для того, щоби забезпечити швидкий потік і скорочений час виконання замовлення, майже завжди потрібно скоротити час очікування, доки черга дійде до завдання.

Типовий сценарій: час розгортання вимагає місяців

У випадку ведення бізнесу звичайними способами ми часто опиняємося в ситуації, коли розгортання займає кілька місяців. Особливо часто це відбувається у великих організаціях зі складною структурою, що використовують тісно пов'язані один з одним монолітні застосунки, котрі зазвичай погано інтегровані в середовище для тестування, мають значну тривалість тестування і тривалий час розгортання в робочому середовищі, високу залежність від тестування вручну і необхідність схвалення численними інстанціями в компанії. Коли щось подібне відбувається, потік створення цінності починає виглядати приблизно так, як показано на рис. 3.

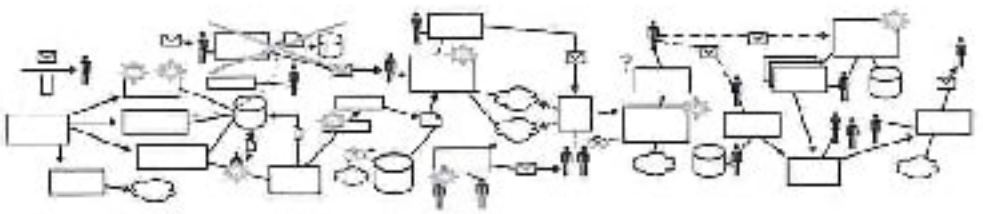

Рис. 3. Потік технологічної цінності із часом розгортання тривалістю три місяці. *(Джерело: Edwards D. DevOps Kaizen. 2015)*

За наявності великої тривалості часу розгортання практично на кожній стадії потоку створення цінності потрібні надзусилля. Може статися і так, що на завершальному етапі проекту, коли всі результати праці інженерів буде зібрано докупи, все перестане працювати, код не буде збиратися правильно або не витримуватиме тестування. Виправлення проблеми разом із виявленням того, хто саме «зламав» код і як це виправити, вимагатиме кількох днів або навіть тижнів, а в результаті віддача для клієнтів виявиться невисокою.

Наш ідеал — DevOps: розгортання за хвилини

В ідеальному випадку при роботі з DevOps розробники швидко й регулярно отримують зворотний зв'язок. Це надає їм можливість швидко й незалежно зпроваджувати, інтегрувати і валідувати код, а також забезпечувати його розгортання у виробничому середовищі (це можуть робити як вони самі, так і інший відділ).

Це досягається шляхом постійної перевірки невеликих змін у коді, зроблених у репозиторії системи контролю версій, виконанням автоматичного і передвиробничого тестування змін, а потім розгортанням у реальному виробничому середовищі. Що й дозволяє нам бути твердо впевненими:

ри завдань та інтервали між ними, забезпечити якість шляхом запобігання потраплянню дефектів на кінцеві етапи і постійно займатися оптимізацією заради досягнення глобальної мети організації.

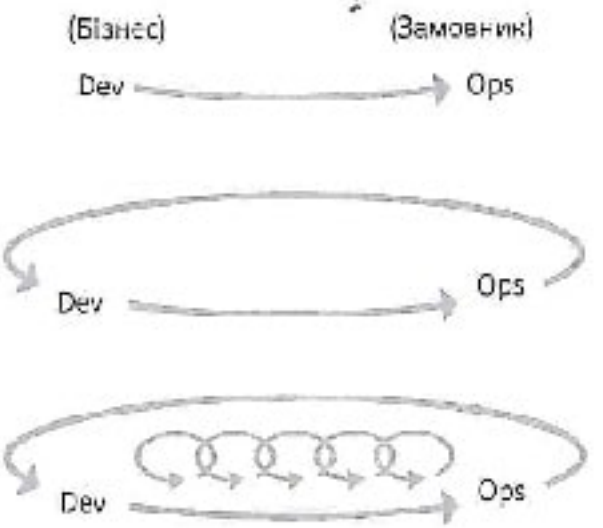

Рис. 5. «Три шляхи» (*Джерело: Kim G. The Three Ways: The Principles Underpinning DevOps.— IT Revolution Press blog, 2016, August 9.— http://itrevolution.com/ the-three-ways-principles-underpinning-devops/*)

За рахунок прискорення проходження роботи через потік створення технологічної цінності ми скорочуємо час виконання запитів внутрішніх або зовнішніх клієнтів, особливо необхідний для розгортання коду у виробничому середовищі. При цьому підвищується якість і результативність, а також здатність перевершити конкурентів.

До розроблених на основі тривалого досвіду методів входять безперервна розробка, інтеграція і тестування, а також процеси розгортання: створення різних середовищ на вимогу, обмеження обсягу незавершеного виробництва (WIP) і побудову систем і організацій, які дозволяють здійснення змін без ризику.

Другий шлях забезпечує швидкий і постійний потік зворотного зв'язку справа наліво на всіх етапах потоку створення цінності. Це вимагає посилення зворотного зв'язку з метою запобігання повторному виникненню проблем, їх швидкому виявленню та усуванню. При цьому ми забезпечуємо якість уже на початковому етапі і створюємо або вбудовуємо знання в ті етапи, де воно необхідне. Це дозволяє нам створювати більш безпечні системи: проблеми виявляються і виправляються задовго до того, як може статися катастрофічний збій.

Якщо розпізнавати проблеми одразу після їх появи і розв'язувати їх негайно, то зникає можливість постійного скорочення петель зворотного зв'язку, що є одним із основних постулатів усіх без винятку сучасних методологій удосконалення процесів.