Engineer

Технічний борг

додаткові витрати згодом за швидке рішення сьогодні

Що це таке

Технічний борг — це майбутня ціна спрощення в дизайні або коді системи: рішення, яке заощаджує час сьогодні, а пізніші зміни робить повільнішими, ризикованішими чи взагалі неможливими, доки це спрощення не приберуть. Англійською це називають technical debt, часто скорочено tech debt. Назва — метафора, і її обрали тому, що таке спрощення поводиться як позика: воно купує щось зараз і бере за це плату згодом, поки лишається на місці.

Ідея походить зі звіту Ворда Каннінгема про досвід розробки WyCash, системи для керування інвестиційними портфелями, написаної на Smalltalk. Звіт він представив на конференції OOPSLA 1992 року. Невелика команда мала загальне уявлення про близько чотирьох мегабайтів вихідного коду, а зрілі частини програми багато разів переглядали чи переписували. Саме ця консолідація, стверджує звіт, зробила можливим рівномірне поступове зростання, а її відсутність і є серйозною пасткою: код, який працює і задовольняє клієнта, але так і не був доведений до ладу. Каннінгем висловив це одним рядком: «Shipping first time code is like going into debt». Сам звіт говорить лише про борг і про свою боргову аналогію. Пізніші праці вживають повну назву, а стаття Google 2015 року називає технічний борг метафорою, яку Ворд Каннінгем запровадив 1992 року.

Дослідники, які зібралися в Дагштулі у квітні 2016 року, щоб домовитися, що охоплює це поняття, сформулювали робоче визначення: сукупність проєктних або реалізаційних конструкцій, зручних у короткій перспективі, які, однак, створюють технічний контекст, здатний зробити майбутні зміни дорожчими або неможливими. Вплив вони обмежують якостями всередині системи, передусім тим, наскільки легко її супроводжувати й розвивати. Це відрізняє борг від функції, якої ніхто не збудував: ціну боргу платять ті, хто змінює систему, а до користувачів вона доходить лише через них. Той самий звіт зберігає фінансову лексику. Основна сума — це те, що заощадило

спрощення, або скільки сьогодні коштувало б краще рішення. Відсотки бувають двох видів: поточні (recurring interest), які сплачують нижчою швидкістю, похідними дефектами й утраченою якістю, і накопичувальні (accruing interest), тобто додаткова ціна кожної нової частини програмного забезпечення, збудованої поверх коду, який не зовсім правильний. Борг живе переважно в коді, але також у документації, тестах і відомих дефектах.

Застосуйте поняття до рішення

Застосуйте «Що таке технічний борг?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Структура системи, спершу продумана на папері, щоб частини пасували одна до одної, а вартість була відома до першого рядка коду.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Один інфраструктурний план для 14 підрозділів — 14 дослідницьких підрозділів, охоплених спільним інфраструктурним планом

Один інфраструктурний план для 14 підрозділів Зв'язатися з нами →

Чому це важливо

Борг, узятий свідомо, може бути добрим рішенням. Звіт Каннінгема каже, що невеликий борг пришвидшує розробку, якщо його швидко повернути переписуванням, і стаття Google погоджується, що для позики часто є вагомі стратегічні причини. Обидва застерігають від боргу, який лишається неповерненим. У звіті кожна хвилина, витрачена на код, що не зовсім правильний, рахується як відсотки, а цілі організації розробників можуть зупинитися під вагою реалізації, яку ніхто не консолідував. Ще до того програма стає чимось, що жодна окрема людина не здатна осягнути, а це веде до програмістів, кожен з яких знає свій куток, і зрештою до продукту, який не може змінити напрям.

Проблема в тому, що борг часто не є свідомим вибором. Звіт із Дагштуля зазначає, що він часто виникає ненавмисно й застає розробників зненацька, тож виявляють його здебільшого під час супроводу, через тривалий час після рішення, яке його створило. Причини буденні: тиск термінів, ключова людина, яка недоступна, або брак знань про якусь технічну властивість на той момент. Стаття Google додає, що прихований борг небезпечний, бо росте непомітно.

Метафора також дає розробникам і керівникам спільну мову. Звіт із Дагштуля називає її потужним механізмом комунікації, а бачення, яке описують учасники, передбачає керувати боргом так само ретельно, як дефектами й новими функціями, і мати спосіб перекладати між тим, що турбує розробників, і тим, що вирішують керівники. Дослідження, представлене на семінарі, показує, що це може означати для команди. Граціела Тонін, Альфредо Голдман і Керолін Сіман попросили студентів у командах, де загалом було 60 людей, явно виявляти й відстежувати свій борг. Видимий список дав командам змогу домовлятися із замовником про час на рефакторинг, слугував пам'яттю про незрілі частини проєкту й показував, чи покращується якість коду. Більшість учасників команд сказали, що стали більше спілкуватися, ретельніше зважувати, чи справді потрібен той чи інший борг, і частіше робити рефакторинг. Оскільки рішення взяти борг обговорювали, додають автори, у деяких командах стало менше експертів, яких ніхто інший не міг замінити.

Як це виглядає на практиці

Борг стає керованим, коли його записують там, де команда планує свою роботу. Запис у реєстрі, складений для цієї статті, може бути таким коротким:

```
TD-14          Підсумки рахунків обчислюються у трьох місцях
Артефакти:     billing/web, billing/export, reports/monthly
Причина:       фіксована дата запуску; скопіювати правило було швидше,
               ніж перенести його до спільного модуля
Основна сума: близько 2 днів, щоб перенести правило в один модуль
               разом із тестами
Відсотки:      кожну зміну податкових правил вносять і тестують тричі, і
               одна з п'яти останніх змін пропустила одну з копій
Далі:          погасити разом із наступною зміною податкових правил;
               експорт електронних рахунків, запланований на цей
               квартал, не має стати четвертою копією
Власник:       команда білінгу
```

Поля відповідають моделі зі звіту Дагштуля: запис, прив'язаний до конкретних артефактів, із причиною, основною сумою й відсотками. Корисним запис робить саме поле відсотків, бо воно перетворює відчуття, що код білінгу заплутаний, на ціну, яку хтось може зважити. Якщо податкові правила змінюються чотири рази на рік і кожна зміна коштує зайвого дня роботи й тестування, два дні основної суми окупаються за пів року. Запланований експорт — це накопичувальні відсотки в сенсі Дагштуля:

кожна нова частина, збудована на скопійованому правилі, робить пізніше виправлення дорожчим. А якби код був заморожений і ніхто не збирався його чіпати, той самий запис був би аргументом залишити борг як є.

Борг стосується не лише коду. Запис у портфоліо на цьому сайті про те, як реорганізувати процеси й заощадити 8 000 годин, документує попередню роботу засновниці у Mappitall. Він питає читача, чи не гальмує його технічний борг, і описує той його різновид, що накопичується навколо самої організації роботи: архітектуру програмного забезпечення, яка розросталася стихійно, а не за задумом, системи, які працювали, але неефективно, і планування, через яке люди то чекали без діла, то були перевантажені. Нічого з цього не «горіло», саме тому його й не чіпали. За роки роботи з 2015 до 2022 перебудова архітектури, систем і планування заощадила близько 8 000 годин, а запис наголошує, що зміни вбудували в повсякденну роботу команди, бо покращення процесів, які не закріплено назавжди, з часом сповзають до старого способу роботи.

Де виникають проблеми

Технічним боргом називають усе. Щойно вираз стає популярним, цю етикетку отримує будь-яке подразнення, від старої бібліотеки й відсутньої функції до стилю колеги. Звіт із Дагштуля прямо називає цей ризик і у відповідь просить чітких визначень. Вживайте це поняття лише для спрощень, ціна яких лягає на майбутні зміни, реєструйте дефекти як дефекти, а побажання як функції, і відхиляйте запис у реєстрі, який не може назвати свою причину й свої відсотки.

Борг існує лише в головах. Борг, який ніхто не записав, сплачує той, хто наступним торкнеться коду, зазвичай не знаючи, чому робота триває так довго. Ведіть реєстр поруч із беклогом і перевіряйте його, перш ніж змінювати частину системи, як студентські команди в дослідженні Тонін використовували свій список. Запис стає видимим для тих, хто вирішує, на що витрачати час, лише тоді, коли він стоїть там, куди вони дивляться.

Основну суму гасять там, де відсотки не нараховуються. Прибирання в модулі, якого ніхто не змінює, здається продуктивним і нічого не повертає, тоді як код, що змінюється щотижня, і далі коштує. Відсотки нараховуються лише там, де люди працюють, тож сортуйте реєстр за тим, як часто змінюється кожна частина і що незабаром будуватимуть поверх неї, і гасіть найдорожчі записи першими.

Погашення відкладають до одного великого переписування. На борг, що лежить роками, часто відповідають проєктом, який має замінити всю систему й сам набирає нового боргу, тоді як стара система і далі коштує. Каннінгем радив повертати борг швидко, а стаття Google перелічує звичні способи його гасити: рефакторинг коду, кращі модульні тести, видалення мертвого коду, менше залежностей, суворіші API і краща документація. Жоден із них не додає функції; суть у тому, щоб наступна зміна стала дешевшою. Плануйте їх як частину звичайної роботи, поруч зі зміною, якій вони потрібні.

Експерименти лишаються в робочому коді. Перевірити альтернативу як умовну гілку в робочому коді дешево для кожного окремого експерименту, і стаття Google описує, як такі гілки накопичуються, доки перевірити всі комбінації не стає важко або неможливо. Вона згадує Knight Capital, чия система, за даними статті, втратила 465 мільйонів доларів за 45 хвилин, судячи з усього, через те, що застарілі експериментальні шляхи коду повелися неочікувано. Переглядайте кожну експериментальну гілку й кожен прапорець за сталим графіком, видаляйте те, що вже не використовується, і цінуйте видалення коду так само, як додавання.

Оцінку інструмента сприймають за сам борг. Статичний аналіз може порахувати дубльовані рядки й довгі функції, а дашборд робить це число схожим на залишок на банківському рахунку. Стаття Google прямо каже, що метафора не дає суворої міри, яку можна відстежувати з часом, і що команда, яка досі рухається швидко, ще не доводить, що боргу мало. У Дагштулі Форрест Шалл розповів, що жоден окремий набір правил для пошуку боргу не корелював послідовно з його симптомами, як-от зростанням вартості змін і падінням швидкості, у різних проєктах. Звіряйте знахідки інструмента з власною історією змін і дефектів, залишайте правила, які передбачають витрати, а решту відкиньте. Кращою перевіркою є власні запитання статті, зокрема те, як швидко новий учасник команди може ввійти в курс справи.

Що почитати далі

- The WyCash Portfolio Management System — звіт Ворда Каннінгема про досвід, представлений на OOPSLA ‘92, де вперше з’являється боргова аналогія.
- Managing Technical Debt in Software Engineering — звіт семінару Дагштуль 16162 із робочим визначенням, основною сумою й відсотками та дослідженням явних списків боргу.

- Hidden Technical Debt in Machine Learning Systems — стаття Скаллі та колег із Google 2015 року про те, як борг накопичується, як його гасити і чому його важко виміряти.
- Refactoring — сайт Мартіна Фаулера про невеликі зміни, що зберігають поведінку, які стаття Google називає першими серед способів гасити борг.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке технічний борг?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/technical-debt/