ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ ПОКАЗНИКІВ ЖИТТЄВОГО ЦИКЛУ ІТ-ТЕХНОЛОГІЙ

М.С. Пасека¹, О.Б. Турчин²

Проаналізовано основні показники життєвого циклу технологій та виділено узагальнену систему для оцінювання етапу життєвого циклу. Показники розглянуто як властивості, якими має володіти та чи інша технологія, щоб відповідати певному етапу свого існування. Досліджено наявні підходи до аналізу розвитку технології. Запропоновано узагальнену класифікацію, тобто перелік показників для відповідного етапу розвитку. Узагальнена система оцінки технологій дасть змогу мінімізувати технологічні ризики, що пов'язані з коротко- та довготерміновими перспективами розроблення програмного забезпечення.

Ключові слова: життєвий цикл, технологія, параметри, етап розвитку, узагальнена класифікація, технологічні ризики, мова програмування.

Вступ. У сучасному світі ІТ-технології розвиваються дуже динамічно і часто вступають в конкуренцію між собою або навпаки – займають різні ніші, не вступаючи у відкриту боротьбу. Але очевидно одне – розвиток технології та рішень, побудованих за допомогою неї, вступає в повну залежність до кількості кваліфікованих фахівців, що безпосередньо її використовують та створюють комерційні рішення на її основі. Варто також враховувати промислове лобі – зацікавленість великих компаній у просуванні власних технологічних рішень з метою встановлення домінантного положення у вже існуючих та нових напрямках розробки.

Термін "технологія" треба розуміти як сукупність знань, відомостей про послідовність окремих виробничих операцій у процесі виробництва чого-небудь [1]. Тому, як і в інших виробничих галузях, в ІТ-галузі базовою технологію потрібно вважати не програмне забезпечення, а засоби, за допомогою яких це програмне забезпечення (ПЗ) створюється, тобто мови програмування та технологічне середовище, яке утворилося довкола них. Водночас, кожна мова програмування розробляється під конкретний напрям розробки, тому безпосереднє порівняння різних за ідеологією мов є не доцільним та під час обґрунтування показників було враховано їх життєвий цикл (ЖЦ).

На ринку існує велика різноманітність технологій для розробки та немає узагальненої системи для оцінювання їх перспективності, можливості впровадження, доцільності використання та кількості працівників які володіють або володітимуть у майбутньому достатньою кваліфікацією. Потреба менеджерів, керівників фірм, компанії, великих корпорації визначити, яку технологію варто використати зі запропонованих, щоб забезпечити добру продуктивність розробки, малі фінансові витрати та в майбутньому отримати достатню кількість кваліфікованих працівників, які могли б підтримувати п/п та удосконалювати його з найменшою кількістю витрат – невпинно зростає і пов'язано зі шаленим розвитком та проникненням програмного забезпечення у життя людей.

¹ доц. М.С. Пасека, канд. техн. наук – Івано-Франківський НТУ нафти і газу;

² бакалавр О.Б. Турчин – Івано-Франківський НТУ нафти і газу

Актуальність. На цей час життєвий цикл технології розглядають у контексті життєвого циклу програмного продукту, тобто не існує певного розділення між технологією і програмним продуктом. Вважають, що технологія – це теж програмний продукт. Технологія володіє більшим терміном використання у зв'язку з ефектом "снігового кому" – кількість продуктів, які були розроблені та слід підтримувати не дають змогу відмовитися від навіть застарілої технології. Наприклад, мови програмування C/C++ є достатньо успішними навіть після 30-40 років свого існування. Їх частка на ринку зменшилася не через те, що вони втратили доцільність, а через те що почали використовуватися у більшому об'ємі новіші технології. Вони стали вузькоспеціалізованими технологіями і водночас перейшли на останній етап свого існування – повільний розвиток та зменшення кількості кваліфікованих спеціалістів. Вважати, що найпопулярніша технологія на цей момент буде популярною через 10-15 років – помилково. Це можливо, але величина ризику такого підходу – колосальна. До 2000 р. C++ вважався еталоном для розробки більшості ПЗ, а кількість програмістів, які ним володіли на високому рівні, була величезна. На той момент Java вважалася молодішою технологією з малою кількістю спеціалістів, що нею володіли, і тяжко було уявити, що через 10 років вона обійде за кількістю спеціальності C++ у кілька разів. Причиною цього була висока швидкість розроблення, адаптивність під різні платформи, легкість освоєння порівняно з C та C++.

Програмні продукти мають високу цінність тільки за умови, якщо вони розробляються на перспективу, тобто на початковому етапі вкладаються ресурси з метою отримання доходу у майбутньому протягом тривалого часу. Вибір технології, за допомогою якої буде розроблятися продукт, дуже важливий. Оскільки немає певних характерних параметрів, на які варто звернути увагу, а тільки розмиті уявлення у різних представників ринку розробки ПЗ, то доцільно теоретично обґрунтувати та виділити основні параметри оцінки життєвого циклу технології, щоб не потрапити в ситуацію переходу від однієї технології до іншої. Отже, під час вибору технології потрібно оцінити, на якому етапі свого існування вона перебуває і чи для розроблюваного продукту цей етап є доцільним.

Постановка завдання. Для оцінювання етапу життєвого циклу технології проаналізовано параметри, за якими можна скласти її якісні та кількісні характеристики. Усі параметри поділено на дві групи: параметри, які безпосередньо та які опосередковано характеризують етапи життєвого циклу. На основі такої класифікації створено узагальнену систему оцінки.

Аналіз наявних показників. Існуючі системи життєвого циклу програмного забезпечення ПЗ відображають етапи розробки, а не стан та доцільність тої чи іншої технології в певний час, тому виділено такі параметри, які характеризують етап життєвого циклу технології:

- період існування на ринку;
- парадигма;
- кількість кваліфікованих спеціалістів;
- популярність засобів розробки на цій технології;
- швидкість адаптації;
- величина спільноти.

Водночас, також виділено опосередковані параметри:

- швидкість освоєння;
- підтримка великими компаніями.

Наявний перелік не можна вважати кінцевим або остаточним при виборі технології, але перераховані показники розкривають "підводне каміння" в процесі оцінки ЖЦ.

Період існування на ринку. Цей параметр безпосередньо характеризує етап життєвого циклу технології. Оскільки зрозуміло, що технології віком 5-10 років можна вважати молодими, 10-20 років – зріли, 20-30 років – стрімко старіючими (рис. 1). У ході дослідження проаналізовано найбільш популярні мови програмування, що представляють різні напрямки у розробленні програмних продуктів: Java і C# інтерпретуються у проміжний код віртуальної машини, Ruby та Perl – безпосередньо інтерпретуються під час виконання, C++ – компілюється. Встановлено, що Java від моменту релізу першої стабільної версії в 1995 р. стрімко зростала, тобто належала до молодого покоління технологій, а з 2005 р. – перейшла у стадію зрілості. C# вийшов на ринок у 2001 р. та в період 2001-2012 рр. демонстрував стрімке зростання (молоде покоління). C++ отримав першу стабільну версію у лютому 1985 р. (перший зовнішній випуск) та демонстрував стрімке зростання майже до 1998 р. (стандартизований Міжнародною організацією стандартизації під номером 14882:1998). Перша загальнодоступна версія мови програмування Ruby побачила світ 1995 р. [5]. Період молодості припадає на 1995-2007 рр. На стадію зрілості вона перейшла у 2007 р. Мова програмування Perl є піонером серед інтерпретованих мов, оскільки перший стабільний реліз отримала у 1987 р. Її період молодості припадає на початок ери мережі Internet, а зрілості – на період 1994-2005 рр. З 2005 р. вона перейшла на задній план та стрімко втрачає популярність.

Рис. 1. Тенденції популярності мов програмування в період 2005-2015 рр., засновані на оцінці кількості запитів у мережі Internet [3]

Цей параметр особливо важливий у контексті перспективи. Наприклад, якщо обрати технологію оперуючись тільки на час присутності на ринку, то присутність більше ніж 10 років – очевидний вибір, але для продуктів, які розробляються на довгу перспективу, це хибний шлях, бо кількість кваліфікованих спеціалістів з цієї технології невпинно знижуватиметься у зв'язку з високою конкуренцією інших технологій, що будуть більш привабливими через свою новизну. Тому для короткотермінових проектів зрілі технології – найкращий варіант, а для довготермінових доцільно обирати технології, що належать до

молодшої категорії, але є перспективними з погляду інших параметрів, що розглядаються.

Парадигма. Парадигма програмування – спосіб концептуалізації, що визначає організацію обчислень і структурування роботи, яку виконує комп'ютер [2]. Встановлено, що парадигма технології характеризує підхід у розробці ПЗ і не є основним фактором у визначенні етапу життєвого циклу технології, оскільки супроводжує її протягом усього існування, але безпосередньо визначає період існування. Тому розроблення мов програмування до 1990 р. велася спираючись на підтримку 1-2 парадигм (імперативний та об'єктно-орієнтований), а після – на 3-5 парадигм (імперативний та об'єктно-орієнтований, функціональний, декларативний, узагальнене програмування). Хоча більшість мов програмування декларують підтримку багатьох підходів – розроблення ведеться на основі того, який розробники визначили як оснований. Сьогодні найбільш популярні мови програмування належать до об'єктно-орієнтованої парадигми (рис. 2). Парадигма безпосередньо не вказує на етап у життєвому циклі, але вказує на тривалість циклу. Об'єктно-орієнтовані мови мають недолік, який забезпечить їм величезний період існування – велика кількість залежностей, що створюються у процесі розробки. Доречним є вислів Джо Армстронг в "Codgers at Work": "Проблема об'єктно-орієнтованих мов в тому, що вони тягнуть з собою все своє неявище оточення. Вам потрібен був банан – а ви отримуєте горилу з бананом, і цілі джунглі на додачу" [6]. Тобто у процесі розроблення програмних продуктів з використанням ООП тривалість підтримки, а значить присутність на ринку – зростає. Виникає парадоксальний висновок: "Парадигми, що тягнуть за собою своє оточення, володіють довшим життєвим циклом".

Кількість кваліфікованих спеціалістів. Цей параметр характеризує як зацікавленість та освіченість програмістів у певній технології, так і рівень конкуренції на ринку. Велика потреба в спеціалістах, що володіють певною технологією, спричиняє появу різноманітних засобів розробки. Але цей зв'язок не є одностороннім, оскільки зацікавлення у спеціалістах входить в пряму залежність від наявних засобів розробки, властивостей технології та кількості спеціалістів. Це замкнене коло спричиняє ефект "вічної молодості" – не здатність технології отримати критичну масу інструментів та спеціалістів, що забезпечать її популярність та перехід у сектор "вічно перспективних" технологій.

Також вважається, що ефект "вічної молодості" не відміння параметр "період існування на ринку", тобто після 20-30 років концепції закладання під час проектування технології стають неактуальними або неінноваційними. Зважаючи, що основна маса спеціалістів зосереджена в межах 10-15 мов – список яких постійно змінюється у процесі еволюції ринку (див. рис. 2).

Рис. 2. Найпопулярніші мови програмування на 2015 р. [4]

Тому під час оцінювання перспективності та етапу життєвого циклу потрібно враховувати як загальну кількість спеціалістів, так і тенденцію зростання їхньої кількості. Якщо кількість спеціалістів є істотною, але ріст їх числа дуже повільний або відсутній – технологія перебуває на піку своєї популярності, водночас мала кількість спеціалістів, проте колосальні темпи зростання – демонструють перехід технології із стадії молодості на стадію зрілості, що є оптимальним варіантом для програмних продуктів, які розробляються на перспективу.

Популярність засобів розробки на цій технології. Цей параметр часто не враховують під час оцінки, оскільки вважається, що популярність засобів розробки напряму залежить від кількості спеціалістів і часу наявності на ринку. В історії розвитку ринку програмного забезпечення часто відбувалася ситуація "другого дихання" мов програмування, тобто ситуації коли популярність зростала не завдяки інноваційним особливостям самої мови, а завдяки появі інноваційних засобів розробки. Для прикладу, Ruby до 2004 р. демонтувала помірковане зростання, але в період 2004-2006 рр. згідно з індексом TIOBE (TIOBE programming community index) збільшила свою присутність на ринку втричі та здобула перемогу в номінації "Мова програмування року" [7]. Аналогічна ситуація повторилася і для Python з її фреймворком Django, після появи якого забезпечив зростання популярності як самої мови, так і спонукав до розробки альтернативних засобів, що, водночас, розширює область застосування та привабливість для розробки. Також варто враховувати, що поява інноваційних засобів розробки позитивно впливає на час присутності на ринку мови програмування. Отже, можна зазначити, що висока популярність та велика кількість засобів розробки відтермінує етап стрімкого старіння, що є важливим в оцінці життєвого циклу мови програмування, як технології, так і її виборі для розробки. У табл. наведено переможців у номінації "Мова програмування року" згідно з індексу TIOBE в період 2003-2015 рр. та рік появи знакових для даних мов програмування засобів розробки.

Табл. "Мова програмування року" згідно з індексом TIOBE та засоби розробки, що сприяли росту популярності

Рік	Мова програмування року	Засіб розробки	Рік
2015	Java	Spring Hadoop	2012
		Spring Boot	2014
2014	JavaScript	AngularJS	2009
		Backbone	2010
		Meteor	2012
		ReactJS	2013
2013	Transact-SQL	–	–
2012	Objective-C	Apportable SDK	–
2011	Objective-C	iOS SDK	2009
2010	Python	Python 3	2009
2009	Go	–	–
2008	C	–	–
2007	Python	Django	2003
2006	Ruby	Ruby on Rails	2004
2005	Java	Spring Framework	2003
2004	PHP	WordPress	2003
2003	C++	Boost (libraries)	2001

Швидкість адаптації. В умовах ринку, що динамічно розвивається, здатність мови програмування пристосовуватися до потреб ринку є неоціненною, але у зв'язку із тим, що швидка адаптація може ламати зворотну сумісність, процес еволюції є дуже повільним. Також потрібно розуміти, що швидкість адаптації до потреб, які виникають у процесі розроблення, накладає на розробників істотні обмеження. Для всіх мов програмування швидкість еволюції напряму залежить від потреб ринку та ніші, у якій вони існують, але можна виділити 2 підходи розвитку: повна зворотна сумісність і зворотна сумісність у межах версії.

Переваги першого підходу: програмне забезпечення, написане за допомогою старішої версії, буде працювати на новішій; відсутність потреби оновлювати програмне забезпечення відповідно до нової версії мови програмування; менша потреба у вивченні спеціалістами нових підходів розробки. Недоліки першого підходу: зменшення швидкості розробки програмних продуктів порівняно з аналогічними, що розробляються на новіших мовах програмування; менша кількість "синтаксичного цукру" (англ. *Syntactic sugar*) порівняно з іншими технологіями [8]; перехід великої кількості спеціалістів на інші технології; низька швидкість адаптації до потреб ринку.

Переваги другого підходу: кодова база програмного забезпечення відповідає найновішим практикам розробки; перехід великої кількості спеціалістів з інших технологій; велика кількість "синтаксичного цукру", що пришвидшує розробку; збільшення терміну ЖЦ мови програмування; висока конкурентоспроможність. Недоліки другого підходу: програмне забезпечення працює тільки в межах версії, за допомогою якої розроблялося; потреба оновлювати код програмного забезпечення під час оновлення до версії мови програмування, яка ламає зворотну сумісність; потреба вивчення нових підходів розробки.

Зважаючи на переваги та недоліки, повна зворотна сумісність є безумовною перевагою на стадії зрілості, але загалом зменшує швидкість розробки та сповільнює розвиток індустрії загалом. Сумісність у межах версії є абсолютною протилежністю першого підходу, оскільки забезпечує підвищення швидкості розробки завдяки застосуванню нових підходів, навіть, якщо вони ламають зворотну сумісність. Сьогодні дедалі частіше використовують другий підхід, але з деякими обмеженнями: адаптацію програмного коду до нової версії мови програмування проводять тільки тоді, якщо це забезпечить кардинальне покращення продуктивності і швидкості розробки. Отже, результат розробки в межах однієї версії забезпечують переваги першого підходу, а перехід на нові версії у разі доцільності – переваги другого підходу. Швидкість адаптації безпосередньо не характеризує ЖЦ, але вказує наскільки довго буде існувати технологія.

Величина спільноти. За цією характеристикою можна оцінити кілька властивостей, якими володіє технологія: популярність, етап ЖЦ, перспективи. Щоб оцінити етап ЖЦ, потрібно поглянути на динаміку росту спільноти, якщо величина спільноти почала рости дуже повільно або взагалі скорочується, то технологія перейшла свій пік і знаходиться рівно на середині періоду зрілості. Вона ще буде популярна в межах зайнятої ніші до моменту поки кількісне відображення об'єму активної спільноти не почне різко скорочуватися. Це є відправна точка етапу старіння. Також характеристика відображає зацікавленість спеціалістів у

цій технології та її перспективи. Щоб оцінити перспективи, достатньо порівняти динаміку росту обраної технології та технології, що вже завоювала прихильність за період етапу молодості. Якщо динаміка схожа, то і перспективи теж схожі, оскільки однакові темпи росту перспективної технології та тої, що вже завоювала ринок, свідчать про можливість циклічного повторення успіху.

У ході проведення дослідження виділено параметри, які не характеризують технології ні з погляду етапу життєвого циклу, ні з погляду доцільності обрання технології на різних етапах існування, але безпосередньо впливають на тенденції розвитку. Такими параметрами є: швидкість освоєння, підтримка великими компаніями.

Швидкість освоєння є дуже критичним параметром, оскільки потреба у спеціалістах з розробки програмних продуктів зростає швидше, ніж освітні установи можуть достатньою мірою випустити кваліфікованих спеціалістів. Тому простота освоєння у багатьох випадках стає тим фактором, який забезпечує ріст спільноти, появу інноваційних засобів розробки, тривалий період існування. Складність розробки зростає швидше, ніж продуктивність програмістів. Велика кількість абстракцій, які були популярні під час розробок програмних додатків останніх 30 років, призвела до того, що багато паттернів проектування є прихованими, а значить незрозумілими. Це породжує складність у розумінні молодими спеціалістами того, як той чи інший механізм працює. В умовах дефіциту професіоналів – це проблема, на яку потрібно зважати в оцінці технології. Перехід багатьох спеціалістів з розробки програмного забезпечення за допомогою C++ на Java технологію відображає цю проблему. За абсолютної переваги C++ у плані продуктивності, вимогами до пам'яті, величиною спільноти та кількістю засобів розробки Java володіє трьома перевагами: швидкість розробки, легкість освоєння, кросплатформленість.

Підтримка великих компаній полягає у стрімкому зростанні на початкових стадіях, зосередження внутрішніх продуктів на використанні власної технології, що загалом її розвиває. Також істотний вплив на життєвий цикл технології відіграють кошти, які вкладаються у популяризацію. Великі компанії здійснюють підтримку мови програмування та зацікавлені у тривалому терміні її існування. Для таких мов, як Java, C#, Swift, Objective-C, підтримка батьківських компаній зіграла важливу роль в їх розвитку та позиції на ринку. Тому при схожості вихідних параметрів різних технологій підтримка компаній є істотною перевагою.

Висновки. Проаналізовано основні показники життєвого циклу технології. Встановлено закономірності притаманності певних показників відповідним етапам життєвого циклу технології. Запропоновано узагальнену класифікацію показників оцінки та перспективності використання програмних технологій. Здійснено теоретичне обґрунтування обраних показників.

Література

1. Великий тлумачний словник сучасної української мови (з дод. і допов.) / укл. і голов. ред. В.Т. Бусел. – Київ-Ірпінь : Вид-во ВТФ "Перун", 2005. – 1728 с.
2. Роганов Е.А. Основы информатики и программирования : учебн. пособ. / Е.А. Роганов. – М. : Изд-во МГИУ, 2001. – Вузол VI.

3. Popularity of Programming Language // GitHub. – 2016. [Electronic resource]. – Mode of access <http://www.pypl.github.io/PYPL.html>.
4. A small place to discover languages in GitHub // GitHub. – 2015. [Electronic resource]. – Mode of access <http://www.github.info>.
5. An Interview with the Creator of Ruby // O'Reilly Media. – 2001. [Electronic resource]. – Mode of access <http://www.linuxdevcenter.com/pub/a/linux/2001/11/29/ruby.html>.
6. Сейбел П. Кодеры за работой. Размышления о ремесле программиста / Питер Сейбел. – СПб. : Изд-во "Символ-Плюс", 2011. – 544 с.
7. TIOBE Index // TIOBE. – 2016. [Electronic resource]. – Mode of access http://www.tiobe.com/tiobe_index.
8. Креншоу Д. Давайте создадим компилятор! / Джек Креншоу. – М. : Изд-во "Наш мир", 2001. – 465 с.

Надійшла до редакції 16.05.2016 р.

Пасека Н.С., Турчин О.Б. Теоретическое обоснование показателей жизненного цикла технологий

Проанализированы основные показатели жизненного цикла технологий и выделена обобщенная система для оценки этапа жизненного цикла. Показатели рассмотрены как свойства, которыми должна обладать та или иная технология, чтобы соответствовать определенному этапу своего существования. Исследованы существующие подходы анализа развития технологии. Предложена обобщенная классификация, что представляет собой перечень показателей для соответствующего этапа развития. Обобщенная система оценки технологий позволит минимизировать технологические риски, связанные с краткосрочными и долгосрочными перспективами разработки программного обеспечения.

Ключевые слова: жизненный цикл, технология, параметры, этап развития, обобщенная классификация, технологические риски, язык программирования.

Paseka M.S., Turchyn O.B. Theoretical Explanation of Technologies Life Cycle Indicators

The basic indicators of the life cycle of technology were analyzed and the generalized system for the evaluation of the life cycle phase is allocated. Indicators are considered as properties that technology must have to fit a certain stage of its existence. The existing approaches of technology development analysis are studied. The generalized classification, which is a list of indexes for the proper stage of development, is proposed. The generalized technologies estimation system will enable to minimize technological risks related to the short-term and long-term software development prospects.

Keywords: life cycle, technology, stage of development, generic classification, technological risks, programming language.

УДК 004.9:530.1

МЕТОДИ ТА ЗАСОБИ ОПРАЦЮВАННЯ ВЕЛИКИХ ДАНИХ У СИСТЕМАХ ТЕРИТОРІАЛЬНОГО УПРАВЛІННЯ

Ю.Я. Болюбаш¹

Подано визначення Великих даних та описано основні характеристики. Проаналізовано математичні засоби подання та опрацювання Великих даних та визначено їх обмеження. Визначено формальний опис Великих даних. Подано моделі асоціацій між сутностями та характеристиками для різних категорій NoSQL баз даних. Розроблено метод обміну різнотипними даними та приведення реляційних даних до моделі "сутність-характеристика". Спроектовано схему даних регіону. Апробовано розроблені методи та алгоритми.

¹ Здобувач Ю.Я. Болюбаш – НУ "Львівська політехніка"