ЛЕКЦІЯ 5. УПРАВЛІННЯ ВАРТІСТЮ ПРОЕКТУ

План лекції:

1. Фундаментальні принципи і проблеми оцінки економічних параметрів програмних проектів.
2. Найпростіша формула розрахунку вартості проекту.
3. Розрахунок трудовитрат на основі хронологічних даних.
4. Одиниці оцінки розміру програмного забезпечення.
5. Кількість рядків коду при оцінці розміру програмного забезпечення.
6. Функціональні точки для оцінки розміру програмного забезпечення.
7. Метод точок властивостей при оцінці розміру програмного забезпечення.
8. Метод об’єктних точок при оцінці розміру програмного забезпечення.
9. Метод побудови бліц-моделі при оцінці обсягу проекту.
10. Методика Wideband Delphi при оцінці розміру програмного проекту.
11. Модель COCOMO для оцінки економічних параметрів програмних проектів.
12. Модель COCOMO II для оцінки економічних параметрів програмних проектів.
13. Модель SLIM для оцінки економічних параметрів програмних проектів.

Питання, що виносяться на самостійне вивчення студентом (4,5 год.):

1. Метричні показники в моделях SEI CMM/CMMI [7, С. 704-710].
2. Показники якості програмного забезпечення [3, С.145-162].

Показники надійності програмного забезпечення і методи доведення правильності програм [3, С. 168-178, 106-117; 7, С. 669-691].

5.1. Фундаментальні принципи і проблеми оцінки економічних параметрів програмних проектів

Побудова ефективної системи управління можлива лише за умови використання кількісних методів прийняття рішень, основаних на здійсненні економічної оцінки параметрів процесу розробки ПЗ.

Питання економічної оцінки параметрів процесу розробки ПЗ має високу актуальність та активно обговорюється у науковій літературі. В цілому, загальна риса досліджень, присвячених даним питанням, полягає в тому, що вони визнають надзвичайно високу складність отримання достовірних оцінок у порівнянні з аналогічними показниками сфери матеріального виробництва.

Згідно з даними SEI, близько 80% усіх систем кількісної оцінки процесу розробки ПЗ виявляються недієздатними протягом перших двох років використання.

5.2. Найпростіша формула розрахунку вартості проекту

У найпростішому випадку оцінити вартість розробки ПЗ можна виходячи з оцінки трудовитрат на розробку та вартості одиниці часу на розробку:

$$В = Т \times Ц \qquad (5.1),$$

де $В$ – вартість розробки ПЗ;

$Т$ – трудовитрати на розробку ПЗ (людино-міс);

$Ц$ – ціна одиниці трудовитрат.

Ціна одиниці трудовитрат звичайно формується з заробітної плати та нарахувань на заробітну плату, оскільки в сучасних умовах інші витрати є порівняно незначними, їх не приймають до уваги.

5.3. Розрахунок трудовитрат на основі хронологічних даних

Простий розрахунок трудовитрат оснований на хронологічних даних:

$$Т = Р \times П \qquad (5.2),$$

де T – трудовитрати на розробку ПЗ;

P – розмір ПЗ;

$П$ – хронологічна продуктивність.

5.4. Одиниці оцінки розміру програмного забезпечення

Для оцінки величини розміру ПЗ (Р) використовуються різні одиниці величин, які залежать від методики оцінки проекту та не можуть довільним чином бути перетворені між собою, оскільки між ними складно встановити однозначну тотожність.

Крім того, одиниці оцінки розміру ПЗ також передбачають використання відповідних методик для проведення оцінки, результатом виконання яких є певне значення, що вимірюється у відповідних одиницях.

Найбільш поширеними є наступні одиниці оцінки розміру ПЗ:

- кількість рядків коду (*Lines Of Code, LOC*);
- функціональні точки;
- точки властивостей;
- кількість «пухирців» на діаграмі потоку даних (Data Flow Diagram, *DFD*);
- кількість сутностей на діаграмі сутностей (Entity Relationship Diagram, *ERD*);
- кількість «квадратиків», що відповідають процесу/контролю на структурному графіку;
- кількість різних елементів у складі управлінської специфікації;
- обсяг документації;
- кількість об’єктів, атрибутів та служб на об’єктній діаграмі [74].

5.5. Кількість рядків коду при оцінці розміру програмного забезпечення

Кількість рядків коду (Lines Of Code, *LOC*; Source Lines of Code, *SLOC*) є найпростішою і найпоширенішою серед зазначених одиниць виміру. У загальному випадку *LOC* означає кількість рядків коду на відповідній мові

програмування, які мають бути написані для того, щоб проект був виконаний. На практиці звичайно використовується похідна від *LOC – KLOC (KSLOC),* що означає кількість тисяч рядків коду.

Оскільки різні мови програмування мають різні можливості і різну продуктивність по вирішенню задач програмування, то одиниці *LOC* мають бути приведені до певної співрозмірної величини. Приведення здійснюється з використанням таблиць перетворень, які дозволяють перевести кількість рядків коду більшості найбільш поширених мов програмування у еквівалентну кількість рядків коду мови асемблера, кожна строчка коду якої відповідає одній машинній команді. Такі таблиці періодично поновлюються, щоб враховувати еволюцію мов програмування.

5.6. Функціональні точки для оцінки розміру програмного забезпечення

Результати дослідження окремих вчених, зокрема К. Джонса встановили, що метрика *LOC* не може бути використана для ефективної оцінки проектів, виконаних з використанням сучасних об'єктно-орієнтованих мов програмування, поширеність яких останнім часом суттєво зростає.

Вирішити проблему обмеженості *LOC* по відношенню до сучасних мов програмування покликана інша одиниця оцінки розміру ПЗ – функціональні точки.

Головна ідея використання функціональних точок полягає в тому, що розмір ПЗ краще оцінювати в термінах кількості та складності функцій, а не кількості рядків коду.

Перша наукова робота, присвячена функціональним точкам, була опублікована в кінці 1970-х рр., її автором був А. Альбрехт, що представляв компанію IBM. Подальші дослідження були здійснені К. Джонсом, який суттєво розвинув дану ідею.

Створена в 1986 р. некомерційна група International Function Point User Group (IFPUG) займається розробкою і підтримкою стандартів, зокрема

IFPUG Standard Function Point Counting Practices Manual та IFPUG Standard Guidelines to Software Measurement.

При використанні методу функціональних точок здійснюється вимірювання категорій бізнес-функцій. На відміну від використання *LOC*, даний метод зосереджений на визначенні не абстрактного розміру ПЗ, а лише таких його показників, які визначають функціональність. Дана особливість дозволяє ефективно використовувати метод функціональних точок на етапі планування розробки.

Характерно, що метод функціональних точок має можливості для переведення кількості функціональних точок в одиниці *LOC*. Подібне переведення дозволяє порівнювати розмір проектів, оцінених за різними методиками, однак слід зазначити, що результатом переведення є величина, яка може зіставлятися для різних методик лише з урахуванням певної погрішності.

Зокрема, використання різних методик для оцінки розміру одного й того ж самого проекту, а потім приведення результату до одиниць *LOC* може дати значення, що суттєво відрізняються для різних методик.

Певним недоліком методу функціональних точок, що обмежує його використання, є порівняно висока складність даного методу. На відміну від оцінки кількості рядків коду, даний метод вимагає кваліфікованого підходу і порівняно значних зусиль при проведенні оцінки.

Також певним недоліком даного методу є його орієнтація на оцінювання бізнес-функцій. Існує певна категорія програмних продуктів, які мають порівняно простий набір входів/виходів, однак характеризуються значною внутрішньою складністю.

5.7.Метод точок властивостей при оцінці розміру програмного забезпечення

Подолати обмеженість методу функціональних точок в оцінюванні розміру складного ПЗ покликаний метод точок властивостей. Даний метод в цілому дуже подібний до методу функціональних точок, однак він більш

адаптований до високої складності алгоритмів, оскільки крім врахування параметрів функціональних властивостей ПЗ, він здатен оцінити складність його внутрішньої організації.

5.8.Метод об'єктних точок при оцінці розміру програмного забезпечення

Результатом адаптації методів функціональних точок і точок властивостей до особливостей об'єктно-орієнтованої технології розробки ПЗ є метод об'єктних точок, який дуже схожий до базових методів, однак при оцінці оперує об'єктними параметрами проекту, такими як класи і об'єкти.

5.9.Метод побудови бліц-моделі при оцінці обсягу проекту

Якщо для розробника ПЗ доступні хронологічні дані по виконаним проектам, то для здійснення оцінки розміру наступного проекту може бути використаний метод побудови бліц-моделі, розроблений Т. ДеМарко.

В основі процесу побудови бліц-моделі знаходиться припущення, що оцінити розмір проекту можна на основі хронологічних даних по проектам, які вже були закінчені. Бліц-модель може бути побудована на будь-якому етапі життєвого циклу проекту і оперувати показниками, що відповідають етапу циклу (наприклад, кількість «пухирців» на діаграмі потоку даних, кількість сутностей на діаграмі сутностей, кількість «квадратиків», що відповідають процесу/контролю на структурному графіку та ін.).

Спрощено побудова бліц-моделі відбувається як процес розрахунку показників розміру проекту, виходячи з відповідних показників раніше закінчених проектів, які коригуються відповідно до вже відомих характеристик проекту, що аналізується. Процес побудови можна здійснювати постійно протягом роботи над проектом і його точність буде зростати одночасно із прогресом у виконанні проекту.

Результативність методики побудови бліц-моделей збільшується при її системному використанні, оскільки вона передбачає можливість коригування параметрів на основі післяпроектного аналізу.

Суттєвим недоліком даної методики є її залежність від хронологічних даних. Вважається, що зростання кількості хронологічних даних за рахунок виконаних проектів підвищує точність методики. Однак, на нашу думку, використання хронологічних даних має свої обмеження, зокрема розвиток технологій і впровадження інновацій компанією-розробником ПЗ змінює процес розробки і самі поняття розміру і складності ПЗ для нього. Оцінка сучасного проекту на основі параметрів проектів, які були виконані з використанням застарілих технологій, втрачає смисл.

5.10. Методика Wideband Delphi при оцінці розміру програмного проекту

Іншим поширеним методом оцінювання розміру ПЗ є методика Wideband Delphi, розроблена як адаптація методики експертної оцінки Delphi до особливостей процесу розробки ПЗ.

В цілому Wideband Delphi дозволяє скоординувати дії експертів для проведення оцінки параметрів проекту точно також, як і при звичайній методиці Delphi. Єдина її суттєва відмінність полягає у тому, що завершальним етапом є групові дискусії експертів, що можуть покращити достовірність оцінки шляхом досягнення консенсусу між експертами.

Перевагою методики є її порівняна простота і можливість проведення оцінки без наявності хронологічних даних по виконаним проектам.

Суттєвим недоліком методики є можливість надання невірної оцінки, коли експерти поступаються своєю думкою і готові прийняти іншу думку без належних на те підстав. Крім того, існує ризик взагалі не досягти єдиної думки і, відповідно, не виробити єдиної оцінки. Важливим питанням також є відповідальний підбір експертів, оскільки, за умов неправильного підбору, група експертів може однобічно оцінювати проект і не враховувати усієї повноти діючих факторів.

5.11. Модель COCOMO для оцінки економічних параметрів програмних проектів

Як зазначалося раніше, лінійна модель припускає, що трудовитрати лінійно залежать від розміру ПЗ, оскільки можна припустити, що продуктивність роботи над проектом є сталою величиною, однак дане припущення було спростовано Б. Боемом, який розробив і опублікував у 1981 р. одну з найбільш популярних моделей оцінки витрат на розробку ПЗ, що отримала назву COCOMO (Constructive COst Model, Конструктивна модель витрат).

Модель COCOMO була розроблена на основі аналізу статистичних параметрів 63 проектів з розробки ПЗ різних типів. Параметрами, що підлягали оцінці були розмір ПЗ (*LOC*), понесені в процесі реалізації трудовитрати, а також фактична тривалість процесу розробки.

Фактично під загальною назвою COCOMO було розроблено три рівня моделі, які передбачають різний ступінь деталізації: базовий, проміжний та деталізований рівні.

Згідно з моделлю COCOMO виділяються також три режими її використання:

- органічний режим – невеликий проект розробляється невеликою командою, для якої не характерні нововведення, середовище розробки залишається стабільним;
- зблокований режим – порівняно невелика команда працює над проектом середнього розміру, в процесі розробки потрібні певні інновації, середовище розробки характеризується незначною нестабільністю;
- впроваджений режим – велика команда розробників працює над великим проектом, потрібен великий обсяг інновацій, середовище розробки складається із значної кількості елементів, які не характеризуються стабільністю.

Для оцінки трудовитрат на базовому рівні моделі COCOMO використовується наступна формула:

$$T = a \times P^{b} \qquad (5.3),$$

де T – трудовитрати на розробку ПЗ;

a, b – константи, які залежать від режиму моделі, що визначається типом проекту;

P – розмір ПЗ, KLOC.

Константи *a* і *b* встановлюються відповідно до режиму використання моделі COCOMO, який визначається типом проекту в залежності від його розміру і середовища розробки.

Значення коефіцієнтів *a* і *b*, що відповідають залежності режиму моделі COCOMO представлені у табл. 1.6.

Таблиця 5.1

Значення коефіцієнтів *a* і *b* в залежності від режиму моделі COCOMO

Назва режиму моделі COCOMO	**Значення коефіцієнту *a***	**Значення коефіцієнту *b***
Органічний	2,4	1,05
Зблокований	3,0	1,12
Впроваджений	3,6	1,20

Відповідно до даних табл. 5.1 значення коефіцієнтів *a* і *b* зростають при переході від органічного до впровадженого режимів моделі COCOMO, що, відповідно, означає експоненціальне зростання розміру трудовитрат при рівних значеннях розміру ПЗ.

Таким чином, модель COCOMO враховує нелінійність зростання трудовитрат на розробку відповідно до зростання розміру проекту.

Також модель враховує необхідність у проведенні інновацій, яка зростає із зростанням розміру проекту. Відповідно до зростання інтенсивності інновацій відбувається зростання і трудовитрат проекту.

Важливий висновок, який можна отримати з аналізу моделі COCOMO, полягає в тому, що зростання трудовитрат на реалізацію проекту при переході до іншого режиму моделі (в тому числі і викликане необхідністю впровадження інновацій) не означає відповідне зростання тривалості виконання проекту. Зокрема, розрахунок тривалості виконання проекту згідно моделі COCOMO виконується відповідно до формули:

$$F = 2{,}5 \times T^{k} \qquad (5.4),$$

де F – час, необхідний на виконання проекту;

T – трудовитрати на виконання проекту, розраховані згідно з формулою (5.2);

k – величина, що залежить від режиму моделі COCOMO і дорівнює для органічного режиму – 0,38, для зблокованого – 0,35, для впровадженого – 0,32.

Відповідно до формули (5.3) величина k зменшується при переході від органічного до впровадженого режиму і, відповідно, зменшується час, необхідний для виконання проекту за умов рівних трудовитрат.

Для проміжної моделі COCOMO формула (5.3) змінюється, до неї вводиться коефіцієнт, що враховує фактор коригування трудовитрат, формула отримує такий вигляд:

$$T = a \times P\,b \times C \qquad (1.5),$$

де T – трудовитрати на розробку ПЗ;

a, b – константи, які залежать від режиму моделі, що визначається типом проекту;

P – розмір ПЗ, KLOC;

C – фактор коригування трудовитрат.

Величина фактору коригування трудовитрат (C) визначається, виходячи з 15-ти показників, які враховують параметри програмного продукту, комп’ютерного забезпечення, персоналу та проекту.

Деталізований рівень моделі COCOMO відрізняється від проміжного тим, що весь проект розбивається на складові на основі ієрархії із трьох рівнів: система, підсистема модуль. Для кожного рівня ієрархії оцінка трудовитрат здійснюється окремо.

Також окремо оцінюються фази життєвого циклу по розробці проекту. Виділяється чотири основних фази: вимоги, розробка проекту продукту, деталізований дизайн продукту, кодування та тестування модулів. Такі складові життєвого циклу як інтеграція та тестування, а також підтримка описуються протягом всього життєвого циклу. Поділ на фази і оцінка кожної з них окремо може бути здійснена для всіх трьох рівнів ієрархії.

Враховуючи той факт, що модель COCOMO була створена на основі хронологічних даних, а значення її факторів і коефіцієнтів були встановлені емпіричним шляхом, то модель передбачає калібрування цих показників, для того, щоб більш точно відповідати особливостям компанії-розробника програмного забезпечення.

Однак для калібрування необхідні хронологічні дані мінімум по п’яти закінченим проектам, а тому цей процес можливий лише у тих умовах, коли компанія-розробник використовує кількісні методи управління процесом розробки і збирає статистичні дані протягом певного часу.

В цілому, модель COCOMO значно краще підходить для визначення трудовитрат на розробку проекту, ніж лінійна модель, однак вона має певні досить суттєві недоліки:

- по-перше, модель побудована емпіричним шляхом і орієнтована на певну категорію «типових» проектів, її не можна в оригінальному вигляді використовувати для проектів, які не підпадають під цю категорію, для таких проектів модель слід калібрувати, для чого потрібна відповідна статистика;
- по-друге, модель COCOMO орієнтована на каскадну модель життєвого циклу і передбачає певне співвідношення трудовитрат

між основними фазами: розробка проекту (30%), кодування (30%), інтеграція та тестування (40%);

- по-третє, модель не враховує певні фактори, що суттєвим чином впливають на процес розробки, зокрема, взаємодію з замовником і його рівень участі у виконанні проекту;
- по-четверте, модель не враховує науково-технічний прогрес, зміну середовища розробки і ріст продуктивності за рахунок впровадження інновацій, вона основана на певних хронологічних даних, які можуть бути застарілими по відношенню до сучасних проектів;
- по-п’яте, модель не охоплює весь життєвий цикл програмного забезпечення, її використання починається з проектування системи і закінчується інтеграцією та тестуванням, фази системного аналізу, розробки вимог, впровадження та супроводу залишаються поза увагою моделі.

Незважаючи на зазначені недоліки, модель COCOMO набула значної популярності і є однією з класичних методик, що використовуються при оцінці економічних параметрів проекту.

5.12. Модель COCOMO II для оцінки економічних параметрів програмних проектів

У другій половині 1990-х рр. була створена модель COCOMO II, що являє собою покращений варіант оригінальної моделі COCOMO, адаптований до сучасних методологій розробки ПЗ, а також придатний для використання зі спіральною та ітераційною моделями життєвого циклу.

Також модель COCOMO II, на відміну від багатофакторного регресійного аналізу у моделі COCOMO, використовує Бейесовський аналіз для обробки статистичних даних, який у більшому ступені відповідає особливостям статистичних даних програмних проектів, що характеризуються неповнотою і неоднозначністю.

У якості одиниці для визначення розміру ПЗ модель COCOMO II поряд із одиницями LOC використовує також вимірювання розміру за допомогою об’єктних точок.

Модель COCOMO II включає три різні моделі:

– композиційна прикладна модель – придатна для використання у проектах, що розробляються з використанням сучасних інструментальних засобів, основаних на об’єктно-орієнтованих технологіях;
– модель ранньої розробки проекту – придатна для приблизного оцінювання витрат на розробку проекту до того, як була визначена його архітектура;
– пост-архітектурна модель – найбільш деталізована модель, що використовується після розробки загальної архітектури проекту і включає нові правила оцінки розміру проекту, а також нові рівняння визначення трудовитрат порівняно до моделі COCOMO.

Модель COCOMO II, на відміну від оригінальної моделі COCOMO, при оцінці розміру ПЗ враховує використання і адаптацію сторонніх компонентів, що відповідає сучасному підходу до створення прикладного ПЗ, який в значній мірі орієнтований на використання сторонніх компонент.

Для оцінки розміру ПЗ (P) у моделі COCOMO II використовується наступна формула:

$$P = KNSLOC + \left[KASLOC \cdot \left(\frac{100 - AT}{100} \right) \cdot \left(\frac{AA + SU + 0{,}4 \cdot DM + 0{,}3 \cdot CM + 0{,}3 \cdot IM}{100} \right) \right] \quad (5.6),$$

де $KNSLOC$ – розмір модуля, тис. строчок коду;

$KASLOC$ – розмір адаптованих компонент, тис. строчок коду;

AT – доля компонент, що були автоматично трансльовані, %;

AA – доля зусиль на оцінку і асиміляцію модуля, %;

SU – розуміння внутрішньої будови модуля (нуль, якщо $CM = 0$ та $IM = 0$), %;

DM – доля дизайну, яка була модифікована, %;

CM – доля коду, яка була модифікована, %;

IM – доля інтеграції та тестів, що була модифікована, %.

При оцінці розміру трудовитрат модель COCOMO II враховує фактори масштабу, сумарна дія яких розраховується згідно з наступною формулою:

$$B = 0{,}91 + 0{,}01 \times \sum_{i=1}^{5} W_i \qquad (5.7),$$

де W_i – *і-й фактор масштабу (модель враховує 5 факторів).*

Фактори масштабу визначають експоненціальну залежність між розміром ПЗ та трудовитратами на його розробку. Фактори масштабу замінюють і розширюють поняття режимів для попередньої моделі COCOMO. Вони охоплюють такі питання, що впливають на трудовитрати процесу розробки, як спадкоємність по відношенню до попередніх проектів, гнучкість середовища розробки, систему виявлення і оцінки ризиків, згуртованість команди, а також зрілість процесу розробки у відповідності з моделями SEI CMM/CMMI.

Трудовитрати на розробку розраховуються згідно з формулою:

$$T = \prod_{i=1}^{17} (EM_i) \times A \times \left[\left(1 + \frac{BRAK}{100} \right) \times P \right]^B + \left(\frac{ASLOC \times \left(\frac{AT}{100} \right)}{ATPROD} \right) \qquad (5.8),$$

де *EM* – фактор, що модифікує трудовитрати (модель враховує 17 факторів);

A – константа, що визначається в процесі калібрування, її початкове значення становить 2,45;

BRAK – доля коду, що був забракований і не ввійшов у кінцевий продукт, %;

P – розмір ПЗ, розраховується згідно з формулою (5.6), тис. строчок коду;

B – сумарна дія факторів масштабу, розраховується згідно з формулою (5.7);

SF – фактори масштабу (модель враховує 5 факторів);

ASLOC – розмір адаптованого коду, тис. рядків коду;

ATPROD – продуктивність автоматичної трансляції коду;

AT – доля коду, що був автоматично трансльований, %.

Слід зазначити, що модель COCOMO II постійно оновлюється, зміні підлягають значення коефіцієнтів, які коригуються з урахуванням даних, отриманих з нових проектів, по яким була зібрана статистика. Зокрема, у варіанті моделі від 1997 р. стале значення формули (5.7) становило не 1,01, а 0,91.

Для розрахунку часу (F), необхідного для виконання проекту, у моделі COCOMO II використовується наступна формула [104]:

$$F = \left[3{,}67 \times T^{(0{,}28+0{,}2\times(B-1{,}01))}\right] \times \frac{SCED}{100} \qquad (1.9),$$

де T – *трудовитрати, розраховується у відповідністю з формулою (1.8);*

B – *сумарна дія факторів масштабу, розраховується згідно з формулою (1.7);*

SCED – *фактор, що визначає запізнення чи випередження графіку виконання проекту, %.*

Як видно з формули (1.9), її загальний вигляд у порівнянні з формулою (1.4) принципово не змінився, формула була доповнена новими коефіцієнтами і до неї був введений показник, що визначає вплив графіку виконання проекту. Однак в цілому формула (1.9) означає експоненціальну залежність між трудовитратами і часом, необхідним на виконання проекту.

Таким чином, модель COCOMO II вирішила основні недоліки моделі COCOMO, вона інтегрується з сучасними моделями розробки ПЗ, такими як спіральна та ітераційна, а також стандартами та методологіями, такими як SEI SW-CMM/CMMI та RUP. Модель враховує науково-технічний прогрес і ріст продуктивності компанії-розробника за рахунок факторів масштабу, що впливають на оцінку трудовитрат.

Основним недоліком нової моделі залишилася необхідність використання статистичних даних, зібраних на основі завершених проектів, які можуть не в повній мірі відповідати особливостям організації процесу компанії-розробника ПЗ. Хоча модель і передбачає калібрування параметрів, для здійснення даної процедури необхідно зібрати відповідні статистичні дані.

5.13. Модель SLIM для оцінки економічних параметрів програмних проектів

Іншою моделлю, яка набула популярності для оцінки економічних параметрів процесу розробки ПЗ, побудованою на основі емпіричних даних, є модель SLIM (Software Life-cycle Model, Модель життєвого циклу ПЗ).

Модель SLIM була розроблена в 1970-х рр. Л. Патнамом, який використав формулу Рейлайха по відношенню до життєвого циклу ПЗ.

Для встановлення зв'язку між розміром ПЗ та трудовитратами і обмеженням на строк поставки була використана наступна формула:

$$S = C \times K^{1/3} \times t_d^{4/3} \qquad (5.10),$$

де S – розмір ПЗ, строчок коду;

C – фактор середовища, що залежить від стану технології;

K – загальні трудовитрати для всього проекту;

t_d – обмеження на строк поставки (графік), років.

Фактор середовища (C) може бути розрахований за наступною формулою:

$$C = \frac{S}{K^{1/3} \times t_d^{4/3}} \qquad (5.11),$$

де S, K, t_d – значення розміру, трудовитрат та обмеження на строк поставки для попередніх проектів.

Для визначення розміру ПЗ (S) у моделі SLIM можуть бути використані різні підходи, а сам розмір може бути визначений не лише як кількість

строчок коду, а як і інші одиниці виміру, такі як, наприклад, функціональні точки.

Один із рекомендованих підходів для визначення розміру ПЗ у моделі SLIM полягає у використанні бета-розподілу:

$$S_n = (S_{min} + 4S_i + S_{max}) / 6 \qquad (5.12),$$

де S_n – прогнозне значення для номінального розміру ПЗ;

S_{min} – мінімально можливий розмір;

S_i – найбільш ймовірний розмір;

S_{max} – максимально можливий розмір.

Якщо відомі значення розміру, обмеження на строк поставки продукту та визначений фактор середовища, розрахунок трудовитрат можна здійснити наступним чином (виходячи з формули 5.10):

$$K = \left(\frac{S}{C}\right)^3 \times \frac{1}{t_d^4} \qquad (5.13).$$

Фактор середовища (C) моделі SLIM враховує також і продуктивність персоналу, яка може суттєво відрізнятися як для різних проектів, так і для різних компаній, що працюють над проектами подібної складності. Врахування продуктивності персоналу було одним з основних відмінностей моделі SLIM від моделі COCOMO, до тих пір, поки не була розроблена модель COCOMO II.

Незважаючи на значні досягнення у побудові математичних параметричних моделей останнього часу, проведене на початку 2000-х рр. канадськими вченими дослідження доцільності використання оцінки складності проекту як величини, що може бути самостійно використана для оцінки його тривалості, показало, що подібну оцінку з прийнятним рівнем вірогідності здійснити дуже складно.

Тому для оцінки економічних параметрів ПЗ активно використовуються також підходи, орієнтовані на експертну оцінку, метод аналогій, побудову нейронних мереж та ін.