ЛЕКЦІЯ 4:

МЕТРИКА ЯК ОСНОВА ВИМІРЮВАННЯ. КІЛЬКІСНЕ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ

NAU

Дишлевий О.П.

- Кількісне забезпечення якості (Частина 1)
  - Зворотній зв’язок в якості (загальний механізм)
  - Моніторинг та вимірювання
  - Аналіз та зворотній зв’язок
  - Застосування інструментів та забезпечення впровадження ПЗ
- Метрика як основа вимірювання (Частина 2)
  - Поняття “метрика”
  - Види метрик якості
  - Ключові метрики для контролю розробки ПЗ

Важливість зворотного зв'язку

Всі заходи якості потребують додаткової підтримки:

- Планування та постановка цілей
- Управління з допомогою

зворотного зв'язку:

- Коли зупинитися?
- Коригування і вдосконалення

і т.д.

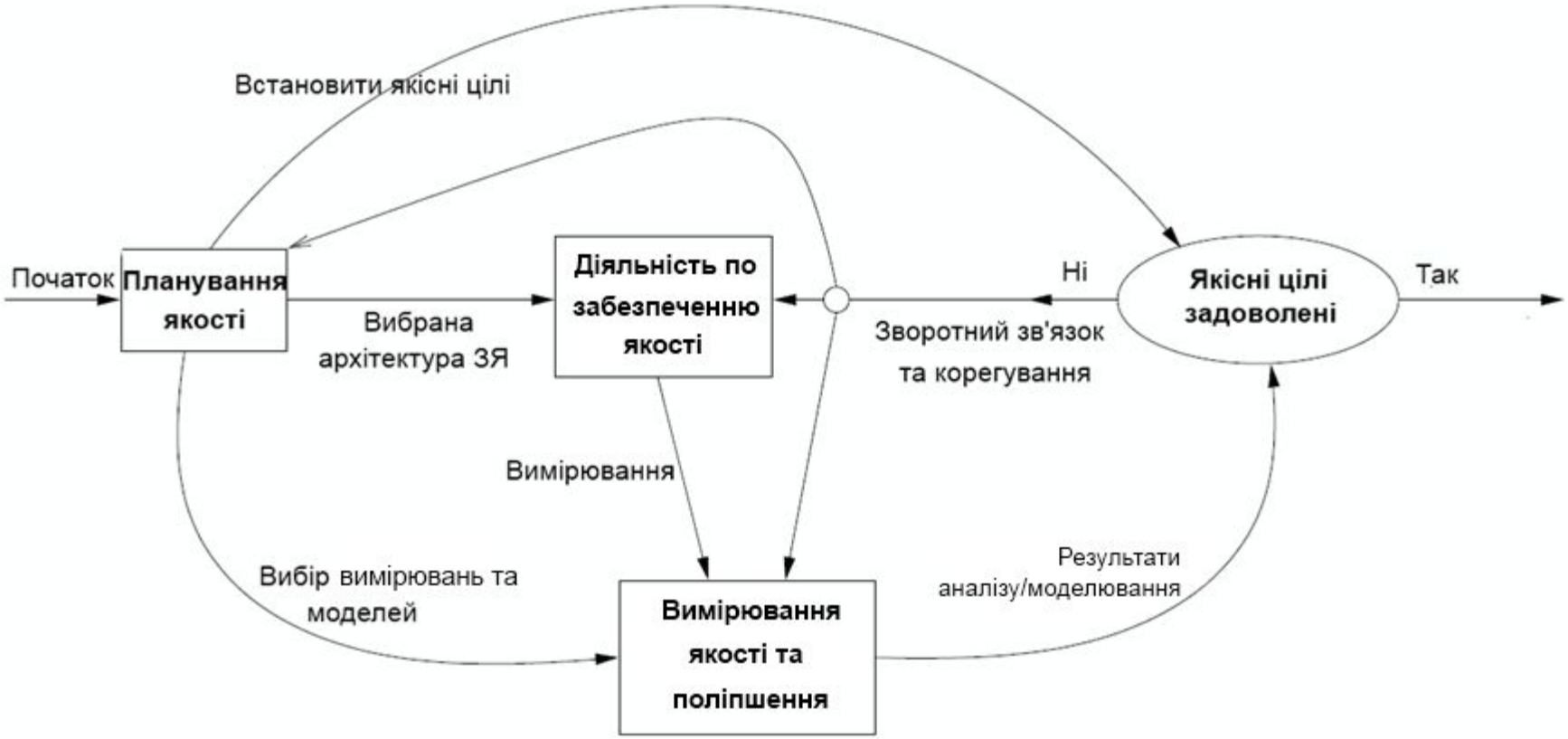
- Все, засноване на оцінках / прогнозах

ЗЯ(QA) діяльності та процес огляду

Основні напрями діяльності:

- Попереднє планування
- Забезпечення якості
- Пост-аналіз і зворотній зв'язок (Можливо, паралельно замість “пост

Інженерія якості програмного забезпечення (SQE)



Діяльність по забезпеченню якості та процес огляду

Діяльність пов’язана зі зворотним зв’язком

- Моніторинг та вимірювання:
  - Моніторинг дефектів ∈ управління процесами.
  - Вимірювання дефектів ∈ обробка дефектів.
  - інші пов'язані вимірювання.
- Моделювання аналізу:
  - Випадки з історії та досвід.
  - Вибір моделей і методик аналізу.
  - Зосередження на аналізах дефектів / ризику / надійності.
  - Мета: оцінка / прогноз / поліпшення.
- Зворотний зв'язок та відсдлідковування:
  - Частота зворотного зв'язку: оцінки / прогнози.
  - Визначення можливих областей розвитку.
  - Загальне управління і розвиток.

Моніторинг якості та вимірювань

Потреби контролю якості:

- Якість як визначення кількісних характеристик протягом часового періоду.
- Здатність оцінювати, прогнозувати і контролювати.
- Необхідні випадково вимірювані дані.
- Відслідковування якості шляхом отримання “прямих” (безпосередніх) даних.
- Інші дослідження для досягнення зворотного зв’язк

Прямі вимірювання якості:

- Результат, наслідки та пов'язана з ними інформація
  - Наприклад, успіх та відмови
  - Класифікація інформації. (Наприклад, ODC)
- Інформація про дефекти: моніторинг.
  - Додатковий аналіз дефектів.
- В основному використовується в контролі якості.

Непрямі вимірювання якості

- Непрямі вимірювання якості: Чому потрібні?
  - Вимірювання якості (надійності) потрібують додаткового аналізу / даних.
  - Відсутність прямого вимірювання якості на ранніх етапах циклу розробки ⇒ ранні (непрямі) показники.
  - Використовується для оцінки / прогнозування / контролю якості.
- Типи непрямих вимірювань якості:
  - Вимірювання, пов’язані з середовищем.
  - Внутрішні вимірювання продукту.
  - Вимірювання діяльності.

Непрямі вимірювання: середовище

- Характеристики процесів
  - Сутності і зв'язки
  - Підготовка, проведення і завершення
  - Використовувані методики
- Характеристики людей
  - Навики та досвід
  - Ролі: планувальники / розробники / тестери
  - Управління процесами і команди
- Характеристики продуктів
  - Середовище продукту / ринку
  - Машинне/ програмне середовище

Непрямі вимірювання: внутрішні

- Внутрішні вимірювання продукту:найбільш вивчені/ зрозумілі в ІПЗ
- Артефакти ПЗ вимірюються:
  - Переважно код
  - Іноді ресурси, дизайн (проект), документи і т.д.
- Атрибути продукту вимірюються:
  - Управління: наприклад, складність McCabe
  - Дані: наприклад, метрики Halstead
  - Типографія (представлення даних): наприклад, правила відступів
- Структури:
  - Неструктуровані: наприклад, LOC
  - Структуровані: наведені вище приклади

Непрямі вимірювання: діяльність

- Вимірювання виконання/активності:
  - Загальне: наприклад, час циклу, загальні зусилля.
  - Поетапне: профільні дані/ гістограма.
  - Деталізоване: транзакції в SRGMs.
- Приклади діяльності по тестуванню:
  - Синхронізація під час тестування / використання
  - Перевірка шляху(білого ящика)
  - Відображення використання компоненту(чорний ящик)
  - Вимірювання по шляху
- Використання спостереження / вимірювання: засновані на спостереженні і прогнозних моделей

Безпосередній супровід і зворотній зв'язок

- Негайне (без аналізів): Чому потрібне?
  - Термінові заходи необхідні саме зараз:
    - Критична проблема ⇒ негайне виправлення
    - Більшість інших проблем: не потрібно чекати
  - Деякі види зворотного зв'язку, як вбудовані різні функції ЗЯ в якості альтернатив і методів.
  - Діяльність, пов'язана з негайним діям.
- Приклади тестування:
  - Зміщення акценту з невдалого запуску/області.
  - Повторний тест для перевірки виправленого дефекту.
  - Інші регулювання пов’язані з дефектами.
- Використання дефектів та вимірювань.

Аналіз, зворотній зв'язок, і супровід

- Більшість зворотних зв'язків / супроводу спирається на аналіз.
- Типи аналізу:
  - Пов’язані з рішенням про випуск продукту.
  - Ті, що стосується інших рішень управління проектами, на певних етапах або на загальному рівні проекту.
  - Довгостроковий або розширений аналіз.
- Типи шляхів зворотного зв'язку:
  - Коротші чи довші петлі зворотного зв'язку.
  - Частота та тривалість відхилень.
  - Повне охоплення зворотного зв'язку.
  - Деталізація джерел даних.
  - Напрямки зворотного зв’язку.

Аналіз для рішення випуску продукту

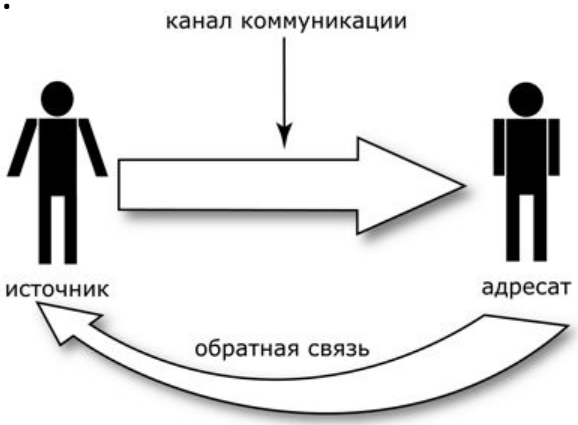
- Найважливіші використання аналізу результатів пов'язані з тим “коли зупинити тестування?”
- Основа для прийняття рішень:
  - Без явної оцінки якості:
    - Неявна: плановані заходи,
    - Непряма: охоплення цілей,
    - Інші фактори: часоорієнтовані / орієнтовані на кошти.
  - При явному оцінюванні якості:
    - Орієнтовані на відмову: надійність,
    - Орієнтовані на дефект: обрахунок дефектів і щільність.
- Критерії переваги: визначена надійність – дефекти – покриття – діяльність.

Аналіз для інших рішень

- Перехід від однієї фази в іншу:
  - Пізніша: за аналогією з випуском продукту.
  - Раніша за часом: невизначена надійність
    - Дефекти – покриття – діяльність,
    - Інспекції та інше ЗЯ
- Інші заходи прийняття рішень/управління:
  - Регулювання графіку.
  - Розподіл ресурсів і регулювання.
  - Планування супроводу після релізу.
  - Планування майбутніх продуктів і оновлень.
- Це діяльність та рішення рівня або підрівня продуктів.

Інший зворотний зв’язок та супровід

- Інший (рідший) зворотній зв'язок / супровід:
  - Управління метою (виправдання/ схвалення).
  - Особисто-зворотний (в компаніях) зв'язок (вимірювання та аналіз)
    - Непридатні вимірюваня і моделі?
    - SRE вимірювання наприклад, в IBM.
  - Довгостроковий, зворотній зв’язок на рівні проекту.
  - Може навіть перенестись на наступні проекти.
- Тривалість/ область за межами одного прое
  - Майбутні поліпшення якості продукції
    - Загальної цілі / стратегії / моделі / дані,
    - Особливо для запобігання дефектів.
  - Процес поліпшення.
  - Більш досвідчені люди.



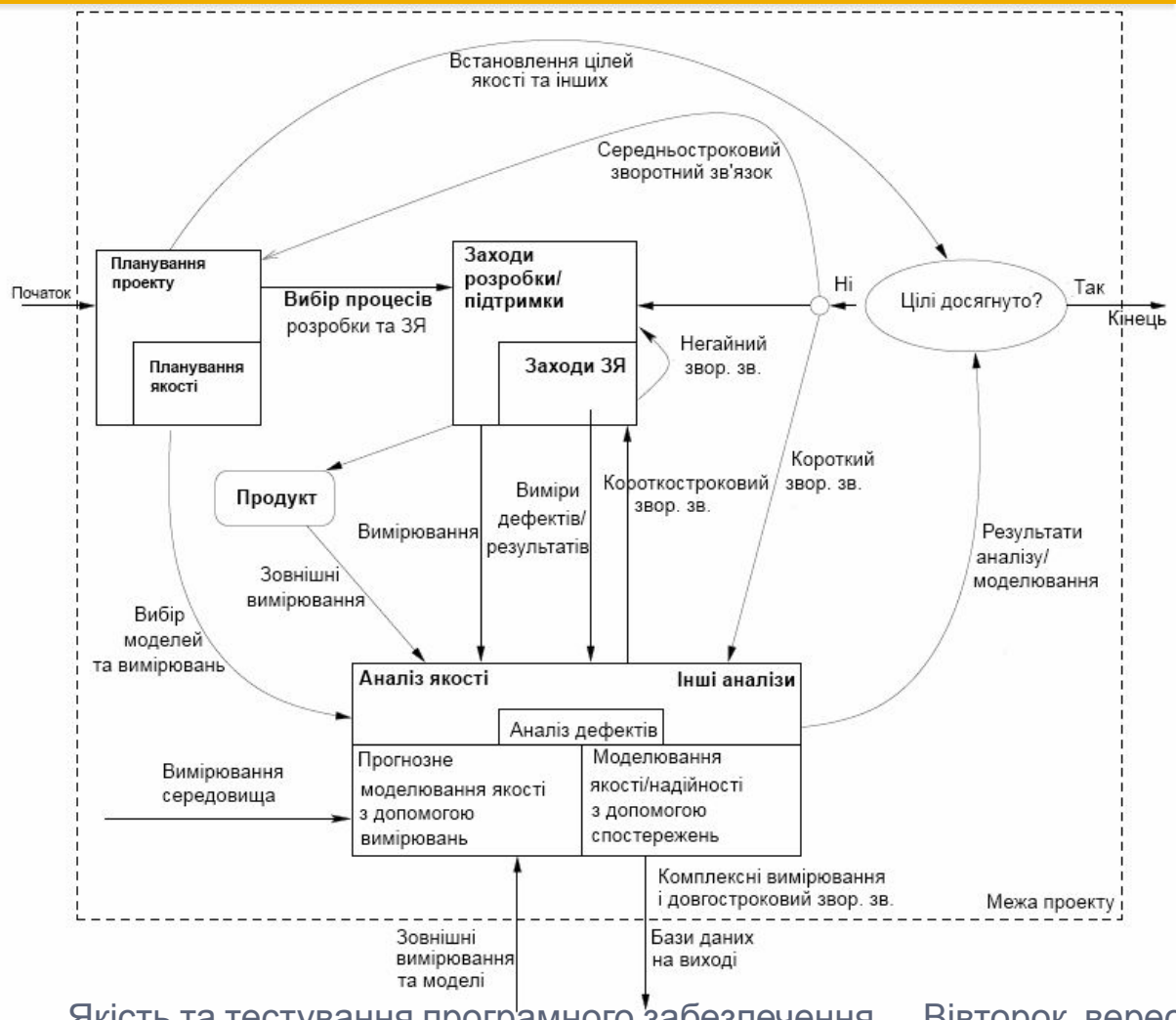
Здійснення зворотного зв'язку

- Ключове питання: джерела та кінцеві точки. (Аналіз та моделювання діяльності взагальному.)
- Джерела зворотного зв'язку = джерела даних:
  - Результат і дефект даних:
    - Діяльність ЗЯ сама по собі.
  - Дані про діяльність:
    - Дії з забезпечення якості та розробки.
  - Внутрішні дані: продукт. (роботи по розробці)
  - Дані: середовище.
- Додаткові джерела зворотного зв'язку:
  - Від проекту / планування ЗЯ.
  - Розширене середовище: дані вимірювань і моделі замежами проекту.

Здійснення зворотного зв'язку

- Петля зворотного зв'язку на різних рівнях тривалості /обсягу
- Безпосередній зворотній зв'язок від поточної діяльності розробки (локально).
- Короткостроковий або суб-проектний рівень зворотного зв'язку:
  - перехід, графік, ресурс,
  - призначення: діяльності розробки.
- Середньостроковий або зворотний зв’язок проектного рівня:
  - загальне коригування проекту і випуск
  - призначення: основні блоки на рис. Наступного слайду
- Довгостроковий або мультипроектний зворотний зв'язок:
  - До зовнішньої (за межами проекту) мети

Здійснення зворотного зв'язку



Засоби підтримки реалізації

- Тип засобу:
  - Інструменти збору даних.
  - Інструменти аналізу та моделювання.
  - Інструменти презентації.
- Інструменти збору даних :
  - Дефекти / прямі вимірювання якості:
    - З інструментів відстеження дефектів.
  - Дані середовища: БД проекту.
  - Вимірювання активності: журнали (log).
  - Внутрішні виміри продукту:
    - Комерційні/ домашні інструменти.
  - Нові інструменти / API можуть бути необхідними.

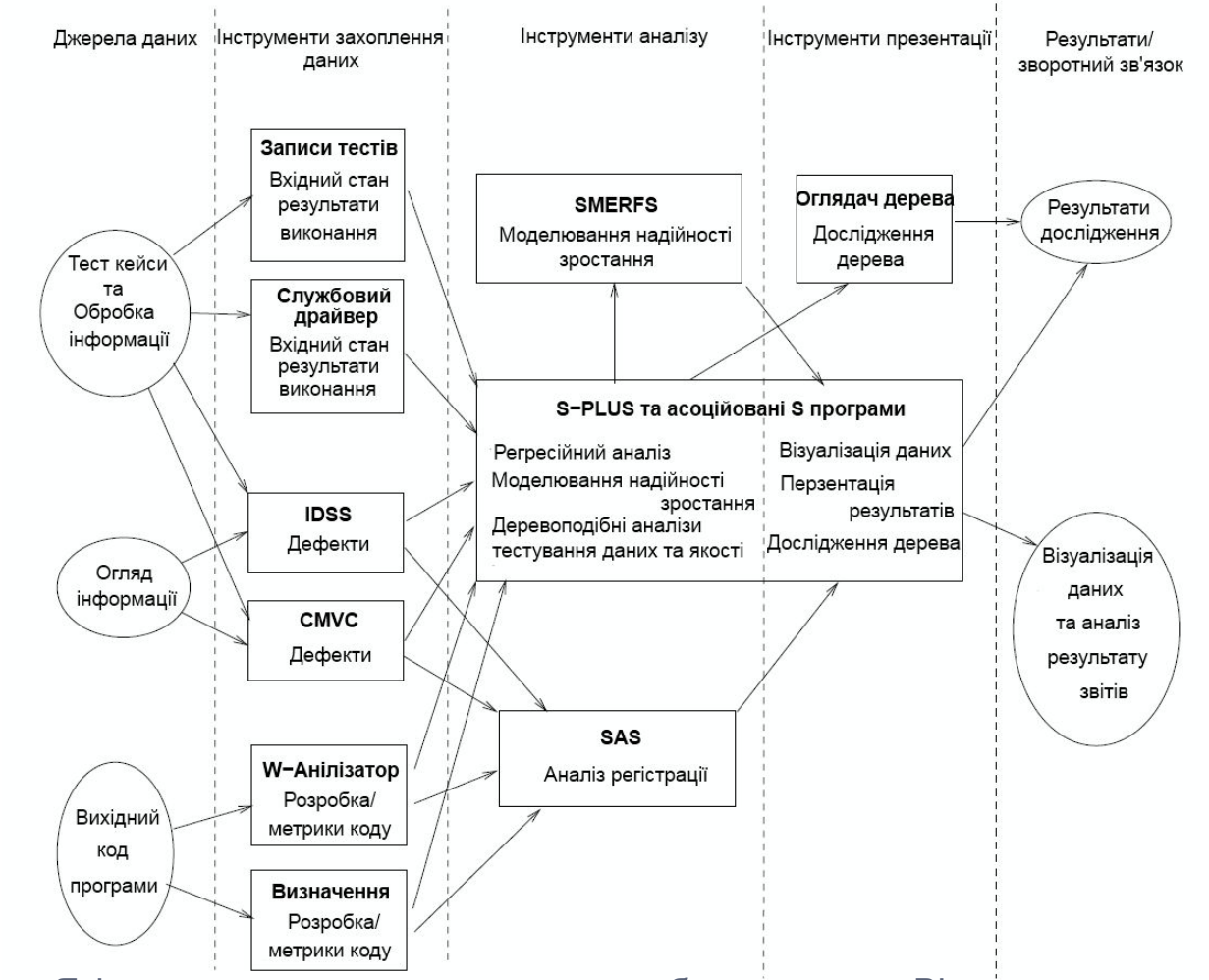
Засоби підтримки реалізації

- Інструменти аналізу та моделювання:
  - Присвячені для моделювання:
    - Наприклад, SMERFS і CASRE для SRE
  - Загальні інструменти/пакети моделювання:
    - Наприклад, багатоцільові S-Plus, SAS.
  - Програми-утиліти часто потрібні для перегляду та обробки даних.
- Інструменти презентації:
  - Мета: легка інтерпретація зворотного зв'язку ⇒ Найбільше схоже що над ним працюють.
  - Переважно графічне представлення.
  - Деякі “що-якщо "/ дослідницькі можливості.

Стратегія для підтримки інструменту

- Використання існуючих інструментів ⇒ Вартість ↓:
  - Функціональність та корисність/ вартість.
  - Зручність використання.
  - Гнучкість і програмування.
  - Інтеграція з іншими інструментами.
- Приклади інструменту інтеграції:
  - Прогнозування: використовуються кілька інструментів. (Інструменти загального призначення не практичні/підходящі)
  - Зовнішні правила сумісності,
    - Загальний формат даних і сховища.
  - Багатоцільовий інструмент.
  - Програми для сумісності.

Приклад інструменту підтримки



Метрика як основа вимірювання (Частина 2)

Метрика програмного забезпечення

Метрика програмного забезпечення (англ. *software metric*) – це міра, яка дозволяє отримати числову оцінку деякої властивості ПЗ або його специфікацій.

У загальному випадку застосування метрик дозволяє керівникам проектів і підприємств вивчити складність розробленого або навіть проекту, що розробляється, оцінити обсяг робіт, стилістику програми і зусилля,

витрачені кожним розробником для

реалізації того чи іншого рішення.

Метрика програмного забезпечення

Метрики складності програм прийнято поділяти на три основні групи:

- метрики розміру програм;
- метрики складності потоку управління програм;
- метрики складності потоку даних програм.

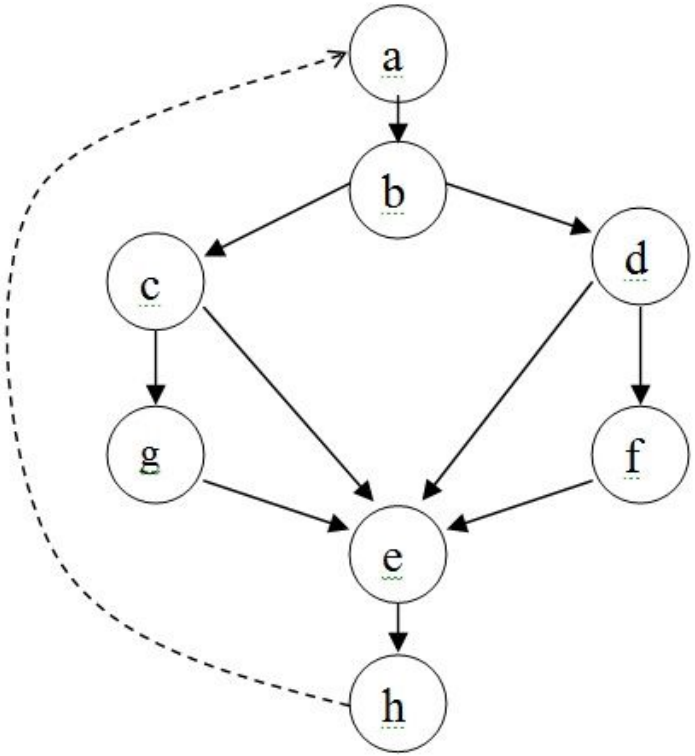
Метрики розміру програм

Метрики першої групи базуються на визначенні кількісних характеристик, пов'язаних з розміром програми, і відрізняються відносною простотою

До найбільш відомих метрика цієї групи відносяться кількість операторів програми, кількість рядків вихідного тексту, набір метрик Холстеда. Метрики цієї групи орієнтовані на аналіз вихідного тексту програм, тому вони можуть використовуватися для оцінки складності проміжних продуктів розробки.

Метрики складності потоку управління програм

Метрики другої групи базуються на аналізі керуючого графа програми. Представником цієї групи є метрика Маккейба. Керуючий граф (блок-схема програми), який використовують метрики даної групи, може бути побудований на основі алгоритмів модулів. Тому метрики другої групи також можуть застосовуватися для оцінки складності проміжних продуктів розробки.



Метрики складності потоку даних програм

Метрики третьої групи базуються на оцінці використання, конфігурації і розміщення даних у програмі. У першу чергу це стосується глобальних змінних. До цієї групи відносяться метрики

Loc- оцінки якості

Розмірно-орієнтовані метрики прямо вимірюють програмний продукт і процес його розробки. Ґрунтуються такі метрики на LOC-оцінках.

Цей вид метрик побічно вимірює програмний продукт і процес його розробки. При цьому замість підрахунку LOC-оцінок розглядається не розмір, а функціональність або корисність продукту. В практиці створення програмного забезпечення найбільше поширення отримали розмірно-орієнтовані метрики.

Loc- оцінки якості

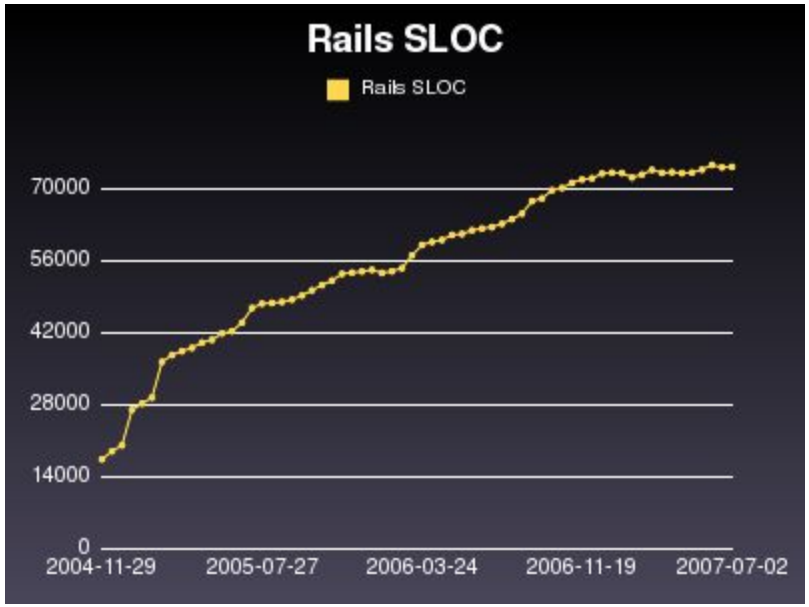
В організаціях, зайнятих розробкою програмної продукції для кожного проекту прийнято реєструвати наступні показники:

- загальні трудовитрати (в людино-місяцях, людино-годинах);
- обсяг програми (в тисячах рядках вихідного коду-LOC);
- вартість розробки;
- обсяг документації;
- помилки, виявлені протягом року експлуатації;
- кількість людей, які працювали над виробом;
- термін розробки.

SLOC

Виділяють два основні показники SLOC:

- кількість «фізичних» рядків коду - - визначається як загальне число рядків вихідного коду, включаючи коментарі і порожні рядки.



- кількість «логічних» рядків коду - визначається як кількість команд і залежить від мови програмування.

SLOC

Для метрики SLOC існує велике число похідних метрик, покликаних отримати окремі показники проекту, основними серед яких є:

- кількість порожніх рядків;
- число рядків, що містять коментарі;
- відсоток коментарів (відношення рядків коду до рядків коментаря, похідна метрика стилістики);
- середнє число строк для функцій (класів, файлів);
- середнє число рядків, що містять вихідний код для функцій (класів, файлів);
- середнє число строк для модулів.

Недоліки SLOC

Потенційні недоліки SLOC, на які орієнтована критика:

- Некрасиво і неправильно зводити оцінку роботи людини до декількох числовим параметрами і по них судити про продуктивність.
- Метрика не враховує досвід співробітників та їх інші якості
- Спотворення: процес вимірювання може бути спотворений за рахунок того, що співробітники знають про вимірюваних показниках і прагнуть оптимізувати ці показники, а не свою роботу.
- Неточність: ні метрик, які були б одночасно і значущі і досить точні.

Метрики стилістики й зрозумілості і складності

Метрика зрозумілості може бути розрахована за формулою:

Fi = SIGN (Nкомм. i / Ni - 0,1)

Суть метрики проста: код розбивається на n-рівні шматки і для кожного з них визначається Fi

Крім показників оцінки обсягу робіт за проектом дуже важливими для отримання об'єктивних оцінок за проектом є показники оцінки його складності. Разом з тим, ці показники дуже важливі для отримання прогнозних оцінок тривалості і вартості проекту, оскільки безпосередньо визначають його трудомісткість.

Об’єктно-орієнтовані метрики

- **Зважена насиченість класу (Weighted Methods Per Class (WMC)**. Відображає відносну міру складності класу на основі циклічної складності кожного його методу.
- **Зважена насиченість класу 2 (Weighted Methods Per Class (WMC2))**. Міра складності класу, заснована на тому, що клас з великою кількістю методів, є більш складним, і що метод з великою кількістю параметрів також є більш складним.
- **Глибина дерева успадкування (Depth of inheritance tree)**. Чим глибше дерево успадкування модуля, тим може бути складніше передбачити його поведінку.

Об’єктно-орієнтовані метрики

- **Кількість нащадків (Number of children)**. Число модулів, безпосередньо успадковують даний модуль.
- **Зв'язність об'єктів (Coupling between objects)**. Кількість модулів, пов'язаних з даним модулем в ролі клієнта або постачальника.

- **Відгук на клас (Response For Class)**. Кількість методів, які можуть викликатися екземплярами класу; обчислюється як сума кількості локальних методів, так і кількості вилучених методів.

Метрики Холстеда

Метрика Холстед відноситься до метрик, що обчислюються на підставі аналізу числа рядків і синтаксичних елементів початкового коду програми. Основу метрики Холстеда складають чотири вимірювані характеристики програми:

- NUOprtr (Number of Unique Operators) - число унікальних операторів програми, включаючи символи-роздільники, імена процедур і знаки операцій (словник операторів);
- NUOprnd (Number of Unique Operands) - число унікальних операндів програми (словник операндів);
- Noprtr (Number of Operators) - загальна кількість операторів в програмі;
- Noprnd (Number of Operands) - загальна кількість операндів в програмі.

Метрики циклічної складності за Мак-Кейбом

Даний показник був розроблений вченим Мак-Кейбом в 1976 р., належить до групи показників оцінки складності потоку управління програмою і обчислюється на основі графа керуючої логіки програми (control flow graph). Даний граф будується у вигляді орієнтованого графа, в якому обчислювальні оператори або вирази представляються у вигляді вузлів, а передача управління між вузлами - у вигляді дуг.

Спрощена формула обчислення циклічної складності представляється наступним чином:

C = e - n + 2, де e - число ребер, а n - число вузлів у графі керуючої логіки.

Метрики циклічної складності за Мак-Кейбом

Цикломатичне число Мак-Кейба показує необхідну кількість проходів для покриття всіх контурів сильно зв’язаного графу або кількості тестових прогонів програми, необхідних для вичерпного тестування за принципом «працює кожна гілка».

Існує значна кількість модифікацій показника циклічної складності:

- модифікована цикломатична складність - розглядає не кожне розгалуження оператора множинного вибору (switch), а весь оператор як єдине ціле;
- строга цикломатична складність - включає логічні оператори;
- спрощена цикломатична складність - передбачає обчислення не на основі графа, а на основі підрахунку керуючих операторів.

Метрика Чепіна

Суть методу полягає в оцінці інформаційної міцності окремо взятого програмного модуля за допомогою аналізу характеру використання змінних зі списку вводу-виводу.

Вся множина змінних, що складають список вводу-виводу, розбивається на чотири функціональні групи.

1. Множина «P» - змінні, що вводяться для розрахунків та для забезпечення виведення. Прикладом може служити змінна, яка використовується в програмах лексичного аналізатора, що містить рядок вихідного тексту програми, тобто сама змінна не модифікується, а лише містить вихідну інформацію.

2. Множина «M» - змінні, що модифікуються або створюються усередині програми.

3. Множина «C» - змінні, що беруть участь в управлінні роботою програмного модуля (керуючі змінні).

4. Множина «T» - змінні, які не використовуються в програмі ("паразитні").

Метрика Чепіна

Далі вводиться значення метрики Чепіна:

Q = a1*P + a2*M + a3*C + a4*T, де a1, a2, a3, a4 - вагові коефіцієнти множин.

Вагові коефіцієнти використані для відображення різного впливу на складність програми кожної функціональної групи.

З урахуванням вагових коефіцієнтів вираз набуде вигляду:

$$Q = P + 2M + 3C + 0.5T.$$

Аналіз ефективності використання метрик

Метрика	Навіщо потрібна	Впливає на…	Аналіз на основі статистичних даних (як тренд, так і прогноз))
Зусилля розробника при реалізації	Для оцінки ефективності роботи розробника	Точність прогнозів оцінки трудомісткості при виконанні організацією типових запитів або запитів , які мало відрізняються	Можна аналізувати зусилля розробника у тимчасовому зрізі або у зрізах по релізах або проектах. Виявляти, на яких завданнях програміст повністю викладається, а які йому не до душі. Тренд дозволить менеджеру краще розуміти, хто і у яких завданнях максимально ефективний при формуванні команди нового проекту, а також які підсистеми складні, а які - прості
Довжина та об’єм програми		Оцінку об’єму змін	Збільшується чи зменшується об’єм програм в часі. Використовується для прогнозу складності на ранніх етапах розробки з допомогою статистики

Аналіз ефективності використання метрик

Аналіз цикломатичної сладності		Оцінку складності змін	Аналізується ріст чи зниження складності. Використовується для прогнозу складності на ранніх етапах розробки з допомогою статистики
Зусилля програмістів при розробці	Для визначення реалізації того чи іншого блоку коду (класу, функції ш т.п.	Розуміння того, на скільки інтелектуально затратною для розробника була та чи інша функція	Аналізується ріст чи зниження в часі. На попередніх етапах метрику можна використовувати для прогнозу
Кількість рядків для реалізації вимоги	Вимірювання загального стану програми, приймається до уваги при аналізі реалізації запитів	Розуміння коефіцієнта корисної дії, виявляються викиди	Сигнал небезпеки при зростанні типового запиту. Використовується для прогнозу складності на ранніх етапах розробки з допомогою статистики.
Аналіз цикломатичної сладності		Оцінку складності змін	Аналізується ріст чи зниження складності. Використовується для прогнозу складності на ранніх етапах розробки з допомогою статистики

Аналіз ефективності використання метрик

Кіль кість коментарів на одиницю	Код повинен бути прокоментований. Якщо відношення об'єму коду до об' єму коментарів більше1/4, програму потрібно доопрацювати	Якість коду, його прозорість	Аналізується ріст чи зниження загальної культури розробки в часі. Якщо зміни стрибкоподібні, співвідносяться стрибки з менеджерами проектів. Виділяється складні проекти, проблемні модулі чи підсистеми
Інші кількісні (число функцій, класів, файлів)	Відношення нових функцій до змінених	Кількість доданих, видалених та змінених рядків відносно попередньої версії	Глибокий аналіз змін у релізах (версіях, збірках). Дає зрозуміти кількість змін і коригувань коду, вузькі місця в програмі, які визначаються кількістю змін у блоках, що може впливати на загальну якість програми – можливо потрібна зміна архітектури блоку.
Щільність дефектів на одиницю коду	Кількість дефектів на один рядок коду	Похідна метрика – кількість рядків/кількість дефектів	Аналізується ріст чи зниження щільності дефектів від версії до версії. Виявляються критичні підсистеми з великою щільністю дефектів. У цьому випадку щільність, звичайно, корелюється з метрикою

Попередня оцінка якості

Статистичний метод добре підходить для вирішення подібних типових завдань і практично не підходить для прогнозу унікальних проектів. У випадку унікальних проектів застосовуються інші підходи, обговорення яких виходить за рамками даного матеріалу.

Виділимо типові етапи розробки ПЗ:

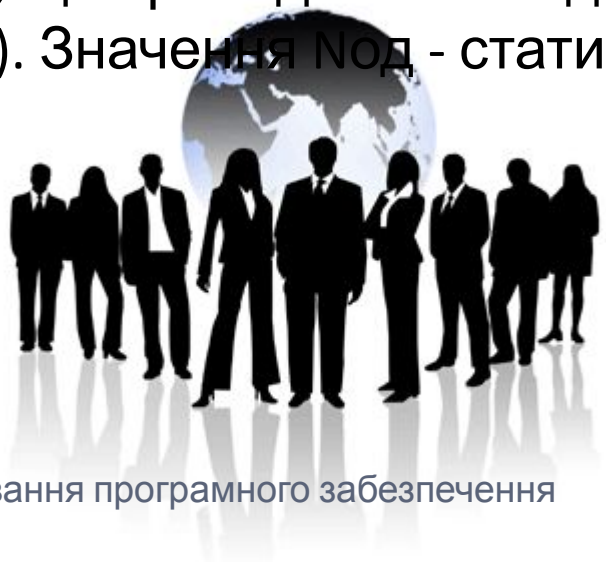
- розробка специфікацій вимог;
- визначення архітектури;
- опрацювання модульної структури програми, розробка інтерфейсів між модулями.
- опрацювання алгоритмів;
- розробка коду і тестування.

На етапі розробки специфікацій вимог до програми

Для оцінки результатів роботи даного етапу може бути використана метрика прогнозованого числа операторів програми:

Nпрогн = NF * Nод, де NF - кількість функцій чи вимог у специфікації вимог до програми, що розробляється;

Nод - одиничне значення кількості операторів (середня кількість операторів, що припадають на одну середню функцію або вимогу). Значення Nод - статистичне.



На етапі визначення архітектури

Ця оцінка визначається формулою:

$$Ci = NI / (NF * NIод * КСЛ)$$

де NI - загальна кількість змінних, що передаються через інтерфейси між компонентами програми (є статистичною);

NIод-одиничне значення кількості змінних, що передаються через інтерфейси між компонентами (середнє число переданих через інтерфейси змінних, що припадають на одну середню функцію або вимогу);

КСЛ - коефіцієнт складності програми , враховує зростання одиничної складності програми (складності, що припадає на одну функцію або вимога специфікації вимог) для великих і складних програм в порівнянні з середніми програмами.

Запитання?