



Лекція 1.4

Основні технічні та управлінські проблеми супроводження програмного забезпечення



Питання до розгляду

☐ Конфігураційне управління.

II. Вартість супроводження.

1. Моделі емпіричної оцінки вартості.

2. Оцінка вартості супроводження програмного забезпечення.

III. Література.



I. Конфігураційне управління

Задачі розробників-супроводжувачів програмного забезпечення

- Зрозуміти систему.
- Розмістити інформацію в документації.
- Постійно тримати програмну документацію в актуальному стані.
- Розширити існуючі функції.
- Додати нові функції.
- Пошук джерел помилок.
- Виправлення помилок.
- Відповідати на операційні запити.
- Реструктуризація структури програми і та програмного коду.
- Видалення застарілих частин структури і коду.
- Управління змінами.



Технічні особливості супроводження програмного забезпечення

- Існують спільні технічні питання як для розробки програмного забезпечення, так і для супроводження
 - **конфігураційне управління** і контроль версій (CVS, RCS)
- Деякі технічні питання стосуються лише супроводження програмного забезпечення
 - визначити вартість внесення змін відповідно до запитів на зміни програмного забезпечення

Конфігураційне управління

- Зміни програмного забезпечення є неминучим процесом.
- Однією з цілей розробки програмного забезпечення є змінювати програмне забезпечення для покращення його зрозумілості.
- Конфігураційне управління це вся інформація про контроль змін.
- Кожен програміст повинен бути стурбований тим, як зміни, внесені до робочих продуктів відстежуються і поширюються протягом усього проекту.
- Для забезпечення якості супроводження повинен бути проведений аудит процесу змін.

Сфери конфігураційного управління

- Комп'ютерні програми
 - вихідний код
 - результати виконання
- Документація
 - технічна
 - користувацька
- Дані
 - які містяться в програмі
 - зовнішні дані (наприклад, файли і бази даних)

Базиси

- Робочий продукт стає базисом тільки після того, як буде розглянутий і затверджений.
- Базис – етап розробки програмного забезпечення, що визначає одну або кілька сфер конфігураційного управління.
- Після того, як базис буде визначено кожен запит зміни повинен бути оцінений і верифікований, перш ніж буде оброблений.

Джерела змін

- Нові ринкові умови призводять до змін вимог до програмного забезпечення або бізнес-правил.
- Новий користувач потребує зміни даних, функціональності або послуг.
- Реорганізація бізнесу призводить до зміни в пріоритетах програмного проекту або структурі команди з розробки програмного забезпечення.
- Зміни у фінансуванні та планах розробки призводять до перегляду вимог до системи.

Запити на зміну

- Запити можуть надходити від користувачів, клієнтів, або керівників.
- Запити на зміну повинні бути ретельно проаналізовані, як частина процесу супроводження перш, ніж вони будуть реалізовані.
- Деякі запити на зміну повинні бути реалізовані в терміновому порядку в силу своєї природи
 - виправлення несправностей
 - зміни системного середовища
 - терміново бізнес-зміни

Прогнозування змін

- Прогнозування кількості змін потребує розуміння зв'язку між системою та її програмним середовищем.
- Тісно пов'язані системи вимагають внесення змін щоразу при зміні середовища їх функціонування.
- Фактори, що впливають на зв'язок системи та програмного середовища
 - кількість і складність інтерфейсів системи
 - кількість і волатильність системних вимог
 - бізнес-процеси, в яких використовується система

Задачі конфігураційного управління

- Ідентифікація
 - відстеження змін для різних версій SCI
- Контроль версій
 - контроль змін до і після користувацьких релізів
- Управління змінами
 - можливість затверджувати та пріоритезувати зміни
- Аудит конфігурацій
 - забезпечити правильності внесення змін
- Звітність
 - Оповіщення всіх користувачів системи про внесення змін

Контроль версій. Основні терміни

□ Сутність

- складається з об'єктів на однаковому рівні перегляду

□ Варіант

- набір із різних об'єктів на однаковому рівні перегляду та співіснуючий із іншими варіантами

□ Нова версія

- виникає, коли основні зміни для одного або декількох об'єктів були проведені

Процес контролю змін - 1

- Запит на зміну подається та оцінюється, щоб оцінити його технічні переваги і вплив на інші об'єкти конфігурації і бюджет.
- Звіт по зміні містить результати оцінки, які генеруються.
- Група контролю за змінами (ССА – Change control authority) приймає остаточне рішення про статус і пріоритет зміни базуючись на звіті.

Процес контролю змін - 2

- Ордер обробки зміни (ECO – Engineering change order) створюється для кожної затвердженої зміни.
 - ECO описує зміну, складає список обмежень, і визначає критерії для аналізу та аудиту
- Об'єкт, що буде змінений, перевіряється з базою даних проекту для отримання доступу до параметрів керування для об'єкта
- Модифікований об'єкт піддається відповідним процедурам SQA і тестування.

Процес контролю змін - 3

- Модифікований об'єкт перевіряється механізмами проектної бази даних і контролю версій, які використовуються для створення наступної версії програмного забезпечення.
- Контроль синхронізації використовується для забезпечення того, щоб паралельні зміни, зроблені різними людьми зовсім не змінювали один одного.

Команда з конфігураційного управління

- Аналітики.
- Програмісти.
- Програмний бібліотекар.

Рада управління змінами

- Представники замовника.
- Деякі члени команди конфігураційного управління.

Робота програміста - 1

- Проблема виявлена.
- Звіт про проблему направляється до групи конфігураційного контролю.
- Група обговорює проблему
 - чи є проблема несправністю?
 - чи є це покращенням?
 - хто повинен за це платити?
- Визначити рівень пріоритетності чи серйозності проблеми, і призначити відповідальних співробітників.

Робота програміста – 2

- Програміст або аналітик
 - знаходить джерело проблеми
 - визначає, що необхідно, щоб її виправити
- Програміст разом із програмним бібліотекарем контролюють інсталяцію змін в операційну систему і в документацію.
- Програміст подає звіт по зміні, в якому задокументовані усі проведені зміни.

Питання контролю змін

- ❑ Синхронізація (коли?)
- ❑ Ідентифікація (хто?)
- ❑ Визначення назв (що?)
- ❑ Аутентифікація (чи вірно зроблено?)
- ❑ Авторизація (хто погоджує?)
- ❑ Маршрутизація (хто повідомив?)
- ❑ Скасування (хто може зупинити його?)
- ❑ Делегування (питання відповідальності)
- ❑ Оцінювання (питання пріоритетності)

Конфігураційний аудит - 1

- Чи була зміна визначена ЕСО зроблена без модифікацій?
- Чи було проведено FTR (First Time Right), щоб оцінити технічну правильність?
- Чи був дотриманий програмний процес і стандарти розробки програмного забезпечення?

Конфігураційний аудит - 2

- Чи атрибути об'єкта конфігурації відображають зміни?
- Чи були дотримані стандарти конфігураційного управління для запису та звітності по змінам?
- Чи були всі пов'язані SCI правильно оновлені?

Конфігураційний статус звіту

- Що сталося?
- Хто це зробив?
- Коли це сталося?
- Що ще буде залежати від зміни?



II. Вартість супроводження

1. Моделі емпіричної оцінки вартості

Кілька моделей виходу з різною успішністю і легкістю/важкістю користування.

- Розглянемо СОСОМО (Модель витрат розробки).
- Розкладемо програмне забезпечення на достатньо малі частини, щоб мати можливість оцінити LOC (лінії коду).
- Визначення:
 - KDSI – кіло поставки інструкцій із вихідних даних (заявки)
 - не включаючи коментарі, тестові випробування, тощо.
 - PM - людиномісяці
- З рівня моделі СОСОМО: Базовий, Середній та, Деталізований (про нього мова в лекції не йтиметься)

COCOMO

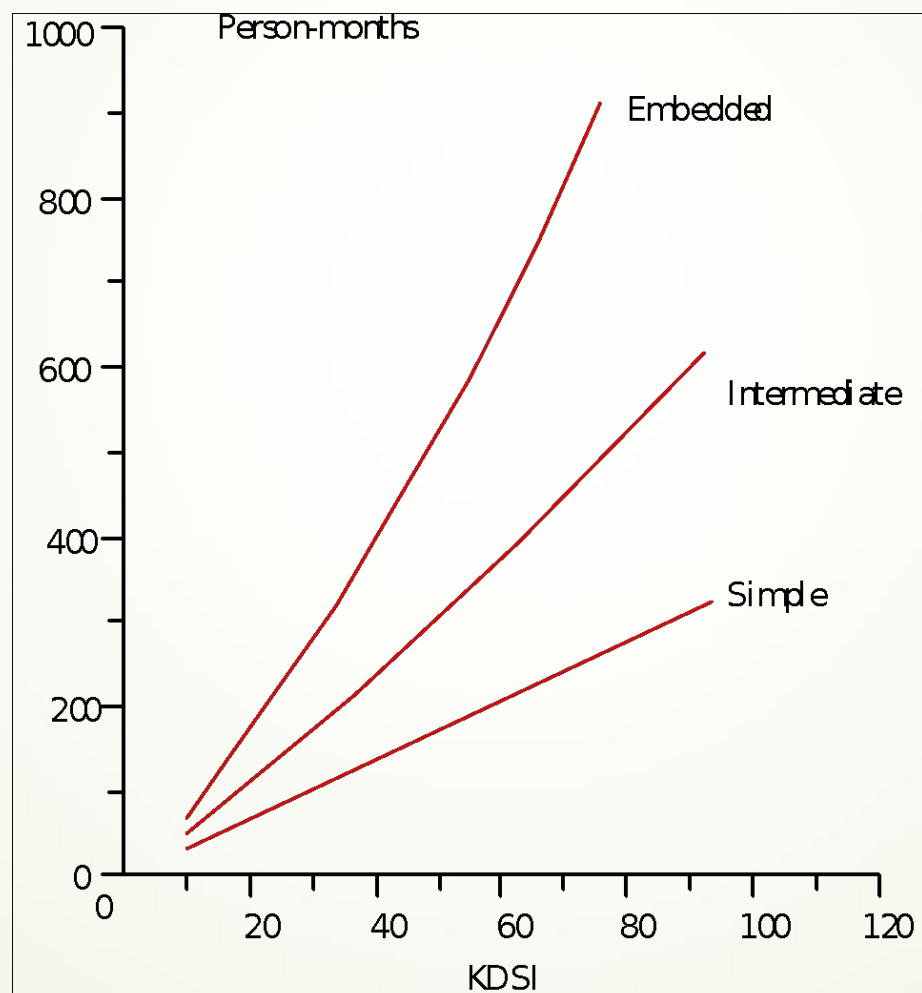
Модель 1: Базова

□ Застосовуємо наступні формули, щоб отримати грубі оцінки

□ $PM = 2.4(KDSI)^{1.05}$

□ $T_{DEV} = 2.5(PM)^{0.38}$ (хронологічні місяці)

Оцінки робіт



СОСОМО приклади

- Органічний клас проекту, 32KLOC
 - $PM = 2.4 (32)^{1.05} = 91$ людиномісяців
 - $TDEV = 2.5 (91)^{0.38} = 14$ місяців
 - $N = 91/14 = 6.5$ людей
- Вбудований клас проекту, 128KLOC
 - $PM = 3.6 (128)^{1.2} = 1216$ людиномісяців
 - $TDEV = 2.5 (1216)^{0.32} = 24$ місяців
 - $N = 1216/24 = 51$ людей

COCOMO

Модель 2: Середній

□ **крок I:** отримуємо номінальну оцінку у вигляді:

□ $PM_{NOM} = a_i (KDSI)^{b_i}$, де

	a_i	b_i
Органічний	3.2	1.05
Напіврозподілений	3.0	1.12
Вбудований	2.8	1.2

SOCOMO

- *Органічні проекти*: маленька команда розробників; знайомі між собою; робота в одному приміщенні; великий досвід; нескладні вимоги.
- *Вбудовані проекти*: фірма, жорсткі обмеження (в т. ч. апаратного забезпечення), в основному малознайома територія.
- *Напіврозподілені проекти*: в обох випадках.
- **крок II**: визначити оцінки факторів вартості:
 - З таблиці з 15 факторів, кожен оцінюється за 6-бальною шкалою.

COCOMO

□ 4 групи факторів:

1. *Характеристики продукту: потрібна надійність, складність продукту,*
2. *Характеристики апаратного забезпечення: обмеження: швидкодії підчас виконання програми, пам'яті, середовища віртуальної машини, волатильності (апаратної та програмної), часу відновлення,*
3. *Характеристики персоналу: аналітичні, програмістські навички, досвід розробки, досвід роботи з віртуальними машинами, досвід розробки на потрібних мовах програмування,*
4. *Характеристики проекту: сучасні методики програмування, використання інструментів розробки програмного забезпечення, реалістичний графік робіт.*

COCOMO

- 15 факторів, кожен оцінений за 6-бальною шкалою:
- дуже низький - низький - середній - високий – дуже високий - критичний
- використовуючи модель COCOMO рахуємо регулюючий фактор трудоемності (EAF):

$$\text{Фактори} \prod_{i=1}^{15}$$

- **крок III:** середня оцінка трудоемності розробки матиме вигляд:

$$PM_{DEV} = PM_{NOM} * EAF$$

COCOMO

□ **крок IV:** оцінка відповідних ресурсів виглядатиме так:

$$T_{DEV} = c_i (PM_{DEV})^d_i$$

	c_i	d_i
Органічний	2.5	0.38
Напіврозподілений	2.5	0.35
Вбудований	2.5	0.32

2. Оцінка вартості супроводження програмного забезпечення

- Основне питання: яка кількість програмістів необхідна для супроводження системи програмного забезпечення?
- Не складно здогадатись, що відповідь матиме вигляд типу:
лінії коду (LOC), що потребують супроводження
оцінка ліній коду (LOC) на кожного програміста

Оцінка вартості супроводження програмного забезпечення

- Альтернативи: за Б.Бемом
 - визначимо АСТ (річний трафік змін) як частку заявок змін на рік:

$$\frac{\text{KLOC}_{\text{added}} + \text{KLOC}_{\text{changed}}}{\text{KLOC}_{\text{total}}}$$

- **1 рівень моделі:**

$$\text{PM}_{\text{AM}} = \text{ACT} \times \text{PM}_{\text{DEV}}$$

де AM = річне супроводження

Оцінка вартості супроводження програмного забезпечення

□ 2 рівень моделі:

$$PM_{AM} = ACT \times PM_{DEV} \times EAF_M$$

де EAF_M може відрізнятися від EAF для розробки з:

- різним персоналом
- рівнем досвіду, мотивації, тощо

Оцінка вартості супроводження програмного забезпечення

□ Фактори, які впливають на продуктивність програмного забезпечення (так само на супроводження):

1. *Людські фактори*: розмір організації і досвід розробки (супроводження).
2. *Проблемні фактори*: складність проблеми і кількість змін в проектних обмеженнях і вимогах.

Оцінка вартості супроводження програмного забезпечення

3. *Процесні фактори*: аналіз, проектування, методи тестування, мови програмування, CASE інструменти, тощо.
4. *Фактори продукту*: потрібна надійність і продуктивність системи.
5. *Ресурсні фактори*: наявність CASE інструментів, ресурсів апаратного та програмного забезпечення.

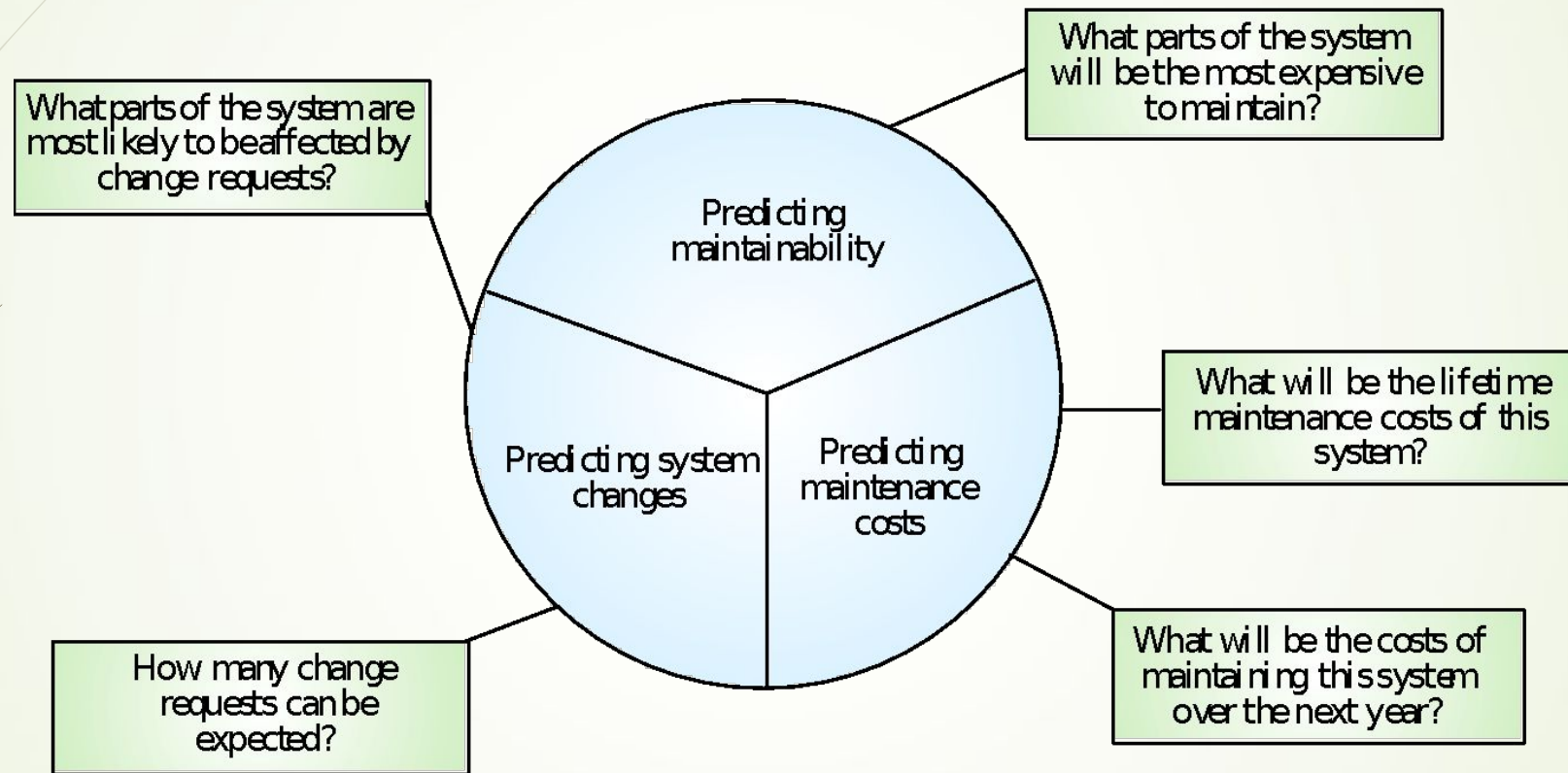
Фактори вартості супроводження

- Плинність кадрів
 - низька плинність означає зниження витрат на супроводження
- Договірна відповідальність
 - розробники можуть не мати договірних зобов'язань супроводжувати розроблювану систему і розбудовувати її структуру для майбутніх змін
- Кваліфікація персоналу
 - персонал супроводження часто недосвідчений і має обмежені знання про домени супроводження
- Вік програми та структура
 - з часом структура програми погіршується, її стає важче розуміти і змінити

Прогнозування супроводження

- ❑ Визначити частини системи, що можуть викликати проблеми і вимагати значні витрати на супроводження
- ❑ Прийняття змін залежить від супроводжуваності компонентів, які зачіпає зміна
- ❑ Проведення змін погіршує систему і знижує її супроводжуваність
- ❑ Вартість супроводження залежить від кількості змін
- ❑ Вартість зміни залежить від супроводжуваності

Прогнозування супроводження



Метрики складності супроводження

- Прогнозування супроводження може бути здійснене шляхом оцінки складності компонентів
- Більшість зусиль спрямованих на супроводження впливає тільки на невелику кількість компонентів системи
- Складність супроводження залежить від
 - складності структури управління
 - складності структури даних
 - розміру модуля

Метрики процесу супроводження

- Показники супроводжуваності
 - кількість запитів на коригуюче супроводження
 - середній час, необхідний для імпакт аналізу
 - середній час для реалізації запиту на зміну
 - кількість розміщених запитів на зміни
- Якщо який-небудь з цих показників збільшився, це може сигналізувати про зниження супроводжуваності

Інструменти супроводження

- Текстові редактори.
- Програми для порівняння файлів.
- Компілятори та редактори зв'язків.
- Програми для проведення дебагінгу (налагодження).
- Генератори перехресних посилань.
- Калькулятори підрахунку складності.
- Контроль бібліотек.
- CASE інструменти для всього життєвого циклу.

III. Література

- Guide to the Software Engineering Body of Knowledge (SWEBOK). – California: IEEE Computer Society, 2001. – 219 p.
- 1. Grubb Penny. Software Maintenance: Concepts and Practice (2nd Edition) / [Penny Grubb, Takang A. Armstrong]. – Singapore: World Scientific, 2003. – 349 p.
- 2. Pigosky M. Thomas. Practical Software Maintenance – Best Practices for Managing Your Software Investment / Thomas M. Pigosky. – Canada: Wiley Computer Publishing, 1997. – 228 p.
- 3. Shari L. Pfleeger, Joann M. Atlee. Software Engineering. Theory and practice.