

OPENUP

- 1. ЩО ТАКЕ OPENUP?
- 2. ПРИНЦИПИ
- 3. ОРГАНІЗАЦІЯ
 - А. МЕТОД
 - (1) РОЛІ, (2) ДИСЦИПЛІНИ,
 - (3) ЗАВДАННЯ, (4)
 - АРТЕФАКТИ, (5)
 - ДИРЕКТИВИ
 - Б. ПРОЦЕС
 - (1) ШАБЛОНИ, (2)
 - ЖИТТЄВИЙ ЦИКЛ
 - ІТЕРАЦІЙ, (3) ПРОЦЕС
 - ДОСТАВКИ
- 4. МІКРО ПРИРІСТ
- 5. ВПЛИВИ

ПЛАН

- 2005 РІК: БАЗОВИЙ ЄДИНИЙ ПРОЦЕС ІВМ
- 2006: OPENUP ТА EPF ВІД ECLIPSE FOUNDATION
- ІТЕРАЦІЙНИЙ ПРОЦЕС НА ОСНОВІ RUP
 - МІНІМАЛЬНИЙ
 - ЗАВЕРШЕНИЙ
 - РОЗШИРЮВАНИЙ
- ШВИДКИЙ ПІДХІД
 - СПІВПРАЦЯ ТА СПІЛКУВАННЯ

ЩО ТАКЕ OPENUP?

1. СПІВПРАЦЮЙТЕ ДЛЯ
УЗГОДЖЕННЯ ІНТЕРЕСІВ
ТА ОБМІНУ ЗНАННЯМИ
2. ЗБАЛАНСУЙТЕ
ПРІОРИТЕТИ ДЛЯ
МАКСИМІЗАЦІЇ ВИГОДИ
СТОРИН
3. ЗОСЕРЕДЬТЕСЯ НА
АРХІТЕКТУРІ, ЩОБ
МІНІМІЗУВАТИ РИЗИК
І ОРГАНІЗУВАТИ
РОЗРОБКУ
4. ЕВОЛЮЦІОНУЙТЕ,
ЩОБ ПОСТІЙНО
ОТРИМУВАТИ ВІДГУКИ
ТА
ПОКРАЩУВАТИСЯ

ПРИНЦИПИ

1. СПІВПРАЦЮЙТЕ ДЛЯ УЗГОДЖЕННЯ ІНТЕРЕСІВ ТА ОБМІНУ ЗНАННЯМИ ->
"ОСОБИ ТА ВЗАЄМОДІЇ НАД ПРОЦЕСОМ ТА ІНСТРУМЕНТАМИ"

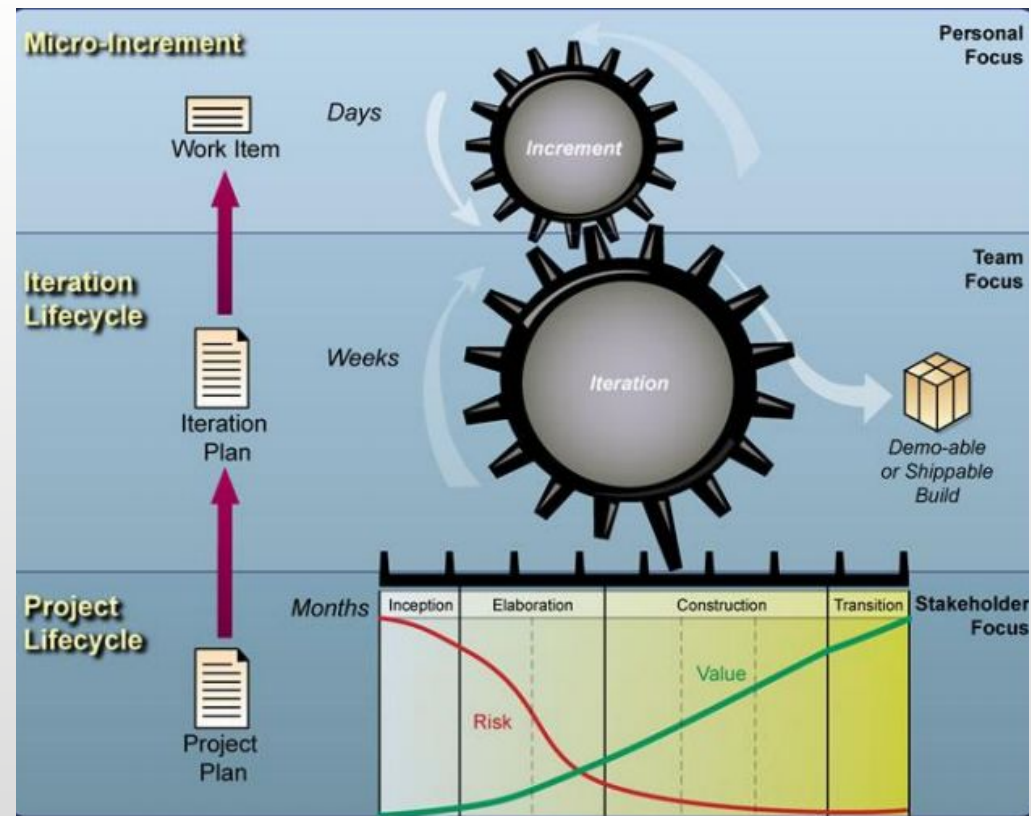
2. ЗБАЛАНСУВАННЯ ПРІОРИТЕТІВ ДЛЯ МАКСИМІЗАЦІЇ ВИГОДИ СТОРІН ->
"СПІВПРАЦЯ З КЛІЄНТАМИ ЩОДО УКЛАДЕННЯ ДОГОВОРІВ"

3. ЗОСЕРЕДЬТЕСЯ НА АРХІТЕКТУРІ , ЩОБ МІНІМІЗУВАТИ РИЗИК І ОРГАНІЗУВАТИ РОЗРОБКУ -> "РОБОЧЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ НАД ВИЧЕРПНОЮ ДОКУМЕНТАЦІЄЮ"

4. ЕВОЛЮЦІОНУЙТЕ, ЩОБ ПОСТІЙНО ОТРИМУВАТИ ВІДГУКИ ТА ПОКРАЩУВАТИСЯ -> "ВІДПОВІДЬ НА ЗМІНИ ПРОТЯГОМ ВИКОНАННЯ ПЛАНУ"

ПРИНЦИПИ // ШВИДКИЙ МАНІФЕСТ

ОРГАНІЗАЦІ Я



- РОЛІ
- ЗАЦІКАВЛЕНІ СТОРОНИ
 - АНАЛІТИК
 - АРХІТЕКТОР
 - РОЗРОБНИК
 - ТЕСТЕР
- КЕРІВНИК ПРОЕКТУ
 - БУДЬ-ЯКА РОЛЬ
 - + 6 КОНКРЕТНИХ РОЛЕЙ ДЛЯ РОЗГОРТАННЯ
- + 2 КОНКРЕТНІ РОЛІ СЕРЕДОВИЩА

МЕТОД: РОЛІ (I)

Метод фокусується на наступних

дисциплінах:

1. Вимоги
2. Архітектура
3. Розвиток
4. Тест
5. Управління проектами
6. Управління конфігурацією та змінами

МЕТОД: ДИСЦИПЛІНИ (2)

МЕТОД: ЗАВДАННЯ (3)

- "Одиниця роботи, яка може бути запропонована для виконання"
- Первинний виконавець проти додаткових виконавців
- Близько 35 завдань, визначених за замовчуванням, розподіляються на 7 категорій (5 дисциплін + розгортання та оточення)

Наприклад:

- Показ архітектури (Architecture)
 - Пакет випуску (Deployment)
 - Створення тестових завдань (Test)
 - Процес розробки (Environment)
-

МЕТОД: АРТЕФАКТИ (4)

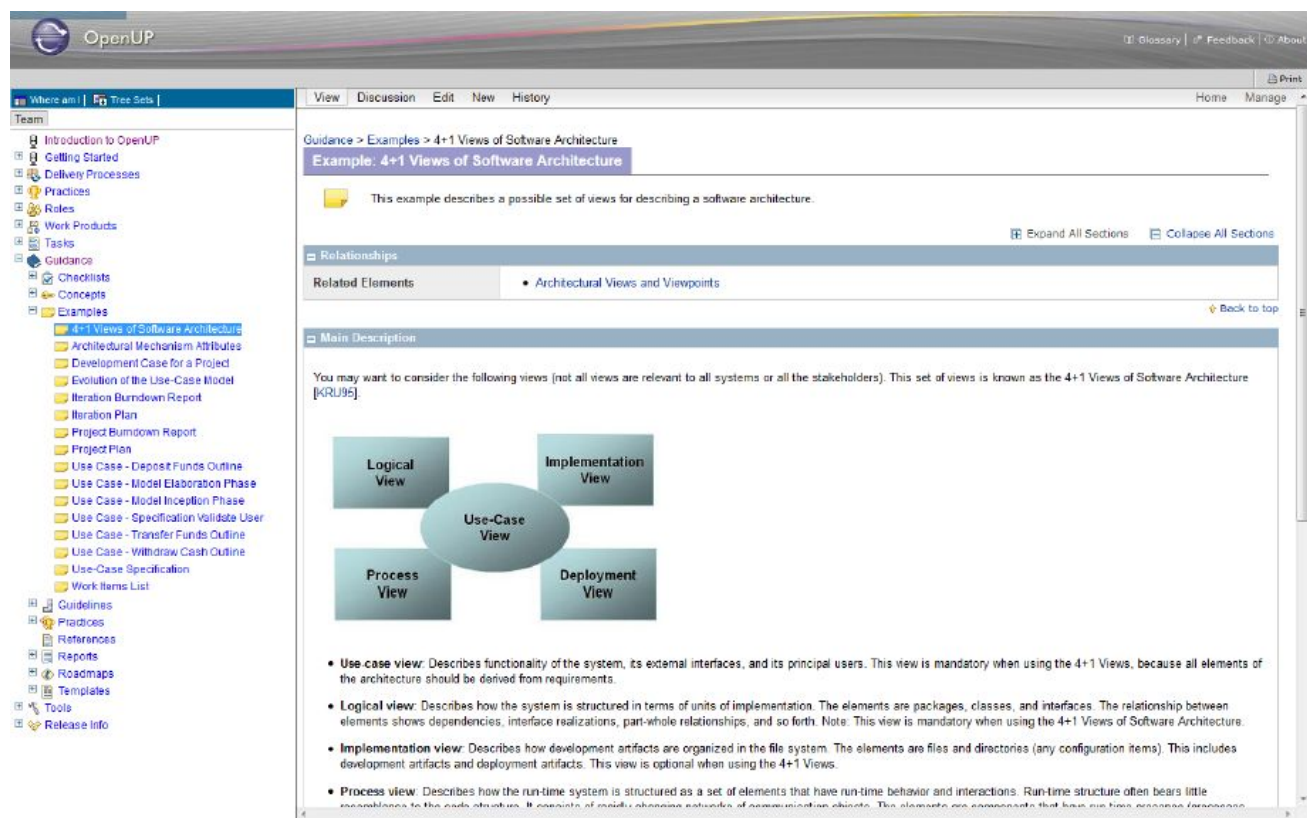
- "Щось, що виробляється, змінюється або використовується завданням."
- Немає формалізму в представленні артефактів
 - Наприклад: Фото дошки, представляє архітектуру
- Близько 30 артефактів, визначених за замовчуванням, розподілені у 7 категорій

Наприклад:

- Glossary (Requirement)
 - Risk list (Project management)
 - Deployment plan (Deployment)
-

МЕТОД: ДИРЕКТИВИ (5)

НАБІР КОРИСНИХ ПОРАД
ТА КОНТРОЛЬНИХ
СПИСКІВ, ЯКІ
ДОПОМОЖУТЬ ВАМ
ДІСТАТИСЯ ЦІЛІ



OpenUP

Where am I | Tree Sets | View | Discussion | Edit | New | History | Home | Manage

Guidance > Examples > 4+1 Views of Software Architecture

Example: 4+1 Views of Software Architecture

This example describes a possible set of views for describing a software architecture.

Expand All Sections | Collapse All Sections

Relationships

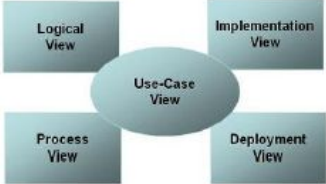
Related Elements

- Architectural Views and Viewpoints

Back to top

Main Description

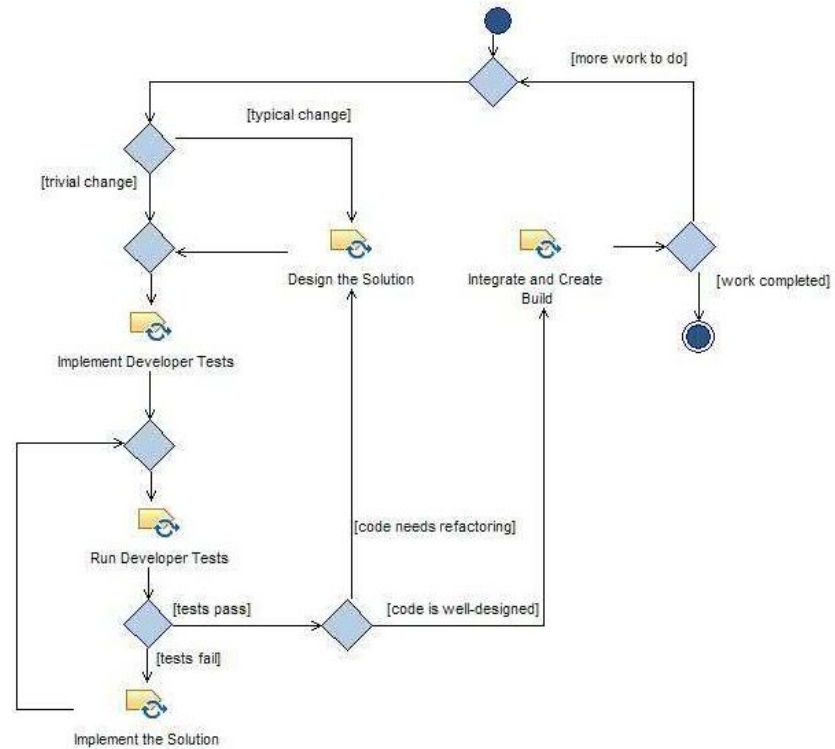
You may want to consider the following views (not all views are relevant to all systems or all the stakeholders). This set of views is known as the 4+1 Views of Software Architecture [KRJ95].



- Use case view:** Describes functionality of the system, its external interfaces, and its principal users. This view is mandatory when using the 4+1 Views, because all elements of the architecture should be derived from requirements.
- Logical view:** Describes how the system is structured in terms of units of implementation. The elements are packages, classes, and interfaces. The relationship between elements shows dependencies, interface realizations, part-whole relationships, and so forth. Note: This view is mandatory when using the 4+1 Views of Software Architecture.
- Implementation view:** Describes how development artifacts are organized in the file system. The elements are files and directories (any configuration items). This includes development artifacts and deployment artifacts. This view is optional when using the 4+1 Views.
- Process view:** Describes how the run-time system is structured as a set of elements that have run-time behavior and interactions. Run-time structure often bears little resemblance to the code structure. It consists of modules, channels, threads of communication, objects. The elements are elements that have run-time process (processes).

ПРОЦЕС: ШАБЛОНИ (I)

ВИЗНАЧАЄ ВСІ ЗАВДАННЯ
ДЛЯ ВИКОНАННЯ ЩОБ
ЗАДОВОЛЬНИТИ
ПОТРЕБУ ПРОЕКТУ.



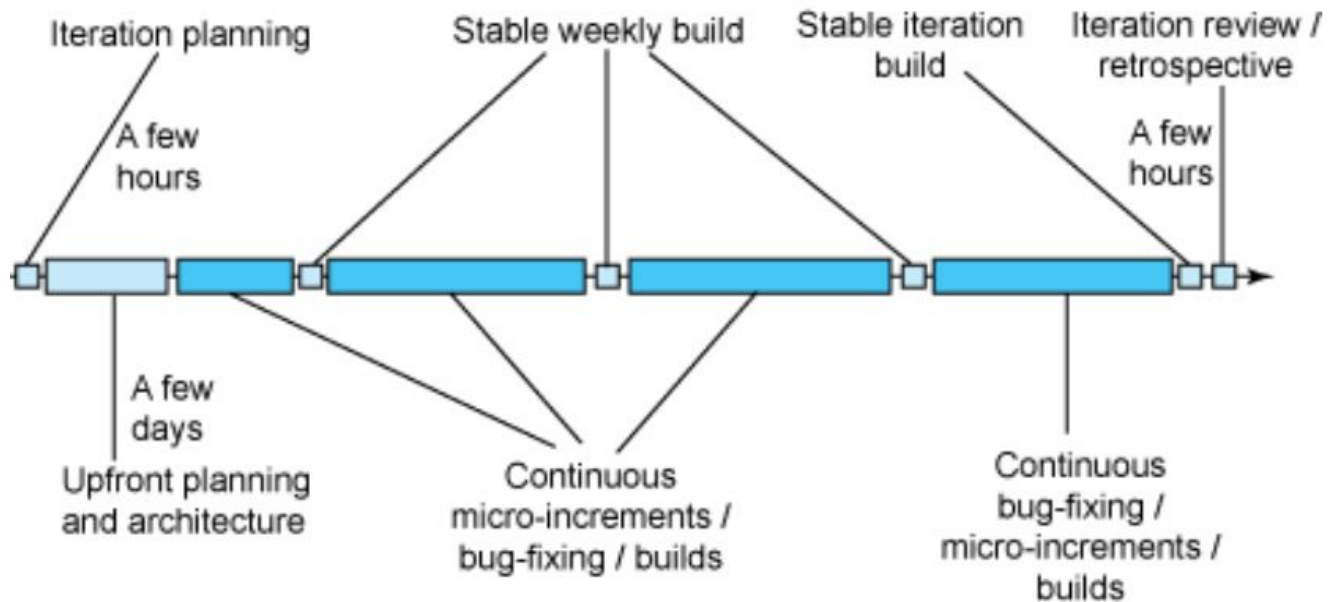
ПРОЦЕС: ЖИТТЄВИЙ ЦИКЛ ІТЕРАЦІЇ (2)

1. ПЛАНУВАННЯ ІТЕРАЦІЇ

2. ЗАПУСТІТЬ І ВИПРОБУЙТЕ
МІКРОПРИРІСТ

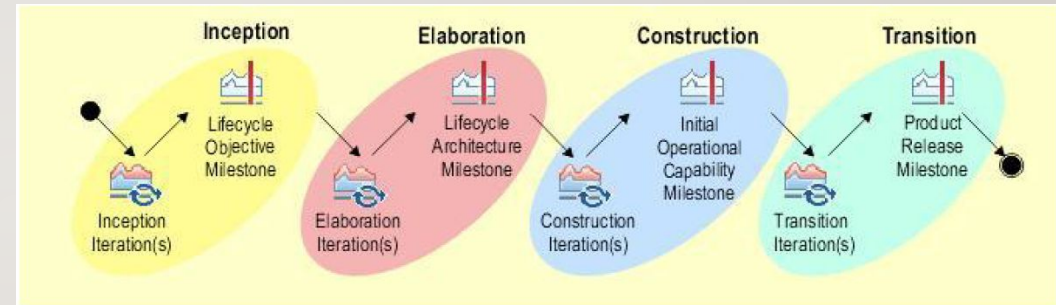
3. ВИРІШЕННЯ ПОМИЛОК ТА МОЖЛИВЕ
ДОДАВАННЯ НОВИХ ХАРАКТЕРИСТИК

4. РЕТРОСПЕКТИВА ТА ОЦІНКА

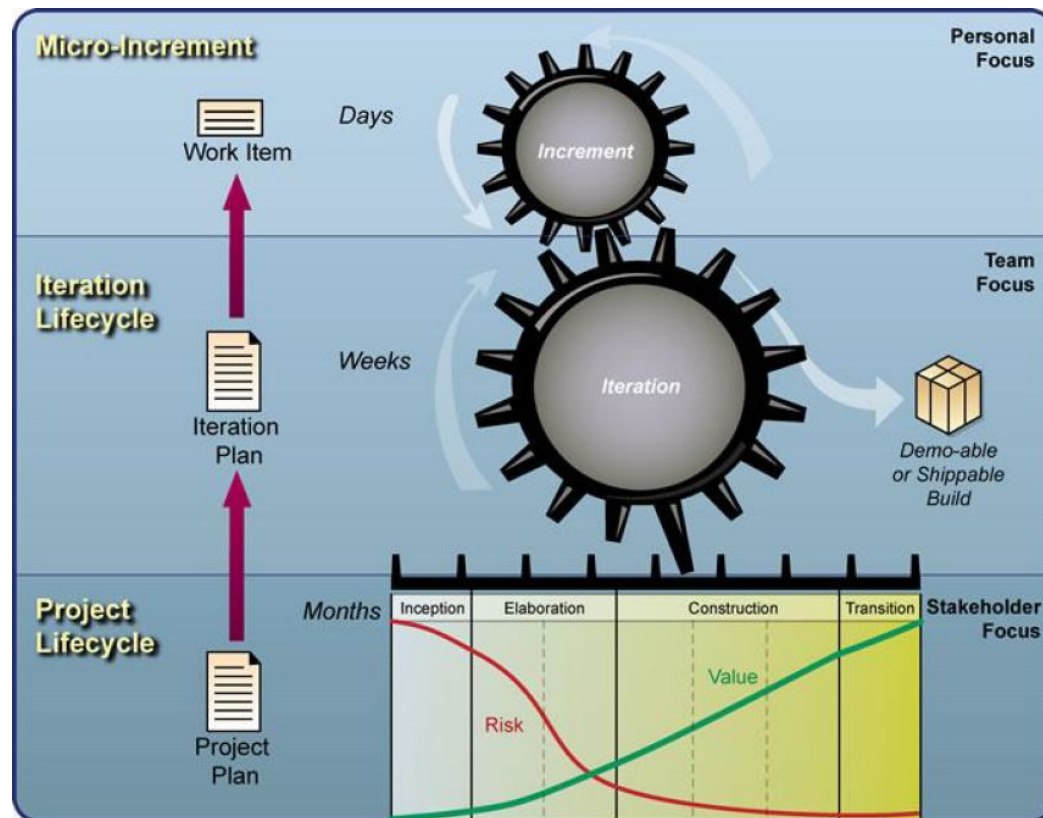


ПРОЦЕС: ПРОЦЕС ДОСТАВКИ (3)

- ОПИСУЄТЬСЯ ПОВНИЙ ТЕРМІН ЕКСПЛУАТАЦІЇ ПРОЕКТУ, АЛЕ НЕ ЗАМІНЮЄ ЙОГО РЕАЛЬНІСТЬ
- СКЛАДЕНО З ШАБЛОНІВ, ЩОБ ПОВТОРИТИ СТІЛЬКИ РАЗІВ, СКІЛЬКИ ПОТРІБНО ДЛЯ КОЖНОЇ З 4 ФАЗ
- КІЛЬКІСТЬ ІТЕРАЦІЙ СИЛЬНО ЗАЛЕЖИТЬ ВІД ТИПУ ПРОЕКТУ



МІКРО ПРИРІСТ



МІКРО ПРИРІСТ

- Результат роботи, який може становити від кількох годин до кілька днів від одного або декількох людей.
 - ЦІЛЬ:
 - Розділіть роботу на менші одиниці, щоб кожен сприяв доданій вартості проекту.
 - Дуже короткий цикл зворотного зв'язку
 - OpenUP не забезпечує повний перелік можливих приростів.
-



ВПЛИВ

- Scrum і XP: коротка ітеративна модель з можливістю доставки в кінці кожної ітерації
 - RUP: приймає концепцію фази, додаючи "Selforganization" і ітерацію
 - Eclipse Way: гнучкий і ітеративний
-

ДЖЕРЕЛА ІНФОРМАЦІЇ

OpenUP

<http://epf.eclipse.org/wikis/openup/>

<http://www.eclipse.org/epf/general/OpenUP.pdf>

<http://www.methodsandtools.com/PDF/mt200801.pdf>

EPF

http://en.wikipedia.org/wiki/Eclipse_Process_Framework

