МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування
імені адмірала Макарова

Т. В. ПОНОМАРЕНКО, С. В. СУСЛОВ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт з дисципліни
«МЕНЕДЖМЕНТ ПРОЄКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Рекомендовано Методичною радою НУК

ВИДАВНИЦТВО
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
КОРАБЛЕБУДУВАННЯ
ІМ. АДМІРАЛА МАКАРОВА
2021

УДК 004.4512(076)
П56

Автори: Т. В. Пономаренко, канд. техн. наук, доцент кафедри ПЗАС;
С. В. Суслов, канд. техн. наук, доцент кафедри ПЗАС

Рецензент Т. А. Фаріонова, канд. техн. наук, доцент, директор ННІКТІТН

Рекомендовано Методичною радою НУК

Пономаренко Т. В.

П56 Методичні вказівки до виконання лабораторних робіт з дисципліни «Менеджмент проєктів програмного забезпечення» / Т. В. Пономаренко, С. В. Суслов. – Миколаїв : НУК, 2021. – 48 с.

Призначено для студентів, які навчаються за напрямом підготовки 6.050103 «Програмна інженерія».

УДК 004.4512(076)

МЕТА І ЗАВДАННЯ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Мета дисципліни – навчання студентів принципам теорії управління проєктами з розробки проєктів програмного забезпечення та набуття практичних навичок планування, контролю й оптимізації процесів розробки програмного забезпечення.

Завданням дисципліни є надання студенту можливості засвоїти сукупність знань та вмінь, що дозволять фахівцю управляти проєктами по створенню програмного забезпечення, працювати з окремими процесами управління проєктами, використовувати спеціальне прикладне програмне забезпечення для моніторингу та планування управління проєктами.

У результаті вивчення навчальної дисципліни студент повинен ***знати*:**

a) процеси менеджменту проєктів, життєвий цикл ПЗ;

b) управління змістом проєкту;

c) управління строком виконання та вартістю проєкту;

d) управління людським потенціалом і комунікаціями;

e) управління якістю проєкту та ризиками;

вміти:

– здійснювати управління проєктами з розробки програмного забезпечення;

– виділяти та класифікувати проєкти і задачі управління проєктами в межах організації;

– визначати ролі учасників проєкту;

– формувати організаційну структуру для управління проєктами;

– використовувати існуючі стандарти з управління проєктами;

– здійснювати вибір програмного забезпечення для задач управління проєктами.

ВИМОГИ ДО ОФОРМЛЕННЯ ЛАБОРАТОРНИХ РОБІТ

Звіт має містити:

– титульний аркуш;

– тему і мету роботи;

– завдання до роботи;

– лаконічний опис теоретичних відомостей;

– опис ходу роботи та результати її виконання;

– аналіз досягнутих результатів та висновки.

Титульний аркуш є першою сторінкою звіту, яка не нумерується.

На титульному аркуші подаються тема роботи, зазначається спеціальність, номер групи та назва проєкту, вказується ВНЗ, прізвища виконавців та перевіряючого.

Звіт виконують на білому папері формату А4 (210×297 мм). Текст розміщують тільки з однієї сторони листа. Поля сторінки з усіх боків – 20 мм. Аркуші скріплюють за допомогою канцелярських скріпок або вміщують у канцелярський файл. Під час співбесіди при захисті лабораторної роботи студент повинен виявити знання про мету роботи, по теоретичному матеріалу, про методи виконання кожного етапу роботи, по змісту основних розділів оформленого звіту з демонстрацією результатів на конкретних прикладах.

Студент повинен вміти правильно аналізувати отримані результати. Для самоперевірки при підготовці до виконання і захисту роботи студент повинен відповісти на контрольні запитання, наведені наприкінці опису відповідної роботи.

Враховуючи, що ми навчаємо студентів програмній інженерії, логічним буде в рамках цього курсу самостійно розробити програмне забезпечення для менеджменту проєктів програмного забезпечення, що з однієї сторони буде знайомити

нас з предметною галуззю менеджменту, а з іншої надасть додаткових навичок створення програмного забезпечення.

ЛАБОРАТОРНА РОБОТА № 1

***Тема*: Управління ініціацією проєкту**

Мета роботи: виходячи зі специфікації на розробку проєкту, визначити необхідні ресурси для виконання проєкту з розробки програмного забезпечення та узгодити графік виконання проєкту з замовником (викладачем), а також провести предпроєктну підготовку робочого місця виконавця (студента).

Завдання: створити проєкт.

Короткі теоретичні відомості

1.1. Основні поняття менеджера проєктів

Проєкт (Project) – це цілеспрямована діяльність тимчасового характеру з метою створення унікального продукту або послуги. Тобто проєкт повинен мати окреслений початок і кінець, він може бути дуже коротким або дуже тривалим, але повинен залишатися тимчасовим етапом. Як правило, проєкти ініціюють сторони, які не беруть безпосередньої участі в проєкті, наприклад, керівник організації або уповноважений представник.

Наука управління проєктами охоплює 10 взаємопов'язаних областей знань і включає в себе використання 47 процесів, організованих у 5 груп процесів.

Керівник проєкту (Project manager/PM)

Проєктний менеджер, проджект-менеджер – це особа, яку призначає компанія або представник компанії, і яка ініціює проєкт. Керівник проєкту відповідає за команду проєкту, веде нагляд за проєктом та надає допомогу й фідбек усім, хто цього потребує, для забезпечення успішних результатів проєкту.

Управління програмами (Program management)

Якщо компанія має подібні проєкти, заходи або підпрограми, вони можуть бути згруповані для керування разом, оскільки це може бути більш ефективним і більш розумним, ніж управління кожним проєктом окремо.

Бізнес потреба (Business need)

Це несформовані побажання до проєкту. Бізнес-потреби впливають на бізнес-рішення. Інколи бізнес-потреби прописуються як додатковий документ у завданні виконання проєкту/звіті про роботу (SOW).

Бізнес-вимоги (Business requirements)

Бізнес-вимоги (business requirements) – це вже опрацьовані бізнес потреби. Бізнес вимоги містять високорівневі цілі організації або замовників системи. Бізнес вимоги уже обов'язково оформлюють у документ, в якому пояснюється, чому організації потрібна саме така система, і деталізовано описують цілі, які організація має намір досягти. Іноді такий документ оформлюють як статут проєкту **(Project Charter)** або документ ринкових вимог **(Market Requirements Document)**.

Зміст проєкту (Project Scope)

Буквально **Project Scope** – це часові рамки, обсяг опису робіт, які потрібно виконати, необхідних ресурсів, очікувані кінцеві результати, критерії оцінки, вимоги до якості – все те, що повинне бути виконане для того, щоб отримати продукт, послугу або результат.

Обсяг продукту (Product scope)

Product Scope – це сукупність продуктів і послуг, виробництво яких має бути забезпечене в рамках здійснюваного проєкту. Наприклад, провести маркетингове дослідження, розробити по проєкту унікальний дизайн, закодити сайт + мобільний додаток. Product Scope найчастіше деталізовано описують у бізнес-плані по розділах разом із необхідною ін-

формацією: що, коли і по якій ціні буде реалізоване. Обсяг продукту також розглядається як додатковий документ у завданні виконання проєкту/звіті про роботу (SOW).

Управління обсягом проєкту (Project scope management)

Ця область знань застосовується в рамках планування, моніторингу та контролю за групами/фазами процесу та охоплює збір і визначення детальних вимог проєкту, створення ключових проєктних документів, а також перевірку та контроль загальних продуктів, послуг або результатів проєкту.

Похибка (Project Deviation)

Deviation – це відхилення, вихід за межі встановлених вимог. До відхилень відносяться випадки, коли результат роботи не задовольняє вимогам або необґрунтовано їх перевищує.

Життєвий цикл проєкту (Project lifecycle)

Включає в себе ряд етапів проєкту: ініціювання, планування, виконання, моніторинг, контроль і закриття. Фази можуть бути ще подрібнені різними способами, а цикл може бути визначений більш, ніж одним фактором, зважаючи на організаційну структуру галузі.

Проєктні ризики (Project Risks)

Проєктні ризики (Project Risks) – можливість виникнення непередбачених ситуацій або ризикових подій у проєкті, які можуть негативно або ж навпаки позитивно впливати на досягнення цілей проєкту.

Ризики проєкту характеризується: джерелами і характеристиками подій, які потенційно можуть впливати на його виконання; ймовірності появи таких подій; можливим збитком проєкту й оцінкою його впливу на проєкт.

Ризик – це потенційна, чисельно вимірювана можливість виникнення несприятливих ситуацій і пов'язаних з ними наслідків у вигляді втрат і збитків.

Проєктний ризик – це небезпека небажаних відхилень від очікуваного результату у майбутньому, з розрахунку яких приймаються рішення в сьогоденні.

Базис проєкту (Project Baseline)

Project Baseline – це стрижневі параметри проєкту, які фіксуються на початковому етапі після узгодження їх з усіма учасниками проєкту, така собі «точка опори» для всього подальшого розвитку проєкту. По ходу зміни до проєкту ще можуть вноситися без проблем, звісно.

Зміни в проєкті (Project Changes)

Зміни в проєкті – це модифікація раніше узгоджених характеристик елементів проєкту. Як ми вже згадували вище, це також є переглядом базового плану проєкту. Важливо усі зміни затверджувати і документально оформлювати.

Інтегрований процес контролю змін (Integrated change control process)

Процес перегляду, затвердження, управління та спілкування, а також формального документування змін у аспектах проєкту, таких як результати, активи, документи, плани.

Календарний план проєкту (Project Schedule)

Календарний план проєкту – це перелік запланованих робіт проєкту з розписаними термінами виконання робіт на планові дати та відповідальними особами, підготовлений у відповідній формі, визначеній планом управління проєктом **(Project management plan)**. Додатково у календарному плані проєкту можуть бути зазначені контрольні (ключові) події («віхи») проєкту.

Контрольні точки (Project Milestones)

Project Milestones – це важливі події проєкту, звершення яких є необхідною і достатньою умовою, що визначає досягнення найважливіших результатів проєкту. Зазвичай **Project Milestones** представляються у вигляді таблиці зі взаємозв'яз-

ками і термінами звершення, таким чином, щоб можна було вивести звіти по віхах **(Milestone Plan, Milestone Schedule, Master Schedule).**

Назви-синоніми: **контрольна подія, ключова подія, контрольна точка.**

Проблеми проєкту (Project Problems)

Project Problems – це будь-які функціональні, технічні або бізнес-питання, які виникли у процесі реалізації проєкту й вимагають вивчення, знаходження оптимального шляху рішення і, врешті-решт, саме вирішення **(Problem Solving)**, для того щоб проєкт міг йти так, як заплановано.

Рішення проблем (Problem Solving)

Вирішення проблем (Problem Solving) – означає побудувати ієрархію послідовних систематичних процедур, за допомогою яких аналізуватимуться і вирішуватимуться різноманітні проблемні ситуації.

Управління проєктами (Project management/PM)

Це процес, коли керівники проєктів та члени команди проєкту виконують свої обов'язки, використовуючи специфічні процеси, свої знання та навички, методи, інструменти та деякі вхідні параметри для успішного виконання завдань і вимог проєкту.

На початку будь-якого проєкту, перш ніж дійсно розпочати будь-яку роботу, існують конкретні процеси, які виконуються з метою визначення нового або існуючого проєкту з метою отримання схвалення для продовження цього проєкту. Це фаза, на якій викладені всі деталі та вимоги, а також визначено та виділено фінансові ресурси. На цьому етапі також визначені всі зацікавлені сторони (особи, які будуть впливати на рішення, діяльність або результати проєкту), які будуть залучені до проєкту. Метою цього етапу є також забезпечення чіткого визначення цілей проєкту та встановлення

очікуваного результату. Ця фаза називається «групою ініціюючих процесів» у сфері управління проєктами і включає кілька конкретних процесів.

Коли розпочинається фаза ініціації, представник замовника проводить перемовини зі спеціалістами з продажів компанії. Кожен з них описує поточний стан – потенційний замовник розказує про предметну галузь та ситуацію, що призвела до виникнення потреби в поточному замовленні, а представник компанії розказує, який у поточної компанії є досвід розробки схожих в якихось пунктах проєктів. Після чого формується специфікація майбутнього проєкту у вигляді його статуту.

Статут проєкту (Project charter) – це перший офіційний документ для прогнозування/оцінки діяльності за проєктом. **Project charter** створюється особою, яка ініціює проєкт. Створення цього документу дозволяє запустити проєкт і розпочати використання ресурсів в організації. У цьому документі встановлюються такі параметри, як дата початку та закінчення, а також ключові деталі проєкту високого рівня (включаючи припущення та обмеження) для отримання офіційного схвалення проєкту від керівництва організації.

План управління проєктом (Project management plan)

Після ухвалення Статуту Проєкту (**Project charter)** створюється цей первинний документ, який містить інформацію про статут проєкту і описує всі параметри проєкту. **Project management plan** може бути дуже деталізованим або, навпаки, узагальненим.

Project management plan повинен містити базову інформацію про проєкт, графік та витрати. Він також повинен включати окремі плани, що деталізують: сфери застосування, вимоги, графік, вартість, якість, вдосконалення процесу, управління людськими ресурсами, зв'язок, ризик, закупівлі та управління акціонерами.

Project management plan також може включати детальну документацію про те, як буде виконуватися робота і ким саме, процеси для впровадження, управління проєктами, прийняття рішень, обрані методи, використані інструменти та методи, специфічні умови, вимоги до конфігурації, способи комунікації, обрані методи зменшення ризику.

Додаткові документи можуть бути включені в допоміжні проєктні документи, однак вони не будуть частиною офіційного плану управління проєктом.

Кожна задача містить у собі невелику частину всієї роботи, яка матиме кінцевий результат. Приділіть увагу формулюванню завдань. Наприклад, назва задачі «Написати додаток» явно не кращий варіант, тому що неясно, яким має бути додаток, кращими будуть чіткіші назви типу «Поміняти форму кнопки з квадрата на коло».

Ще одна перевага задач – вони показують, в якому стані знаходиться робота по проєкту. Завжди можна подивитися, що вже зроблено, а що залишилося. Можливість фільтрації задач по типу допоможе визначити те ж саме, але вже для якогось конкретного напрямку. У той час як фільтрація по пріоритетності показує, як багато ключових завдань було зроблено.

1.2. Встановлення Yii

Ви можете встановити Yii двома шляхами: використовуючи менеджер пакунків Composer або завантаживши файл архіву. Перший варіант є бажанішим, тому що дозволяє встановлювати нові розширення або оновлювати Yii простим виконанням однієї команди.

Після стандартного встановлення Yii ми отримуємо як фреймворк, так і шаблон проєкту. Шаблон проєкту – це робочий проєкт Yii, в якому реалізовано деякий базовий функціонал, такий як система входу/виходу користувачів, форма зво-

ротного зв'язку і т. д. Його код організовано в рекомендований спосіб. Таким чином, це може слугувати гарною відправною точкою для ваших проєктів.

У цьому та наступних кількох розділах буде описано, як встановити Yii з так званим *Базовим шаблоном проєкту* та як реалізувати нові можливості на базі цього шаблону. Також Yii надає інший шаблон із назвою Розширений шаблон проєкту, який краще використовувати у середовищі розробки в команді для розробки складних додатків.

Info: Базовий шаблон проєкту підходить для розробки 90 відсотків веб-додатків. Він відрізняється від Розширеного шаблону проєкту в основному організацією коду. Якщо ви малознайомі з Yii, наполегливо рекомендується використовувати Базовий шаблон проєкту через його простоту, але й достатньою функціональністю.

Встановлення за допомогою Composer

Якщо у вас все ще не встановлено Composer, то це можна зробити за допомогою інструкції на getcomposer.org. Користувачам Linux та Mac OS X потрібно виконати наступні команди:

```
curl -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer
```

При роботі з Windows, необхідно завантажити та запустити Composer-Setup.exe.

В разі наявності проблем або якщо вам необхідна додаткова інформація, зверніться до документації Composer.

Якщо ж Composer вже було встановлено раніше, переконайтесь, що використовуєте його останню версію. Ви можете оновити Composer простою командою `composer self-update`.

Після встановлення Composer, встановити Yii можна виконавши наступну команду з директорії, яка доступна через Web:

```
composer global require "fxp/composer-asset-plugin:^1.4.1"
composer create-project —prefer-dist yiisoft/yii2-app-basic basic
```

Перша команда встановить плагін ресурсів composer (composer-asset-plugin), що дозволить керувати залежностями пакунків Bower та NPM за допомогою Composer. Цю команду потрібно виконати лише один раз. Друга команда встановить Yii у директорію під назвою basic. За бажанням, ви можете обрати інше ім'я для директорії.

Note: Під час встановлення може статися так, що Composer запитає облікові дані від вашого профілю на GitHub, через встановлені обмеження запитів GitHub API. Це є нормальним, оскільки Composer повинен отримати багато інформації для всіх пакунків із GitHub. Надання облікових даних профілю GitHub збільшить кількість запитів до API, потрібних для подальшої роботи Composer. Для більш детальної інформації, будь ласка, зверніться до документації Composer.

Tip: Якщо ви хочете встановити останню нестабільну версію Yii, ви можете виконати наступну команду, яка додає опцію stability:

```
composer create-project —prefer-dist —stability=dev yiisoft/yii2-app-basic basic
```

Варто зауважити, що нестабільну версію Yii не можна використовувати на робочому сервері, оскільки вона може порушити виконання робочого коду.

Встановлення з архіву

Встановлення Yii з архіву складається з трьох кроків:

1. Завантажте архів за адресою yiiframework.com.
2. Розпакуйте архів в директорію, доступну через Web.
3. Відредагуйте файл конфігурації config/web.php – необхідно ввести таємний ключ до пункту cookieValidationKey (це виконується автоматично при вставленні Yii через Composer):

```
4. // !!! встановити таємний ключ до наступного пункту-
(якщо порожній) – це необхідно для перевірки кукі
'cookieValidationKey' => 'enter your secret key here',
```

Інші параметри встановлення

Наведені вище інструкції показують, як встановити Yii та створити базовий веб-додаток, який працює «з коробки». Цей підхід є гарною відправною точкою для більшості проєктів, як малих, так і великих. Це особливо підходить для тих, хто тільки розпочинає вивчати Yii.

Але є ще й інші варіанти встановлення:

- Якщо вам потрібен тільки один фреймворк і ви хотіли б створити додаток з нуля, використовуйте інструкцію, що описана в розділі Створення додатка з нуля.
- Якщо ви хочете розпочати з більш складного додатка, який краще підходить для роботи в команді, використовуйте Розширений шаблон проєкту.

Перевірка встановлення

Після успішного встановлення ви можете налаштувати свій веб-сервер (див. наступний розділ) або використати вбудований веб-сервер PHP, виконавши наступну консольну команду із директорії web:

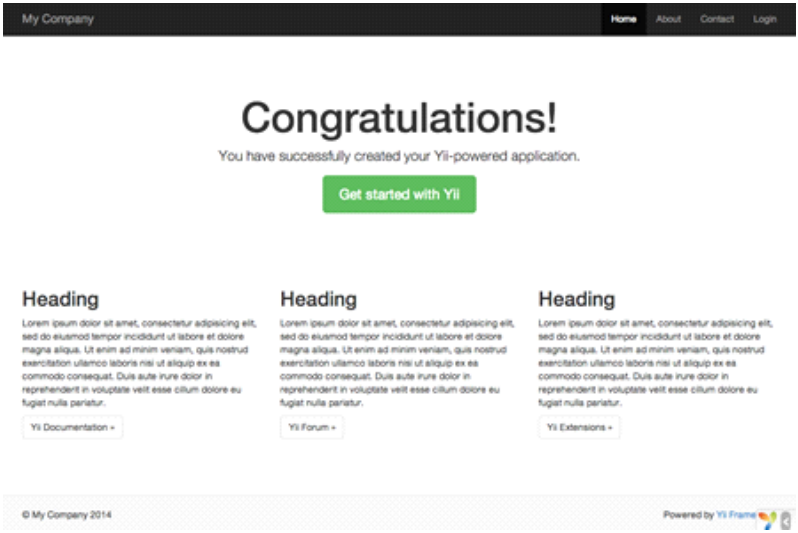
```
php yii serve
```

Note: За замовчуванням, HTTP-server буде прослуховувати порт 8080. Проте, якщо цей порт вже використовується або ви бажаєте таким чинов використовувати кілька додатків одразу – ви можете встановити, який саме порт використовувати. Тільки додайтие аргумент —port:

```
php yii serve —port=8888
```

Тепер ви можете використати свій браузер для доступу до встановленого Yii додатку за наступним посиланням:

```
http://localhost:8080/
```



Ви повинні побачити сторінку із привітанням «Congratulations!» у вашому браузері. Якщо ж ні – будь ласка, перевірте, чи задовольняють налаштування PHP вимоги Yii. Це можна зробити одним із наведених способів:

- Скопіюйте файл /requirements.php до /web/requirements.php та використайте браузер для доступу до URL http://localhost/requirements.php
- Виконайте наступні команди в консолі:

```
• cd basic
• php requirements.php
```

Необхідно налаштувати PHP таким чином, щоб він відповідав мінімальним вимогам Yii. Основна вимога – PHP версії 5.4

або вище. Якщо ваш додаток працює з базою даних, необхідно встановити розширення PHP PDO та відповідний драйвер (наприклад, pdo_mysql для MySQL).

Налаштування веб-серверів

Info: Можете пропустити даний підрозділ, якщо ви тільки розпочали знайомитися з фреймворком і не розгортаєте його на робочому сервері.

Додаток, встановлений за інструкціями, наведеними вище, буде працювати одразу як з Apache HTTP server, так і з Nginx HTTP server на Windows, Mac OS X чи Linux із встановленим PHP 5.4 або вище. Yii 2.0 також сумісний із віртуальною машиною Фейсбука HHVM, однак є деякі крайні випадки, де HHVM поводиться інакше, ніж рідний PHP, тому ви повинні бути дуже уважними при використанні HHVM.

На робочому сервері вам напевно захочеться змінити URL додатку з http://www.example.com/basic/web/index.php на http://www.example.com/index.php. Для цього необхідно змінити кореневу директорію в налаштуваннях веб-сервера на basic/web. Додатково можна сховати index.php із URL, як це описано у розділі Маршрутизація та створення URL. Далі буде описано, як налаштувати Apache або Nginx для цих цілей.

Info: Встановлюючи basic/web кореневою директорією веб-сервера, ви забороняєте кінцевим користувачам доступ до приватного коду додатка та важливих даних, які знаходяться на одному рівні з basic/web. Це робить додаток більш захищеним.

Info: Якщо додаток працює на хостингу, де немає доступу до налаштувань сервера, ви все ще можете змінити структуру додатка для покращення безпеки, як описано в розділі - Робота на віртуальному хостингу.

Рекомендовані налаштування Apache

Додайте наступний код до файлу конфігурації httpd.conf веб-сервера Apache або в конфігурацію віртуального хоста.

Не забудьте замінити path/to/basic/web на коректний шлях до basic/web.

```
    # Встановлюємо кореневою директорією "basic/web"
    DocumentRoot "path/to/basic/web"

    <Directory "path/to/basic/web">
        # використаємо mod_rewrite для підтримки гарних URL
        RewriteEngine on
          # Якщо запитуваний файл або директорія існують - звертаємось до них напряму
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteCond %{REQUEST_FILENAME} !-d
         # Якщо ні - перенаправляємо запит на index.php
        RewriteRule . index.php

        # ...інші налаштування...
    </Directory>
```

Рекомендовані налаштування Nginx

Для використання Nginx вам потрібно встановити PHP як FPM SAPI. Використовуйте наступні параметри Nginx, замінивши path/to/basic/web на коректний шлях до basic/web, а mysite.test на актуальний домен.

```
    server {
        charset utf-8;
        client_max_body_size 128M;

        listen 80; ## "слухаємо порт" для ipv4
        #listen [::]:80 default_server ipv6only=on; ## "слухаємо порт" для ipv6
```

```
        server_name mysite.test;
        root       /path/to/basic/web;
        index      index.php;

        access_log  /path/to/basic/log/access.log;
        error_log  /path/to/basic/log/error.log;

        location / {
            # Перенаправляємо всі запити на index.php, якщо це не наявна директорія або файл
            try_files $uri $uri/ /index.php?$args;
        }

        # розкоментуйте рядки нижче для запобігання обробки звернень Yii до не наявних статичних файлів
        #location ~ \.(js|css|png|jpg|gif|swf|ico|pdf|mov|fla|zip|rar)$ {
        #    try_files $uri =404;
        #}
        #error_page 404 /404.html;

        location ~ \.php$ {
            include fastcgi.conf;
            fastcgi_pass   127.0.0.1:9000;
            #fastcgi_pass unix:/var/run/php5-fpm.sock;
            try_files $uri =404;
        }

        location ~ /\.(ht|svn|git) {
            deny all;
        }
    }
```

Використовуючи дану конфігурацію, встановіть cgi.fix_pathinfo=0 у файлі php.ini, щоб запобігти зайвим системним викликам stat().

Врахуйте також, що при використанні HTTPS необхідно задавати fastcgi_param HTTPS on; щоб Yii міг коректно визначати захищене з'єднання.

Виконання додатків

Після встановлення Yii базовий додаток буде доступний або по URL http://hostname/basic/web/index.php, або по http://hostname/index.php, в залежності від налаштування веб-сервера. Даний розділ – загальне введення в організацію коду, вбудований функціонал і опрацювання запитів додатком Yii.

Info: Для спрощення далі у вказівках припускається, що Yii встановлений в директорію basic/web, яка, в свою чергу, встановлена, як коренева директорія в налаштуваннях веб-сервера. В результаті, звернувшись до URL на зразок http://hostname/index.php, ви отримаєте доступ до додатка. Будь ласка, відредагуйте URL-адреси, наведені у прикладах відповідно до ваших потреб.

Функціонал

Встановлений базовий додаток складається з чотирьох сторінок:

- домашня сторінка, відображається при переході по URL http://hostname/index.php;
- сторінка «About» ("Про нас");
- сторінка «Contact», що відображає форму зворотнього зв'язку, через яку користувач може звернутися до розробника за допомогою електронної пошти;
- сторінка «Login», на якій відображається форма для аутентифікації користувачів. Спробуйте увійти з логіном/паролем «admin/admin». Зверніть увагу на зміну пункту «Login» в головному меню на «Logout».

Ці сторінки використовують спільну шапку та футер. Шапка містить головне меню, за допомогою якого здійснюється навігація по сайту.

У нижній частині вікна ви зможете бачити системні повідомлення Yii – налагоджувальну інформацію, повідомлення про помилки, запити до бази даних і т. п. Відображенням даної інформації керує вбудований налагоджувач, він записує і відображає інформацію про хід виконання додатка.

Крім веб-додатка, існує консольний скрипт Yii, що розташований у базовій директорії додатка. Цей скрипт може бути використаний для виконання фонових завдань або завдань обслуговування додатка. Все це описано у розділі Консольні команди.

Структура додатка Yii

Нижче наведений перелік основних директорій і файлів вашого додатка (вважаємо, що додаток встановлений в директорію basic):

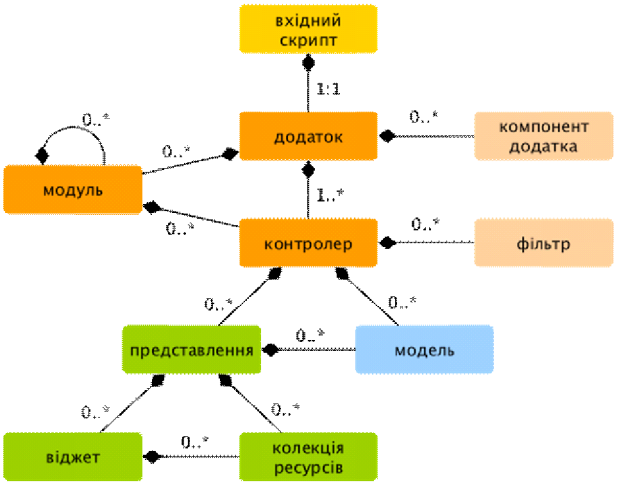
basic/	базова директорія додатка
composer.json	використовується Composer'ом, містить опис додатка
config/	містить конфігураційні файли
console.php	конфігурація консольного додатка
web.php	конфігурація веб-додатка
commands/	містить класи консольних команд
controllers/	містить класи контролерів
models/	містить класи моделей
runtime/	містить файли, які генерує Yii під час виконання додатку (журнали, кеш і т. п.)
vendor/	містить пакунки Composer'а і, власне, сам фреймворк Yii
views/	містить представлення додатка

web/	базова веб-директорія, містить файли, доступні через Web
assets/	містить файли ресурсів (javascript та css)
index.php	вхідний (або bootsrap) скрипт для додатка Yii, з нього розпочинається виконання додатка
yii	скрипт виконання консольних команд Yii

У цілому, додаток Yii можна розділити на дві категорії файлів: розміщені в basic/web і розміщені в інших директоріях. Перша категорія доступна через HTTP (наприклад, браузером), друга недоступна зовні, та і не повинна бути, оскільки містить службову інформацію.

В Yii реалізований шаблон проєктування модель-представлення-контролер (MVC), що відображається на вищезазначеній структурі директорій додатка. В директорії models знаходяться всі класи моделей, у views розміщені всі скрипти представлень, а в директорії controllers – всі класи контролерів додатка.

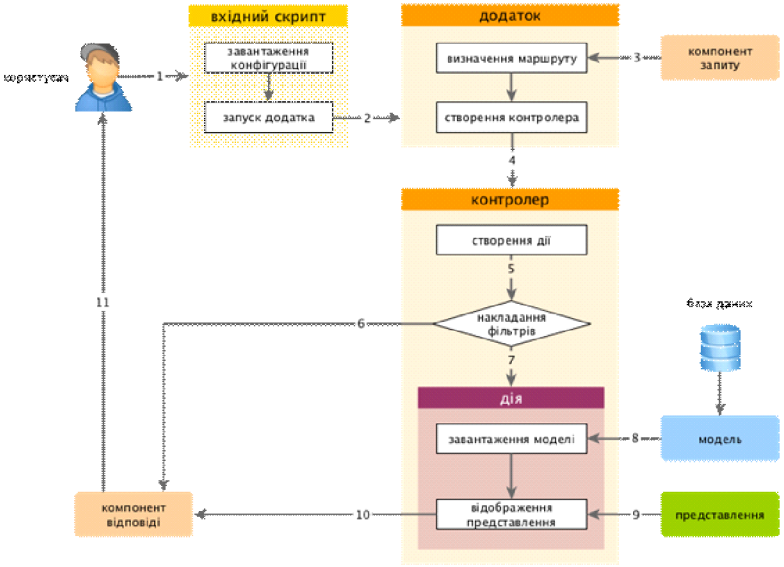
Діаграма демонструє статичну структуру додатка.



У кожному додатку Yii є місце входження в додаток, web/index.php – це єдиний PHP-скрипт, доступний для виконання через Web. Він отримує вхідний запит і створює екземпляр додатка. Додаток опрацьовує запит з допомогою його компонентів і відправляє запит елементам MVC. Віджети використовуються у представленнях для побудови складних та динамічних елементів інтерфейсу користувача.

Життєвий цикл запиту

На діаграмі показано, як додаток відпрацьовує запит.



1. Користувач робить запит до вхідного скрипту web/index.php.

2. Вхідний скрипт завантажує конфігурацію додатка та створює екземпляр додатка для наступного опрацювання запиту.

3. Додаток визначає маршрут запиту за допомогою компонента запиту додатка.

4. Додаток створює екземпляр контролера для виконання запиту.

5. Контролер, у свою чергу, створює дію і накладає на неї фільтри.

6. Якщо хоч один фільтр поверне помилку – виконання додатка зупиняється.

7. Якщо всі фільтри пройдені – додаток виконується.

8. Дія завантажує модель даних, скоріше за все, з бази даних.

9. Дія формує представлення та відображає в ньому дані (в т. ч. й отримані із моделі).

10. Сформований результат передається компоненту відповіді додатка.

11. Компонент «відповіді» відправляє готовий результат роботи додатка браузеру користувача.

Етапи роботи

1. Проаналізувати вимоги до проєкту розробки програмного інструментарію менеджера проєктів за допомогою фреймворку Yii2, створити роадмап у будь-якому інструменті, естиміювати необхідні до реалізації функції, визначити пріоритети реалізації з замовником та узгодити milestones.

2. Встановити фреймфорк Yii2 та composer за інструкцією https://yiiframework.com.ua/uk/doc/guide/2/start-installation/.

3. Навести результати у звіті з указанням теми, мети, завдання, теоретичних відомостей, що знадобилися у ході виконання роботи, ходу роботи та результатів виконання, використаної літератури.

4. Зробити висновки про здобуті в ході роботи навички.

Контрольні питання

1. Які функції виконує менеджер проєкту на етапі ініціації?

2. Які документи містять вимоги до проєкту?

3. Назвіть основні принципи ефективної роботи з замовниками.

ЛАБОРАТОРНА РОБОТА № 2

***Тема*: Управління плануванням проєкту**

Мета роботи: вивчити основні сутності проєктного менеджменту та створити повну атрибутивну модель бази даних для їх реалізації.

Завдання: створити повну атрибутивну модель (Fully Attributed model) бази даних програмного забезпечення для управління проєктами, що буде містити наступні сутності: проєкти, користувачів, ролі, таски (features, bugs, support), статуси тасків, ризики, спринти, а також усі необхідні сутностям атрибути та зв'язки між ними.

Короткі теоретичні відомості

Після ухвалення Статуту Проєкту (Project charter) створюємо План управління проєктом (Project management plan), який містить інформацію про статут проєкту і описує всі параметри проєкту. Project management plan може бути дуже деталізованим або узагальненим.

Project management plan повинен містити базову інформацію про проєкт, графік та витрати. Project management plan також може включати детальну документацію про те, як буде виконуватися робота і ким саме, процеси для впровадження, управління проєктами, прийняття рішень, обрані методи, використані інструменти та методи, специфічні умови, вимоги до конфігурації, способи комунікації, обрані методи зменшення ризиків.

Управління комунікаціями проєкту (Project communications management). Ця область знань охоплює планування, виконання і моніторинг контроль груп/фаз процесів, а також відповідає за планування, управління, моніторинг і контроль комунікацій проєкту.

Управління витратами проєкту (Project cost management) – область знань, що охоплює планування, оцінювання, бюджетування та контроль усіх витрат у рамках проєкту, а також поєднує етапи планування, моніторингу та контролю за проєктом.

Управління проєктом (Project governance) – важливий елемент у будь-якому проєкті і є моделлю організації, яка включає життєвий цикл проєкту. Ця структура окреслює проєктним групам процеси, інструменти, структури та механізми прийняття рішень для управління, підтримки, моніторингу та контролю проєктів для успішних результатів.

Управління людськими ресурсами проєкту (Project human resource management). Планування, придбання, розробка та управління всіма людськими ресурсами потрапляє в цю область знань. Область знань з управління людськими ресурсами пересікається з групою процесу планування та виконання. Управління інтеграцією проєкту (Project integration management). Ця область знань перекривається у всіх п'яти (5) групах процесів (ініціювання, планування, виконання, моніторинг/контроль і закриття). Вона передбачає створення початкових проєктних документів, а також керівництво, управління та контроль всієї проєктної роботи, здійснення зміни контролю під час закриття проєкту/фази. Управління проєктами закупівель (Project procurement management). Ця область знань охоплює планування, проведення, контроль і закриття всіх закупівель проєкту. Вона поширюється на всі етапи планування, виконання, моніторингу та контролю та закриття. Управління якістю проєкту (Project quality management). Три групи/фази процесу (планування, виконання та моніторинг та контроль) перетинаються цією областю знань. Більшість гнучких методологій націлені на мінімізацію ризиків шляхом зведення розробки до серії коротких циклів, званих ітераціями

або спринтами, які зазвичай тривають два-три тижні. Кожна ітерація – це програмний проєкт у мініатюрі, який включає в себе всі завдання, необхідні для видачі приросту за функціональністю: планування, аналіз вимог, проєктування, програмування, тестування і документування. Хоча окрема ітерація, як правило, недостатня для випуску нової версії продукту, мається на увазі, що гнучкий програмний проєкт готовий до випуску в кінці кожної ітерації. Після закінчення кожної ітерації команда виконує переоцінку пріоритетів розробки.

Етапи роботи

1. Скласти опис до декількох систем комунікацій, проаналізувати надані можливості та ефективність даних систем.

2. Вибрати систему комунікації для свого проєкту, обґрунтувати вибір тієї або іншої системи комунікацій.

3. Створити проєкт, зареєструвати в ньому команду проєкту.

4. Створити спринти, занести туди роботи, розподілити відповідальність між працівниками.

5. Виконати перший спринт, продемонструвати спринт-беклог.

Контрольні питання

1. Як здійснюються комунікації проєктної команди? Назвіть основні принципи ефективної комунікації. Як ви плануєте їх реалізовувати?

2. Що таке гнучка методологія розробки програмного забезпечення? Назвіть її основні елементи.

3 Які ролі використовуються в проєктній команді? Чи можуть вони змінюватись від спринту до спринту?

4. Хто відповідає за результат командної роботи?

5. Які технології документування комунікацій ви можете назвати?

ЛАБОРАТОРНА РОБОТА № 3

***Тема*: Управління виконанням проєкту**

Завдання: командно розробити Project Backlog, розбити задачі на спринти та скласти для кожного з них Sprint Backlog.

Короткі теоретичні відомості

Власники продуктів мають складну задачу організації зворотного зв'язку і спілкування між учасниками команди проєкту або гілками проєктної команди, а також його представлення до збрігання в зручному для пошуку форматі. Зворотний зв'язок є найважливішою частиною життєвого циклу продукту. Ми не можемо впевнено переходити до наступної ітрації без того, щоб набувати досвіду з попредніх. Просте резервування призводить до хаосу некерованих даних, тобто ми втрачаємо владу над майбутнім напрямком розвитку нашого продукту.

Беклог (backlog) – це список усіх робіт. Можна сказати, що це щоденник загального користування.

Sprint backlog – це простий список завдань, які повинні бути виконані командою за один спринт (2 тижні) для того, щоб доставити достатнє прирощення функціональності програмного забезпечення в кінці цього спринту для справляння враження на власника продукту. Створення Sprint backlog відбувається у другій частині наради з планування спринту за участю кожного члена команди. Приділяти увагу цьому процесу дуже важливо для кращого розуміння по команді того, що повинно бути зроблено. Це дозволяє більш ефективно планувати розробку. Незважаючи на це, багато команд все ще борються з цією діяльністю. В процес залучається кожен член команди. Для цього, як показано на рис. 1, скраммайстер вибирає пункт create issue і заносить завдання, які називають учасники команди. Потім, на вкладці Agile за допомогою

Isuue Navigator, задачі розставляються за порядком виконання, будується діаграма Ганта, де визначаються критичні задачі. Після цього скрам-майстер виставляє потрібні пріоритети для задач.

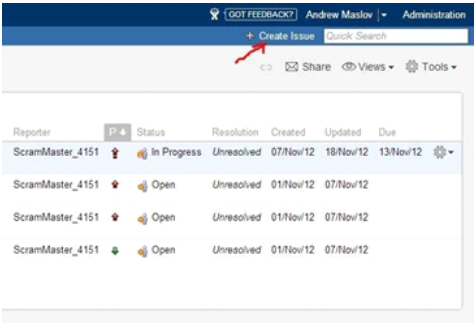


Рис. 1. Створення задач для спринт-беклогу

На цій же вкладці переходимо до вкладки Agile та вибираємо задачі на наступний спринт (рис. 2).

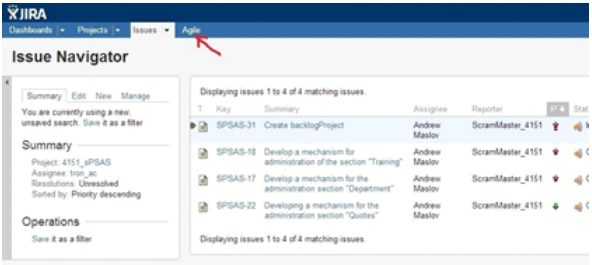


Рис. 2. Задачі на наступний спринт

Резерв спринту (Sprint backlog) – містить функціональність, обрану власником проєкту з резерву проєкту. Всі функції розбиті по задачах, кожна з яких оцінюється скрам-командою. Щодня команда оцінює обсяг роботи, який потрібно виконати для завершення спринту.

Контрольні питання

1. Що таке backlog? Які види backlog ви знаєте? Для чого вони використовуються?

2. Що таке спринт? Яка оптимальна довжина спринту? Хто її визначає? Чи може вона змінюватись під час проєкту?

3. Що важливіше за принципами гнучкої методології – якість, взаєморозуміння із замовником або добре документування?

4. Які інструменти можливі до застосування у цій роботі? Перелічіть їх основні особливості. Обґрунтуйте, чому ви використовували саме обраний Вами.

ЛАБОРАТОРНА РОБОТА № 4

***Тема*: Управління моніторингом та контролем проєкту**

Мета роботи: набути навички командного планування за допомогою використання систем планування задач.

Завдання: вибрати систему планування задач для проєкту, враховуючи особливості вашого проєкту; обґрунтувати вибір; показати хід командної роботи з системою планування задач.

Короткі теоретичні відомості

Проєкт JIRA є набором запитів і визначається згідно з вимогами організації.

Компоненти – це свого роду категорії, що дозволяють розділяти запити JIRA по різних галузях. Приклад: ведеться в JIRA проєкт з розробки комп'ютерної гри. У цьому проєкті будуть наступні компоненти: документація, користувальницький інтерфейс, ядро.

При створенні запиту/завдання в системі користувач може віднести його до одного або більшої кількості компонент. Так

само можна призначити відповідального співробітника за компонент і тоді запити будуть відразу йти на нього, а не на керівника всього проєкту.

Запит може бути представленим як помилка, задача і т. п. Кожен запит відноситься до проєкту. Приклад запиту представлений на рис. 3.

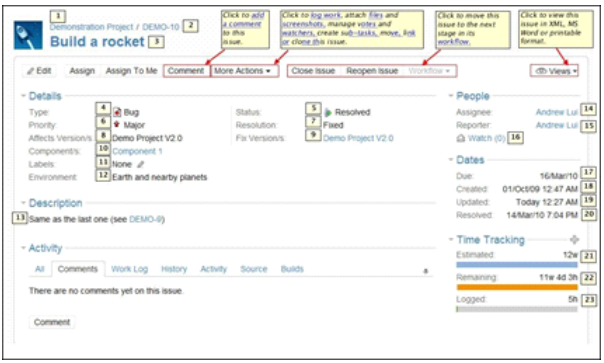


Рис. 3. Запити проєкту

Контрольні питання

Керівник проєкту дивиться на потреби проєкту в ресурсах і переглядає історію виконаних проєктів. Отримана інформація змушує його замислитися про можливі проблеми з отримання достатньої кількості ресурсів через півроку. Що з перерахованого нижче є найменш ефективним рішенням цієї проблеми?

1. Розіслати кожному функціональному менеджеру копію ресурсної гістограми.

2. Показати спонсору дані, пояснивши свою стурбованість.

3. Розробити показники, які б забезпечили раннє оповіщення про те, що проєкт залишиться скоро без ресурсів.

4. Запитати у функціональних менеджерів рекомендацій про превентивні заходи.

Великий проєкт у процесі виконання, і один з членів команди переглядає звіт за статусом проєкту. Із звіту він розуміє, що проєкт спізнюється. Подальше вивчення звіту показує, що проєкт зсувається так, що та операція, яку він повинен виконувати, пересувається на той час, коли його не буде в країні і працювати в цей час він не зможе. Для нього це дуже важливо: він зав'язаний на цьому проєкті, успіх проєкту для нього дуже важливий, і він не хоче бути причиною ще більшого запізнення. Що в такій ситуації краще всього зробити?

1. Негайно зв'язатися з керівником проєкту, повідомити йому про своє відставання від графіка і попросити рішення.
2. Включити інформацію у свій наступний звіт.
3. Додати питання в журнал відкритих питань по проєкту.
4. Негайно зв'язатися з керівником проєкту, повідомити йому про своє відставання від графіка і порекомендувати превентивні дії.

ЛАБОРАТОРНА РОБОТА № 5

***Тема*: Управління версіями проєкту**

Мета роботи: набути навички використання командної роботи з керування версіями програмного забезпечення при роботі в команді проєкту.

Завдання: вибрати систему для свого проєкту, обґрунтувати свій вибір, показати хід роботи комадою проєкту з вибраною СКВ.

Короткі теоретичні відомості

Система керування версіями (СУВ) – це система, яка зберігає зміни в одному або декількох файлах так, щоб потім можна було відновити певні старі версії. Можна керувати версіями

практично будь-яких типів файлів. Система керування версіями дозволяє повернути файли до колишнього вигляду, повернути до колишнього стану весь проєкт, порівняти зміни з якогось часу, побачити, хто останнім змінював модуль, який дав збій, хто створив проблему тощо. Взагалі, якщо користуючись СКВ, ви все зіпсували або втратили файли, все можна легко відновити.

Локальні системи управління версіями

Багато людей, щоб керувати версіями, просто копіюють файли в інший каталог (розумні ще пишуть поточну дату в назву каталогу web). Такий підхід дуже поширений, тому що простий, але він часто дає збої. Дуже легко забути, що ти не в тому каталозі, і випадково змінити не той файл, або скопіювати і перезаписати файли не туди, куди хотів.

Щоб вирішити цю проблему, програмісти розробили локальні СКВ з простою базою даних, в якій зберігаються всі зміни потрібних файлів. Однією з найбільш популярних СКВ даного типу є RCS, яка до цих пір встановлюється на багато комп'ютерів. Навіть у сучасній операційній системі Mac OS X утиліта RCS встановлюється разом з Developer Tools. Ця утиліта заснована на роботі з наборами патчів між парами змін (патчфайл, що описує відмінність між файлами), які зберігаються в спеціальному форматі на диску. Це дозволяє перестворити будь-який файл на довільний момент часу, послідовно накладаючи патчі.

Централізовані системи управління версіями

Наступною великою проблемою виявилася необхідність співпрацювати з розробниками за іншими комп'ютерами. Щоб вирішити її, були створені централізовані системи управління версіями (ЦСУВ). В таких системах, наприклад CVS, Subversion і Perforce, є центральний сервер, на якому зберігаються всі відстежувані файли, і ряд клієнтів, які отримують

копії файлів з нього. Багато років це був стандарт управління версіями. Такий підхід має безліч переваг, особливо над локальними СУВ. Приміром, всі знають, хто і чим займається в проєкті. У адміністраторів є чіткий контроль над тим, хто і що може робити, і, звичайно, адмініструвати ЦСУВ набагато легше, ніж локальні бази на кожному клієнті.

Однак при такому підході є й кілька серйозних недоліків. Найбільш очевидний – централізований сервер є вразливим місцем всієї системи. Якщо сервер вимикається на годину, то протягом години розробники не можуть взаємодіяти, і ніхто не може зберегти нові версії. Якщо ж пошкоджується диск з центральною базою даних і немає резервної копії, ви втрачаєте абсолютно все – всю історію проєкту, хіба що за винятком кількох робочих версій, збережених у робочих машинах користувачів. Локальні системи управління версіями схильні до тієї ж проблеми: якщо вся історія проєкту зберігається в одному місці, ви ризикуєте втратити все [4].

Розподілені системи контролю версій

У такій ситуації в гру вступають розподілені системи управління версіями (РСКВ). У таких системах, як Git, Mercurial, Bazaar або Darcs клієнти не просто забирають останні версії файлів, а повністю копіюють репозиторій. Тому в разі, коли «вмирає» сервер, через який йшла робота, будь клієнтський репозиторій може бути скопійований назад на сервер, щоб відновити базу даних. Щоразу, коли клієнт забирає свіжу версію файлів, створюється повна копія всіх даних.

Крім того, в більшій частині цих систем можна працювати з декількома віддаленими репозиторіями, таким чином, можна одночасно працювати по-різному з різними групами людей в рамках одного проєкту. Так, в одному проєкті можна одночасно вести кілька типів робочих процесів, що неможливо в централізованих системах [4].

Git та Subversion. У даний час більшість проєктів з відкритим вихідним кодом, а також велика кількість корпоративних проєктів використовують Subversion для управління своїм вихідним кодом. Це найпопулярніша на поточний момент система управління версіями з відкритим вихідним кодом. Крім того, вона дуже схожа на CVS – систему, яка була найпопулярнішою до Subversion. Одна з чудових особливостей Git – можливість двостороннього обміну з Subversion через інтерфейс, званий Git SVN. Цей інструмент дозволяє вам використовувати Git в якості коректного клієнта при роботі з сервером Subversion. Таким чином, ви можете користуватися всіма локальними можливостями Git, а потім зберігати зміни на сервері Subversion так, якщо б використовували Subversion локально. Тобто, ви можете робити локальне розгалуження і злиття, використовувати індекс, переміщення і відбір патчів для переносу з однієї гілки в іншу (зняття вершків) і т. д., у той час, як ваші колеги будуть продовжувати використовувати в розробці підхід часів кам'яного віку. Це хороший спосіб протягти Git в робоче оточення своєї компанії, щоб допомогти колегам розробникам стати більш ефективними, в той час як ви будете лобіювати перехід повністю на Git. Інтерфейс обміну з Subversion – це ворота в світ розподілених систем керування версіями.

Базовою командою в Git для всіх команд, що працюють з мостом, до Subversion є Git SVN. Їй передує будь-яка команда. Вона приймає досить порядне число команд, тому ми вивчимо з них ті, які найбільш часто використовуються, розглянувши кілька невеликих варіантів роботи. Найкраще тримати свою історію в якомога більш лінійному вигляді, використовуючи переміщення (перебазуватися) і уникаючи таких речей, як одночасний обмін з віддаленим репозиторієм Git. Чи не переписуйте свою історію, спробуйте відправити зміни

ще раз, а також не відправляйте зміни в паралельний Git-репозиторій, використовуваний для спільної роботи, одночасно з іншими розробниками, які використовують Git. Subversion може мати тільки одну єдину лінійну історію змін, збити з пантелику яку дуже і дуже просто. Якщо ви працюєте в команді, в якій деякі розробники використовують Git, а інші Subversion, переконайтеся, що для спільної роботи всі використовують тільки SVN-сервер. Для того, щоб спробувати цей функціонал в дії, вам знадобиться доступ з правами на запис до звичайного SVN-репозиторію. Якщо ви хочете повторити аналізовані приклади, вам потрібно зробити доступну на запис копію мого тестового репозитория. Це можна зробити без праці за допомогою утиліти SVNSync, що входить до складу останніх версій Subversion (принаймні після версії 1.4). Для цих прикладів був створений новий Subversion репозиторій на Google Code, який був частковою копією проєкту protobuf (утиліта шифрування структурованих даних для їх передачі по мережі).

Насамперед створіть новий локальний репозиторій Subversion:

```
$ mkdir /tmp/test-svn.
$ svnadmin create /tmp/test-svn.
```

Потім дозвольте всім користувачам змінювати revprops – найпростішим способом зробити це буде додавання сценарію pre-revprop-change, який завжди завершується з кодом 0:

```
$ cat /tmp/test-svn/hooks/pre-revprop-change #!/bin/sh exit 0;
$ chmod +x /tmp/test-svn/hooks/pre-revprop-change.
```

Тепер ви можете синхронізувати проєкт зі своєю локальною машиною, викликавши SVNSync ініціалізації з параметрами, які задають вихідний і цільовий репозиторій:

```
$ svnsync init file:///tmp/test-svn http://progit-example.googlecode.com/svn/
```

Ця команда підготує процес синхронізації. Потім треба клонувати код, виконавши:

```
$ svnsync sync file:///tmp/test-svn Committed revision 1.
Copied properties for revision 1. Committed revision 2.
Copied properties for revision 2.
Committed revision 3.
...
```

Хоча виконання цієї операції і може зайняти всього кілька хвилин, однак, якщо ви спробуєте скопіювати початковий репозиторій в інший віддалений репозиторій, а не в локальний, то процес займе майже годину, хоча в цьому проєкті менше ста коммітів. Subversion змушений клонувати ревізії по одній, а потім відправляти їх в інший репозиторій – це неефективно, проте це єдиний простий спосіб виконати дану дію. Тепер, коли у вашому розпорядженні є SVN-репозиторій, для якого ви маєте право на запис, давайте виконаємо типові дії по роботі з СУВ. Почнемо з команди git svn clone, яка імпортує весь SVN-репозиторій у локальний Git-репозиторій. Пам'ятайте, що якщо ви робите імпорт зі справжнього віддаленого SVN-репозиторія, вам треба замінити файл: /// TMP / ТестSVN на реальну адресу вашого SVN-репозиторія:

```
$ git svn clone file:///tmp/test-svn -T trunk -b branches -t tags
Initialized empty Git repository in /Users/schacon/projects/testsvnsync/
svn/.git/ r1 = b4e387bc68740b5af56c2a5faf4003 ae42bd135c
(trunk)
A m4/acx_pthread.m4
A m4/stl_hash.m4
...
r75 = d1957f3b307922124eec6314e15bcda59e3d9610 (trunk)
Found possible branch point: file:///tmp/test-svn/trunk => \
file:///tmp/test-svn /branches/my-calc-branch, 75
Found branch parent:
(my-calc-branch)
```

d1957f3b307922124eec6314e15bcda59e3d9610
Following parent with do_switch Successfully followed parent r76 = 8624824ecc0badd73f40ea2f01fce51894189b01 (my-calc-branch)
Checked out HEAD:
file:///tmp/test-svn/branches/my-calc-branch r76

Ця команда еквівалентна виконанню для зазначеного вами URL двох команд – git SVN ініціалізації, а потім git SVN вибірки. Процес може зайняти деякий час. Тестовий проєкт має всього лише близько 75 коммітів, і коду там не дуже багато, так що, швидше за все, вам доведеться почекати всього декілька хвилин. Однак Git повинен окремо перевірити і виконати commit для кожної версії. Для проєктів, що мають історію з сотнями і тисячами змін, цей процес може зайняти кілька годин або навіть днів. Частина команди – -T trunk -b branches -t tags – повідомляє Git, що цей SVN-репозиторій відповідає стандартним угодам про розгалуження і мітки. Якщо ви використовуєте нестандартні імена: trunk, branches і tags, а якісь інші, то повинні змінити ці параметри відповідним чином. У зв'язку з тим, що такі угоди є загальноприйнятими, ви можете використовувати короткий формат, замінивши всю цю частину на -s, яка замінює собою всі ці параметри. Наступна команда еквівалента попередній:

$ git svn clone file:///tmp/test-svn -s

Таким чином, ви повинні були отримати коректний Git-репозиторій з імпортованими гілками і мітками:

$ git branch -a * master my-calc-branch tags/2.0.2 tags/release-2.0.1 tags/release-2.0.2 tags/release-2.0.2rc1 trunk

Важливо відзначити, що ця утиліта іменує ваші посилання на віддалені ресурси по-іншому. Коли ви клонуєте звичайний репозиторій Git, ви отримуєте всі гілки з віддаленого сервера на локальному комп'ютері у вигляді: походження/[філіал] – в просторі назв з ім'ям віддаленого сервера. Однак

git SVN вважає, що у вас не буде безлічі віддалених джерел даних і зберігає всі посилання на всяке, що знаходиться на віддаленому сервері, без простору назв. Для перегляду всіх імен посилань ви можете використовувати службову команду Git show-ref.

Контрольні питання

1. Що таке версія програмного забезпечення?
2. Назвіть кілька систем керування версіями. Які переваги використання кожної з них?
3. Які принципи групової динаміки ви використовували в цій роботі?

ЛАБОРАТОРНА РОБОТА № 6

***Тема*: Управління завершенням проєкту**

Мета роботи: набути навички використання засобів для командної роботи.

Завдання: продемонструвати фрагмент командної роботи за допомогою інструментів вибраного командою фреймворку.

Короткі теоретичні відомості

Сучасні середовища розробки дають можливість користуватися великою кількістю засобів для прискореної роботи та великим ступенем візуалізації, що дає можливість додавати нові компоненти простим перетягуванням в робочу область. Інтеграція редакторів і компіляторів для найрізноманітніших мов, засоби для роботи з базами даних, інструменти розробки специфічні для того чи іншого фреймворка, а також багато інших є вже звичними для нас і засновані на більш дрібних, але ключових речах. Створені різні інструменти розробки, з одного боку надійно працюють, а з іншого, грамотно вбудовані в інтерфейс, допомагають розробнику бути більш

продуктивним. Тим не менше, зі зростанням складності інтегрованих засобів розробки, тільки наявність подібних невеликих інструментів допомагає орієнтуватися в можливостях цих систем. У зв'язку з цим, сучасним розробникам необхідна така середа, що дозволить зосередитися тільки на тих артефактах, які становлять інтерес для користувача, і, крім цього, буде відображати тільки той функціонал системи, який дійсно важливий користувачеві. Для цього, наприклад, у фреймворці Eclipse існує плагін під назвою Mylyn. Замість того щоб встановлювати масу інструментів для роботи з артефактами, що виникають в процесі розробки, Mylyn допомагає обробляти ці артефакти в момент створення.

Створення підсумкової презентації проєкту програмного забезпечення

Мета роботи: набути навички представлення результатів роботи проєктної команди та ознайомитися з сучасними презентаційними світовими практиками.

Завдання: представити результати командної розробки проєкту ПЗ у презентаційній формі, використовуючи кращі світові практики.

Короткі теоретичні відомості

Презентація (з латинського «представлення») – один із засобів маркетингових комунікацій, метою якого є рекламне просування товарів, послуг, компанії на ринку.

Презентація – форма ділових комунікацій, спрямована на демонстрацію кінцевому споживачеві можливостей фірми, товару, послуги, з рекламною демонстрацією їх властивостей, переваг, особливостей і формування позитивного образу, напрям дій.

При розробці даного заходу необхідно враховувати так звані 5 «С» презентації, що визначають її результативність. Кожна з 5 «С» презентацій має особливе значення. Чим більше

приділяється уваги і часу даному алгоритму, тим результат буде більш ефективний.

1. *Структура презентації* – це компоненти, з яких вона складається:

- привернення уваги;
- вступна частина;
- основна частина;
- огляд;
- висновок (спонукання).

Якщо яка-небудь з перерахованих частин відсутня, ефективність презентації знижується. Якщо ж порушення допущені відразу в декількох частинах, то презентація перестає діяти.

2. *Зміст презентації* включає багатоаспектну характеристику об'єкта просування.

3. *Стиль презентації* може бути різним: високим, діловим, дружнім і т. п. Як правило, стиль визначається наступними факторами:

- зовнішній вигляд учасників;
- манера подачі матеріалу;
- атмосфера, яка панує в приміщенні;
- тема презентації.

4. *Супровід презентації*. До цього аспекту відноситься все, що оточує презентацію, всі її зовнішні складові: організація залу, розстановка столів і стільців, оптимальна кількість запрошених; використання фліпчартів, дощок, ноутбуків, слайдів. Все це суттєво впливає на ефект презентації.

5. *Ситуативне керування презентацією* передбачає встановлення контакту з аудиторією.

Техніка підготовки успішної презентації досягається за рахунок послідовного відпрацювання трьох етапів:

- планування презентації;
- підготовка і проведення репетиції перед презентацією;
- проведення презентації.

Кращі світові практики презентацій

У ролі кращих світових практик можна розглянути ***секрети*** презентацій Стіва Джобса. Він слідував деяким презентаційним принципам, які перетворили продукти Apple на справжні витвори мистецтв за ціною та продажами.

Стів Джобс використовує короткі заголовки, що зручно для преси та його клієнтів. Ці заголовки містять короткий опис всієї суті того, що відбувається, в межах 140 символів, що чимось схоже на твіти. Наприклад, «What's an iPod? 1,000 songs in your pocket» (Що таке iPod? Це 1000 пісень у вашій кишені).

27 січня на презентації iPad Стів Джобс використав наступний опис: «Our most advanced technology in a magical and revolutionary device at an unbelievable price» (Наші найпередовіші технології в магічному і революційному пристрої за неймовірною ціною). Як тільки Джобс вимовив це, на сайті Apple оновилася головна сторінка: з'явилося велике зображення iPad з цими словами поруч. Ці ж заголовки використовуються і в рекламах, і в прес-релізах.

У кожній презентації Стіва Джобса є «погані» і «хороші». На презентації iPad «поганим» був нетбук – проблема, яку треба вирішити. Джобс показав слайд з iPhone на одній стороні і MacBook на інший. У середині з'явився знак питання. Він поставив риторичне питання: чи існує пристрій, який може знаходитися між ними. «Цей новий пристрій має набагато краще виконувати деякі ключові завдання», – сказав Стів. «Деякі люди думають, що це нетбук. Але проблема нетбуків у тому, що вони не кращі. Вони повільні, мають низьку якість дисплеїв і низьку швидкість старого програмного забезпечення». Тільки після того, як він сформував проблему, Джобс представив «хорошого»: «У нас є дещо краще. Ми називаємо його iPad».

Правило трьох. Нейрофізіологи довели, що люди можуть сприймати лише три або чотири пункти інформації в короткочасній пам'яті одночасно. Тому Стів Джобс ділить продукти, ідеї та повідомлення на три частини і робить це в кожній презентації.

Ось кілька прикладів з презентації iPad:

• Apple отримує дохід від трьох продуктів: iPhone, iPod і Mac.

• У Apple три основних конкуренти в класі мобільних пристроїв: Nokia, Samsung і Sony.

• У нетбуків є три проблеми: повільність, низька якість дисплея і запуск важкого програмного забезпечення.

Візуальна простота

Усі слайди Стіва Джобса вельми наочні – це фотографії і зображення. Людина краще зберігає інформацію, якщо вона представлена в поєднанні тексту і зображень, ніж просто суцільний текст. Кожна функція представлена графічно, одна думка – один слайд.

Емоційна мова

Стів Джобс використовував образну, емоційну промову, яка діє зухвало на нашу праву півкулю мозку. Деякі критики говорять, що це межує з перебільшенням, але краще використовувати емоції, ніж застарілі, жаргонні "розумні слівця". Ось кілька прикладів з презентації iPad:

• Він більш особистий, ніж ноутбук, і розумніший, ніж смартфон.

• Це найкраще, що ви коли-небудь бачили.

• Це мрія.

• Він дуже простий.

Репетиції і вдосконалення

Стів Джобс репетирував свої презентації протягом багатьох годин протягом багатьох тижнів, щоб все було бездоган-

но. Він уважно переглядає кожен слайд і точно знає, що і як він збирається сказати. У результаті виходять легкі для сприйняття презентації. Але це приходить тільки після кількох тижнів тренування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. **Highsmith, Jim.** E-Project Management: Harnessing Innovation And Speed, Cutter consortium [Text] / J. Highsmith 2001, VOL 1, No. 1.

2. **Cockburn, A.** Agile Software Development, 2001, Addison Wesley.

3. Martin Fowler The New Methodology [Електронний ресурс]. – Режим доступу: www.martinfowler.com/articles/newMethodology.html

4. Полное руководство по Yii 2.0 | Yii PHP Framework [Електронний ресурс]. – Режим доступу: https://www.yiiframework.com › doc › guide

5. **Скляр, Д.** PHP. Сборник рецептов [Текст] : [пер. с англ.] / Д. Скляр, А. Трахтенберг. – СПб. : СимволПлюс, 2005. – 672 с.

6. **Шаркоан, С.** Progit [Текст] / С. Шаркоан. – GP : Apress, 2009. – 262 с.

7. **Канер, С.** Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений [Текст] : [пер. с англ.] / С. Канер, Д. Фолк, Е. Нгуен. – К. : ДиаСофт, 2001. – 544 с.

8. Trac and Sunversion [Электронный ресурс]. – Режим доступа: http://trac.edgewall.org.

9. **Вельма, А. М.** Вибір системи відстеження помилок в залежності від конфігурації програмного забезпечення [Текст] / А. М. Вельма, Є. Ю. Лактіонов // Вісник НТУУ "КПІ".

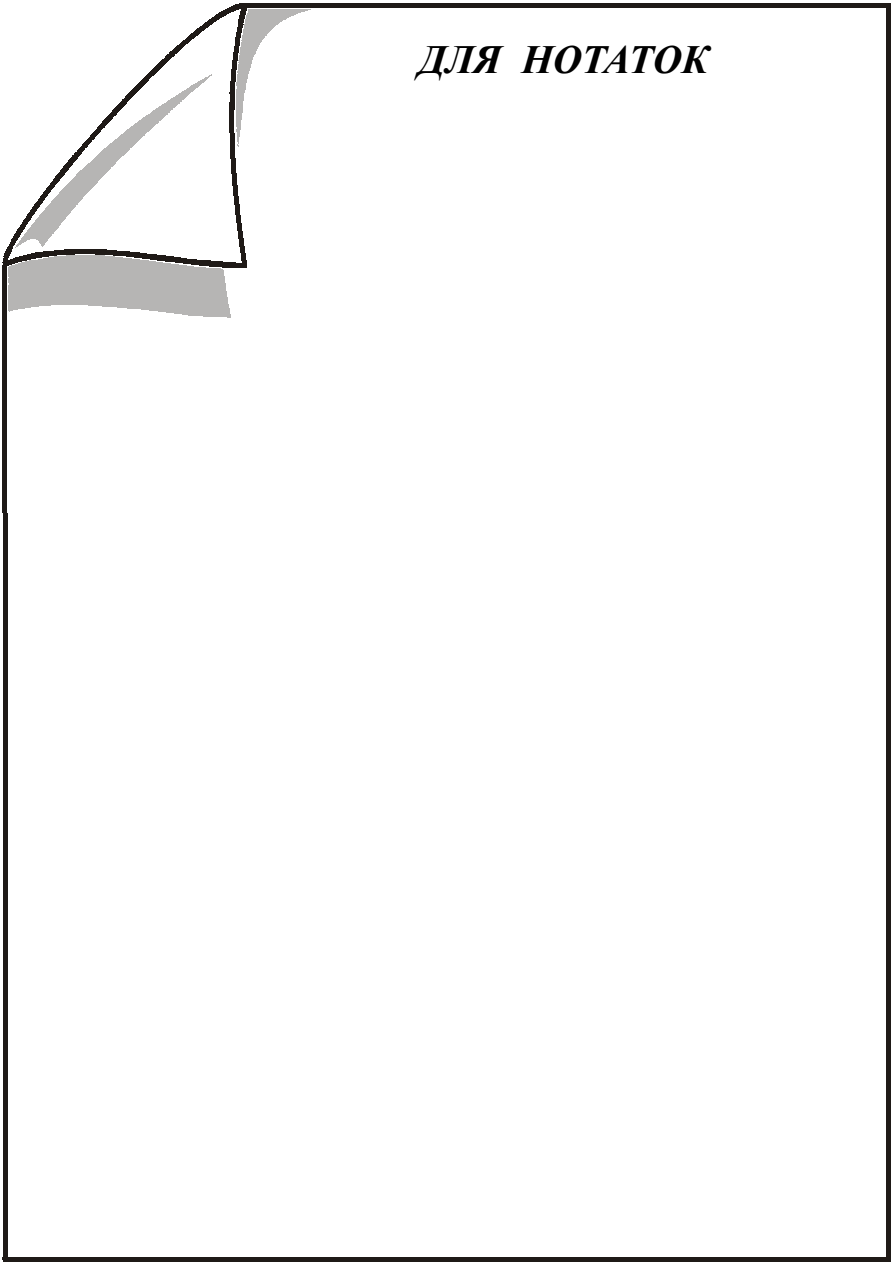
Інформатика, управління та обчислювальна техніка : зб. наук. пр. – К. : Век+, 2010. – № 52. – С. 137–141.

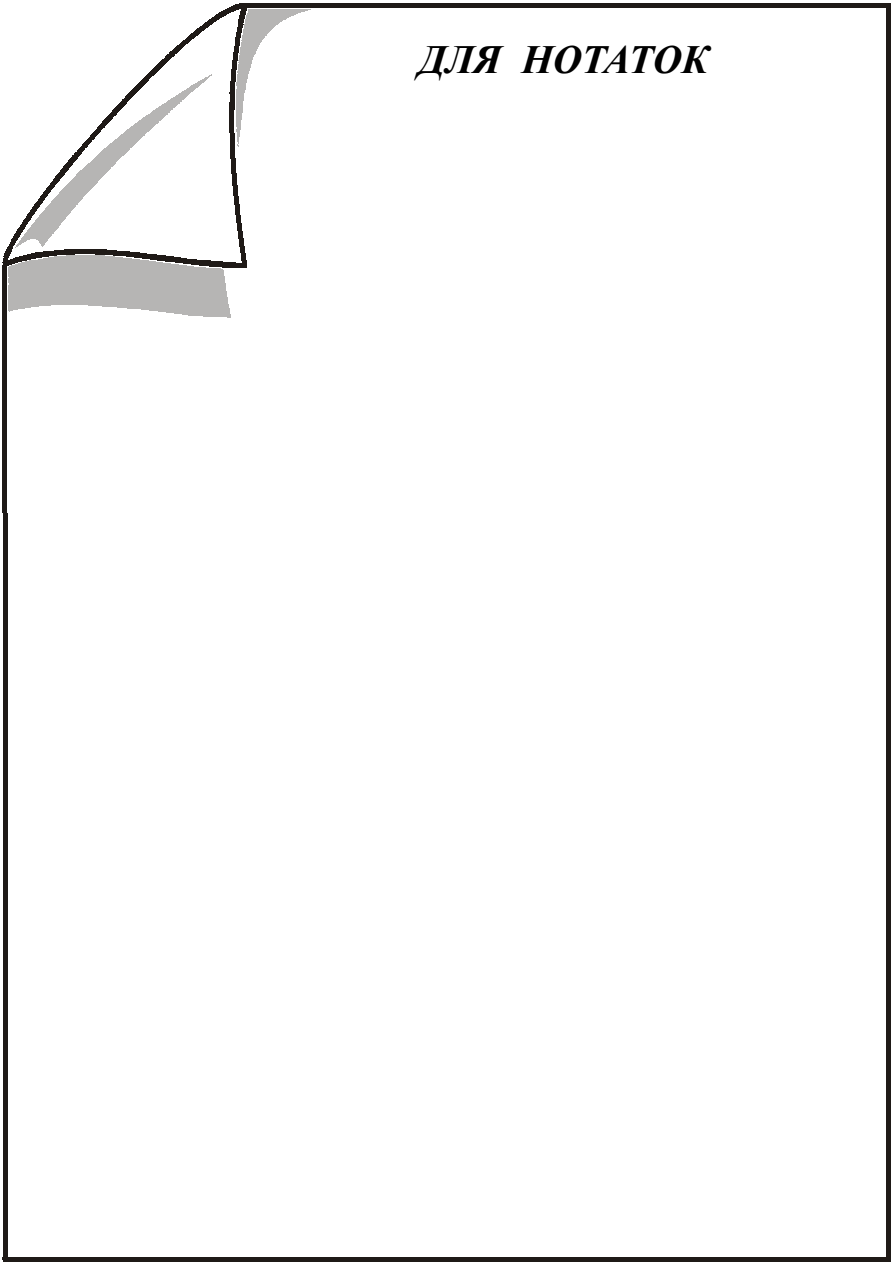
10. Пересекая границы: специфика разработки ПО распределенной командой [Электронный ресурс]. – Режим доступа: http://citforum.ck.ua.

11. Управління людськими ресурсами [Електронний ресурс]. – Режим доступу: http://kn-grup.com.

12. Управління розробкою ПЗ [Електронний ресурс]. – Режим доступу: http://www.unicyb.kiev.ua.

ЗМІСТ


ДЛЯ НОТАТОК


ДЛЯ НОТАТОК

Навчальне видання

ПОНОМАРЕНКО Тетяна Вікторівна
СУСЛОВ Сергій Віталійович

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт
з дисципліни

«МЕНЕДЖМЕНТ ПРОЄКТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Комп'ютерна правка *М. О. Паненко*
Комп'ютерне верстання *В. В. Москаленко*

Формат 60×84/16. Ум. друк. арк. 2,8. Тираж 100 прим. Вид. № 19. Зам. № 2306-54.

Видавець і виготівник Національний університет кораблебудування
імені адмірала Макарова
просп. Героїв України, 9, м. Миколаїв, 54025
E-mail : publishing@nuos.edu.ua
Свідоцтво суб'єкта видавничої справи ДК № 6402 від 19.09.2018 р.