УДК 004.891

к.т.н., доц. **Джулій В.М.** (ХмНУ)
д.т.н., доц. **Гунченко Ю.О.** (ОНУ)
Чешун Д.В. (ХмНУ)

МЕТОДИ ТА АЛГОРИТМИ РОЗРОБКИ WEB-ДОДАТКІВ

Веб-додатки оперують величезними обсягами даних, в тому числі і персональними даними (такими як номери кредитних карт), що пред'являє високі вимоги до програмного коду. У той же час, велика тривалість розробки, погодження та затвердження міжнародних і національних стандартів призводить до їх консерватизму, а також до хронічного відставання вимог і рекомендацій цих документів від сучасної практики та технології створення складних систем. Сучасні методи розробки програмних засобів, їх уніфікація та стандартизація не повною мірою враховують специфіку розробки веб-додатків з використанням вільних інтернет-технологій. Проблема полягає у відсутності чітких рекомендацій по розробці файлів вихідного коду контролерів, моделей і представлень та специфікації їх взаємозв'язків. Структурний синтез веб-додатка є багатогранною проблемою, що містить повністю невирішені питання синтезу як всієї архітектури в цілому, так і вихідного коду рівнів моделей, представлень і контролерів. Існуючі стандарти розробки регламентують тільки протоколи взаємодії веб-додатків і недостатньо враховують їх специфіку. Низький поріг входження в технологію для розробника створює можливість появи веб-додатків, програмний код яких не відповідає вимогам якості. Внаслідок цього, розроблюваний програмний код веб-додатків не уніфікований і не стандартизований. Розглянуті методи хоч і покращують такі показники якості, як структурність і повторюваність, але при цьому погіршують критерії простоти конструкції та можливість модифікацій.

На основі проведеного аналізу можна зробити висновок, що існує необхідність створення методів і алгоритмів оптимізації розробки веб-додатків, які включали б в себе питання зберігання (синтез рівня моделей) і взаємодії (синтез рівня контролерів).

Ключові слова: веб-додатки, контролери, моделі, представлення, програмний код, метод, алгоритм, структурний синтез, інтернет-технології.

Вступ. Останнім часом зростає роль інформаційних технологій в житті суспільства. На даний момент мережа Інтернет являє собою сукупність веб-додатків. Основна функція веб-додатка полягає в обробці запитів і генерації відповіді користувачеві. На процес взаємодії з користувачем накладається ряд обмежень згідно з протоколом прикладного рівня HTTP. HTTP відноситься до протоколів без встановлення з'єднання і не забезпечує зберігання стану між запитами. Найпоширенішою платформою розробки веб-додатків є LAMP, до складу якої входить операційна система Linux, веб-сервер Apache, реляційна база даних MySQL та інтерпретатор PHP. PHP як мова програмування, що входить до складу платформи, володіє низьким порогом входження, що також збільшує поширеність платформи, проте саме це є причиною створення програмного коду, що не задовольняє вимогам якості. Разом з тим, на даній платформі розробляються багатокомпонентні програмні рішення з участю багатьох десятків розробників. Веб-додатки починають оперувати величезними обсягами даних, в тому числі і надважливими персональними даними (такими як номери кредитних карт), що зумовлює високі вимоги до програмного коду. У той же час, велика тривалість розробки, погодження та затвердження міжнародних і національних стандартів призводить до їх консерватизму, а також до хронічного відставання вимог і рекомендацій цих документів від сучасної практики та технології створення складних систем. Сучасні методи розробки програмних засобів, їх уніфікація та стандартизація, не повною мірою враховують специфіку розробки на платформі LAMP з використанням вільних інтернет-технологій. В цих умовах створення методів розробки веб-додатків є актуальною задачею.

Постановка задачі. Найбільш поширена методика розробки веб-додатків – MVC. Програмні системи, що розробляється за даною методикою, діляться на три частини: класи-контролери, що реалізують взаємодію з користувачем; класи-моделі, що представляють об'єктну модель даних та методи роботи з ними; класи-представлення, що реалізують

інтерфейс. Зв'язки між зазначеними частинами програмної системи були описані, але без конкретизації. Крім того, на момент створення архітектури такі програмні системи, як веб-додатки були відсутні в принципі. У зв'язку з цим поширеним явищем стало невірне трактування архітектури, що призвело до появи великої кількості не уніфікованого програмного коду, що не задовольняють вимогам якості.

Проблема полягає у відсутності чітких рекомендацій з розробки файлів вихідного коду контролерів, моделей і представлень та специфікації їх взаємозв'язків. На основі проведеного аналізу можна зробити висновок, що існує необхідність створення методів і алгоритмів оптимізації розробки веб-додатків, які включали б в себе питання зберігання (синтез рівня моделей) і взаємодії (синтез рівня контролерів).

Напрямок створення веб-додатків розвивається надзвичайно швидко. Протягом останніх років з'явилися нові мови програмування (Ruby, Dart, Go), розмітки (HTML5, SASS, SCSS), нові платформи (Ruby on Rails, Hivex, PhpFog) і технології (хмарні сховища та обчислення, нові нереляційні бази даних). Разом з розвитком клієнтських частин веб-браузерів) розвиваються і серверні. Платформи розробки поділяються на власні й вільні. До перших належать: Microsoft .NET Framework (мови ASP, ASP.NET, C++); Java Enterprise Edition (Java), яка також поставляється у варіанті з відкритим вихідним кодом; Hivext (доступний у вигляді веб-сервісу) та ін. З відкритим вихідним кодом активно розвиваються платформи: LAMP (PHP), Ruby on Rails (Ruby), Mono, Zope (Plone), Django (Python).

Основна частина. Структурний синтез веб-додатків заснований на певній класифікації вихідного (скомпільованого) коду веб-додатка. Зараз прийнято кілька класифікацій вихідного коду. Відповідно до клієнт-серверної технології веб-додатка файли програмного коду ділиться на клієнтські файли і серверні. До складу клієнтських файлів входять JavaScript-бібліотеки, зображення, файли таблиць стилів (CSS), що відповідають за відображення інтерфейсу. До складу серверних файлів входять бібліотеки інтерпретатора PHP (для платформи LAMP), файли конфігурацій і файли тестування, тобто, це код, який виконується на сервері. В основі такої класифікації лежить відділення логіки від представлення або інтерфейсу від реалізації.

Веб-додаток (ВД) можна представити як сукупність файлів, що реалізують набір функцій

$$ВД=< КК,\ КМ, ФП, Ф3 >,$$

де КК - множина класів-контролерів, файлів ВД, які відповідають за логіку поведінки; КМ - множина класів-моделей сутностей предметної області; ФП - множина файлів представлень, що описують інтерфейс ВД (графічний для користувача і програмний для взаємодії з іншим ВД); Ф3 - множина файлів завантажувача, в обов'язки якого входить завантажити всі інші частини ВД в пам'ять сервера.

В основі такого поділу лежить принцип відділення логіки від представлення, оформлений у вигляді парадигми «Модель-Представлення-Контролер» (МПК).

Однією з найбільш простих архітектур ВД (рис. 1) є монолітна архітектура. Вона характеризується тим, що всі частини ВД зібрані воєдино в монолітне ядро (МВД). Це може бути один файл, який отримує запит користувача і реалізує функції завантажувача (Ф3), обробляє запит за допомогою конструкцій мови (представляють контролер К К), отримує доступ до даних (модель КМ) і повертає в певному вигляді (ФП) користувачеві. Дана модель архітектури характеризується високим ступенем зв'язаності між всіма частинами, однак, у випадку невеликого проекту це не є проблемою. Як правило, за допомогою такої моделі будуються програми-утиліти, що реалізують обмежений набір функцій. При збільшенні реалізованих функцій виявляються недоліки такої архітектури: висока зв'язаність, відсутність диференціації програмного коду, відсутність виділеного завантажувача і конфігурації.

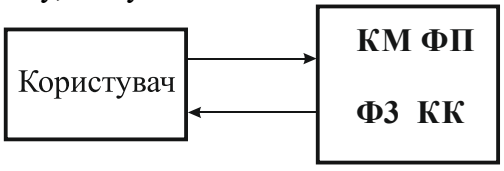

Рис. 1. Архітектура монолітного ВД

Першочерговим завданням при експлуатації даної архітектури є забезпечення завантаження ВД в пам'ять. Доцільно провести виділення завантажувача з ВД (рис. 2).

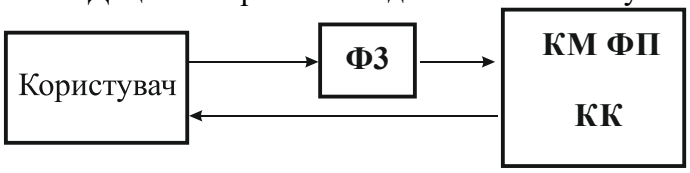

Рис. 2. Архітектура МВД з виділеним файловим завантажувачем (МВДВФЗ)

Дана архітектура надає можливість розділити весь процес роботи ВД на два підпроцеси: ініціалізація і робота. Це дозволяє реалізовувати у завантажувачі комплексну логіку завантаження ВД. Застосування даного підходу зменшує займаний ВД об'єм оперативної пам'яті сервера і дозволяє обслуговувати більшу кількість запитів. При застосуванні до даної моделі методів декомпозиції та реінжинірингу здійснюється класифікація реалізованих функцій. Функції групуються в модулі і з програмного коду виділяються алгоритми ініціалізації та завантаження (рис. 3). В цьому випадку зміни вносяться в конкретний модуль ВД і не зачіпають інші модулі. Для забезпечення легкості внесення змін модулі повинні мати низьку зв'язаність (один з одним) і високу зв'язність (в наборі функцій всередині модуля). Група програмного коду представлення (ФП), роботи з даними (КМ) і логіки предметної області (КК) складають множину МПК. Прикладами ВД такої архітектури є Drupal, ImageCMS, InstantCMS.

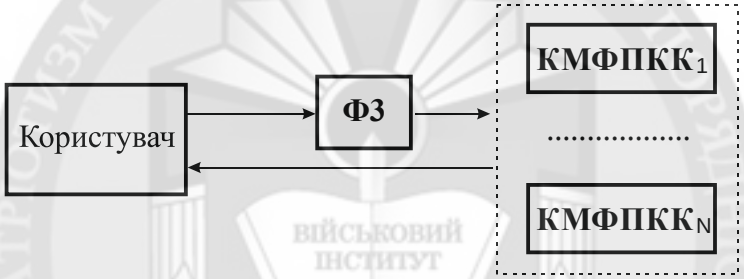

Рис. 3. Архітектура моно-модульного ВД (ММВД)

Ще одним типом архітектури ВД є модель з виділеним завантажувачем і контролером (рис. 4).

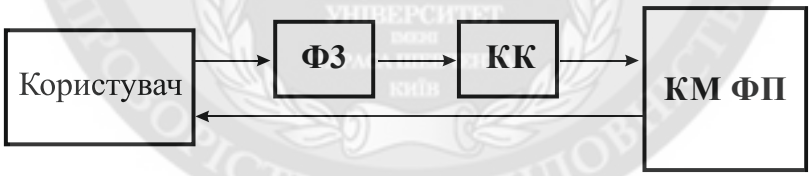

Рис. 4. Архітектура ВД з виділеним завантажувачем і контролером

У даній архітектурі відділяється логіка предметної області (КК) і завантажувач ВД (ФЗ), а вихідний код рівня представлення (користувацький інтерфейс) змішується з кодом рівня доступу до даних. В якості прикладу такої архітектури можна привести систему управління контентом WordPress, в якій у файлах рівня представлення викликаються методи класів-моделей. Це є основним недоліком такого підходу до розробки ВД.

Одним з найпоширеніших на даний час методів розробки ВД є метод структурного синтезу «Модель-Представлення-Контролер» (рис. 5).

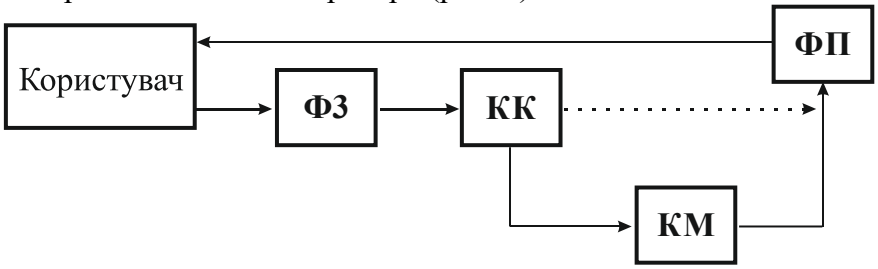

Рис. 5. Архітектура «Модель-Представлення-Контролер»

У даному підході програмний код поділяється на три підмножини: множина коду рівня логіки предметної області (рівень контролерів КК), множина коду рівня роботи з даними (рівень моделі КМ), множина коду зовнішнього представлення (рівень представлення ФП). Аналогічно попереднім моделям архітектури виділяється завантажувач (Ф3). Модель надає дані (зазвичай для рівня представлення), а також реагує на запити (зазвичай, від контролера), змінюючи свій стан (рис. 6).

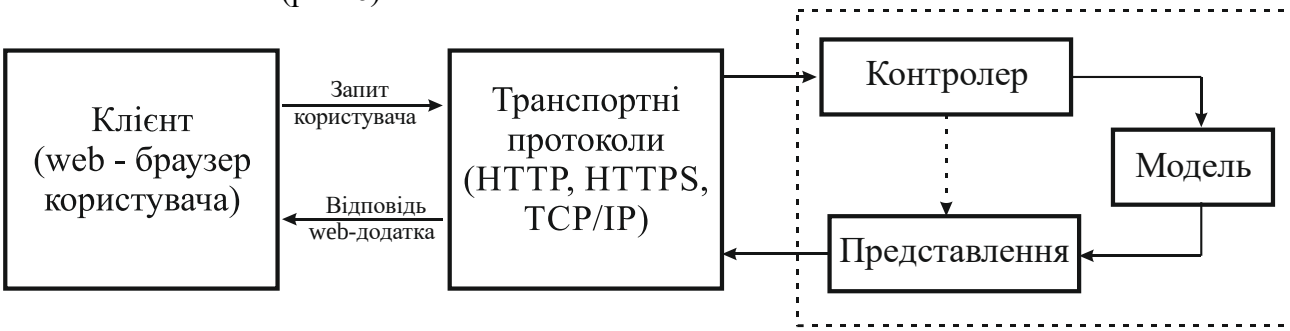

Рис. 6. Загальний вид ВД архітектури «Модель-Представлення-Контролер»

Множина моделей у пам'яті становлять стан ВД, а представлення реалізує відображення інформації. Поведінка (контролер) інтерпретує дані, введені користувачем, та інформує модель і представлення про необхідність відповідної реакції. В якості прикладу ВД такої архітектури можна привести Joomla, Lifestreet, ModX.

Зі зростанням обсягу програмного коду росте навантаження на завантажувачі ВД і тому доцільним є проведення подальшої декомпозиції, яка дає систему з двоетапною ініціалізацією (рис. 7).

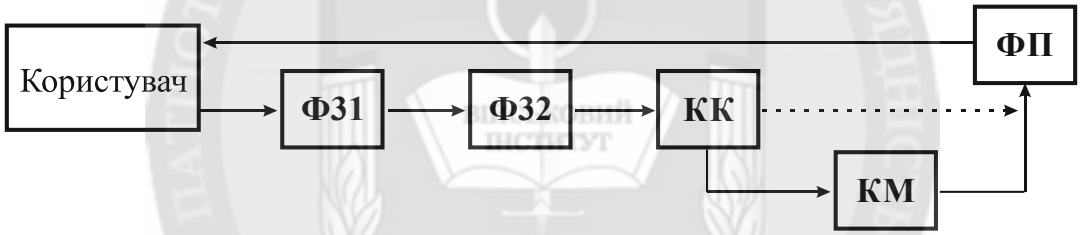

Рис. 7. Архітектура ВД з двоетапною ініціалізацією

У такому ВД завантажувач (Ф31) здійснює попереднє завантаження, що відповідає мінімуму необхідних операцій для ініціалізації ВД. Він не виконує операції установки з'єднання із зовнішніми серверами (базами даних) і підключення файлів інтернаціоналізації. Ці функції переходять в основний завантажувач (Ф32). Наприклад, Ф31 на платформі LAMP реалізує ініціалізацію системи автозавантаження класів, що дозволяє при першому зверненні до класу системи автоматично завантажувати файл з його вихідним кодом. При розробці ВД декомпозиції піддається не тільки виконуваний код, але і конфігурація. В МВД архітектурі конфігурація неявно закодована в тексті ВД за допомогою параметрів застосовуваних функцій, які не змінюються залежно від стану S або переданих вхідних впливів *v*. При виділенні конфігурації (конф.) операція її завантаження розташовується в Ф32 (рис. 8).

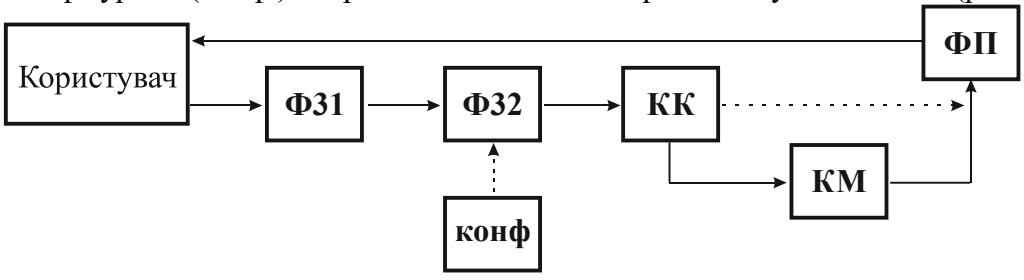

Рис. 8. Архітектура ВД з виділеною конфігурацією

У даній архітектурі класи-контролери отримують повністю готове до роботи оточення, яке ініціалізували завантажувачі ФЗ1 і ФЗ2.

Запропонована класифікація враховує підходи до синтезу архітектури ВД на концептуальному рівні (синтез загальної архітектури ВД).

Рівень контролерів реалізує логіку обробки запиту користувача, а також може реалізовувати логіку предметної області. Існує кілька підходів до синтезу контролерів. Компромісною методикою є введення додаткового шару абстракції, так званого сервісного шару (рис. 9).

Рис. 9. Синтез контролерів на основі сервісного шару

Введення сервісного шару дозволяє покращити уніфікацію програмного коду і такі критерії якості, як структурність та можливість модифікацій, не погіршивши при цьому інші. Основним завданням класифікації є визначення якісних характеристик вихідного коду, в залежності від яких слід поміщати його в контролер, модель або сервіс. На основі аналізу множини ВД формалізація задачі синтезу контролерів буде наступною:

$$(пкк \in ПКК) \vee (пкк \in СК) \vee (ппк \in M);\ |ПКК| \to \min;\ |СК| \to \min;\ |M| \to \min,$$

де *ПКК* - множина програмного коду контролерів, *СК* - множина сервісів, *М* - множина моделей, *пкк* - ділянка програмного коду, що аналізується.

Крім методів декомпозиції коду контролерів, необхідно визначити концептуальний алгоритм синтезу і залежності контролерів від інших елементів ВД. При використанні незв'язаного контролера, тобто відсутності архітектури контролерів, користувач надсилає запит контролеру, який виконує всі необхідні дії. Рівень контролерів у даному випадку не піддається декомпозиції. Найчастіше контролери даного типу зустрічаються в МВД з виділеним завантажувачем і контролером.

Іншим методом синтезу контролерів є створення контролерів на основі форм користувальницького інтерфейсу (рис. 10).

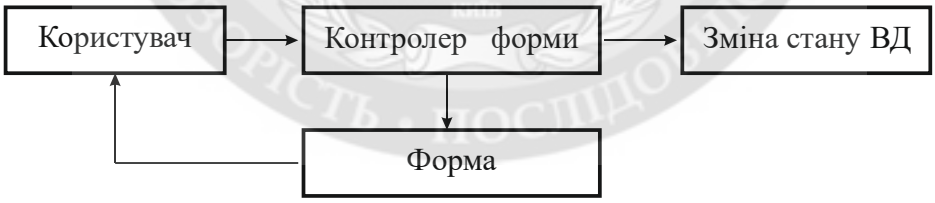

Рис. 10. Контролер на основі форми

Користувач отримує від ВД сторінку з формою, заповнює її, відправляє на сервер, а сервер викликає відповідний контролер. Контролер проводить валідацію форми і, в разі позитивного результату, змінює стан ВД або надсилає форму користувачеві ще раз. Функції-дії, що отримані таким чином, є проекцією представлень (форм) на контролери. Основним недоліком даного методу є те, що логіка ВД будується на основі інтерфейсу, а отже, зміна інтерфейсу спричинить за собою зміну логіки. Це суперечить концепції МПК і визначень синтезу контролерів. Метод синтезу контролерів на основі моделі представлений на рис. 11.

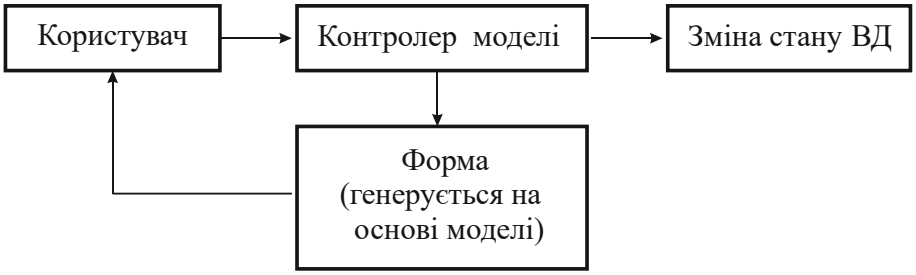

Рис. 11. Контролер на основі моделі

Синтез контролерів йде не від представлення (форми), а від даних (моделі). При наявності моделей, що повністю формалізовані в термінах рівня зберігання даних, існує можливість за цією специфікацією автоматично створювати контролери. Даний метод відноситься до метапрограмування (розробка програм, що генерують інші програми). Програма на основі специфікації моделей створює відповідні контролери для них і при цьому формує СЧМВ-дії контролерів: створення, читання, модифікацію, видалення (СЧМВ) . Функції-дії, отримані таким чином, є проекцією моделей контролерів.

Даний метод використовується для швидкого створення рівня логіки предметної області разом з методом створення форм. Існує можливість швидкої генерації всього додатка на основі лише структури бази даних. Такі проекти існують в даний момент в фреймворку Ruby on Rails. Даний підхід має недоліки. В першу чергу, це відсутність гнучкості генеруючих процедур. Вимога однозначності до них не дозволяє створювати різні контролери для одних і тих же моделей і впливати на процес їх генерації. Тому цей метод використовується лише для прототипування ВД на початкових етапах розробки.

Розроблений лінгвістичний підхід до синтезу контролерів (рис. 12). У цьому підході методами абстрагування створено універсальний контролер, який отримує запити від користувача на спеціальній мові дій. Виступаючи в ролі транслятора, він перетворює його в виконуваний код ВД, виконуючи зазначені в мові дії. За рахунок цього зовнішній інтерфейс користувача та ВД уніфікується, що є безумовною перевагою.

Рис. 12. Лінгвістичний підхід до синтезу контролерів

При цьому виникає ряд проблем:

- розробку форм необхідно проводити з урахуванням вимог інтерфейсу;
- введення в проект нової мови підвищує його складність;
- з появою нових вимог виникає необхідність розширювати мову дій і сам універсальний контролер, в результаті чого він може втратити свою універсальність;
- можливе зниження продуктивності внаслідок необхідності операції розбору команд нової мови.

Розглянуті методи хоч і покращують такі показники якості, як структурність і повторюваність, при цьому погіршують критерії простоти конструкції та можливість модифікацій.

Розробка ВД для конкретної предметної області відбувається за наступним алгоритмом:

1. Визначення значущих моделей предметної області, сукупності їх властивостей та взаємозв'язків у відповідності з однією із методик моделювання предметної області.

2. Класифікація моделей за функціональною ознакою. Формування однієї або декількох груп моделей. Число модулів, що додаються буде відповідати числу груп моделей. При формуванні груп необхідно дотримуватися принципу мінімізації зв'язків між групами моделей:

$$|Li_{IN}| > |Li_{OUT}|; \sum_{i \in M} |Li_{OUT}| \to \min,$$

де Li_{IN} - множина зв'язків між моделями всередині групи, Li_{OUT} - множина зв'язків між моделями поточної групи і моделями інших груп, M - множина груп моделей.

3. Визначення характеру взаємодії моделей і самих операцій взаємодії. Формування сервісного шару і шару контролерів з допомогою відображення СЧМУ-впливів (впливи створення, читання, модифікації, видалення) на ПСП-ресурси (ресурси передачі стану представлення) на основі інформації класифікації моделей за функціональною ознакою та графічного інтерфейса.

Висновки. Структурний синтез веб-додатка є багатогранною проблемою, що містить повністю невирішені питання синтезу як всієї архітектури в цілому, так і вихідного коду рівнів моделей, представлень і контролерів. Існуючі стандарти розробки регламентують тільки протоколи взаємодії веб-додатків, недостатньо враховують специфіку веб-додатків. Низький поріг входження в технологію для розробника створює можливість появи веб-додатків, програмний код яких не відповідає вимогам якості. Внаслідок цього, розроблюваний програмний код веб-додатків не уніфікований і не стандартизований. На сьогодні існує необхідність створення методів і алгоритмів оптимізації розробки веб-додатків, які включали б в себе питання зберігання (синтез рівня моделей) і взаємодії (синтез рівня контролерів).

ЛІТЕРАТУРА:

1. Майк Кон. Scrum: Гибкая разработка ПО. / Майк Кон. - Изд-во: Диалектика-Вильямс, 2016. – 576с.

2. Лєнков С.В. Концептуальна схема системи інтелектуальної обробки даних / С.В. Лєнков, В.М. Джулій, О.М. Горбатюк, Н.М. Берназ // Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка. – К.: ВІКНУ, 2014. – Вип. № 46. – С.181-190.

3. Роберт Мартин Быстрая разработка программ. Принципы, примеры, практика. /Роберт Мартин, Джеймс Ньюкирк, Роберт Косс - Изд-во: Диалектика-Вильямс, 2004. - 752с.

4. Сэм Руби. Rails 4. Гибкая разработка веб-приложений / Сэм Руби, Дэйв Томас, Дэвид Ханссон - Изд-во: Питер, 2014. - 448с.

5. Роберт Мартин Гибкая разработка программ на Java и C++. Принципы, паттерны и методики. /Роберт С. Мартин, Джеймс Ньюкирк, Роберт Косс - Изд-во: Диалектика-Вильямс, 2016. - 704с.

REFERENCES:

1. Mayk Kon. Scrum: Gibkaya razrabotka PO. / Mayk Kon. - Izd-vo: Dialektika-Vilyams, 2016. – 576s.

2. Lienkov S.V. Kontseptualna skhema systemy intelektualnoi obrobky danykh / S.V. Lienkov, V.M. Dzhulii, O.M. Horbatiuk, N.M. Bernaz // Zbirnyk naukovykh prats Viiskovoho instytutu Kyivskoho natsionalnoho universytetu imeni Tarasa Shevchenka. – K.: VIKNU, 2014. – Vyp. № 46. – S.181-190.

3. Robert Martin Byistraya razrabotka programm. Printsipyi, primeryi, praktika. /Robert Martin, Dzheyms Nyukirk, Robert Koss - Izd-vo: Dialektika-Vilyams, 2004. - 752s.

4. Sem Rubi. Rails 4. Gibkaya razrabotka veb-prilozheniy / Sem Rubi, Deyv Tomas, Devid Hansson - Izd-vo: Piter, 2014. - 448s.

5. Robert Martin Gibkaya razrabotka programm na Java i C . Printsipyi, patternyi i metodiki. /Robert S. Martin, Dzheyms Nyukirk, Robert Koss - Izd-vo: Dialektika-Vilyams, 2016. - 704s.

Рецензент: д.т.н., проф. Лєнков С.В., начальник науково-дослідного центру Військового інституту Київського національного університету імені Тараса Шевченка

к.т.н. Джулий В.Н., д.т.н., доц. Гунченко Ю.А., Чешун Д.В.

МЕТОДЫ И АЛГОРИТМЫ РАЗРАБОТКИ WEB - ПРИЛОЖЕНИЙ

Веб-приложения оперируют огромными объемами данных, в том числе и персональными данными (такими как номера кредитных карт), что предъявляет высокие требования к программному коду. В то же время, большая длительность разработки, согласования и утверждения международных и национальных стандартов приводит к их консерватизму, а также к хроническому отставанию требований и рекомендаций этих документов от современной практики и технологии создания сложных систем. Современные методы разработки программных средств, их унификация и стандартизация не в полной мере учитывают специфику разработки веб-приложений с использованием свободных интернет-технологий. Проблема заключается в отсутствии четких рекомендаций по разработке файлов исходного кода контроллеров, моделей и представлений, а также спецификации их взаимосвязей. Структурный синтез веб-приложения является многогранной проблемой, что содержит полностью нерешенные вопросы синтеза как всей архитектуры в целом, так и исходного кода уровней моделей, представлений и контроллеров. Существующие стандарты разработки регламентируют только протоколы взаимодействия веб-приложений и недостаточно учитывают их специфику. Низкий порог входа в технологию для разработчиков создает возможность появления веб-приложений, программный код которых не соответствует требованиям качества. Вследствие этого, разрабатываемый программный код веб-приложений не унифицирован и не стандартизирован. Рассмотренные методы хоть и улучшают такие показатели качества, как структурность и повторяемость, но при этом ухудшают критерии простоты конструкции и возможность модификаций.

На основе проведенного анализа можно сделать вывод, что существует необходимость создания методов и алгоритмов оптимизации разработки веб-приложений, которые включали бы в себя вопросы хранения (синтез уровня моделей) и взаимодействия (синтез уровня контроллеров).

Ключевые слова: веб-приложения, контроллеры, модели, представления, программный код, метод, алгоритм, структурный синтез, интернет-технологии.

Ph.D. Dzhuliy V.M., GunchenkoYu.A., Cheshun D.V.

METHODS AND ALGORITHMS WEB APPLICATION DEVELOPMENT

Web applications operate with large amounts of data, including personal data (such as credit card numbers), which makes high demands on the software. At the same time, the long duration of the development, approval and adoption of international and national standards leads to conservatism and chronic backlog of requirements and recommendations in these documents to current practices and technology of complex systems. Modern software development methods, their unification and standardization do not fully take into account the specific web application development using available Internet technologies. The problem is the lack of clear guidelines on the development of source code files of controllers, models and views and specifications of their relationships. Structural synthesis of Web application is a multifaceted problem that includes a outstanding issues about the whole synthesis of architecture in general and about the source code level models, views and controllers. Existing standards governing the development of interaction protocols only web applications and do not define specification. The low threshold of entry into the technology developer creates the possibility of web applications with software code, which does not meet quality requirements. As a result, developed the code of web applications are not standardized. Existing methods are improve structural and repeatability, but worsen the criteria of simplicity of design and the possibility of modifications.

Based on the analysis, we can conclude, that there is a need to establish methods and algorithms for optimization of web application development, which would include a storage issue (synthesis of models) and interaction (synthesis of controllers).

Keywords: web applications, controllers, models, views, code, methods, algorithms, structural synthesis, Internet technology.