УДК 004.891

С.Р. КРАСИЛЬНИКОВ
Хмельницький Національний університет

ОПТИМІЗАЦІЯ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА РОЗПОДІЛУ ЗАДАЧ СЕРЕД WEB - РОЗРОБНИКІВ

В статті запропоновано підхід до оптимізація процесу розробки програмного забезпечення та розподілу задач серед розробників.

Проблема полягає у відсутності чітких рекомендацій по розробці файлів вихідного коду контролерів, моделей і представлень і специфікації їх взаємозв'язків. Для оптимізації процесу розробки програмного забезпечення необхідною умовою є оптимізація процесу роботи всіх програмістів – членів команди. 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 і специфікації їх взаємозв'язків.

Низький поріг входження в технологію для розробника створює можливість появи веб-додатків, програмний код яких не задовольняє вимогам якості. Внаслідок цього, розроблений програмний код веб-додатків не уніфікований і не стандартизований. За допомогою застосування СКР-функції (середньостатистичні коливання працездатності) до визначення денного фокус-фактора розробника обчислено значення фокус-фактора для нових команд. Це значення може використовуватися при плануванні ітерації нової команди, або індивідуальному плануванні для нового члена команди розробки, вирішена оптимізаційна задача по визначенню такого розподілу завдань по програмістам, яка призведе до максимізації показника продуктивності (Ррозр) всієї команди, підвищення ефективності вирішення практичних завдань розробки веб-додатків.

Вирішено оптимізаційну задачу по визначенню розподілу задач серед розробників, яка призведе до максимізації показника продуктивності всієї команди.

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

S.R. KRASILNIKOV
Khmelnitsky National University

OPTIMIZATION OF PROCESS OF DEVELOPMENT OF SOFTWARE AND DISTRIBUTION OF TASKS AMONG WEB - DEVELOPERS

Abstract - The article suggests an approach to optimizing the process of software development and distribution of tasks among developers.

The problem is that there are no clear recommendations for the development of source code files for controllers, models and views, and specification of their interrelationships. To optimize the process of software development, it is necessary to optimize the work of all programmers-team members. Web applications operate with huge amounts of data, including personal data (such as credit card numbers), which places high demands on the software code. At the same time, the lengthy development, harmonization and approval of international and national standards leads to their conservatism, as well as to the chronic lag in the requirements and recommendations of these documents from modern practices and the technology of complex systems. Modern methods of software development, their unification and standardization do not fully take into account the specifics of developing web applications using free Internet technologies. The problem is that there are no clear recommendations for the development of source code files for controllers, models and views, and specification of their interrelationships.

A low threshold of entry into technology for the developer creates the possibility of the appearance of web applications, the program code of which does not meet the quality requirements. Because of this, the developed code of web applications is not unified and not standardized. By using the TFR-function (average statistical fluctuation of working capacity) to determine the day-to-day focus-factor of the developer, the value of the focus factor for new teams was calculated. This value can be used when planning the iteration of a new team, or for individual planning for a new member of the development team, an optimization task has been solved to determine this distribution of tasks by programmers, which will lead to maximization of the performance score (RRAS) of the entire team, applications.

The optimization task to determine the distribution of tasks among developers, which will lead to the maximization of the performance of the whole team, is solved.

Keywords: Web applications, controllers, models, views, program code, method, algorithm, structural synthesis, Internet technologies.

Вступ. 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. Найпоширенішою платформою розробки веб-додатків є LAMP – в склад якої входить – операційна система Linux, веб - сервер Apache, реляційна база даних MySQL та інтерпретатор PHP. На даній платформі розробляються багатокомпонентні програмні рішення з участю багатьох десятків розробників. Веб-додатки починають оперувати 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.

Протягом декількох десятиліть вирішується задача пошуку повторюваного, передбаченого процесу або методології, яка б покращала продуктивність, якість і надійність розробки програмного забезпечення. Для коректної постановки задачі, перш за все, необхідно сформулювати опис предметної області, її характеристик і одиниць вимірювання. В процесі створення або модифікації веб-додатка розробник виконує роботу, яка характеризується трудомісткістю, що виражається (вимірюється) в *sp (story point)*. Для оптимізації процесу розробки програмного забезпечення необхідною умовою є оптимізація процесу роботи всіх програмістів – членів команди. 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. Архітектура програмного забезпечення або комп'ютерної системи являє собою сукупність структур, що складаються з програмних елементів, зовні видимих властивостей цих елементів та взаємозв'язків між ними. Архітектура – це незмінна глибинна структура веб-додатка, помилки, закладені в архітектуру при її проектуванні, будуть призводити до ще більших помилок в процесі її реалізації. Зв'язки між частинами програмної системи були описані 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.

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 і 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).

Основна частина. При розробці програмного забезпечення постають проблеми якості, вартості та надійності. Деякі програми містять мільйони рядків коду. Як очікується, цей код буде правильно виконуватись у змінних умовах. Протягом декількох десятиліть вирішується задача пошуку повторюваного, передбаченого процесу або методології, яка б покращила продуктивність, якість і надійність розробки.

Для коректної постановки задачі, перш за все, необхідно сформулювати опис предметної області, її характеристик та їх одиниць вимірювання.

В процесах створення або модифікації веб-додатка розробник виконує роботу, яка характеризується трудомісткістю, що вимірюється в SP *(story point)*. SP – загальнокомандна оцінююча одиниця роботи, що на першій ітерації приймається за людино-годину. Під розробкою веб-додатка розуміється ітераційний процес. Загальний час розробки ділиться на інтервали (спринти). SP_i – інтервал розробки, одна ітерація циклу. $T_{SPR}(S_i)$ – довжина інтервалу розробки, що вимірюється в годинах. При ітераціях однакової довжини, яка рівняється одному тижню, $T_{SPR} = 40$ [годин]. $A_i^{Fixed}(S_i)$ – тудомісткість ітерації, виміряна в інтервалах розробки - SP. Розробник працює над конкретними задачами («тікетами») tk_i, які також характеризуються $T(tk_i)$ – часом, витрачений на виконання задачі, $A(tk_i)$ – її трудомісткість і $V(tk_i)$ – її корисність (важливість). Задача tk_i для розробника вважається виконаною, якщо вона переходить із стану *new* (задача формалізована) в стан *fixed* (завершена). Для цього розробник виконує роботу в кількості $A(tk_i)$ за час $T(tk_i)$ *[SP]*. Загальний об'єм роботи, що виконується *i*-тим програмістом за одну ітерацію розраховується наступним чином:

$$A_i^{Fixed} = \sum_{i=1}^{n_{Fixed}} A_i^{Fixed} \left[SP\right],$$

де n_{Fixed} – кількість виконаних задач. Продуктивність розробника (P_i) визначається відношенням виконаних задач до загальної кількості часу в спринті:

$$P_i = \frac{A_i^{Actual}}{T_{SPR}} \left[SP / час\right]$$

Продуктивність команди розраховується наступним чином:

$$P_{розр} = \sum_{i=1}^{N_P} P_i$$

де N_P – кількість розробників у команді, M_N – кількість часу в ітерації (при тижневій ітерації M_N рівняється 40). Оскільки, зазвичай за SP приймають величину роботи, що виконується за один час $\left(t(SP)=1\left[час\right]\right)$, то допустима область існування критерію $P_{розр}$ виражається так: $\left(0, M_N \times N_P\right)$.

Фокус-фактор $\left(FF_{SPR}\right)$ – це величина, яка характеризується відношенням виконаної роботи

$\left(A_{SPR}^{Actual}\right)$ до запланованої $\left(A_{SPR}^{Estimated}\right)$:

$$FF_{SPR} = \frac{A_{SPR}^{Actual}}{A_{SPR}^{Estimated}}$$

діапазон значень (0,1). Таким чином, задача зводиться до максимізації показника продуктивності команди розробників.

Нехай V_{SPR}^{A} – кількість доступних (*available*) людино-годин ітерації, а V_{SPR}^{E} – кількість очікуваних (*estimated*) людино-годин ітерації, V_{D}^{A} – кількість доступних годин в день для одного програміста (рівняється 8), V_{D}^{E} – кількість очікуваних робочих годин в день для одного програміста.

$$\mathrm{V_{SPR}^{E} = V_{SPR}^{A} \times FF_{SPR}} \quad V_{SPR}^{A} = M_N \times N_P; \ де \ V_{SPR}^{E} = V_{SPR}^{A} \times FF_{SPR}.$$

Для оптимізації процесу розробки команди необхідною умовою є оптимізація процесу розробки всіх членів команди. Нехай $FF_D(t)$ – *ФФ*-функція (залежність фокус-фактору від часу протягом дня), а $ff_D(t)$ – *СКП*-функція (середньостатистичні коливання працездатності).

Отже вони виражають зосередженість програміста на вирішенні задачі. Основний робочий час знаходиться в інтервалі 10.00 – 18.00 годин. Графік *СКП*-функції виглядає наступним чином (рис. 1):

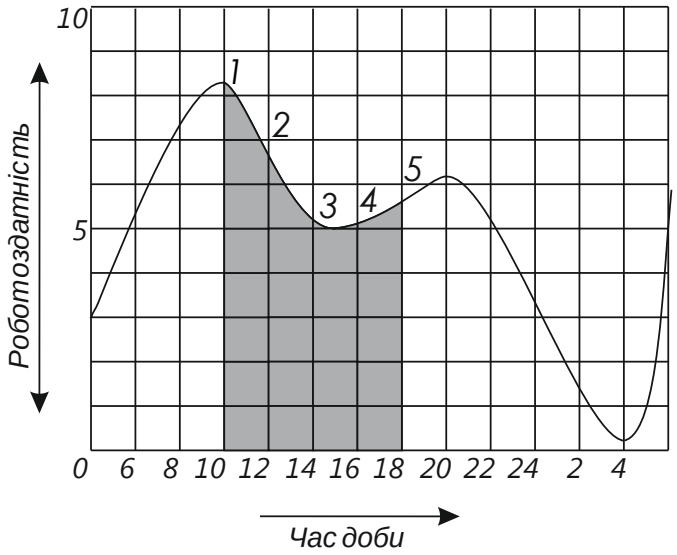

Рис. 1. Графік СКП-функції

Дані функції (*ff* та *FF*) володіють наступними властивостями:

$$FF_D(t) \notin (0,1); \ \ ff_D(t) \notin (0,10); \ \min(ff) = 0.2; \ \max(ff) = 8.1;$$

$$V_D^A = 8; \ V_D^E = \int_{t_1}^{t_2} ff(t), \quad t_1 = 10, \ t_2 = 18$$

Визначимо в табл. 1 вузли інтерполяції (Δt = 2 год.) і екстремуми СКП-функції ($\min(ff) = 0.2; \ \max(ff) = 8.1$) на інтервалі (10,18).

Таблиця 1

Аргументи та значення СКП та ФФ функцій

№	*T*	$ff(t)$	$ff'(t)$	$ff''(t)$
1	*10*	*8.1*	*7.9*	*1*
2	*12*	*6.5*	*6.5*	*0.82*
3	*14*	*5.1*	*4.9*	*0.62*
4	*16*	*5.1*	*4.9*	*0.62*
5	*18*	*5.7*	*5.5*	*0.7*

Функція $ff'(t)$ отримана зсувом $ff(t)$ на 0.2 вниз, а функція $ff''(t)$ отримана масштабуванням $ff'(t)$ на інтервалі (0,1):

$$\min(FF) = 0, \ \max(FF) = 1; \ \min(ff) = 0.2, \ \max(ff) = 8.1$$

$$K_{МАСШТ} = \frac{\max(FF)}{\max(ff')} = \frac{1}{7.9} = 0.126$$

Далі здійснимо інтерполяцію функції *FF(t)* за допомогою полінома Лагранжа. Поліном представлено наступним чином:

$$L(x) = l_0(x)+0.81\cdot l_1(x) + 0.62\cdot l_2(x) + 0.62\cdot l_3(x)+ 0.7\cdot l_4(x)$$

Кількість очікуваних годин роботи розробника за день визначається як

$$V_D^E = \int_{t_1}^{t_2} FF(t), \quad t_1 = 10, \ t_2 = 18; \ V_D^E = 5.81125$$

При кількості доступних людино-годин рівних $V_{SPR}^A = V_D^A \times M_N \times N_P = 8 \times 4 \times 5 = 160$ для команди з чотирьох людей (N_P=4) при тижневій ітерації (M_N=5) кількість очікуваних людино-годин для першої ітерації розраховується, як $V_{SPR}^E = V_D^E \times M_N \times N_P = 5.81123 \times 4 \times 5 = 116.225[SP]$. Таким чином фокус-фактор розраховується, як відношення очікуваної кількості людино-годин до наявної

$$FF = \frac{V_D^E}{V_D^A} = \frac{5.81125}{8} = 0.72640625$$

В результаті розрахунків вдалось отримати значення фокус-фактору першої ітерації для нових команд. На сьогодні використовується значення 0.70, проте у роботі вдалося здійснити уточнення значення даного параметру з використанням СКП-функції. Це значення може використовуватись при плануванні ітерації нової команди чи при індивідуальному плануванні для нового члена команди.

Завдання оптимізації процесу розробки при денному плануванні роботи розробника визначається наступним чином. Нехай є *n* завдань у списку завдань на виконання (*product backlog)*. Завдання *i* (*i* = 1, 2 ... *n*) характеризується обсягом роботи a_i (вираженої в *SP*) і корисністю v_i, яку призначає керівник проекту (або замовник). Нехай V_D^E – загальний максимально допустимий обсяг робіт, що виконується розробником за день. З раніше проведеного розрахунку V_D^E = 5.81124 *[SP]* (при *SP* прийнятій за одну людино-годину роботи). Потрібно обрати завдання таким чином, щоб їх загальний обсяг робіт$\left(\sum_{i=1}^{n} a_i \div\right)$ не перевищував максимально допустимий обсяг (V_D^E), при цьому сумарна корисність набору завдань була максимальною. Нехай x_i = 1, якщо завдання призначене розробнику, та x_i = 0 в іншому випадку. Математичне формулювання завдання буде таким:

$$F(x) = \sum_{i=1}^{n} v_i \times a_i \to \max; \ \sum_{i=1}^{n} v_i \times a_i \le V_D^E; \ x_i \notin \{0,1\}, i = 1,2,...,n.$$

Додаткові обмеження:

1) V_D^E = 5.81124 *[SP]*;

2) кожне завдання можна обрати тільки один раз.

Дану оптимізовану задачу можливо вирішити шляхом динамічного програмування, шляхом розбиття її на більш прості підзадачі, результати вирішення яких зберігаються в локальній пам'яті. Це різко зменшує необхідний обсяг обчислень.

Загальний алгоритм оптимізації процесу розробки команди буде наступним:

- для першої ітерації значення фокус-фактора приймається рівним 0.72640625. Вирішується завдання денного планування для кожного розробника окремо. Завдання з множини T(*product backlog*) сортується по зменшенню корисності. Узагальнюючи частні рішення, обчислюється значення V_D^E для команди. Процес повторюється кожен день протягом ітерації;
- після ітерації розраховуються значення фокус-фактора FF_{SPR} для кожного розробника на основі отриманих значень $\sum_{i=1}^{n} a_i$;
- на наступній ітерації, поставлені завдання знову сортуються по зменшенню корисності, а програмісти – по зменшенню V_D^E. Завдання денного планування вирішується за допомогою обчислених на минулій ітерації значень фокус-фактора розробників, а обмеження обчислюються за методом «вчорашньої погоди» з $\sum_{i=1}^{n} a_i$.

Таким чином планування оптимізується, максимізуючи при цьому корисність виконаних завдань.

У стандарті ДСТУ 2850-94 «Програмні засоби ЕОМ. Показники і методи оцінювання якості» веб-додатки не виділені як окремий клас програмного забезпечення (так як їх не було на момент створення стандарту), але вони відповідають підкласу «інше програмне забезпечення». Вибір показників якості програмного забезпечення для цього підкласу здійснюється в залежності від їх призначення з урахуванням вимог областей застосування.

З множини критеріїв якості було обрано ті критерії, які впливають на продуктивність розробки веб-додатку: *С1, С2, С4, ГЗ*. Для незлічених критеріїв оптимізації використовуються експертні оцінки, для злічених – відповідні формули розрахунку.

Для характеристики продуктивності розробки ($P_{РОЗР}$) використовується інтервальна абсолютна критеріальна шкала (*ІАКШ*). Критерій розраховується наступним чином:

$$P_{РОЗР} = \sum_{i=1}^{N_P} \left(\frac{\sum_{j=1}^{n_{Fixed}} A_j^{Fixed}}{M_N} \right), \text{ де } P_{РОЗР} \notin (0,1)$$

Після декількох ітерацій продуктивність розробки збільшується (завдяки використанню запропонованих у роботі методів) і показник *SP* вже не відображає реальність. Тобто час виконання одиниці роботи після ітерацій займає менше часу, ніж до них $t'(SP) < t(SP)$. Динаміка зміни продуктивності характеризується їх відношенням $E_v = \frac{t(SP)}{t'(SP)}$. Оскільки $P_{РОЗР} \to \max$, *то* $t_{РОЗР} \to \min$, $Q \to \min$.

Повторюваність (*С4*) характеризується з використанням *ІАКШ* за такою формулою:

$$K_{nВД} = \frac{V - V_0}{V} \times 100\%$$

де *V* - загальний обсяг файлів програмного коду, V_0 - обсяг оригінальних файлів програмного коду (розроблених вперше). Чим більше сторонніх бібліотек використовується в ВД, тим більше коефіцієнт стандартизації та уніфікації (коефіцієнт застосування) ВД. Область визначення: [0, 1] або [0,100%].

Структурність (*С1*) характеризується використанням дискретної абсолютної критеріальної шкали (*ДАКШ*): відсутність архітектури програмного забезпечення. Код практично не структурований, ієрархічно не розділений; наявність деякого принципу упорядкування програмного коду; достатня структурованість. Відсутність єдиної системи найменувань більш ніж в 50% вихідного коду; хороша структурованість. Передбачуваність розташування реалізацій набору функцій. В програмному забезпеченні використовується єдина система найменувань; висока глибина ієрархії класів. Чітке розмежування реалізованих функцій.

Простота конструкції (*С2*) за *ДАКШ* поділяється на: надзвичайно складна конструкція системи; складна конструкція архітектури програмного забезпечення; помірна складність конструкції; проста конструкція; дуже проста конструкція.

Можливість модифікацій (*ГЗ*) за *ДАКШ* поділяється на: надзвичайно низька можливість модифікації. Вихідний код неможливо модифікувати під вимоги нового завдання. Будь-яке невелике виправлення викликає крах всієї системи. Немає можливості відстежити залежності і передачі управління в програмному коді; низька можливість модифікації. Для додавання підтримки нового завдання необхідно написати значну кількість нового програмного коду. При цьому відбувається втручання в сторонні підсистеми, які безпосередньо не пов'язані з множиною функцій, що додаються; середня можливість модифікації. Для додавання підтримки нових можливостей необхідно написати не тільки реалізацію, але і провести узгодження програмних інтерфейсів; хороша можливість модифікації. Для додавання підтримки нового завдання необхідно написати незначну кількість нового програмного коду; висока можливість модифікації. Вихідний код легко модифікується під вимоги нового завдання.

В більшості випадків немає необхідності вносити новий програмний код, підтримку нового набору функцій дає конфігурація.

Глобальні параметри оптимізації представлені такою множиною.

$$X = \left\{ \vec{T}_{АРХ}, \vec{T}_{КОНТР}, \vec{T}_{СКОНТР}, \vec{T}_{ВИДІВ}, \vec{T}_{МОД} \right\},$$

де $\vec{T}_{АРХ}$ – тип архітектури, $\vec{T}_{КОНТР}$ – тип алгоритму синтезу класів-контролерів по реалізованим функціям, $\vec{T}_{СКОНТР}$ – тип алгоритму синтезу контролерів по зв'язності з іншими підсистемами, $\vec{T}_{ВИДІВ}$ – тип алгоритму синтезу представлень (видів), $\vec{T}_{МОД}$ – тип алгоритму розробки рівня моделі.

$$\vec{T}_{АРХ} = \{ МВД, МВДВЗ, ММВД, МВДВЗК, МВК, МВКДІ, МВКВК, ПА \},$$

де *МВД* – монолітний ВД, *МВДВЗ* – монолітний ВД з виділеним завантажником, *ММВД* – моно-модульний ВД, *МВДВЗК* – монолітний ВД з виділеним завантажником і контролером, *МВК* – архітектура «Модель-Вид-Контролер», *МВКДІ* – архітектура МВК з двоетапною ініціалізацією, *МВКВК* – архітектура МВК з виділеною конфігурацією, *ПА* – пропонована архітектура.

Тип алгоритму синтезу контролерів по реалізованим функціям:

$$\vec{T}_{КОНТР} = \{ ТлК, ТнК, Серв \},$$

де *ТлК* – «товсті» класи-контролери, *ТнК* – «тонкі» класи-контролери, *Серв* – використання тонких контролерів в поєднанні з сервісним шаром.

Тип алгоритму синтезу контролерів по зв'язності з іншими підсистемами:

$$\vec{T}_{СКОНТР} = \{ НсК, КоФ, КоМд, КоМ, КоР \},$$

де *НсК* – незв'язний контролер, *КоФ* – контролер на основі форми (представлення), *КоМд* – контролер

на основі моделі, *КоМ* – контролер на основі мови (лінгвістичний підхід), *КоР* – пропонований метод (контролер на основі ресурсу).

Тип алгоритму синтезу представлень (видів):

$$\vec{T}_{ВИДІВ} = \{ МВ, ВВ, ШВ \},$$

де *МВ* – монолітна модель видів, *ВВ* – модель з відокремленими видами, *ШВ* – шаблонно-видова архітектура.

Тип алгоритму розробки рівня моделі:

$$\vec{T}_{МОД} = \{ РЗ, КЗ, КМЛМ, ОРП \},$$

де *РЗ* – «ручне» виконання запитів, *КЗ* – використання конструктора запитів, *КМЛМ* - використання відділення концептуальної від логічної моделі, *ОРП* - використання системи об'єктно-реляційного перетворення.

Завдання оптимізації полягає в збільшенні продуктивності створення веб-додатків $P_{РОЗР} \to \max$ при достатній структурності ($C1 > 3$), помірної складності конструкції ($C2 > 2$), високою повторюваності ($C4 > 0.5$) і високій здатності до модифікації ($Г3 > 4$).

$$Y = \left\{ \left(P_{РОЗР}, C1, C2, C4, Г3 \right) : P_{РОЗР} \to \max, C1 > 3, C2 > 2, C4 > 0.5, Г3 > 4 \right\}$$

$$X = \left\{ \vec{T}_{АРХ}, \vec{T}_{КОНТР}, \vec{T}_{СКОНТР}, \vec{T}_{ВИДІВ}, \vec{T}_{МОД} \right\}$$

Веб–додаток відноситься до клієнт-серверного типу додатків, в якому клієнтом виступає браузер або інший веб–додаток, а обслуговуючим сервером – веб-сервер. Інформаційна взаємодія між ними здійснюється в мережі пакетної комутації на основі таких протоколів, як HTTP та TCP / IP. Основна функція ВД полягає в обробці призначених для користувача запитів (які можуть спричинити зміну його внутрішнього стану) і генерації відповіді користувачеві. Особливістю веб–додатка, в порівнянні з іншими клієнт-серверними технологіями, є присутність «тонкого клієнта». Це комп'ютер, або програма-клієнт, який переносить всі або більшу частину завдань пов'язані з обробкою інформації на сервер. У випадку з веб–додатком тонкий клієнт – це веб-браузер. Це означає, що при розробці клієнт-серверної системи завдання по створенню клієнтської частини можна виключити, однак залишаються завдання щодо інтеграції конкретного веб–додатка з безліччю типів клієнтів.

Весь інформаційний процес веб–додатка можна розбити на кілька етапів: відображення запиту користувача на інформаційні структури веб–додатка; завантаження стану веб–додатка на основі даних запиту користувача; зміна стану веб–додатка; генерація відповіді веб–додатка для клієнта; відправка відповіді клієнту.

Формалізувати модель веб–додатка можна за допомогою моделі «чорного ящика» типу «трансформатор» (перетворювач), оскільки веб–додатка виконує перетворення запиту клієнта (v) з множини можливих запитів (V) на вході чорного ящика у відповідь (w) з множини відповідей (W) на його виході. Запит клієнта визначається наступним чином:

$$v = \left\{ \vec{H}, \vec{P}_{GET}, \vec{P}_{POST} \right\}, \partial e \ v \in V ,$$

де $\vec{H}$ – вектор заголовків запиту по протоколу HTTP (включаючи короткостроковий стан веб–додатка, збережений в Cookies), $\vec{P}_{GET}$ –вектор параметрів запиту з GET-множини, $\vec{P}_{POST}$ - вектор параметрів запиту з POST-множини.

Таким чином:

$$w = \left\{ \vec{H}_w, C \right\}, \partial e \ w \in W ,$$

де $\vec{H}_w$ – вектор заголовків відповіді по протоколу HTTP (включаючи заголовки установки короткострокового стану Set-cookie); C – текстовий вміст відповіді в форматі HTML (для браузера в якості клієнта) або JSON / XML для іншого веб–додатка в якості клієнта.

Отже, функціонування веб–додатка є операцією перетворення вхідних значень у вихідні, тобто відображення v на w з використанням s:

$$w = f\left(\vec{v}, \vec{s}\right), \partial e \vec{s} = s_s \bigcup s_{DB},$$

де $f\left(\vec{v}, \vec{s}\right)$ - генеруюча функція веб–додатка.

Таким чином, у даній роботі буде досліджено існуючі ВД, методи їх розробки, що зводяться до реалізації генеруючої функції ВД, їх переваги та недоліки.

Висновки. Для оптимізації процесу розробки програмного забезпечення необхідною умовою є оптимізація процесу роботи всіх програмістів – членів команди. 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. Проведено порівняльний аналіз існуючих архітектур веб-додатків та їх класифікацію. Структурний синтез веб-додатка є багатогранної проблемою, що містить повністю невирішені питання синтезу як всієї архітектури в цілому, так і вихідного коду рівнів моделей, видів і контролерів зокрема.

Низький поріг входження в технологію для розробника створює можливість появи веб-додатків, програмний код яких не задовольняє вимогам якості. Внаслідок цього, розроблений програмний код веб-додатків не уніфікований і не стандартизований. За допомогою застосування СКР-функції (середньостатистичні коливання працездатності) до визначення денного фокус-фактора розробника обчислено значення фокус-фактора для нових команд. Це значення може використовуватися при плануванні ітерації нової команди, або індивідуальному плануванні для нового члена команди розробки. вирішена оптимізаційна задача по визначенню такого розподілу завдань по програмістам, яке призведе до максимізації показника продуктивності ($P_{розр}$) всієї команди, підвищення ефективності вирішення практичних завдань розробки веб-додатків.

Література

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

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. Джулій В.М. Методи та алгоритми розробки web-додатків / В.М. Джулій, Ю.О. Гунченко, Д.В. Чешун // Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка. – К.: ВІКНУ, 2017. – Вип. № 56. – С.107-115

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

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

6. 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. — 704 с.

7. Гарднер Л. Разработка веб-сайтов для мобильных устройств; / Л. Гарднер, Д.Григсби —Питер - Москва, 2013. - 448 с.

8. Дакетт Джон HTML и CSS. Разработка и дизайн веб-сайтов (+ CD-ROM); / Джон Дакетт — Эксмо - Москва, 2013. - 480 с.

9. Китинг Flash MX. Искусство создания web-сайтов; / Китинг, Джоди —ТИД ДС - Москва, 2012. - 848 с.

10. Кузнецов М. PHP. Практика создания Web-сайтов; / М. Кузнецов, И.Симдянов—БХВ-Петербург - Москва, 2012. - 347 с.

11. Митчелл 5 проектов Web-сайтов от фотоальбома до магазина; / Митчелл, Скотт —М.: НТ Пресс - Москва, 2013. - 224 с.

12. Фрейен Бен HTML5 и CSS3.Разработка сайтов для любых браузеров и устройств; / Бен Фрейен— Питер - Москва, 2014. - 304 с.

13. Чебыкин Ростислав Разработка и оформление текстового содержания сайтов; / Ростислав Чебыкин —БХВ-Петербург - Москва, 2014. - 528 с.

14. Энж Эрик. SEO. Искусство раскрутки сайтов; / Эрик Энж , Стефан Спенсер , Рэнд Фишкин , Джесси Стрикчиола —БХВ-Петербург - Москва, 2014. - 668 с.

References

1. Majk Kon. Scrum: Gibkaja razrabotka PO. / Majk Kon. — Izd-vo: Dialektika-Vil'jams, 2016. — S. 576.

2. Lєnkov S.V. Konceptual'na shema sistemi intelektual'noї obrobki danih / S.V. Lєnkov, V.M. Dzhulij, O.M. Gorbatjuk, N.M. Bernaz // Zbirnik naukovih prac' Vijs'kovogo institutu Kiïvs'kogo nacional'nogo universitetu imeni Tarasa Shevchenka. – K.: VIKNU, 2014. – Vip. № 46. – C.181-190

3. Dzhulij V.M. Metodi ta algoritmi rozrobki web-dodatkiv / V.M. Dzhulij, Ju.O. Gunchenko, D.V. Cheshun // Zbirnik naukovih prac' Vijs'kovogo institutu Kiïvs'kogo nacional'nogo universitetu imeni Tarasa Shevchenka. – K.: VIKNU, 2017. – Vip. № 56. – C.107-115

4. Robert Martin Bystraja razrabotka programm. Principy, primery, praktika. /Robert Martin, Dzhejms N'jukirk, Robert Koss — Izd-vo: Dialektika-Vil'jams, 2004. — 752 s.

5. Sjem Rubi. Rails 4. Gibkaja razrabotka veb-prilozhenij / Sjem Rubi, Djejv Tomas, Djevid Hansson — Izd-vo: Piter, 2014. — 448 s.

6. Robert Martin Gibkaja razrabotka programm na Java i C++. Principy, patterny i metodiki. /Robert S. Martin, Dzhejms N'jukirk, Robert Koss — Izd-vo: Dialektika-Vil'jams, 2016. — 704 s.

7. Gardner L. Razrabotka veb-sajtov dlja mobil'nyh ustrojstv; / L. Gardner, D.Grigsbi —Piter - Moskva, 2013. - 448 c.

8. Dakett Dzhon HTML i CSS. Razrabotka i dizajn veb-sajtov (+ CD-ROM); / Dzhon Dakett — Jeksmo - Moskva, 2013. - 480 c.

9. Kiting Flash MX. Iskusstvo sozdanija web-sajtov; / Kiting, Dzhodi —TID DS - Moskva, 2012. - 848 c.

10. Kuznecov M. PHP. Praktika sozdanija Web-sajtov; / M. Kuznecov, I.Simdjanov—BHV-Peterburg - Moskva, 2012. - 347 c.

11. Mitchell 5 proektov Web-sajtov ot fotoal'boma do magazina; / Mitchell, Skott —M.: NT Press - Moskva, 2013. - 224 c.

12. Frejen Ben HTML5 i CSS3.Razrabotka sajtov dlja ljubyh brauzerov i ustrojstv; / Ben Frejen— Piter - Moskva, 2014. - 304 c.

13. Chebykin Rostislav Razrabotka i oformlenie tekstovogo soderzhanija sajtov; / Rostislav Chebykin —BHV-Peterburg - Moskva, 2014. - 528 c.

14. Jenzh Jerik. SEO. Iskusstvo raskrutki sajtov; / Jerik Jenzh , Stefan Spenser , Rjend Fishkin , Dzhessi Strikchiola —BHV-Peterburg - Moskva, 2014. - 668 c.

Рецензія/Peer review : 10.04.2018 р. Надрукована/Printed :17.05.2018 р.
Стаття рецензована редакційною колегією