ТЕХНОЛОГІЇ РОЗРОБКИ ТА СУПРОВОДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 004.413
Сидоров М.О., Костів М.М.
Національний авіаційний університет

МЕТОД СТВОРЕННЯ СТИЛЮ ЕФЕКТИВНОГО ПРОГРАМУВАННЯ

У статті розглянута задача створення стилю ефективного програмування, описано виконання реверсивної інженерії і проведення експериментів для її вирішення, наведено результати експериментальних досліджень для створення рекомендацій стилю.

В статье рассмотрена задача разработки стиля эффективного программирования, описано выполнение реверсивной инженерии и проведение экспериментов для ее решения, представлены результаты экспериментальных исследований для создания рекомендаций стиля.

In the article the task of creation of style of effective programming is considered, the method of reverse engineering is used and the experiments are conducted for its solving, the results of experimental research for creation of the recommendations are represented.

Ключові слова: *стиль ефективного програмування, реверсивна інженерія, конструкція мови програмування, домен.*

Постановка проблеми

У зв'язку з використанням інженерних методів розробки програмного забезпечення, повторного використання і екстремального програмування при корпоративній розробці ПЗ особливого значення набуває використання стилів програмування під час написання програм[1, 2].

Правила і рекомендації щодо написання програм використовували ще в епоху «до структурного програмування». В цей час комп'ютери були повільними і пам'ять обмеженою, а всі ідеї спрямовані на розробку ефективних по швидкості виконання або пам'яті програмам.

Поява потужних комп'ютерів четвертого покоління дозволяє створювати програми, які раніше неможливо було реалізувати. Це призвело до того, що програмне забезпечення досягло розмірів і рівня складності, які в багато разів перевищують аналогічні показники у програмних системах створених на обчислювальній техніці попередніх поколінь.

У зв'язку з підвищенням складності програм в сучасному світі розробникам ПЗ необхідно знову приділяти увагу ефективності, створювати відповідні технології, стилі і методи управління та розробки великих програм.

Стиль ефективного програмування необхідно використовувати для створення великих і складних програм тому, що час виконання таких програм може бути великим. Тоді програмісту, у більшості випадках, доводиться виконувати аналіз коду і його оптимізацію з точку зору швидкості його виконання.

Застосування стилю ефективного програмування допомагає зробити програму швидшою ще на етапі написання її тексту і саме тому розробникам програмного забезпечення необхідно дотримуватись правил стилю [3].

Швидкість виконання програми можна підвищити не тільки застосуванням стилю, який базується на ефективних конструкціях мови програмування, а й покращенням існуючих алгоритмів і видаленням вузьких місць в коді (bottleneck) [4]. Такі місця в тексті програми або критичні частини коду споживають до 20% необхідного ресурсу (часу, пам'яті). Іноді розробники при створенні програми думають лише про виконання функціональних вимог, що призводить до створення функціональних, проте неефективних програм чи появі незапланованих «вузьких місць» в ресурсах.

В даній статті розглянуто завдання створення стилю. Розв'язання завдання продемонстровано на прикладі мови програмування PHP.

Орієнтовані на ефективність стилі поки що представлені тільки у вигляді розмірковувань деяких авторів про те, як писати ефективні програми. Наприклад, Б. Страуструп визначає які структури даних ефективніше використовувати для вирішення певних задач і відповідно створює рекомендації [4].

Ефективність програми можна визначити використанням двох ресурсів. По-перше, це необхідний час для виконання програми і, по-друге, пам'ять, яка потрібна програмі. Далі буде розглянуто створення стилю ефективного програмування спрямованого на прискорення часу роботи ПЗ.

Для Web - застосувань швидкість виконання тексту програми важливий фактор, тому що з появою технології WEB 2.0 Web-застосування стають інтерактивними і доступними для мільйонів людей по всьому світу.

Використання стилю ефективного програмування зменшує час виконання програми, що в свою чергу призводить до зниження завантаженості сервера і, відповідно, збільшення кількості користувачів, які можуть одночасно взаємодіяти з сервером. Крім цього використання стилю ефективного програмування відповідає вимогам екології програмного забезпечення.

Для алгоритмів ранжування пошукові системи використовують значення швидкості Web-застосування як головну характеристику для визначення його позиції у результатах пошуку. Розробники ПЗ, які планують досягти високих позицій створюють ефективне ПЗ з удосконаленням його в процесі розробки.

Зараз розробники надають користувачу програмне забезпечення разом з системними вимогами (до частоти процесора, об'єму оперативної пам'яті, тощо) для можливості його використання на даному апаратному забезпеченні. Застосування стилю ефективного програмування допомагає створити зелену програму і знизити мінімальні вимоги до апаратного забезпечення та втрати [5].

Розв'язання проблеми

Досягти зростання швидкості виконання коду розробник ПЗ може двома шляхами: по-перше, обрати більш ефективні алгоритми, а по-друге, дотримуватись правил стилю ефективного програмування. У статті розглянуто саме створення даного стилю (рис.1).

Рис.1. Схема побудови стилю ефективного програмування

Стиль ефективного програмування буде розроблений для мови програмування PHP, і тому далі розглядаються конструкції до цієї мови.

Схему побудови стилю ефективного програмування зображено на моделі наукового методу (Рис.1).

Алгоритм побудови гіпотез поділяється на дві складові частини: розвязання задач до обраного домену та створення гіпотез на основі вирішених задач (Рис. 2).

Спочатку необхідно визначити конструкції для вирішення задач, які відносяться до певного домену і виникають найчастіше у веб-програмуванні (табл. 1).

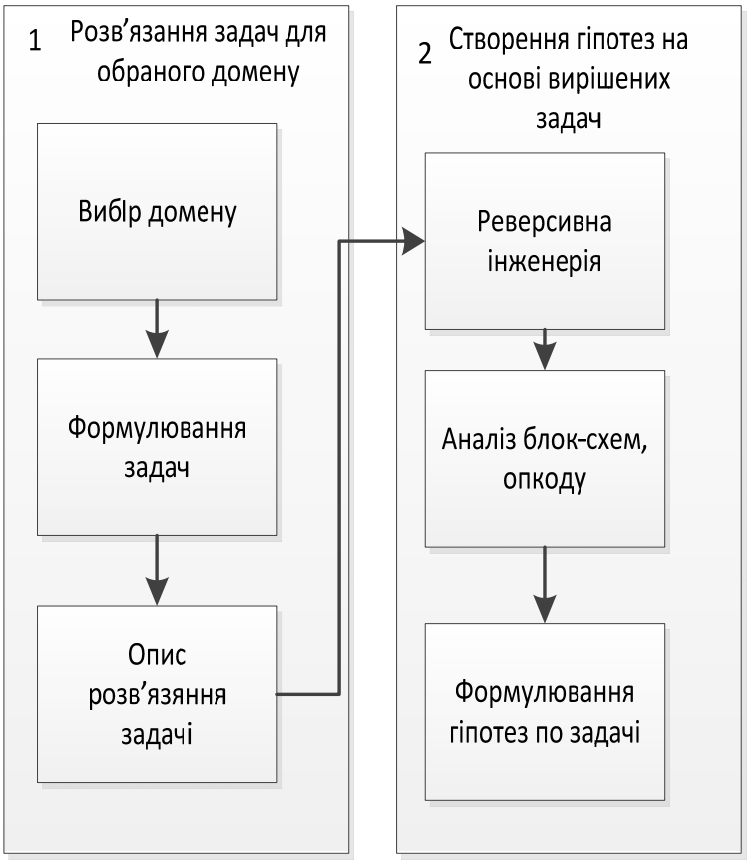

Рис. 2. Алгоритм побудови гіпотез

Таблиця 1.
Можливі задачі згруповані по доменах

Домен	Задача
Сортування і пошук в масивах	Перевірка масиву на пустоту
	Перевірка існування елементу масиву
Асоціативний масив	Отримання доступу до елементу асоціативного масиву
	Прохід по асоціативному масиву
Робота з символьними значеннями	Виведення тексту
	Перевірка довжини рядка

Для кожної задачі створимо алгоритми її вирішення і реалізуємо їх на мові програмування PHP у вигляді конструкцій програмного коду, який використовує операторний базис мови PHP.

Конструкції згрупуємо по спільній задачі, яку вони вирішують (дві конструкції в одну групу). Кількість конструкцій може бути збільшена у процесі розширення й удосконалення стилю. Класифікацію, яку створено в результаті групування конструкцій наведено в табл. 2.

Таблиця 2.
Класифікація конструкцій

№ групи	Задача	Перша конструкція	Друга конструкція
1	Перевірка масиву на пустоту	$items =array(); if(empty($items)) { print 1; }	$var = true; if (true == $var) { print 1; }
2	Перевірка існування елементу масиву	$array = array('a' => 1,'b' => 2); if(isset($array['a')) {}	$array = array('a' => 1, 'b' => 2); if(array_key_exists('a', $array)) {}
3	Прохід по асоціативному масиву	for ($i = 0; $i<$arrLength; $i++) { echo $array[$i]; }	foreach ($array as $value) { echo $value; }
4	Отримання доступу до елементу асоціативного масиву	$array = array('a' => 1, 'b' => 2); echo $array['a'];	4$array = array('a' => 1, 'b' => 2); echo $array[a];
5	Виведення тексту	$string = ‘text’; echo $string;	$string = ‘text’; print $string;
6	Перевірка довжини рядка	if (!isset($str[2])){ echo ‘Length less than 3 symbols’; }	if (strlen($str) < 3) { echo ‘Length less than 3 symbols’; }

Формулюючи гіпотези для кожної задачі у групі визначимо теоретично, яка з конструкцій для її вирішення працює швидше, що необхідно для підвищення рівня розуміння стилю і можливості кращого пояснення його правил в процесі навчання людей стилю ефективного програмування.

Теоретичне підтвердження або спростування гіпотез здійснимо за допомогою виконання реверсивної інженерії.

У реверсивній інженерії використаємо опкод, в який будемо транслювати PHP скрипт за допомогою інструменту Vulcan Logic Disassembler [6]. Аналіз опкоду та побудову блок-схем для кожної конструкції у групах необхідно здійснювати під час виконання реверсивної інженерії.

На рис.3 зображена блок – схема для опкоду першої конструкції, а на рис.4, відповідно, блок-схема для опкоду другої конструкції для задачі “Перевірка масиву на пустоту” з табл. 2.

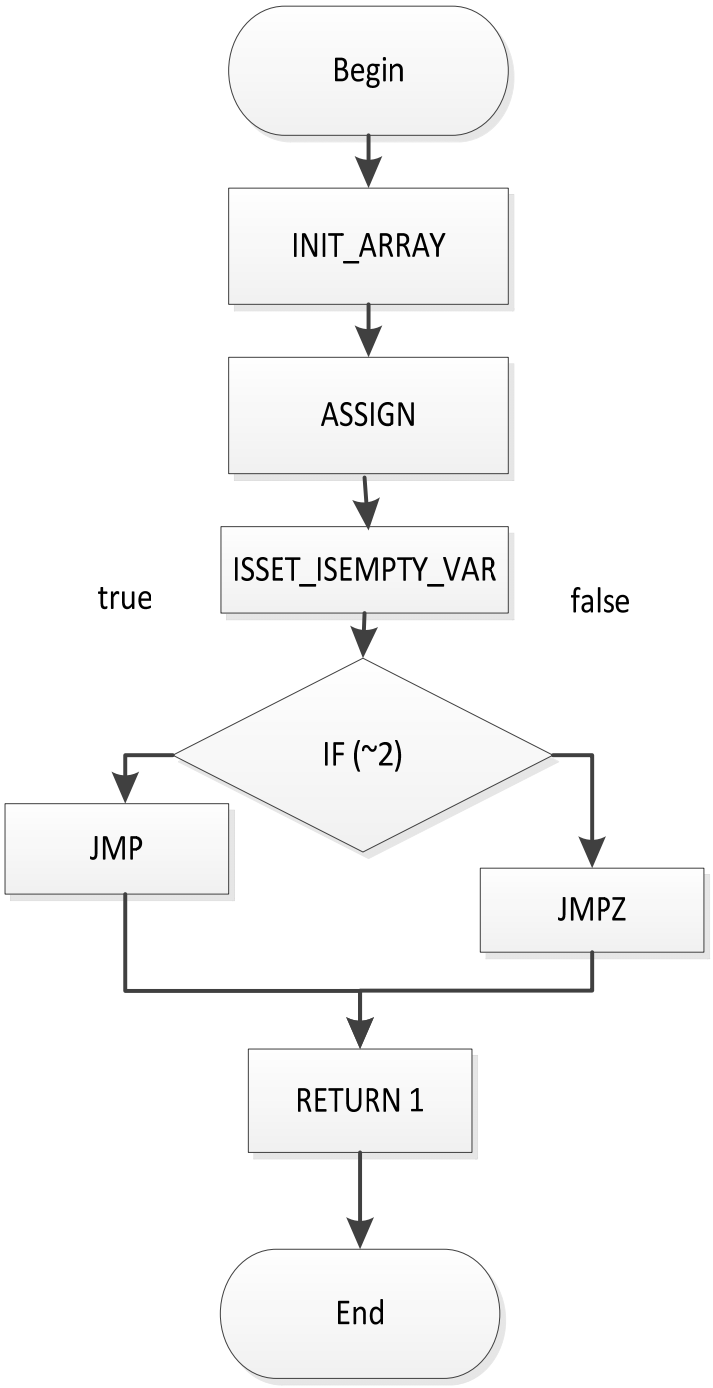

Рис.3. Блок-схема опкоду для першої конструкції

Після застосування реверсивної інженерії необхідно проаналізувати і порівняти блок-схеми для конструкцій в одній групі.

Припускаємо, що чим більше команд містить конструкція, тим більше часу необхідно витратити для їх виконання. Відповідно, якщо блок-схема для однієї конструкції містить більше команд ніж інша у групі, то вона буде визначатись як повільна, а інша – як швидка

Аналізуючи отримані блок-схеми, можна створити та обґрунтувати гіпотези, які перевіримо виконанням експерименту.

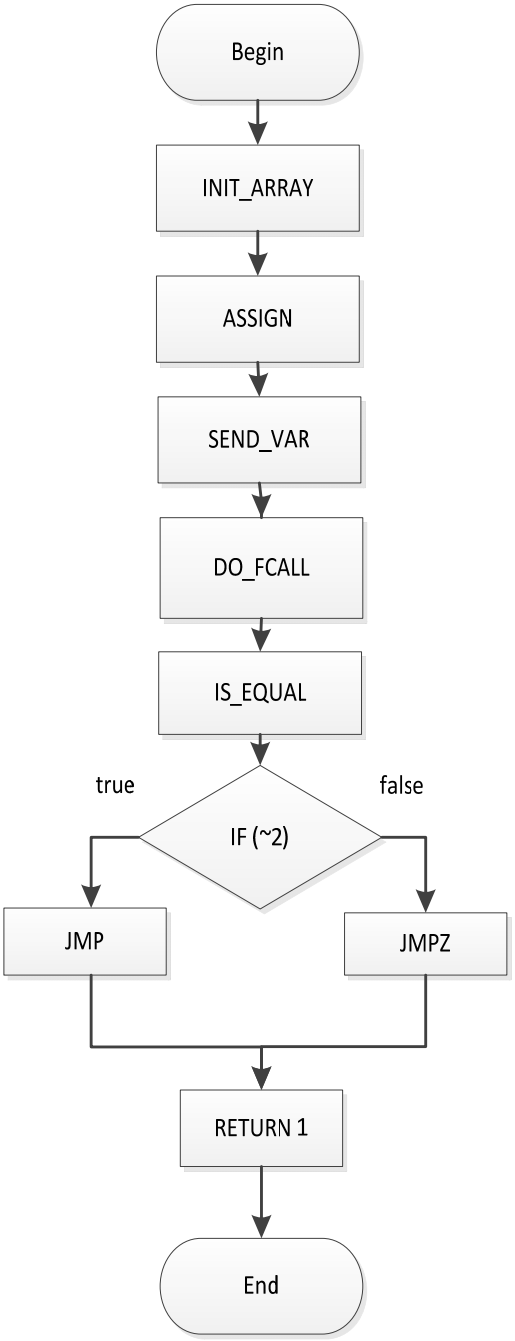

Рис.4. Блок-схема опкоду для другої конструкції

Таблиця 3. Результат роботи програми для першої групи конструкцій.

Конструкція	Код конструції	Середній час виконання	Відхилення від мінімального	Відсоток від максимального	Більше ніж мінімальне	Більше ніж максимальне	Час виконання
1	$items =array(); if(empty($items)) {print 1;}	0.0010	0.0000	32.2581	1.0000	3.1000	
2	$var = true; if (true == $var){print 1;}	0.0031	210.0000	100.0000	3.1000	1.0000	

Таким чином, блок-схема для першої конструкції (рис. 3) містить сім команд, а для другої (рис. 4), відповідно, містить дев'ять команд. У першій конструкції потрібно виконати одну команду ISSET_ISEMPTY_VAR замість трьох команд SEND_VAR, DO_FCALL, IS_EQUAL у другій конструкції. У першій конструкції потрібно виконати на дві команди менше у порівнянні з другою.

Тепер можна створити гіпотезу, що перша конструкція буде працювати швидше за другу.

Для проведення експерименту, який допоможе підтвердити або спростувати гіпотезу, потрібно створити інструмент, що вимірює час виконання кожної конструкції у групі і дає оцінки їх швидкості та провести вимірювання (рис. 5).

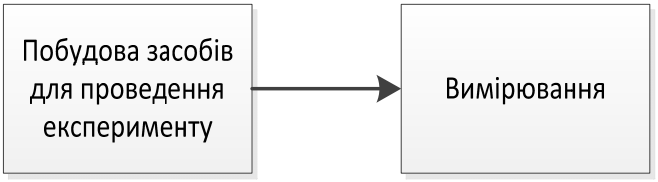

Рис.5. Проведення експерименту

Результати експерименту для конструкцій, сформованих у групу відносно спільної задачі "Перевірка масиву на пустоту", наводяться в табл.3. Отримані дані вказують на те, що друга конструкція є швидшою в порівнянні з першою в 3.1 раз. Таким чином, висунута гіпотеза підтверджується експериментально.

Виконавши усі попередні кроки для інших груп конструкцій, можна розробити стиль ефективного програмування, що містить рекомендації для підвищення швидкості роботи програм (рис. 6).

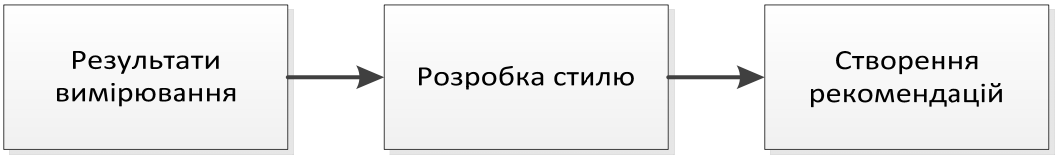

Рис.6. Побудова стилю програмування

Наприклад, слід використовувати такі рекомендації при написанні програм на мові програмування PHP:

- для перевірки масиву на пустоту використовуйте конструкцію if (empty($array)) {} замість if (count($array) == 0) {};
- для проходу по елементам асоціативному масиву і роботою з кожним його значенням використовуйте foreach замість for;
- для перевірки існування елементу асоціативного масиву за індексом використовуйте функцію isset із вказанням індексу елемента;
- використовуйте функцію isset[$array['key']] замість функції array_key_exists('key', $array);
- для порівняння змінної та значення true використовуйте конструкцію if ($var){} замість if (true == $var){};
- для виведення тексту використовуйте фунцію echo замість print.

Висновки

Розв'язання розглянутої задачі у статті має наукову і практичну цінність. Наукова цінність полягає в використанні наукового методу для створення стилю ефективного програмування в інших мовах програмування. Можливість використання стилю для написання ефективного програмного забезпечення з точку зору швидкості є практичною цінністю.

У статті створення стилю ефективного програмування розглянуто на прикладі мови програмування PHP для використання у розробці Web-застосувань. Розроблений стиль ефективного програмування може бути вдосконалений і розширений шляхом додавання нових задач і конструкцій.

.

Список використаних джерел

1. Сидоров Н.А. Стилистика программного обеспечения / Н.А. Сидоров // Проблемы программирования. − 2006. − № 2 − 3. − С. 245 − 255.

2. Sidorov N.A. Software stylistics / N.A. Sidorov // Вісник НАУ. – №2. – 2005. – С. 98 – 103

3. Крамар Ю. М. Применение стилей программирования в конструировании программного обеспечения / Ю.М. Крамар // Інженерія програмного забезпечення. – 2011. – № 1(5). – С. 46 – 53.

4. Schlossnagle G. Advanced Php Programming: A Practical Guide to Developing Large-Scale Web Sites and Applications with Php 5/ George Schlossnagle – Sams Publishing, 2004. – 534 p.

5. Stroustrup B. Software Development for Infrastructure. Computer / B. Stroustrup. – 2012. – Vol. 45, no. 1. – P. 47 – 58.

6. Сидоров Н.А. Экология программного обеспечения / Н.А. Сидоров // Інженерія програмного забезпечення. – 2010. – №1. – С.53 – 61.

7. Vulcan Logic Disassembler [Электронный ресурс] – Режим доступа: http://pecl.php.net/package/vld

Відомості про авторів:

Сидоров Микола Олександрович – доктор технічних наук, професор, завідувач кафедри інженерії програмного забезпечення Інституту комп'ютерних інформаційних технологій Національного авіаційного університету. Наукові інтереси: інженерія програмного забезпечення.

E-mail: nikolay.sidorov@livenau.net

Костів Мілана Миколаївна – студентка 5 курсу Інституту комп'ютерних інформаційних технологій Національного авіаційного університету. Наукові інтереси: інженерія програмного забезпечення.

E-mail: milana.kostiv@livenau.net