ИНТЕГРАЦИОННОЕ ТЕСТИРОВАНИЕ НЕПРЕРЫВНО РАЗВЕРТЫВАЕМЫХ ВЕБ-СИСТЕМ С ПОМОЩЬЮ ЭКСПЕРИМЕНТАЛЬНЫХ ВЫКЛАДОК

INTEGRATION TESTING OF CONTINUOUSLY DELIVERED WEB-SYSTEMS USING EXPERIMENTAL DEPLOYMENT

I. Skorokhodov
A. Tikhomirova

Annotation

Continuous integration, delivery and deployment of web systems becomes more and more popular nowadays, because it makes company more flexible and lets it react on changes more rapidly. While using such a practice system testing, especially integration testing becomes much more important. The classic ways to test web–systems are manual testing and writing automated tests. Both approaches are too time–consuming and have great limitations in scaling. So in these latter days experimental deployment gains popularity in system integration testing. It is a procedure of distribut–ing company's product on a separate share of users and analyzing their behavior to make conclusions about quality of the new product version. Despite of popularity of this kind of approach its theoretical reasoning is almost completely absent. In this paper there are developed a model to determine an experiment which a user should be distributed in and a crite–ria of experiment termination.

Keywords: integration testing, continuous deployment, continuous integration, experimental deployment.

Скороходов Иван Сергеевич
Магистрант, ФГАОУ ВО "Национальный исследовательский ядерный университет "МИФИ"
Тихомирова Анна Николаевна
К.т.н., доцент, ФГАОУ ВО "Национальный исследовательский ядерный университет "МИФИ"

Аннотация

Непрерывная интеграция, доставка и развертывание веб–систем становится все более популярной, так как она позволяет компании быть максимально гибкой и быстрее реагировать на изменения. При данном подходе важную роль начинает играть тестирование компонентов системы, особенно интеграционное. Классическими способами проведения данной процедуры являются ручное тестирование и написание интеграционных тестов. Оба этих подхода несут существенные временные затраты и имеют ограничения в масштабировании. Поэтому в последнее время все больше компаний прибегают к экспериментальной проверке изменений, суть которой заключается в выкладки новой версии приложения на определенную долю пользователей, отслеживание их поведенческих метрик и выявлении на их анализе качества внесенных изменений. При этом сегодня практически отсутствуют теоретические обоснования данного подхода, особенно выявление баланса между продолжительностью эксперимента и рисками потерь от недоброкачественных изменений. В данной работе предложена модель определения экспериментальной выборки для пользователя, а также критерий терминации эксперимента.

Ключевые слова:

Интеграционное тестирование, непрерывная интеграция, непрерывное развертывание, экспериментальное развертывание.

Введение

Непрерывное развертывание по своей сути заключается в максимальном сокращении релизных циклов продукта [1]. Тестирование продукта при этом становится особенно затратным: так как необходимо проверять каждый релиз, то тестирование становится крайне узким местом масштабирования бизнеса. Поэтому компании стараются максимально упростить и ускорить данную процедуру.

Традиционно существует два основных подхода проведения тестирования системы:

- ручное тестирование мануальными QA–специалистами [2];
- автоматизированное тестирование с помощью интеграционных и юнит–тестов [3].

Ручное тестирование является крайне затратным по времени, поэтому плохо масштабируется. Написание интеграционных тестов, в свою очередь, также требует существенных временных затрат на их создание, что также несет большие издержки для компаний. Таким образом, существует необходимость каким–либо способом удешевить данную процедуру.

Поэтому в последнее время все популярнее становится подход, заключающийся в развертывание новой вер–

сии системы в экспериментальном режиме и последующем определении ее качества на основе анализа изменений различных пользовательских метрик [4].

Ключевыми вопросами при этом являются следующие:

- выбор нужных метрик для отслеживания;
- определение времени прекращения эксперимента.

Несмотря на огромную важность и актуальность данной темы теоретические обоснования поставленных вопросов сегодня практически отсутствуют. Каждая компания вынуждена самостоятельно выбирать метрики для отслеживания и критерий терминации, как правило, без какой-либо заранее определенной методологии.

Современные подходы к проведению экспериментального развертывания

Существует два типа проведения экспериментов: параллельные эксперименты (сплит-тесты) и последовательные (A/B-тестирование).

Параллельные эксперименты, которые также называют сплит-тестами, заключаются в развертывании двух версий системы одновременно: эталонной и экспериментальной. Как правило, их могут позволить себе только крупные компании, так как необходима достаточно большая база пользователей, чтобы выделить из них достаточно большую репрезентативную долю. Кроме того, инфраструктура для проведения сплит-тестов достаточно сложна: необходима специальная прослойка (например, L7-балансер [5]), который будет распределять пользователей по экспериментальным сплитам, а также возможность запуска нескольких версий системы одновременно.

Последовательные эксперименты, которые иногда называют A/B-тестированием, предполагает выкладку эксперимента на 100% пользователей и отслеживании на них изменений. Данная процедура является более рискованной, чем сплит-тесты: неудачные эксперименты повлияют сразу на всех пользователей сервиса. Инфраструктура для A/B-тестирования значительно проще, что дает возможность проводить его небольшим компаниям.

В настоящее время существуют предприятия, которые поставляют систему экспериментального развертывания (как параллельного, так и последовательного) на аутсорсинг, и все больше компаний прибегают к их услугам [6]. При этом данные аутсорс-организации оставляют на усмотрение клиента решение по определению критерия терминации эксперимента. Также они оставляют закрытой методологию распределения пользователей по экспериментальным сплитам.

Технические аспекты работы непрерывно развертываемых веб-систем

По своей сути, непрерывное развертывание заключается в частых релизных циклах продукта: каждая небольшая задача (или несколько задач), выполненная разработчиком, собирается в отдельный релиз. Данная практика используется для любого типа ПО, а не только веб-систем, но у последних есть существенное преимущество: они могут форсировать обновление клиентской части сервиса [1].

При проведении экспериментов ключевым является сохранении информации на клиентской части об экспериментальном сплите. Это необходимо для того, чтобы отслеживать пользовательские данные, правильно распределять их по экспериментальным сплитам, а также направлять пользователя в эксперимент, если он зашел на сайт повторно.

Для веб-систем существует два основных способа сохранять информацию на клиенте:

- в клиентском хранилище localStorage [7];
- в cookie-заголовках запросов [8].

Клиентское хранилище localStorage имеет больший, чем cookie-заголовки, объем, но менее надежно и доступно только для клиентских скриптов, поэтому больше подходит для хранения больших и не очень важных объектов. Cookie-заголовки доступны в запросе клиента, поэтому намного более предпочтительны для хранения ключевых данных пользователя.

Взаимодействие с существующими веб-системами осуществляется в основном по протоколу HTTP [9]. Он предполагает наличие на сервере определенных ресурсов, которые запрашивает клиент. Данные ресурсы могут быть как статическими, так и динамическими. В последнем случае сервер имеет возможность модифицировать ресурс в зависимости от различных параметров запроса, которые хранятся в заголовках, теле и адресе. Именно это и позволяет удобно делить пользователей по экспериментам в веб-системах.

Определение пользователя в экспериментальный сплит

При получении запроса сервер должен направить пользователя в определенный тест-сплит.

В общем виде схема обработки сервером запроса представлена на **рис. 1**.

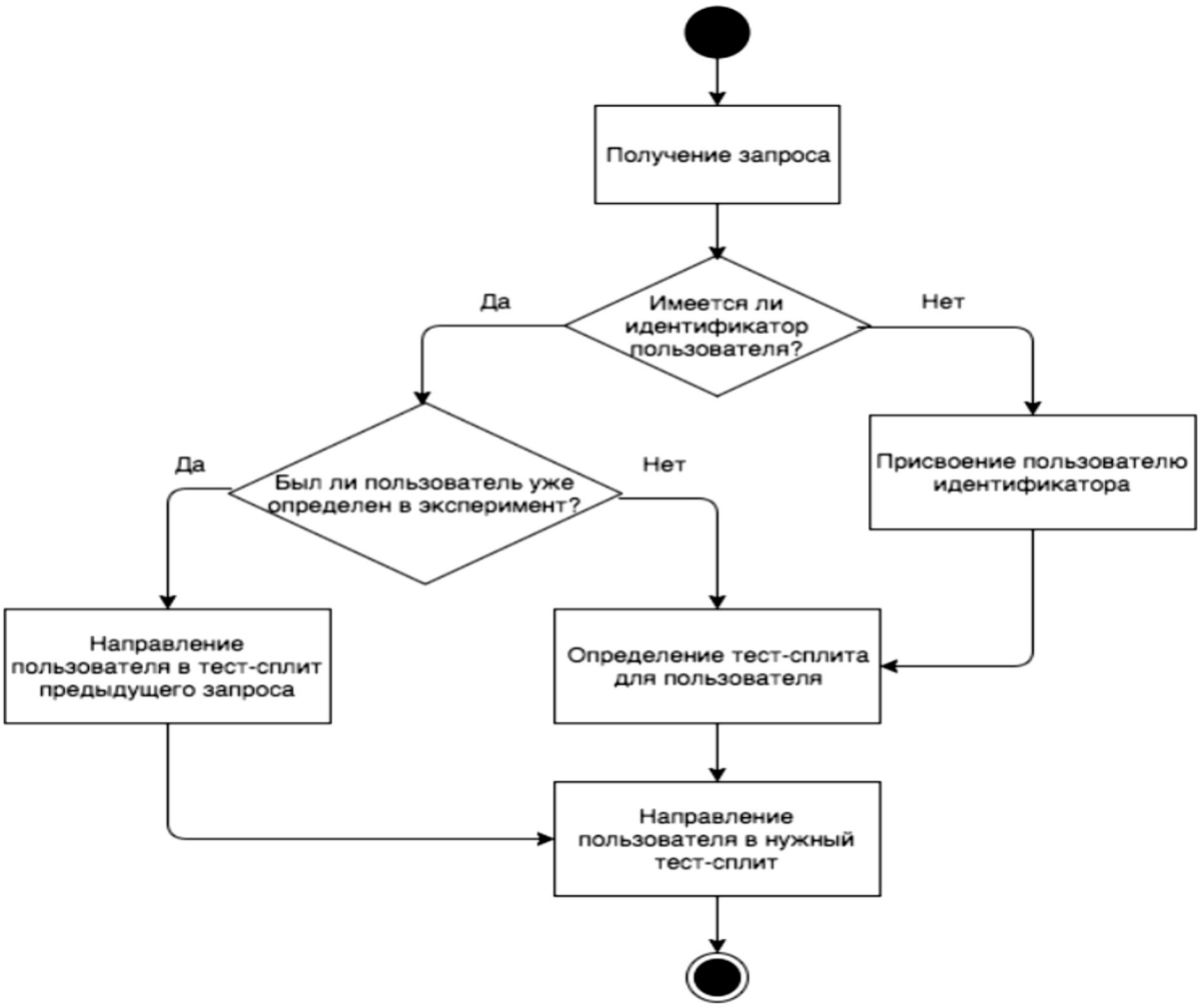

Рисунок 1. Схема обработки пользовательского запроса.

Выбор экспериментального сплита определяется на основе существующего распределения пользователей по экспериментам и того факта, был ли данный пользователь уже распределен при его предыдущих запросах. Для определения экспериментального сплита для пользователя необходимо основываться на существующем распределении пользователей по экспериментам.

Распределение значений признаков запросов в эксперименте должно совпадать с распределением аналогичных значений в эталонном сплите. То есть, функции распределения признаков в различных сплитах должны максимально точно друг друга приближать:

$$\mathcal{F}_{s_i} \sim \mathcal{F}_{s_k} \text{ для } i,k = 1,2 \ldots n,$$

где: $\mathcal{F}_{s_i}$ – распределение случайной величины в –ом экспериментальном сплите;

n – количество экспериментальных сплитов.

Таким образом, можно сформулировать задачу определения экспериментального сплита для запроса.

Пусть:

$$\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$$

– множество пользователей;

$$\mathcal{S} = \{s_1, s_2, \ldots s_m\}$$

– множество экспериментальных сплитов;

$$\mathcal{F} = \{\mathcal{F}_{s_1}, \mathcal{F}_{s_2}, \ldots, \mathcal{F}_{s_n}\}$$

– существующее семейство распределений в экспериментальных сплитах;

$$\Psi: \Omega \to \mathcal{S}$$

– Сюръективное отображение определяет пользователя в нужный экспериментальный сплит.(Об)

Тогда выбор экспериментального сплита должен основываться на том, насколько похожими станут распре–

деления признаков в сплитах:

$$\Psi(\omega) = \underset{\hat{s}\in\mathcal{S}}{\operatorname{argmax}} \frac{\sum^{s\in\mathcal{S}} r\left(\mathcal{F}_{\hat{s}}, \mathcal{F}_{s}\right)}{|S|}, \quad (1)$$

где r – это определенная функция корреляции, выбор которой зависит от априорных предположениях о шуме в признаках: например, для гауссовского шума можно использовать коэффициент корреляции Пирсона.

Важно отметить, что признаки пользователя должны быть "устойчивыми" по отношению к функционалу сервиса, т.е. должны сохранять одинаковое распределение для любой его версии. К таким признакам можно отнести следующие наиболее популярные:

- случайно генерируемые идентификаторы пользователей, если произвести их хеширование в равномерно распределеный массив чисел на заданном дискретном диапазоне;
- IP–адреса пользователей, если произвести их хеширование в категориальную переменную, характеризующую регион пользователя.

Таким образом, можно правильно определять пользователей в экспериментальные сплиты вне зависимости от их размера.

Определение критерия терминации эксперимента

Открытым вопросом при этом остается критерий терминации эксперимента, то есть определения момента, когда необходимо его прервать.

Можно выделить два типа критерия терминации:

- Критерий терминации первого рода: была собрана вся необходимая статистика о качестве сплита;
- Критерий терминации второго рода: были выявлены отклонения, и эксперимент несет убытки для компании.

Определение первого критерия является достаточно нетривиальным: необходимо быть уверенным, что в каждом сплите находится одинаковое количество целевых пользователей – то есть, пользователей, которые осуществляют целевые для компании действия (например, покупку). Для этого требуется отслеживать суммарное количество целевых действий и сходимость функционалов в каждом сплите. Пусть $D_{si}(t)$ – суммарное количество целевых действий в сплите Si за временной промежуток t.

Утверждение

При правильном распределении пользователя по сплитам (выполнено условие из формулы (1)) и одинаковых версий системы в сплитах S_i и S_k выполнено:

$$\frac{\mathcal{D}_{s_i}(t)}{\mathcal{D}_{s_k}(t)} \to 1 \text{ при } t \to \infty.$$

Доказательство

По условию (1) пользовательские признаки в сплитах S_i, S_k являются независимыми одинаково распределенными случайными величинами (многофакторными). При одинаковой версии системы в сплитах, они имеют одинаковые условия для распределения поведенческих факторов. Тогда данная ситуация ложится в условияк лассической центральной предельной теоремы, и сумма их целевых действий имеет нормальное распределение. Следовательно, функционалы имеют сходимость по распределению [10].

Таким образом, если система работает без отклонений, то она будет порождать одинаковые условия для формирования действий пользователей, и будет наблюдаться сходимость функционалов. При этом период сходимости является уникальным для каждой компании и должен подбираться самостоятельно.

Критерий терминации второго рода будет обуславливать расходимость функционалов D_{Si} и D_{Sk} (t).

При этом, если

$$\frac{\mathcal{D}_{s_i}(t)}{\mathcal{D}_{s_k}(t)} > 1$$

то S_i является более прибыльным, чем S_k.

В случае, если

$$\frac{\mathcal{D}_{s_i}(t)}{\mathcal{D}_{s_k}(t)} < 1$$

то S_i является более убыточным.

Заключение

Существующие подходы интеграционного тестирования являются крайне затратными по времени и плохо масштабируются. Особенно остро встает подобная проблема для компаний, разрабатывающих свои веб–системы по практике непрерывного развертывания.

Это обуславливает тот факт, что все больше компаний в последнее время прибегают к выкладке новых версий системы в экспериментальном режиме и анализе качества изменений на основе поведения пользователей в экспериментальном сплите.

В данной работе была предложена модель определения пользователя в экспериментальный сплит вне зави–

симости от типа эксперимента.

Также был предложен подход определения качества новой версии системы, который позволяет определить момент, когда необходимо свернуть эксперимент.

ЛИТЕРАТУРА

1. Джез Хамбл and Дэвид Фарли. Непрерывное развертывание ПО. Автомати– зация процессов сборки, тестирования и внедрения новых версии? программ. Вильямс, 2011.
2. Rick David Craig and Stefan P. Jaskiel. Systematic Software Testing. Artech House, 2002.
3. Elfriede Dustin, Jeff Rashka, and John Paul. Automated Software Testing: Introduction, Management, and Performance. Addison–Wesley, 1999.
4. Dan Siroker and Pete Koomen. A/B Testing: The Most Powerful Way to Turn Clicks Into Customers. Wiley, 2013.
5. Tony Bourke. Server Load Balancing. O'Reilly Media, 2001.
6. Optimizely – главная страница. https://www.optimizely.com/, 2016.
7. Ian Hickson. Web storage. https://www.w3.org/TR/webstorage/, 2016.
8. Http state management mechanism. Technical report, Internet Engineering Task Force (IETF), 2011.
9. David Gourley, Brian Totty, Marjorie Sayer, Anshu Aggarwal, and Sailu Reddy. HTTP: The Definitive Guide. O'Reilly Media, 2002.
10. Севастьянов Борис Александрович. Курс теории вероятностеи? и математи– ческои? статистики. Наука, 1982.