УДК 004.9 DOI 10.5862/JCTCS.212.8

В.В. Брекелов, Е.А. Борисов, И.А. Барыгин

АВТОМАТИЗАЦИЯ ИНТЕГРАЦИОННОГО ТЕСТИРОВАНИЯ НА ПРИМЕРЕ МОДУЛЕЙ ОБМЕНА ДАННЫМИ ПО FIX-ПРОТОКОЛУ

V.V. Brekelov, E.A. Borisov, I.A. Barygin

INTEGRATION TESTING AUTOMATION: CASE STUDY OF FINANCIAL DATA EXCHANGE MODULES BASED ON FIX-PROTOCOL

В трейдинговых системах в качестве транспортного протокола наиболее распространенным является FIX-протокол. Ручное тестирование модулей, интегрирующих финансовые системы посредством FIX-протокола, – весьма трудоемкий процесс.

Рассмотрены автоматизация интеграционного тестирования упомянутых модулей, подход к написанию тестовой документации, возможные проблемы интегрируемых компаний и их решение, временная оценка выполняемых тестов и достигнутое покрытие функциональности тестовыми сценариями.

Созданы тестовая документация и автотесты с общей структурой, не только обеспечивающие быстрое тестирование, но и позволяющие быстро адаптироваться к новым финансовым системам или к новым требованиям.

FIX-ПРОТОКОЛ; АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ; ИНТЕГРАЦИОННОЕ ТЕСТИРОВАНИЕ; ТЕСТ-КЕЙС; ТРЕЙДИНГОВАЯ СИСТЕМА; БИРЖА.

The majority of modern trading systems use FIX-protocol as a transport protocol for data services. Manual testing of trading system integration modules responsible for FIX messaging is an overly laborious process. The paper describes a complex automated testing approach for this type of integration testing, which incorporates improvements to test the documentation structure, and tackles the problem of vendors diversity, as well as the resulting functional coverage and timing estimates of the tests. The major outcome of this work is a complete and unified auto tests set with associated documentation, which sufficiently accelerates testing procedures and allows fast incorporation of new vendors and fast adaptation to changes in existing specifications.

FIX-PROTOCOL TEST; AUTOMATION; INTEGRATION TESTING; TEST CASE; TRADING SYSTEM; EXCHANGE.

Современная трейдинговая система является сложным программным продуктом, предоставляющим участникам торгов различные сервисы. Такая система передает всю финансовую информацию брокерам (вендорам), используя FIX-протокол (Financial Information eXchange protocol – протокол обмена финансовой информацией), – протокол передачи данных, являющийся международным стандартом для обмена данными между участниками биржевых торгов в режиме реального времени. Протокол FIX поддерживается большинством крупнейших банков и электронными трейдинговыми системами, а также крупнейшими биржами мира [1].

Любое програмное обеспечение требует проверки качества. Для финансовых систем особенно критичной областью является передача информации о торговых заявках. В данной статье рассматривается решение ряда проблем интеграционного тестирования компонент системы, отвечающих за передачу и получение финансовой информации. Основные трудности заключаются в недостатках ручного тестирования (большое время выполнения, человеческий фактор, необходимость обучения персонала), а также в неполной тестовой документации. Для решения перечисленных выше задач использовался подход обобщения тегов относительно инструментов и вендоров. Его цель – одинаково структурировать и удобно поддерживать автотесты и документацию, а также оценивать покрытие тестами функциональной части компонент.

Для разработки автотестов использовался язык Groovy [3], служащий для написания функциональных тестов в проекте. Для хранения документации использовалась система Polarion [2], применяемая внутри всех проектов компании.

Особенности работы трейдинговой системы с FIX-протоколом. Рассматриваемая тестируемая система предполагает взаимодействие с 15 различными вендорами посредством обмена FIX-сообщениями. На рис. 1 изображено взаимодействие торгового приложения с биржами.

Для совершения сделки клиенту необходимо создать заявку на покупку или продажу выбранного финансового инструмента в системе. Эта заявка обрабатывается на стороне пользователя и посылается на сервер. Затем она пересылается FIX-модулям, которые используют FIX-протокол для кодирования информации, и, наконец, передается вендору. Последний, в свою очередь, обрабатывает полученное FIX-сообщение и отправляет ответ, в котором содержится информация о статусе ордера клиента.

Особенность использования рассматриваемого протокола брокерами и торговыми системами заключается в различных реализациях. Этот факт усложняет проверку качества при одинаковом подходе для каждого из вендоров. Именно по этой же причине нет единого программного обеспечения, которое применимо для автоматического тестирования компонент, использующих FIX-протокол.

FIX-сообщение. Формат передаваемого сообщения о заявке представляется в виде строчки, которая состоит из набора полей тег = значение. Поля разделяются ASCII кодом SOH — Start of Header (0x01). Например, при покупке опциона IBM отправляется следующее сообщение:

```
8=FIX.4.1|9=169|35=D|52=20140911-22:27:
34|34=403|56=test|49=test|11=2014091-
3000265043|167=OPT|55=IBM|201=1|202=190|
200=201410|205=18|38=10|54=1|77=O|
40=2|44=5.3|59=0|204=0|439=777|47=A|10=212
```

Объяснение некоторых тегов:

55(Symbol) = IBM – показывает по какому инструменту отправлен ордер;

167(SecurityType) = OPT – означает, что ордер отправлен по опциону;

44(Price) = 5.3 – указывает на цену, по которой был издан ордер.

В ответ клиент получает сообщение такого же формата.

Интеграционное тестирование. Интеграционное тестирование – это процесс проверки взаимодействия различных частей системы. В этом случае объектами тестирования являются не функции, непосредственно выполняемые отдельными компонентами (модульное тестирование), а любые вызовы, передачи контроля и качественные характеристики в происходящем между этими компонентами взаимодействии.

Стандартные подходы к интеграционному тестированию предполагают как проверки работы модулей с помощью заглушек и драйверов в изоляции от контекста всей системы, так и тестирование на полностью собранной системе. В контексте рассматриваемой задачи нас интересует функциональное взаимодействие интерфейсов клиента, сервера и FIX-компонент, происходящее при обработке ордеров в торговом приложении, полноценно функционирую-

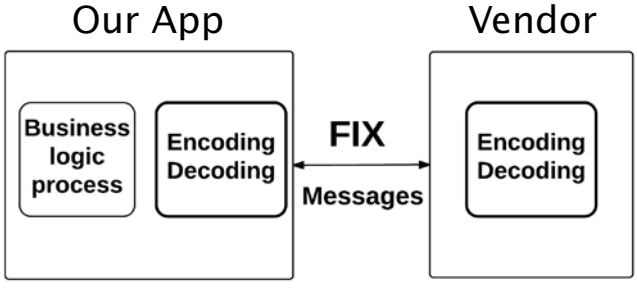

Рис. 1. Взаимодействие с вендорами

Рис. 2. Процесс ручного тестирования

щем в тестовой среде.

Ручное тестирование компонент обмена финансовой информацией. На рис. 2 показан цикл ручной проверки одного тега в сообщении о заявке. Инженеру по контролю качества необходимо сделать следующие шаги:

выполнить вход в трейдинговое приложение;

открыть систему Polarion для хранения тестовой документации;

отправить ордер согласно тест-кейсу;

подключиться к серверу по SSH, на котором запущены FIX-модули;

найти FIX-сообщения для отправленного ордера;

сравнить проверяемый тег с значением в тест-кейсе.

Тест-кейс – это набор действий с ожидаемым результатом, необходимый для проверки части функциональности приложения. Приведенный цикл необходимо проделывать для каждого тега тестируемого вендора.

Необходимо отметить, что тестирование FIX-модулей в рассматриваемом проекте проходит с использованием соединения к демо-платформе вендора, т. е. без использования эмуляций. Такой способ позволяет найти дефекты при изменениях на стороне интегрируемой финансовой организации, что как показывает практика, очень важно.

Проблемы. Использование ручного тестирования содержит ряд проблем.

Человеческий фактор. В ходе описанного выше рутинного процесса инженер может ошибиться, что влияет на качество тестирования.

Трудоемкость. Для проведения регрессионных тестов (данный тип тестов проводится в каждом цикле разработки приложения) для многих вендоров требуется большое количество времени, что задерживает выпуск продукта.

Обучение инженера. Тестируя данную область, необходимо знать бизнес-логику приложения, в т. ч. процесс взаимодействия с вендорами. Это несет дополнительные временные затраты.

Помимо недостатков ручного тестирования существуют сложности с тестовой документацией. Тестовая документация – это набор тест-кейсов, который проверяет функциональную область приложения.

В данном проекте были определены следующие проблемы:

частичное отсутствие тестовой документации (некоторые вендоры полностью недокументированны);

устаревшая информация (часто разработка под новые требования ведется быстрыми темпами, поэтому некоторые части документации не успевают обновиться);

неоптимальная структура (сложная поддержка актуальной тестовой документации связана с неправильным выбором ее структуры).

Автоматизация тестирования и документация. Поскольку функционирование FIX-компонент является критически важным условием при каждом выпуске торговой системы, разрабатываемой по итеративной методологии, и характер тестов для различных вендоров имеет идентичную структуру, эти тесты являются идеальными кандидатами для автоматизации.

Для выполнения данной задачи были рассмотрены существующие решения, такие как QuickFix [9], fiximulator [10], mini-FIX [11]. Но поскольку они предполагают дополнительные затраты на внедрение и адаптацию к реализациям FIX-протокола, а также не решают проблему структуризации тестовой документации, было принято решение раз-

работать автотесты, основываясь на внутреннем фреймворке компании, и подготовить общую структуру для документации.

Подход. Проанализировав все возможные сообщения для различных типов инструментов, а также для различных вендоров [6–8], эмпирическим путем были выведены наборы тегов и разделены на группы. На рис. 3 продемонстрировано выбранное разбиение.

Таким образом получилось выделить четыре группы тегов.

- Общие теги для различных инструментов. Например: 8(FixProtocol), 35(MsgType).
- Специфичные теги для каждого инструмента. Например: 202(MaturityMonth Year).
- Общие теги для различных вендоров. Например: 167(SecurityType).
- Специфичные теги для каждого вендора. Каждая интегрируемая финансовая система имеет свои особенности формата FIX-сообщений. Например, для некоторых необходимо отправлять тег 439(ClearingFirm).

Полученное разбиение использовано для формирования структуры автотестов и тестовой документации.

Автоматизация. Для автоматизации тестирования используется внутренняя разработка нашей компании [5], написанная на языке программирования Groovy [3]. Для управления запуском автотестов применяется TeamCity – серверное программное обеспечение для непрерывной интеграции [4].

Фреймворк имеет возможность «действовать» как обычный пользователь, т. е. позволяет использовать методы и классы клиентского приложения. Автотесты оперируют следующими основными объектами:

Order – объект приложения, содержащий информацию об ордере (используемом инструменте, состоянии ордера, аккаунте, цене и т. д.);

FixOrder – объект фреймворка, содержащий объект Order и ему соответствующие FIX-сообщения;

FixOrderService – объект фреймворка, сервис, отправляющий ордера, используя OrderService, читающий FIX-сообщения с использованием DelayedTailLog и формирующий FixOrder;

Validator – объект фреймворка, содержащий основные методы и DataProvider для тестов.

Структура классов автотестов для каждо-

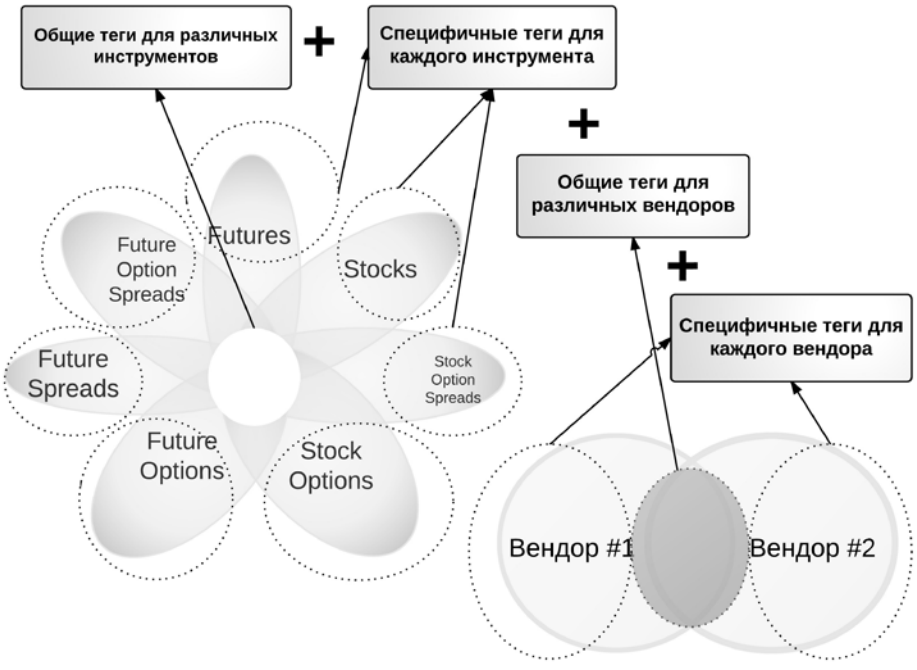

Рис. 3. Выбранное разбиение тегов

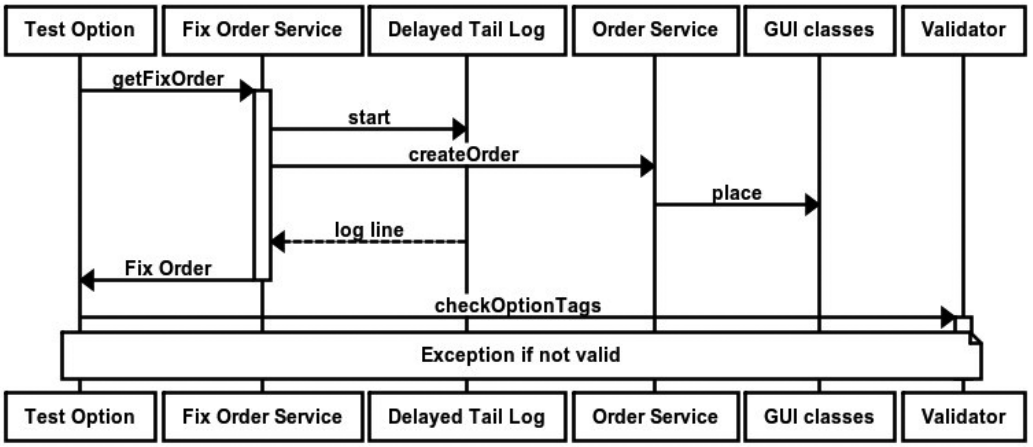

Рис. 4. Схема работы автотеста

го из вендоров соотносится с упомянутым разбиением тегов на группы. К примеру, для вендора, поддерживающего стоковые инструменты, классы автотестов хранятся в пакете <название вендора> и называются в соответствии с типом инструмента:

Common – общие и специфичные теги вендора;

Stock – тесты для стоковых инструментов;

Option – тесты для стоковых опционов;

OptionSpread – тесты для стоковых опционных спредов.

Схема работы автотеста для опционов представлена на рис. 4.

На рис. 5 приведен пример тест-кейса. В разделе Preconditions дана ссылка на общую тестовую процедуру, свойственную именно для проверки FIX-компонент. Input specifications состоит из входных данных: типов ордеров, которые необходимо создавать из клиентского приложения. Output specification содержит ожидаемые результаты теста: проверямые тэги с соответсвую-

Description

Test Items
FIX demoCITI: Ability for sending option order and receiving answer from vendor. Checking set of tags which is the same for all option type orders for CITI.
Preconditions
Use Test Procedure:
/polarion/#/project/TOS_CORE/wiki/FIX/FixTags Test Procedure

Input Specifications
1. Send call Option order, e.g. BUY +10 IBM 100 AUG 14 195 CALL @2.40 LMT
2. Send put Option order e.g. BUY +10 IBM 100 AUG 14 195 PUT @3.20 LMT

Output Specifications
1. Sent, first received with type execution **New**, second received with type execution **Fill** messages all contain:

- '201=1' (PutOrCall)
- '202=195' (StrikePrice)
- '55=IBM' (Symbol)
- '200=201408' (MaturityMonthYear)
- '205=16' (MaturityDay)

2. Sent, first received with type execution **New**, second received with type execution **Fill** messages all contain:

- '201=0' (PutOrCall)
- '202=195' (StrikePrice)
- '55=IBM' (Symbol)
- '200=201408' (MaturityMonthYear)
- '205=16' (MaturityDay)

Рис. 5. Пример тест-кейса

```
@DataProvider(name = «stockOptionCallOrPut»)
    public Object[][] stockOptionCallPut() {
        return [
                [FIX_CALL],
                [FIX_PUT],
        ]
    }
```

Листинг 1. Пример DataProvider

```
protected void checkOptionParameters(String symbol, double quantity) {
FixOrder fixOrder =
 getFixOrder(symbol: symbol, quantity: quantity, type: OrderType.LIMIT)
validator.checkOptionTags(fixOrder)
 }
```

Листинг 2. Пример тестового метода

```
public void checkOptionTags(FixOrder fixOrder) {
String symbol = fixOrder.order.symbol
fixOrder.checkSentAndReceivedTags([
(FixTag.PUT_OR_CALL):   [routingInfo.getSentPutOrCall(symbol),   routingInfo.
getReceivedPutOrCall(symbol)],
(FixTag.STRIKE_PRICE): [routingInfo.getSentConvertedStrike(symbol), routing-
Info.getReceivedConvertedStrike(symbol)],
(FixTag.SYMBOL): routingInfo.getConvertedSymbol(symbol),
(FixTag.MATURITY_MONTH_YEAR):  [routingInfo.getSentMaturityMonthYear(symbol),
routingInfo.getReceivedMaturityMonthYear(symbol)],
(FixTag.MATURITY_DAY):
[routingInfo.getSentMaturityDay(symbol), routingInfo.getReceivedMaturityDay(s
ymbol)],
        ])
    }
```

Листинг 3. Пример внутреннего тестового метода

щими значениями.

Для тестового сценария представлена часть программного кода, исполняющая действия тестовой процедуры (листинг 1).

В части с аннотацией @DataProvider содержатся данные из Input Specifications. Переменная FIX_CALL соответствует инструменту Stock call Option, FIX_PUT – Stock Put Option.

Тест имеет аннотацию @Test и ссылку на тест-кейс в системе Polarion. В тесте вызывается метод checkOptionParameters (листинг 2).

Метод checkOptionTags проверяет ожидаемые значения тегов из Output specification (Листинг 3).

Результаты работы теста представляются с помощью библиотеки TestNG (рис. 6).

Документация. Для хранения тестовой документации используется система Polarion [2]. Она содержит удобный инструментарий для инженера по качеству. Так, например, для процедуры, описывающей одинаковые шаги, необходимые для выполнения тестирования каждого из сценариев (см. рис. 2), была создана отдельная wiki-страница. Сам тест-кейс содержит ссылку на эту страницу. Это позволяет минимизировать текст каждого из тестовых сценариев, оставляя только входные и выходные параметры. Входными параметрами являются типы ордеров, влияющие на значения тегов, что соотносится с DataProvider, используемым в автотестах. Выходными параметрами являются строч-

Single Class

Test duration: 106,990s

Passed Tests		
tests.tos.feature.fix.nknight.Common		
TOSCore-5575 - FIX - NKNIGHT - Clearing Firm	2,257s	*Test method:* `clearingFirm ( PENSON, "234" )`
TOSCore-5575 - FIX - NKNIGHT - Clearing Firm	1,392s	*Test method:* `clearingFirm ( TDA, "188" )`
TOSCore-5575 - FIX - NKNIGHT - Clearing Firm	1,448s	*Test method:* `clearingFirm ( TDW, "615" )`
TOSCore-5573 - FIX - NKNIGHT - Heartbeat to demoNKNIGHT	51,254s	
TOSCore-5574 - FIX - NKNIGHT - Security type	1,381s	*Test method:* `securityType ( "AAPL", 100.0, {}, "CS" )`
TOSCore-5574 - FIX - NKNIGHT - Security type	1,358s	*Test method:* `securityType ( ".AAPL140920C102", 10.0, {}, "OPT" )`
TOSCore-5574 - FIX - NKNIGHT - Security type	1,368s	*Test method:* `securityType ( ".AAPL140920P102", 10.0, {spread=VERTICAL}, "MLEG" )`
tests.tos.feature.fix.nknight.Stock		
TOSCore-5576 - FIX - NKNIGHT - Stock - Constant tags	1,354s	
TOSCore-5581 - FIX - NKNIGHT - Stock - Execution Id	1,391s	
TOSCore-5579 - FIX - NKNIGHT - Stock - Buy and sell orders	1,371s	*Test method:* `side ( 0, -100.0, "5" )`
TOSCore-5579 - FIX - NKNIGHT - Stock - Buy and sell orders	1,669s	*Test method:* `side ( -100.0, 100.0, "1" )`
TOSCore-5579 - FIX - NKNIGHT - Stock - Buy and sell orders	1,382s	*Test method:* `side ( 0, 100.0, "1" )`
TOSCore-5579 - FIX - NKNIGHT - Stock - Buy and sell orders	1,448s	*Test method:* `side ( 100.0, -100.0, "2" )`
TOSCore-5580 - FIX - NKNIGHT - Stock - Symbol	1,365s	
TOSCore-5578 - FIX - NKNIGHT - Stock - Different TIF's	1,525s	*Test method:* `timeInForce ( DAY, "0" )`
TOSCore-5578 - FIX - NKNIGHT - Stock - Different TIF's	1,352s	*Test method:* `timeInForce ( GTC, "1" )`
TOSCore-5577 - FIX - NKNIGHT - Stock - Different order types	1,339s	*Test method:* `type ( MARKET, "1" )`
TOSCore-5577 - FIX - NKNIGHT - Stock - Different order types	1,529s	*Test method:* `type ( LIMIT, "2" )`

Рис. 6. Пример результата работы теста

ки тег=<ожидаемое значение>, что соотносится с кодом автотестов: с параметрами в исполняемых методах.

При реструктуризации и написании новых тест-кейсов была выбрана общая форма названий, которая имеет вид <Тестируемый вендор> - <Проверяемый тип инструмента> - <Проверяемая группа тегов>. Это позволяет быстрее ориентироваться в тестовой документации, что удобно при возникновении изменений согласно новым требованиям заказчика. Необходимо отметить, что названия соотносятся с автотестами – каждый автотест содержит его в описании (рис. 5).

Чтобы восстановить тестовую документацию по каждому из FIX-компонент, отвечающему за взаимодействие с одной финансовой организацией, было необходимо «зафиксировать состояние» приложения и методом отправки различных типов ордеров и получения на них ответов от вендора, а также взаимодействием с представителями вендоров составить список возможных тегов для каждого из сценариев.

Результаты. Основные преимущества использования разработанного инструмента:

расширяемость набора автоматических тестов с точки зрения как добавления новых тегов и усложнения логики сценариев, так и интеграции с новыми вендорами, что весьма важно при работе с изменяющимися требованиями от заказчика;

в отличие от ручного тестирования появилась возможность перебора большого количества комбинаций тегов;

реализация запуска автоматических тестов с использованием системы постоянной интеграции TeamCity [4] позволяет проводить процедуру регрессионного тестирования по заданному заранее расписанию, хранить статистику и снизить нагрузку на инженера по качеству.

Инструмент имеет следующие количественные показатели результатов работы для проекта с 15 вендорами:

новая процедура тестирования в 48 раз быстрее среднего времени аналогичного ручного тестирования, проводится в течение часа;

общий объем обновленной документации составляет 298 тест-кейсов, каждый из которых включает в себя перебор различных торговых инструментов и их производных и имеет один соответствующий автоматический тест.

Использование разработанного инструмента интеграционного тестирования FIX-компонент в повседневной работе отдела тестирования показало правильность пред-

посылок для его создания.

Время обучения сотрудников работе с инструментом сравнительно невелико, поскольку процедура запуска тестов интуитивна и не требует глубокого знания механизмов его работы.

Расширяемость инструмента и ожидаемое постоянство использования трейдинговой системой протоколов FIX позволяют судить о долгосрочности характера его применения, а точное знание области покрытия тестов дает возможность лучше оценивать риски при составлении тест-планов новых релизов системы.

СПИСОК ЛИТЕРАТУРЫ

1. Информация про FIX-протокол [электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Financial_Information_eXchange (дата обращения 02.09.2014).

2. Система Polarion [электронный ресурс] / URL: https://www.polarion.com/ (дата обращения 02.09.2014).

3. Язык программирования Groovy [электронный ресурс] / URL: http://groovy.codehaus.org/ (дата обращения 02.09.2014).

4. TeamCity [электронный ресурс] / URL: https://ru.wikipedia.org/wiki/TeamCity (дата обращения 02.09.2014).

5. Devexperts [электронный ресурс] / URL: http://www.devexperts.com/ (дата обращения 02.09.2014).

6. Darren DeMarco Exploiting Financial Information Exchange (FIX) Protocol? [электронный ресурс] / URL: http://pen-testing.sans.org/resources/papers/gcih/exploiting-financial-information-exchange-fix-protocol-126181 (дата обращения 02.09.2014).

7. Спецификация FIX-протокола от компании Devexperts [электронный ресурс] / URL: http://ftp.micex.com/pub/support/FIX/old/fixgate-protocol.pdf (дата обращения 02.09.2014).

8. Спецификация FIX-протокола от London Stock Exchange [электронный ресурс] / URL: http://www.londonstockexchange.com/products-and-services/millennium-exchange/millennium-exchange-migration/mit202issuev11-1new.pdf (дата обращения 02.09.2014).

9. QuickFix [электронный ресурс] / URL: http://www.quickfixengine.org/ (дата обращения 02.09.2014).

10. FixImulator [электронный ресурс] / URL: https://code.google.com/p/fiximulator/ (дата обращения 02.09.2014).

11. Mini-FIX [электронный ресурс] / URL: http://elato.se/minifix/ (дата обращения 02.09.2014).

REFERENCES

1. *FIX-protokol information*. Available: https://ru.wikipedia.org/wiki/Financial_Information_eXchange (Accessed 02.09.2014).

2. *Polarion system*. Available: https://www.polarion.com/ (Accessed 02.09.2014).

3. *Groovy*. Available: http://groovy.codehaus.org/ (Accessed 02.09.2014).

4. *TeamCity*. Available: https://ru.wikipedia.org/wiki/TeamCity (Accessed 02.09.2014).

5. *Devexperts company*. Available: http://www.devexperts.com/ (Accessed 02.09.2014).

6. *Darren DeMarco Exploiting Financial Information Exchange (FIX) Protocol?* Available: http://pen-testing.sans.org/resources/papers/gcih/exploiting-financial-information-exchange-fix-protocol-126181 (Accessed 02.09.2014).

7. *FIX-protokol specification from Devexperts*. Available: http://ftp.micex.com/pub/support/FIX/old/fixgate-protocol.pdf (Accessed 02.09.2014).

8. *FIX-protokol specification from London Stock Exchange*. Available: http://www.londonstockexchange.com/products-and-services/millennium-exchange/millennium-exchange-migration/mit202issuev11-1new.pdf (Accessed 02.09.2014).

9. *QuickFix*. Available: http://www.quickfixengine.org/ (Accessed 02.09.2014).

10. *FixImulator*. Available: https://code.google.com/p/fiximulator/ (Accessed 02.09.2014).

11. *Mini-FIX*. Available: http://elato.se/minifix/ (Accessed 02.09.2014).

БРЕКЕЛОВ Всеволод Владимирович — *аспирант кафедры математической теории игр и статистических решений факультета прикладной математики процессов управления Санкт-Петербургского государственного университета.*

199034, Россия, Санкт-Петербург, Университетская наб., д. 7-9.

E-mail: vsevolod.brekelov@gmail.com

BREKELOV, Vsevolod V. *St. Petersburg State University.*
199034, Universitetskaya emb. 7-9, St. Petersburg, Russia.
E-mail: vsevolod.brekelov@gmail.com

БОРИСОВ Егор Александрович – *инженер по контролю качества ООО «Эксперт Система».*
197110, Россия, Санкт-Петербург, ул. Барочная, д. 10.
E-mail: borisov@devexperts.com

BORISOV, Egor A. *Devexperts LLC.*
197110, Barochnaya Str. 10, St. Petersburg, Russia.
E-mail: borisov@devexperts.com

БАРЫГИН Илья Алексеевич – *руководитель группы ООО «Эксперт-Система», кандидат физико-математических наук.*
197110, Россия, Санкт-Петербург, ул. Барочная, д. 10.
E-mail: ibarygin@devexperts.com

BARYGIN, Ilya A. *Devexperts LLC.*
197110, Barochnaya Str. 10, St. Petersburg, Russia.
E-mail: ibarygin@devexperts.com