Автоматизация интеграционного тестирования на примере модулей обмена данными по FIX-протоколу

Брекелов В.В[1]., Барыгин И.А.[2], Борисов Е.А.[3]
[1] vbrekelov@devexperts.com, Devexperts LLC, СПбГУ
[2] ibarygin@devexperts.com, Devexperts LLC
[3] borisov@devexperts.com, Devexperts LLC

Аннотация. В трейдинговых системах в качестве транспортного протокола наиболее распространен FIX-протокол. Ручное тестирование модулей, интегрирующих финансовые системы посредством FIX-протокола, – весьма трудоемкий процесс. В данной статье рассматривается 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.
Результатами являются: написанная тестовая документация и автотесты с общей структурой, обеспечивающие не только быстрое выполнение тестов, но также позволяющие быстро адаптироваться к новым финансовым системам или к новым требованиям.

Ключевые слова. FIX-протокол, автоматизация тестирования, интеграционное тестирование, тест кейс, трейдинговая система, биржа.

1. Введение

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 (FIX) 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]
В данной работе 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 так же в неполной тестовой документации. Для решения вышеперечисленных задач был использован подход обобщения тегов относительно инструментов и вендоров. Его целью является одинаково структурировать автотесты и документацию, также их поддерживать, и оценивать покрытие тестами функциональной части компонент.
Для разработки автотестов был использован язык Groovy[3], который используется в фреймворке для написания функциональных тестов в проекте. Для хранения документации использовалась система Polarion[2], используемая внутри всех проектов компании.

1.1 Работа трейдинговой системы с FIX протоколом

Тестируемая система предполагает взаимодействие с 15 различными вендрами посредством обмена FIX-сообщениями.

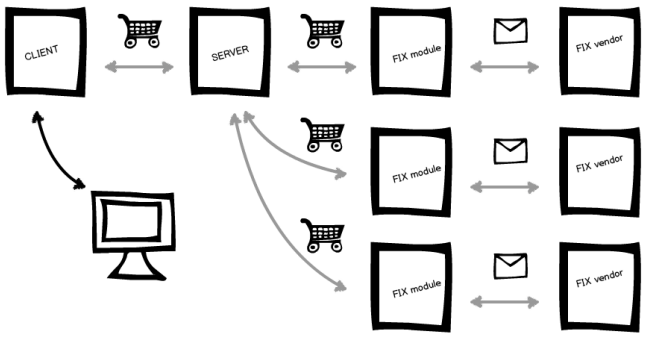


Рис.1. Взаимодействие с вендорами.

Предположим, клиент создает заявку на покупку или продажу в системе. Эта заявка обрабатывается на стороне клиента и посылается на сервер. Затем пересылается FIX-модулям, которые используют FIX-протокол для кодирования информации, и передается вендору. Последний, в свою очередь, обрабатывает полученное FIX-сообщение и отправляет ответ, в котором содержится информация о статусе ордера клиента.

1.2 FIX-сообщение

Формат передаваемого сообщения представляется в виде строчки, которая состоит из набора полей “тег=значение”.

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

Поля разделяются ASCII кодом SOH — Start of Header (0x01). Например, издавая ордер по опциону IBM, отправляется следующее сообщение:
Объяснение некоторых тегов:

- 55(Symbol) = IBM – показывает по какому инструменту оптравлен ордер;
- 167(SecurityType) = OPT – означает, что ордер отправлен по опциону;
- 44(Price) = 5.3 – указывает на цену, по которой был издан ордер.

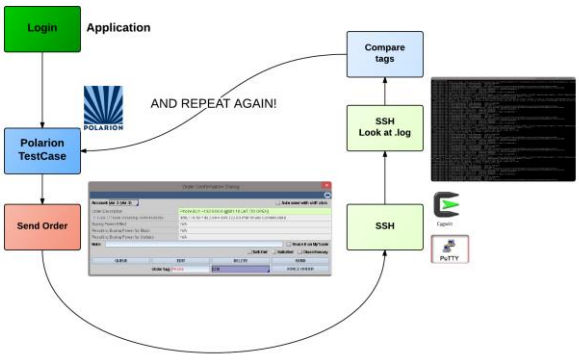
В ответ приходит такого же формата сообщения от вендора.

2. 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 функционирующем в тестовой среде.

2.1 Ручное тестирование компонент обмена финансовой информацией



.

Рис.2. Процесс ручного тестирования.

На рис.2 показан цикл ручной проверки одного тега. 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

Приведенный цикл необходимо проделывать для каждого тега тестируемого вендора.

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

2.2 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 для многих вендоров требуется большое количество времени, что задерживает процесс разработки
- Обучение инженера. Тестируя данную область, необходимо знать бизнес-логику приложения, в том числе, процесс взаимодействия с вендорами. Это несет дополнительные временные затраты.

Помимо недостатков ручного тестирования в каждом проекте существуют сложности с тестовой документацией. В данном проекте были определены следующие:

- 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.

3. Автоматизация тестирования и документация

Поскольку функционирование FIX-компонент является критически важным условием при каждом релизе разрабатываемой по итеративной методологии торговой системы, и характер тестов для различных вендоров имеет идентичную структуру, эти тесты являются идеальными кандидатами для автоматизации. Автоматические тесты позволяют формализовать и структурировать не только сам процесс проверки, но и связать с ним обновлённую, аналогично структурированную документацию, актуальность которой легко поддерживать.

3.1. Подход

Проанализировав все возможные сообщения для различных типов инструментов, а так же для различных вендоров, эмпирическим путем были выведены наборы тегов и разделены на группы. На рис.3 продемонстрировано выбранное разбиение, а так же соответствие с отправленным FIX-сообщением (рис.4).

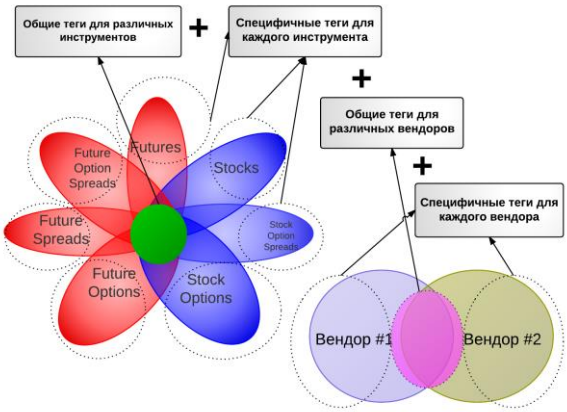


Рис.3. Выделение общих тегов.

8=FIX.4.1|9=181|35=D|52=20140904-17:04:55|34=400|56=test|49=test|
11=20140904-15804136|167=OPT|55=IBM|201=1|202=192.5|
200=201409|205=20|38=10|54=1|77=O|40=2|
44=1.61|59=0|204=0|439=777|47=A|1=17770650|10=028

Рис.4. Пример сообщения с выделенными тегами.

Таким образом получилось выделить четыре группы тегов.

- Общие теги для различных инструментов. Например: 8(FixProtocol), 35(MsgType).
- Специфичные теги для каждого инструмента. Например – 202(MaturityMonthYear).
- 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 применено для формирования структуры автотестов и тестовой документации.

3.2. Автоматизация

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

Фреймворк имеет возможность “действовать” как обычный пользователь, т.е. позволяет использовать методы и классы клиентского приложения. Перечислим основные объекты, которыми оперируют автотесты.

- Order – объект приложения. Содержит информацию об ордере: используемом инструменте, состоянии ордера, аккаунте, цене и т.д..
- FixOrder – объект фреймворка, который содержит объект Order, а также FIX-сообщения ему соответствующие.
- FixOrderService – объект фреймворка, сервис, который отправляет ордера, используя OrderService, читает FIX-сообщения с использованием DelayedTailLog, формирует FixOrder.
- Validator – объект фреймворка, содержит основные методы и DataProvider’ы для тестов.

Структура классов автотестов для каждого 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.5.

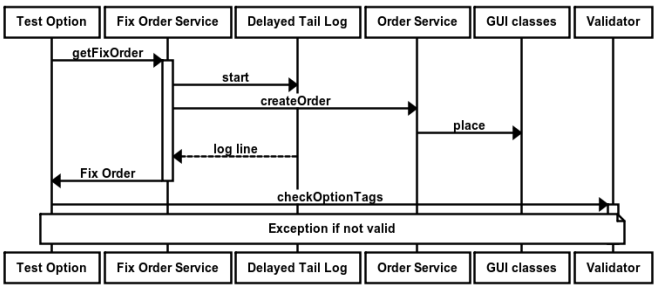


Рис.5. Пример работы автотеста.

3.3. 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 (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’ом, используемым в автотестах.(рис.6,7). Выходными параметрами являются строчки “тег=<ожидаемое значение>”, что так же соотносится с кодом автотестов, а именно 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, который отвечает 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.

4 Результаты

Основными преимуществами, которые могут быть извлечены из использования разработанного инструмента, являются:

1. Расширяемость набора автоматических тестов, как с точки зрения добавления новых тегов и усложнения логики сценариев, так и интеграции с новыми вендорами, что весьма важно при работе с изменяющимися требованиями от заказчика.
2. В отличие от ручного тестирования, появилась возможность перебора большого количества комбинаций тегов.
3. Реализация запуска автоматических тестов с использованием системы постоянной интеграции TeamCity[ссылка], что позволяет проводить процедуру регрессионного тестирования по заданному заранее времени, хранить статистику и снизить нагрузку на инженера по качеству.

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

1. Новая процедура тестирования в 48 раз быстрее среднего времени аналогичного ручного тестирования и занимает один час времени.
2. Общий объем обновленной документации составляет 298 тест кейсов, каждый из которых включает перебор различных торговых инструментов и их производных и имеет один соответствующий автоматический тест.

5 Заключение

Использование разработанного инструмента интеграционного тестирования FIX-компонент в повседневной работе отдела тестирования показало правильность предпосылок, использованных для его создания. Время обучения сотрудников использованию инструмента 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 даёт возможность лучше оценивать риски при составлении тест-планов новых релизов системы.

6 Литература

[1] Информация про FIX-протокол: [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Financial_Information_eXchange (Дата обращения: 02.09.2014)

[2] Система Polarion: [Электронный ресурс] – Режим доступа: https://www.polarion.com/ (Дата обращения: 02.09.2014)

[3] Язык программирования Groovy: [Электронный ресурс] – Режим доступа: http://groovy.codehaus.org/ (Дата обращения: 02.09.2014)

[4] Официальный сайт TeamCity: [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/TeamCity (Дата обращения: 02.09.2014)

[5] Официальный сайт компании "Devexperts": [Электронный ресурс] – Режим доступа: http://www.devexperts.com/ (Дата обращения: 02.09.2014)

[6] Darren DeMarco Exploiting Financial Information Exchange (FIX) Protocol?. 03.03.2013 – Режим доступа: 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] – Режим доступа: 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] – Режим доступа: http://www.londonstockexchange.com/products-and-services/millennium-exchange/millennium-exchange-migration/mit202issuev11-1new.pdf (Дата обращения: 02.09.2014)

Приложение

В данной работе упоминалось о соответствии написанных автотестов с тестовой документаций. На следующих рисунках изображен пример кода автотеста, пример тест-кейса и результат работы автотеста.

На рис. 7 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, а именно типов ордеров, которые необходимо создавать из клиентского приложения. "Output specification" содержит ожидаемые результаты теста, а именно проверямые тэги с соответсвующими значениями.

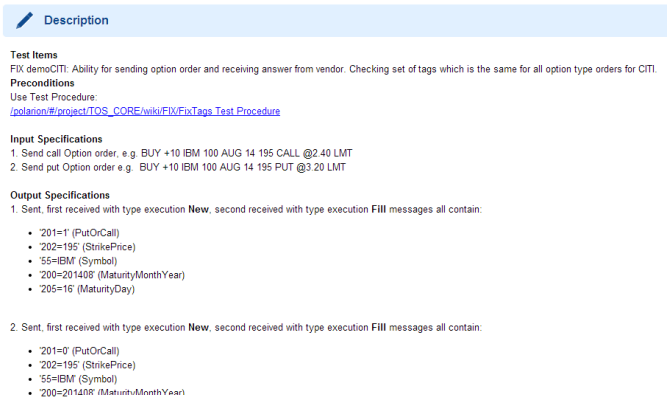


Рис.7. Пример тест-кейса.

Для представленного тестового сценария на рис. 8 содержится часть программного кода, которая исполняет действия тестовой процедуры. В части с аннотацией "@DataProvider" содержатся данные из "Input Specifications". Переменная FIX_CALL соответствует инструменту "Stock call Option", FIX_PUT – "Stock Put Option".

```
@DataProvider(name = "stockOptionCallOrPut")
public Object[][] stockOptionCallPut() {
    return [
            [FIX CALL],
            [FIX PUT],
    ]
}
```

Тест имеет аннотацию “@Test” и ссылку на тест-кейс в системе Polarion. В тесте вызывается метод checkOptionParameters.

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

Метод checkOptionTags проверяет ожидаемые значения тегов из “Output specification”. (рис. 7)

```
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),
routingInfo.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(symbol)],
        ])
}
```

Результаты работы теста представляются с помощью библиотеки TestNG. (рис. 9)

```
@DataProvider(name = "stockOptionCallOrPut")
public Object[][] stockOptionCallPut() {
    return [
            [CALL_OPTION_SYMBOL],
            [PUT_OPTION_SYMBOL],
    ]
}

@Test(description = "TOSCore-5222 - FIX - CITI - Sending option - Option parameters",
        dataProvider = "stockOptionCallOrPut")
public void optionParameters(String symbol) {
        checkOptionParameters(symbol, CITI_OPTION_QUANTITY)
}

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

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),
                    routingInfo.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(symbol)],
    ])
}
```

Рис.8. Программный код автотеста.

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}, "WLEG" )`
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.9. Результаты автотеста.