Инструмент для автоматизированного тестирования систем проведения расчетов и клиринга ClearTH

Анна Торопова
Exactpro Systems
anna.toropova@exactprosystems.com

Екатерина Димова
Exactpro Systems
ekaterina.dimova@exactprosystems.com

Иосиф Иткин
Exactpro Systems
iosif.itkin@exactpro.com

***Аннотация*—Системы, обеспечивающие проведение расчетов и клиринга, называемые также post trade, составляют существенную часть инфраструктуры финансовых рынков. Эти системы ответственны за большую часть доходов и расходов участников электронной торговли. Подобно торговым платформам, отвечающим за сведение транзакций в реальном режиме времени, большинство платформ для проведения расчетов и клиринга представляют собой распределенный диверсифицированный высоконагруженный набор взаимодействующих подсистем. Однако, системы для проведения расчетов и клиринга также характеризуются дополнительным набором свойств. Структура данных в таких системах часто приводит к невозможности проверки корректности их функционирования с помощью индивидуальных независимых сценариев тестирования. В статье рассматриваются особенности систем для проведения расчетов и клиринга и анализируется применимость существующих инструментов по автоматизации тестирования для их динамической верификации. Авторы описывают разработанный ими инструмент ClearTH, позволяющий структурировать библиотеку тестов в соответствии с расписанием операционного дня/цикла систем для проведения расчетов и клиринга. Описанный подход может быть применен и к другим типам систем, обладающим аналогичными свойствами.**

***Ключевые слова*— инструменты тестирования, автоматизация тестирования, ClearTH, клиринг, расчеты, расчетно-клиринговые системы, обеспечение качества**

I. Введение

В современной финансовой индустрии торговые и расчетно-клиринговые системы являются двумя зависящими друг от друга компонентами, которые беспрерывно взаимодействуют между собой. Торговые системы отвечают за заключение сделки, в то время как расчетно-клиринговые системы отвечают за успешный перевод денег и ценных бумаг, а также обеспечивают анонимность сделок, своевременное проведение корпоративных действий и многое другое[1,2,3,4]. Проблема обеспечения надёжности и качества программного обеспечения является одной из ключевых при разработке расчетно-клиринговых платформ, и решения, обеспечивающие качество систем, становятся все более востребованными [5]. Любая ошибка в программном обеспечении является дорогостоящей, а любая задержка с обнаружением дефекта вносит дополнительные затраты, как времени, так и финансов.

Заказчики программного обеспечения хотят быть уверенными, что новые изменения, внесенные в разрабатываемую платформу, не будут негативно сказываться на качестве, а наоборот, расширят функциональность продукта, что поспособствует увеличению в будущем количества клиентов, использующих данную платформу. При тестировании ставится цель максимально точно имитировать реальную активность системы и пользователей в тестовой среде. Использование тестовых инструментов позволяет быстро и точно обнаружить проблемный участок программного кода, обеспечив полное покрытие всей ключевой функциональности тестовыми сценариями.

Во второй части статьи описывается усредненная расчетно-клиринговая система, выделяются её особенности и отличия от торговых платформ. В третьем разделе проводится анализ распространенных тестовых подходов и описание тестового подхода, который наиболее целесообразно использовать при тестировании систем проведения расчетов и клиринга. Четвертая часть содержит статистический анализ использования тестовых инструментов на основе ресурсов, посвященных тестированию и разработке программного обеспечения, и статистике поисковых запросов, а также сравнение с наиболее известными тестовыми инструментами. Созданный инструмент и его особенности описаны в пятой части статьи.

II. Особенности систем проведения расчетов и клиринга

Системы проведения расчетов и клиринга являются важной частью современной инфраструктуры финансовых

рынков, но их архитектура, как правило, не рассматривается в научных публикациях. На Рис. 1 демонстрируется абстрактная внутренняя структура расчетно-клиринговых систем, релевантная для европейских и американских рынков[6,7,8], а также отображаются связи с внешними системами [9], с которыми, как правило, происходит взаимодействие в процессе клиринга или проведения расчетов.

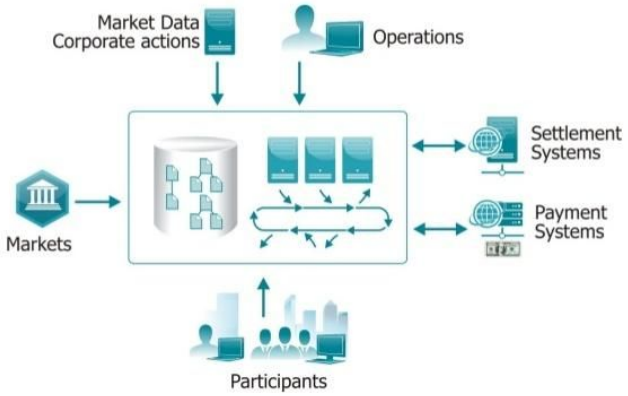


Рис. 1. Схематичное представление расчетно-клиринговой системы

Количество внешних систем, взаимодействующих с системами расчета и клиринга, может варьироваться в зависимости от особенностей конфигурации окружения[10,11], от структуры самой организации, занимающейся расчетами и клирингом, а также от законодательства страны, в которой происходят данные процессы. Однако, некая усредненная система для проведения расчетов и клиринга, схема которой представлена на Рис. 1, чаще всего взаимодействует с рынками (Markets), от которых она получает информацию о совершенных сделках (Trades, deals), с пользователями системы (Participants), с операторами (Operations), которые обеспечивают поддержку системы на том или ином уровне, с внешними расчетными (Settlement Systems) и/или платежными системами (Payment Systems) и получает рыночные данные о внешних событиях (Market Data, Corporate Actions). Связь может осуществляться через различные протоколы: MQ[12] (менеджер очередей сообщений), файловые[13] (FTP, SFTP), бинарные и многие другие. За счет этого система постоянно должна оперировать большим объемом данных[14], накопленных за разные операционные дни.

Сама же система, как правило, основывается на двух элементах: справочных данных (Reference Data/Static Data) и системном расписании (Schedule).

Справочные данные настраиваются под нужды определенных пользователей и впоследствии оказывают большое влияние на все процессы в системе. Как правило, из-за большого количества процессов в системах проведения расчета и клиринга, объем справочных данных в несколько раз превышает их количество в торговых платформах. Объем статичных данных пропорционален количеству пользователей и инструментов в системе, но они не меняются так активно, как индивидуальные транзакции, количество которых может быть не ограничено. Преимущественно в системе хранятся такие данные, как параметры рынков, информация о инструментах и пользователях, различные календари и справочные таблицы, обеспечивающие жизнедеятельность системы.

Системное расписание определяет жизненный цикл объектов в системе (как правило, он может меняться от системы к системе, но обычно длится от нескольких дней до месяцев), а также порядок процессов в течение операционного дня.

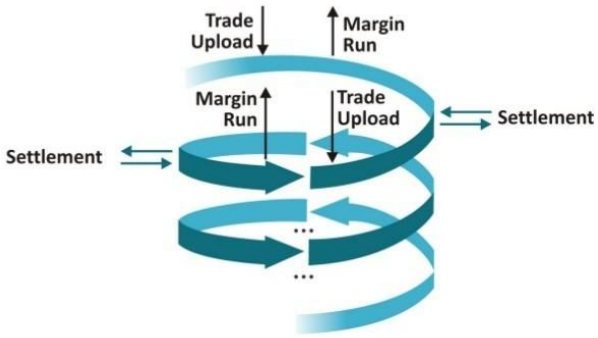


Рис. 2. Жизненный цикл системы расчета и клиринга

Как видно на Рис. 2, каждый день в расчетно-клиринговой системе происходит множество разнообразных процессов: загрузка сделок (trade upload), подсчет рисков (margin computations, margin runs), расчетные сессии (settlement sessions, settlement runs), загрузка залогового обеспечения (collateral uploads) и многие другие.

Также необходимо отметить, что очень часто изменения в законодательстве и международной политике приводят к кардинальной перестройке систем[1,2,15].

Всё это является спецификой систем для обеспечения расчетов и клиринга и находит отражение при верификации программного обеспечения на том или ином уровне.

III. Подходы к тестированию

Перед тем как определять инструменты тестирования или приступать к их проектированию, следует определить задачи, возникающие при тестировании, и подходы к тестированию, которые будут наиболее рациональны и эффективны при тестировании определенной системы.

A. Задачи, возникающие при тестировании сложных систем

При тестировании сложных систем с большим количеством данных и подключений, как правило, возникают следующие задачи:

- Сложный алгоритм подсчета проектных рисков;
- Необходимость тестировать сложные, многоступенчатые и многодневные сценарии;
- Необходимость обеспечить быстрое исполнение регрессивных тестов;

- Тестирование различных способов взаимодействия с системой: API, обмен файлами, графический интерфейс (GUI);
- Симуляция поведения внешних систем;
- Настройка справочных данных и их миграция из других систем.

B. Существующие подходы к тестированию

При тестировании, как правило, используется несколько методов, среди которых выделяются:

- Автоматическое (automation) и ручное (manual) тестирование;
- Функциональное (functional) тестирование и тестирование нагрузки/производительности (load/performance testing);
- Компонентное (unit), интеграционное (integration) тестирование и тестирование полного жизненного цикла системы (end-to-end testing).

Также существует несколько подходов к организации инструментов для тестирования:

- Data-driven testing[16] и Keyword-driven testing[17];
- Параллельный и последовательный запуск тестов.

C. Наиболее удобные подходы для тестирования расчетно-клиринговых систем

Как было сказано выше, системы проведения расчетов и клиринга представляют собой сложные структуры, которые требуют всеобъемлющего тестирования в достаточно сжатые сроки, поэтому самым эффективным решением данной задачи является автоматизация всевозможных сценариев.

При разработке тестов функциональное тестирование всегда имеет большой приоритет и практически никогда не исключается из набора тестовых сценариев. Тестирование нагрузки и/или производительности является важной частью тестирования, особенно в случях, когда предполагается большая нагрузка на компоненты и когда система разрабатывается для большого количества пользователей.

Такие подходы как unit-, integration- и end-to-end тестирование являются основой тестирования любой функциональности и считаются неотъемлемой частью обеспечения качества программ.

Также, при выборе тестовых подходов, следует определиться с особенностями работы инструмента. На данный момент выделяются два популярных метода организации: data-driven testing и keyword-driven testing.

При data-driven testing только тестовые данные отделяются от исполняемого скрипта, тогда как при keyword-driven testing отделяются и тестовые данные, и сам скрипт[16,17]. Если для небольших программ с однотипными действиями достаточно просто отделить тестовые данные, то для расчетно-клиринговых систем, содержащих различные действия (как правило более пятидесяти [18,19]), удобнее использовать keyword-driven testing.

Системы проведения расчетов и клиринга обычно имеют определенный порядок событий и периодов в течение операционного дня[20]. При тестировании данных систем необходимо учитывать все их особенности, а также максимально точно имитировать бизнес-сценарии в тестовой среде. На рис. 3 изображено несколько сценариев, которые проверяются при тестировании систем проведения расчетов и клиринга. Они охватывают сценарии, релевантные для бизнеса, такие как:

- Дефицит средств на залоговом счете (Collateral Deficit);
- Проведение расчетов при условии, что участник торгов не выполнил своих обязательств (Delivery Default);
- Проведение корпоративных действий (Corporate Action).

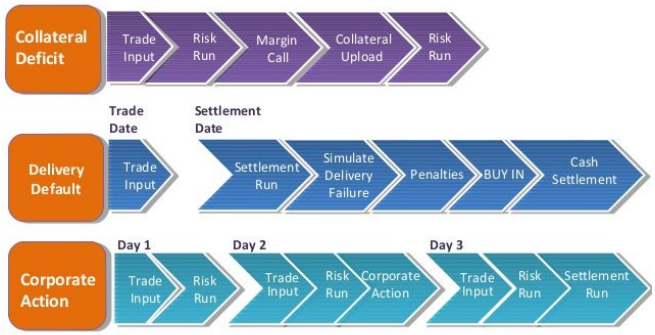


Рис. 3. Бизнес-сценарии, характерные для систем расчета и клиринга

Существующие подходы к тестированию не всегда могут быть эффективно использованы для обеспечения качества систем. Поэтому был разработан метод “concurrent test execution and scheduling”, представляющий собой гибрид популярных методов тестирования с поправкой на особенности систем проведения расчетов и клиринга.

Метод заключается в организации объекта scheduler, который представляет из себя набор тестовых keyword-driven скриптов (matrix) и расписания (sheduler config).

На Рис 4. отображена структура тестового скрипта (Matrix) для “concurrent test execution and scheduling”. Наименьшей единицей скрипта является действие (Action), которое может являться отдельным тестом или его частью. Одно действие выполнется целиком и в один момент времени. Группа действий объединяется в шаги (Global Step). Шаг, в свою очередь, является привязкой к расписанию (sheduler config), на уровне которого регулируется порядок и время их исполнения. Тестом в данном случае будет набор действий из разных шагов. Шаги объединяются в тестовый скрипт, который может содержать один или несколько тестов.

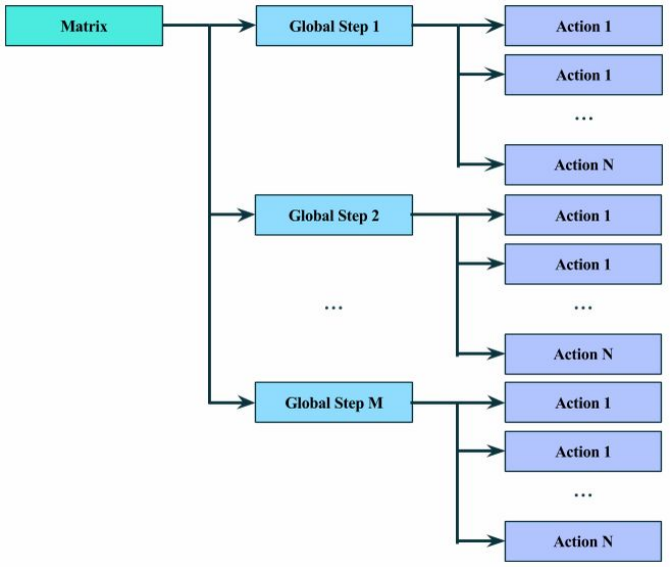


Рис. 4. Схематичное представление структуры тестового скрипта

Итак, в широком смысле подход, названный авторами “concurrent test execution and scheduling”, можно разбить на следующие субподходы:

1) Запуск тестов по расписанию

в данном пункте имеется в виду, что инструмент самостоятельно, по заданным параметрам или времени инициирует запуск необходимых тестовых скриптов без непосредственного вмешательства тестировщика;

2) Одновременное выполнение тестов в одном тестовом окружении

под одновременным исполнением здесь понимается не фактическое параллельное исполнение, которое возможно только в распределённых вычислительных системах, а скорее многозадачность, то есть в один промежуток времени исполняется часть теста (одно действие), симулируя этим параллельное выполнение тестов;

3) Привязка к системному расписанию

тут подразумевается привязка выполнения того или иного действия к определенному периоду или событию в системе.

IV. Существующие инструменты для обеспечения качества программного обеспечения

Обеспечение качества программного обеспечения является неотъемлемой частью жизненного цикла разработки, и с усложнением тестируемых программ появилась необходимость автоматизировать и контролировать процесс тестирования. Многие крупные компании, занимающиеся разработкой программного обеспечения имеют собственные наработки в данной области. Ярким примером является компания Hewlett-Packard Company (HP) имеющая большой набор решений для тестирования программного обеспечения[21]. С выделением процесса обеспечения качества как отдельной части разработки программного обеспечения вырос и спрос на различные инструменты для тестирования[22,23]

На текущее время создано большое количество тестовых инструментов: ресурсы, посвященные тестированию и/или разработке программного обеспечения, содержат описания до нескольких сотен различных инструментов[24,25], применимых в той или иной области.

Естественно, что при таком количестве различных решений встает вопрос о рациональном выборе инструментов для тестирования, но в случае тестирования систем, имеющих ряд специфических особенностей, выбор следует делать учитывая все особенности архитектуры системы, а также входящих в нее подсистем.

Ранее в статье мы выявили особенности систем проведения расчетов и клиринга, выбрали подходы, использование которых наиболее рационально для обеспечения качества систем, а также описали подход к тестированию, учитывающий специфику тестируемой области.

Список инструментов тестирования инструментов, реализующих распространенные тестовые подходы, достаточно обширен. Определяя решения, рациональные для описанного подхода, было решено провести анализ наиболее часто используемых тестовых инструментов.

Исследование инструментов для функционального тестирования, проведенное на различных ресурсах[26,27,28], а также статистика поисковых запросов (см. Рис 5 и Рис. 6) по заданной теме позволила выделить десять инструментов (пять коммерческих и пять с отрытым исходным кодом). Ниже приведен перечень инструментов, по которым была собрана статистика:

1) Коммерческие инструменты:
- *a) HP Quick Test Professional (HP QTP);*
- *b) SilkTest;*
- *c) Telerik TestStudio;*
- *d) TestComplete;*
- *e) Ranorex.*

2) Инструменты с открытым исходным кодом:
- *a) Selenium;*
- *b) AutoIt;*
- *c) SoapUI;*
- *d) Sahi;*
- *e) Watir ;*

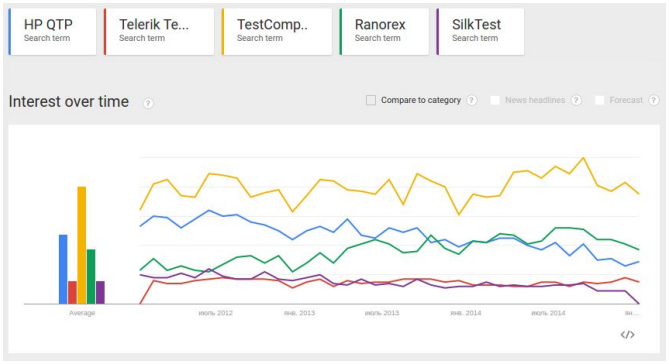


Рис. 5. Статистика поисковых запросов по коммерческим инструментам тестирования, сформированная с помощью сервиса Google Trends

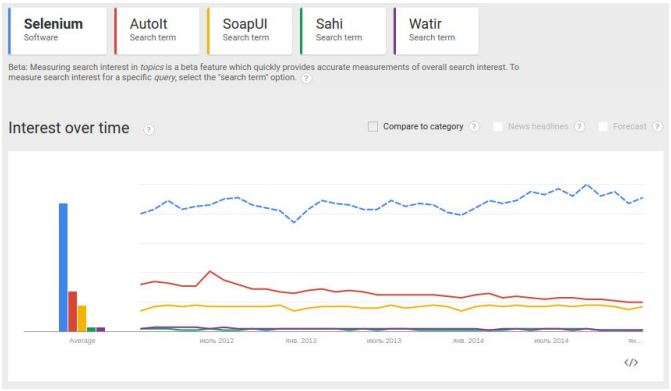


Рис. 6. Статистика поисковых запросов для инструментов тестирования с открытым исходным кодом, сформированная с помощью сервиса Google Trends

На основе имеющихся данных было решено выбрать пять из них, а именно:

- Selenium[29];
- HP Quick Test Professional (HP QTP)[30];
- Telerik TestStudio[31];
- TestComplete[32];
- Ranorex[33].

Авторами было решено провести сравнительный анализ по суб-подходам “concurrent test execution and scheduling” в связи с тем, что для определения подхода, аналогичного “concurrent test execution and scheduling”, в других инструментах могла быть выбрана другая терминология

A. Запуск тестов по расписанию и с привязкой к системным событиям

Практика использования планировщиков для организации расписания выполнения тех или иных приложений не нова и находит отражение во многих программных продуктах. Все анализируемые инструменты имеют возможность работы по расписанию. Например, Telerik TestStudio, TestComplete и HP QTP позволяют выполнять тесты в запланированное время без дополнительных плагинов или надстроек. А расписание в Selenium и Ranorex может быть реализовано при помощи серверов непрерывной интеграции (CI-серверов), например, Jenkins[34]. Привязка к системным событиям может быть осуществлена с помощью внешних плагинов и/или утилит. Например, с помощью Skybot Scheduler существует возможность организовать запуск тестов без жесткой привязки ко времени, а по какому-либо событию[35].

Несмотря на большое количество способов реализации привязки выполнения тестов ко времени или системным событиям, все способы имеют ряд недостатков, таких как сложная конфигурируемость дополнительных утилит и относительно небольшой функционал встроенных инструментов планирования.

B. Одновременное выполнение тестов

Все рассмотренные инструменты позволяют выполнять тестовые сценарии параллельно в нескольких тестовых окружениях, что позволяет распределять нагрузку при тестировании, а также сокращать время прогона независимых тестов.

Например, для выбранных коммерческих инструментов данная функция включена в поставку по умолчанию. В случае Selenium необходима интеграция с внешними модулями. Так, данное решение в сочетании с библиотекой TestNG позволяет выполнять несколько тестов параллельно, но с некоторыми ограничениями:

- Требуется сложная сетевая архитектура типа клиент-сервер;
- Тесты фактически будут выполняться в разных окружениях.

Однако с одновременным выполнением тестовых скриптов все немного сложнее, так как для всех инструментов, тест является наименьшей неделимой единицей автоматизации. Конечно, можно организовать тестовый скрипт как псевдопараллельный, но это существенно усложняет работу тестировщиков, а в случае с тестированием графического интерфейса становится практически невозможным.

Также следует проанализировать эти инструменты относительно общих требований:

- Возможность взаимодействия с системой через графический интерфейс;
- Возможность взаимодействия с системой через API;
- Тестирование Back-End;
- Гибкость и универсальность тестовых скриптов;
- Простота использования.

В Test Complete тесты могут создаваться в обоих форматах при помощи встроенных редакторов. Этот инструмент поддерживает такие языки, как: VBScript, JScript, DelphiScript и другие. Quick Test Professional позволяет записывать действия пользователя в тестируемом приложении. Эти действия хранятся в виде скриптов, которые могут быть выполнены в дальнейшем и могут быть представлены в двух форматах: как в виде кода на VBScript (expert view), так и в качестве набора последовательных шагов с действиями (keyword view).

Selenium IDE также предоставляет возможность записывать действия пользователя, в данном случае, в виде пар Command - Target, которые могут быть экспортированы в нескольких форматах: HTML, Java, Groovy, C#, Python и другие. В Ranorex и Test Studio используется Keyword-driven подход.

Несмотря на то, что все рассмотренные инструменты являются мощными и комплексными решениями, они разработаны с уклоном на тестирование широко используемых программ и приложений. Данная направленность, безусловно, важна на рынке программного обеспечения, но это значительно усложняет их конфигурирование для более специфичных систем проведения расчетов и клиринга, тем самым уменьшая эффективность процесса тестирования и повышая затраты на обеспечения качества программного обеспечения

(время, обучение персонала, настройка окружений и инструментов).

V. Инструмент для автоматизированного тестирования систем проведения расчетов и клиринга ClearTH

ClearTH - это специальный инструмент, предназначенный для автоматизации тестирования, написанный на языке Java, первичной целью которого является тестирование различного рода систем проведения расчетов и клиринга. В Таблице 1 описаны основные характеристики ClearTH.

Таблица I. Основные характеристики ClearTH

	Описание
Тип тестирования	Active Batch
Целевые системы	Расчетно-клиринговые платформы и Middle Office
Интерфейс тестируемой системы	Back-end (обычно подключается к шлюзу сообщений/APIs, и DBs); Тестирование графического интерфейса возможно через плагины или подключение в другим тестовым инструментам
Метод взаимодействия с системой	Прогон нескольких процессов по расписанию (зачастую захватывающий несколько бизнес-дней). Внутренний планировщик синхронизирован с планировщиков тестируемой системы. Запросы в базу данных для проверки данных
Протоколы	Содержит плагины для популярных протоколов (FIX и диалекты, FAST, SWIFT, ITCH, HTTP, SOAP, FTP и так далее) и частных протоколов. Новые плагины для других протоколов могут быть добавлены по требованию
Тестовый скрипт	Удобные для понимания CSV файлы. Скрипты создаются вручную аналитиками или создаются автоматически на основе системных данных.
Контроль тестирования и построение отчетов.	Интегрированный (Web фронт-энд), разрешающий несколько одновременных разнородных соединений, одновременное выполнение множества запланированных скриптов, связанных с основными глобальными шагами, краткое резюмирование результатов тестирования и детальные отчеты о прогоне. Базируется на принципе прогона всех скриптов одним нажатием - принцип “Большой кнопки”
Требования к платформе	Low footprint cross-platform application, MySQL или другие RDBMS

В ClearTH реализован подход “concurrent test execution and scheduling”. Основным объектом инструмента является Sheduler, который связывает между собой тестовые скрипты (matrices) и расписанием (scheduler config).

Принцип по которому происходит выполнение тестов приведен на Рис 7.

Как видно из схемы, после инициации запуска выполнения скрипта ClearTH читает имя первого Global Step, далее приступает к поиску во всех тестовых скриптах невыполненного Action, привязанного к данному Global Step, и если Action найден, то он выполняется, если нет - ClearTH по такому же принципу обрабатывает следующий Global Step. Как только все Global Steps выполнены, ClearTH формирует отчет и завершает работу.

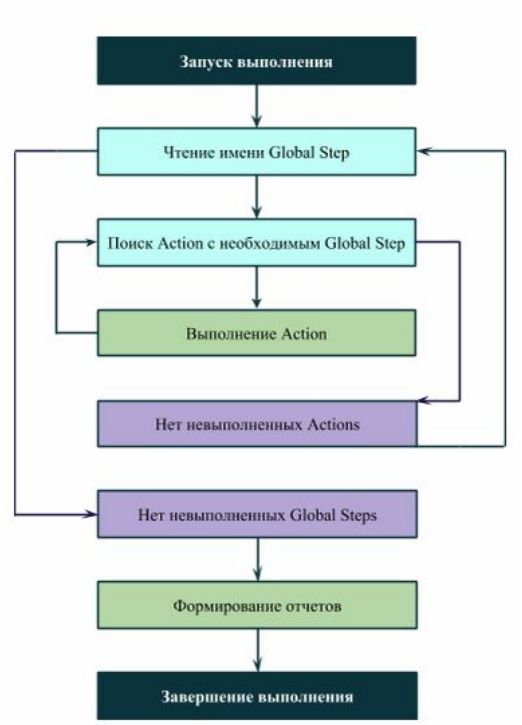


Рис. 7. Схематичное изображение работы ClearTH при выполнении теста

ClearTH имеет простой графический интерфейс, конфигурируемый для каждой системы, но базовая комплектация всегда включает в себя:

- Automation - вкладка на которой настраиваются тестовые скрипты и расписание;
- Tools - вкладка с дополнительными тестовыми инструментами;
- Connectivity - вкладка, содержащая информацию о подключениях.

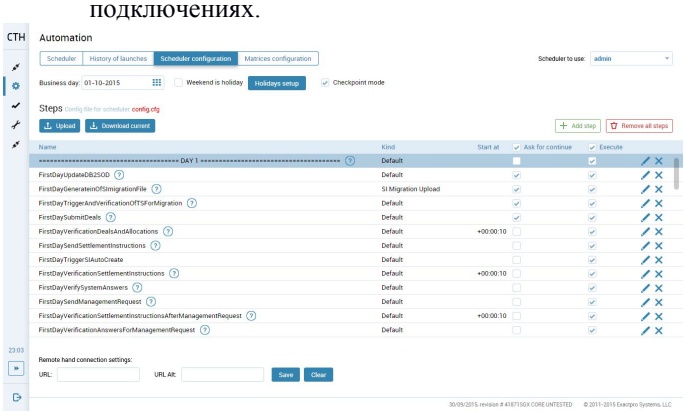


Рис. 8. Графическое представление Sheduler Config в ClearTH

Раздел Automation является базовым: планирование, запуск и выполнение тестовых сценариев происходит именно здесь. Он содержит в себе несколько Sheduler, которые в свою очередь состоят из Sheduler, History of launches, Scheduler configuration и Matrices Configuration.

Sheduler Configuration отображает Sheduler config, загруженный из файла или созданный вручную, и возможность сохранить конфигурацию в файл. Также есть возможность редактировать все поля Sheduler config через графический интерфейс, а именно:

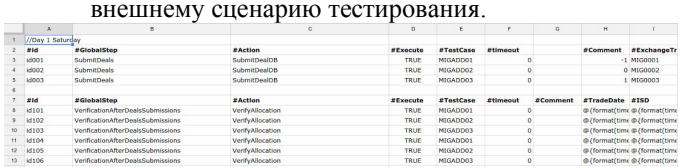
- Имя глобального шага;
- Тип глобального шага;
- Время запуска (относительное или абсолютное);
- Запрос для продолжения (планировщик приостановит исполнение матриц после нужного шага и будет запрашивать разрешение на продолжение или остановку прогона);
- Исполнение (указывает, следует ли выполнять этот глобальный шаг или нет).

Вкладка Matrices configuration содержит список тестовых скриптов (matrices), загруженных пользователем в данный Sheduler. На этой вкладке можно загрузить, удалить или проверить matrix на предмет синтаксических ошибок, а также задать исполнение матрицы при прогоне. Matrix - это текстовый файл в формате CSV(comma separeted values). Файл в таком формате легко читается, его легко редактировать, а программ, поддерживающих данный формат, огромное количество.

Рис. 9. Вкладка Matrices Configuration

Для каждого Action или группы задается свой заголовок, в котором указывается имя параметра, но есть и группа параметров, которая является служебной:

- #id - идетификатор Action;
- #GlobalStep - имя глобального шага;
- #Action - имя действия;
- #Enable - указывает на то, будет ли выполнятся данное действие или нет;
- #Timeout - задержка перед выполнением действия;
- #TestCase - идентификатор теста привязанный к внешнему сценарию тестирования.



	A	B	C	D	E	F	G	H	I
1	//Day 1 Saturday								
2	#id	#GlobalStep	#Action	#Execute	#TestCase	#timeout		#Comment	#ExchangeTr
3	id001	SubmitDeals	SubmitDealDB	TRUE	MIGADD01	0		-1	MIG0001
4	id002	SubmitDeals	SubmitDealDB	TRUE	MIGADD02	0		0	MIG0002
5	id003	SubmitDeals	SubmitDealDB	TRUE	MIGADD03	0		1	MIG0003
6									
7	#id	#GlobalStep	#Action	#Execute	#TestCase	#timeout	#Comment	#TradeDate	#ISD
8	id101	VerificationAfterDealsSubmissions	VerifyAllocation	TRUE	MIGADD01	0		@{format(tim	@{format(tim
9	id102	VerificationAfterDealsSubmissions	VerifyAllocation	TRUE	MIGADD02	0		@{format(tim	@{format(tim
10	id103	VerificationAfterDealsSubmissions	VerifyAllocation	TRUE	MIGADD03	0		@{format(tim	@{format(tim
11	id104	VerificationAfterDealsSubmissions	VerifyAllocation	TRUE	MIGADD01	0		@{format(tim	@{format(tim
12	id105	VerificationAfterDealsSubmissions	VerifyAllocation	TRUE	MIGADD02	0		@{format(tim	@{format(tim
13	id106	VerificationAfterDealsSubmissions	VerifyAllocation	TRUE	MIGADD03	0		@{format(tim	@{format(tim

Рис. 10. Тестовый скрипт (matrix) открытый через табличный редактор

В разделе Sheduler производится запуск тестов, а также, во время прогона запланированных тестовых сценариев, ClearTH показывает все данные о прогоне:

- Текущий исполняемый глобальный шаг;
- Количество удачно завершенных и всех выполенных действий;
- Время начала и завершения каждого шага.

Затем несколько матриц, выполнение которых будет происходить синхронно, загружаются в инструмент. В конфигурационном файле прописывается порядок, в котором будут выполняться все глобальные шаги и действия, описанные в них (например, 1,3,5,2 и т.д.). Все матрицы будут проверяться одновременно, в зависимости от наличия в матрице того или иного глобального шага.

Проверка может быть остановлена в любой момент, после чего сразу же будут составлены отчеты о прогоне. Чтобы сделать перерыв в проверке, необходимо нажать на кнопку Паузы или же задать время “запроса для продолжения” после определенных глобальных шагов в файле конфигурации. В последнем случае результатом будет автоматическая остановка прогона после соответствующего глобального шага.

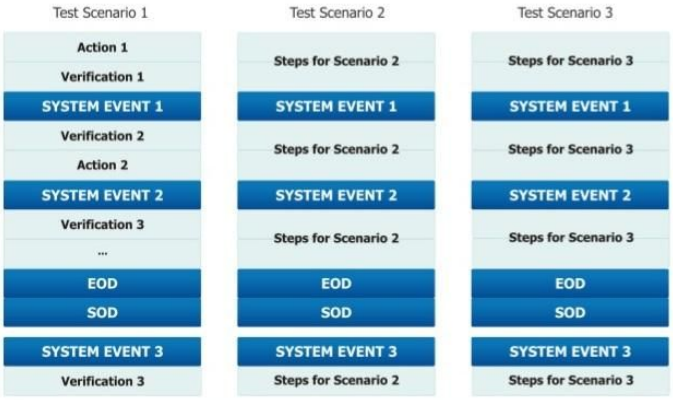


Рис. 11. Схематичное представление выполнение тестовых сценариев, привязанное к системным событиям

History of launches содержит ссылки на все отчеты, созданные в данном Sheduler, а также общую статистику по количеству удачных и неудачных тестов для каждого прогона.

Отчет содержит информацию о результатах выполнения всех Actions. Для действий он содержит информацию о входящих параметрах, а для проверок еще и таблицу сравнения ожидаемого и фактического результатов.

Если по каким либо причинам невозможно выполнить тот или иной Action типа “действие” (посылка файла, сообщения, изменения базы данных и так далее), то Action присваивается статус FAILED (неудача) и указывается причина, в обратном случае - PASSED (успех). Для присвоения статуса FAILED Action типа “проверка” есть несколько причин, например: объект не найден, фактический результат не совпадает с ожидаемым, технические проблемы.

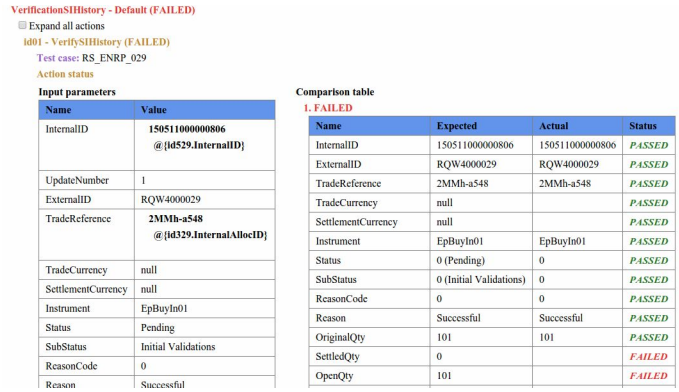


Рис. 12. Пример сформированного отчета

Вкладка Tools в ClearTH содержит набор инструментов, который реализует дополнительный функционал, упрощающий работу тестировщика:

- Matrix from Report - генерирует на основе отчета матрицу, после выполнения которой он был получен;
- Config maker - позволяет создать sheduler config для конкретной матрицы;
- Message parser - обрабатывает сообщение, предоставляя его пользователю;
- Message to script - позволяет преобразовать сообщение в матрицу;

При необходимости могут быть добавлены специальные функции для каждого отдельного проекта или системы.

Кроме возможностей, доступных из графического интерфейса, есть набор “пассивных” функций:

- Имитация компонентов, взаимодействующих с системой;
- Подключение к базе данных;
- Анализ синтаксиса матрицы; в случае, если будут найдены ошибки, пользователь получит уведомление с возможностью просмотреть их список;
- Работа с динамическими данными с помощью mvel-выражений, позволяющих использовать java-синтаксис для автоматического заполнения полей в матрице;
- Возможность установки подключений к нескольким базам данных;
- Сохранение состояния выполнения тестовых сценариев с возможностью продолжить тестирование в дальнейшем.

Вкладка Connectivity содержит информацию о подключениях, использующихся для симуляции пользователей или внешних систем.

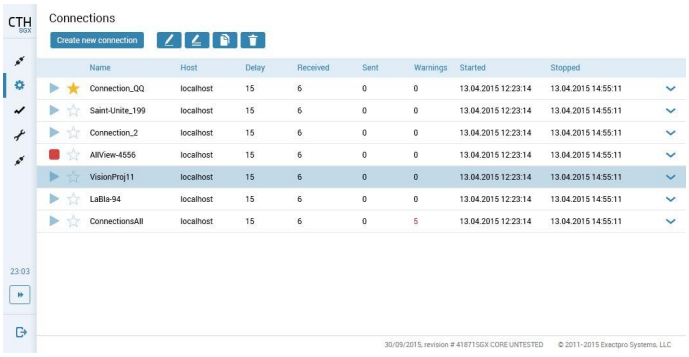


Рис. 13. Вкладка Connectivity

Кроме графического интерфейса, взаимодействие с инструментом ClearTH может быть осуществлено через API, что позволяет написать скрипт, который, например, запускает выполнение тестов, ждет его завершения, собирает отчеты в необходимую папку. Все вызовы API - это HTTP-запросы, которые отправляются по адресу, заданному при конфигурировании инструмента. В базовой комплектации поддерживаются запросы:

1) Отображение статуса планировщика (scheduler);

2) Выполнение того или иного действия для заданного планировщика:

a) Старт;

b) Остановка;

c) Удаление заданного тестового скрипта (matix);

d) Удаление всех тестовых скриптов;

e) Удаление всех шагов из конфигурации планировщика (scheduler cofig);

f) Отображение статуса прогона;

g) Получение отчета о прогоне;

h) Установка нужной даты;

3) Отображение списка тестовых сценариев;

4) Отображение всех шагов (Global Steps) в планировщике;

5) Загрузка тестового скрипта;

6) Загрузка шагов.

Все запросы возвращают результат в виде XML + HTTP-код статуса. Для отправки HTTP-запросов из скриптов можно использовать встроеные Linux-программы, например программу cURL[36].

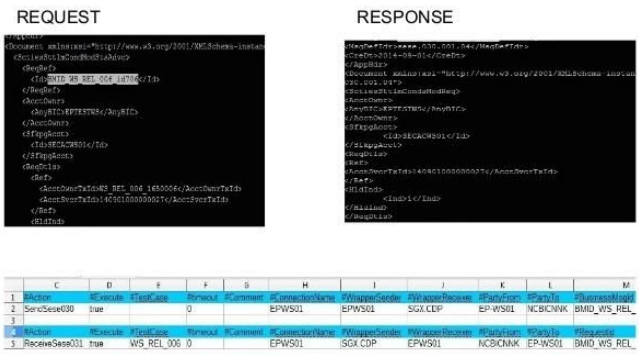


Рис. 14. Пример API сообщений и матрицы к ним

Что касается взаимодействия с системой через API, то в ClearTH есть возможность преобразовать информацию, посылаемую в систему, в сообщение, которое примет система, и наоборот, возможность преобразовывать сообщения системы в понятную для пользователя информацию и сравнивать ее с ожидаемым результатом.

В связи с тем, что доступ к реальным внешним системам ограничен, в ClearTH реализована функциональность, которая позволяет смоделировать поведение всех необходимых внешних систем, взаимодействующих с расчетно-клиринговой системой, сымитировать любой вид активности: от действий пользователя до ответов тестируемой системы (см. Рис. 15). Этот подход позволяет подойти к тестированию с особой тщательностью, позволяет контролировать выполнение тестовых сценариев, максимально точно приблизив к реальности весь процесс.

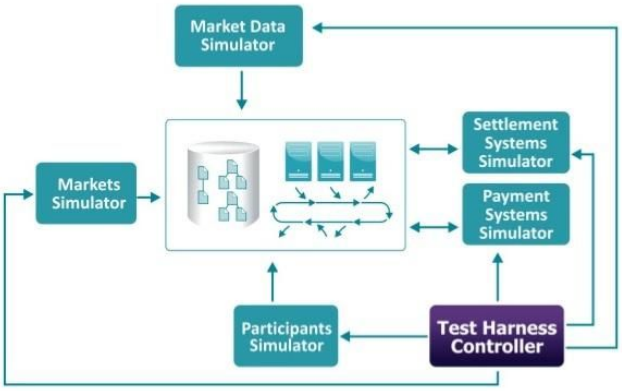


Рис. 15. Симулирование ClearTH реальных внешних систем

Авторы полагают, что с учетом всех особенностей систем проведения расчетов и клиринга ClearTH является крайне удобным и эффективным инструментом для организации тестирования подобных систем за счет того, что при описании подхода к тестированию и проектировании инструмента были учтены основные особенности расчетно-клиринговых систем, а также простоты представления скриптов и использования.

Заключение

С развитием цифровых технологий и, следовательно, увеличением количества транзакций, проведенных через электронные торговые и расчетно-клиринговые системы, стабильность финансовой инфраструктуры будет все больше проистекать от корректной работы систем, обеспечивающих проведение расчета и клиринговых операций. Представленный в статье инструмент для автоматизации тестирования и обеспечения качества систем проведения расчетов и клиринга используется в целях комплексной верификации программных продуктов на различных уровнях, тем самым снижая риск появления дефектов и возникающих с ними затрат.

В статье описаны распространенные методы обеспечения качества программного обеспечения, а также путем анализа сделаны выводы о наиболее рациональном и удобном подходе к тестированию расчетно-клиринговых систем. Рассмотрены также основные особенности и применения инструмента для автоматизированного тестирования систем проведения расчетов и клиринга ClearTH.

Дальнейшие исследования будут направлены на оптимизацию описанного подхода к тестированию и самого инструмента: в частности, на расширение разработанных методов на некоторые торговые, расчетно-клиринговые и другие системы, использующие подобную архитектуру и методы взаимодействий с пользователями, подсистемами и внешними программами.

Источники

[1] Nasdaq Dubai Business Rules [электронный ресурс] // Nasdaq Dubai Ltd. URL: http://www.nasdaqdubai.com/assets/docs/regulation/Nasdaq%20Dubai%20Business%20Rules%20-%20Rulebook%201%20V10.1%20for%20Equities%20(29%20March%202015).pdf (дата доступа: 02.09.2015)

[2] Post-trading/Settlement (SFD, CSDR, T2S) [электронный ресурс] // esma.com. URL: http://www.esma.europa.eu/page/Post-tradingSettlement-SFD-CSDR-T2S2 (дата доступа: 02.09.2015)

[3] Rulebooks [электронный ресурс] // LCH.Clearnet. URL: http://www.lchclearnet.com/rules-regulations/rulebooks (дата доступа: 02.09.2015)

[4] CDP Clearing Rules [электронный ресурс] // SGX.com. URL: http://rulebook.sgx.com/en/display/display_main.html?rbid=3271&element_id=359 (дата доступа: 02.09.2015)

[5] Reengineering Enterprise Wide Legacy BFSI Systems: Industrial case study, Prabhakar Cherukupalli, Y. Raghu Reddy, ISEC '15: Proceedings of the 8th India Software Engineering Conference

[6] Payment, clearing and settlement systems in the United Kingdom [электронный ресурс] // bis.org. URL: http://www.bis.org/cpmi/publ/d105_uk.pdf (дата доступа: 13.09.2015)

[7] Payment, clearing and settlement systems in the United States [электронный ресурс] // bis.org. URL: http://www.bis.org/cpmi/publ/d105_us.pdf (дата доступа: 13.09.2015)

[8] PAYMENT AND SECURITIES SETTLEMENT SYSTEMS IN THE EUROPEAN UNION VOLUME 1 EURO AREA COUNTRIES [электронный ресурс] // EUROPEAN CENTRAL BANK. URL: http://www.ecb.europa.eu/pub/pdf/other/ecbbluebookea200708en.pdf?73f510349d74efaf9d6599048bb32796 (дата доступа: 13.09.2015)

[9] Redesigning business networks: reference process, network and service map, Stefan Reitbauer, Falk Kohlmann, Clemens Eckert, Ken Mansfeldt, Rainer Alt, SAC '08: Proceedings of the 2008 ACM symposium on Applied computing

[10] Singapore Exchange (SGX) [электронный ресурс] // The Global provider of security financial messaging services. URL: http://www.swift.com/products_services/industry_initiatives/SGX/overview?lang=en# (дата доступа: 12.09.2015)

[11] About OeKB [электронный ресурс] // OeKB. URL: http://www.oekb.at/en/about-oekb/pages/default.aspx (дата доступа: 02.09.2015)

[12] SWIFT post-trade services for investment managers [электронный ресурс] // The Global provider of security financial messaging services. URL: http://www.swift.com/assets/swift_com/documents/products_services/SWIFT_post_trade_services_for_investment_managers_56249.pdf (дата доступа: 02.09.2015)

[13] COMMISSION DELEGATED REGULATION (EU) No 148/2013 [электронный ресурс] // Official Journal of the European Union. URL: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2013:052:0001:0010:EN:PDF (дата доступа: 25.09.2015)

[14] Comparative tables [электронный ресурс] // bis.org. URL: http://www.bis.org/cpmi/publ/d135a.pdf (дата доступа: 21.09.2015)

[15] О рынке ценных бумаг (с изменениями на 13 июля 2015 года) (редакция, действующая с 1 октября 2015 года) [электронный ресурс] // Электонный фонд правовой и нормативно-технической документации. URL: http://docs.cntd.ru/document/9018809 (дата доступа: 13.09.2015)

[16] Data-driven test cases [электронный ресурс] // boost.org. URL: http://www.boost.org/doc/libs/1_59_0/libs/test/doc/html/boost_test/tests_organization/test_cases/test_case_generation.html (дата доступа: 09.09.2015)

[17] Test Language - Introduction to Keyword Driven Testing [электронный ресурс] // Software Development Magazine - Programming, Software Testing, Project Management, Jobs. URL: http://www.methodsandtools.com/archive/archive.php?id=108 (дата доступа: 17.09.2015)

[18] CESR Technical Advice to the European Commission in the Context of the MiFID Review – Equity Markets Post-trade Transparency Standards [электронный ресурс] // COMMITTEE OF EUROPEAN SECURITIES REGULATORS. URL: http://www.esma.europa.eu/system/files/10_882.pdf (дата доступа: 22.09.2015)

[19] ISO 20022 Securities messages [электронный ресурс] // ISO 20022 Universal financial industry message scheme. URL: https://www.hkex.com.hk/eng/rulesreg/clearrules/ocp/documents/chap15.pdf (дата доступа: 15.09.2015)

[20] TYPHOONS AND RAINSTORMS [электронный ресурс] // OPERATIONAL CLEARING PROCEDURES. URL: http://www.bis.org/cpmi/publ/d135a.pdf (дата доступа: 22.09.2015)

[21] FUNCTIONAL TESTING [электронный ресурс] // hp.com. URL: http://www8.hp.com/us/en/software-solutions/functional-testing-software-testing/index.html?jumpid=hpr_r1002_usen_link1 (дата доступа: 21.09.2015)

[22] Gelperin, D.; B. Hetzel (1988). "The Growth of Software Testing". CACM 31 (6). ISSN 0001-0782

[23] A (Very) Brief History of Test Automation. Geoff Horne [электронный ресурс] // linkedin.com. URL: https://www.linkedin.com/pulse/20141007123253-16089094-a-very-brief-history-of-test-automation (дата доступа: 13.09.2015)

[24] List of Testing Tools [электронный ресурс] // guru99.com. URL: http://www.guru99.com/list-of-testing-tools.html (дата доступа: 18.09.2015)

[25] Test Tools [электронный ресурс] // ministryoftesting.com. URL: http://www.ministryoftesting.com/resources/software-testing-tools/ (дата доступа: 21.09.2015)

[26] 10 REGRESSION/FUNCTIONAL WEB TESTING TOOLS [электронный ресурс] // TECH BLOG. URL: http://www.hurricanesoftwares.com/10-regressionfunctional-web-testing-tools/ (дата доступа: 02.09.2015)

[27] 5 Best Test Automation Tools [электронный ресурс] // automated-360.com. URL: http://automated-360.com/automation-tools/5-best-test-automation-tools (дата доступа: 21.09.2015)

[28] Category:Software testing tools [электронный ресурс] // wikipedia.org. URL:https://en.wikipedia.org/wiki/Category:Software_testing_tools (дата доступа: 11.09.2015)

[29] What is Selenium? [электронный ресурс] // seleniumhq.org. URL: http://docs.seleniumhq.org/ (дата доступа: 21.09.2015)

[30] HP Unified Functional Testing software [электронный ресурс] // Web-site of Hewlett-Packard Company. URL:http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA4-8360ENW.pdf (дата доступа: 21.09.2015)

[31] Test Studio [электронный ресурс] // telerik.com. URL: http://www.telerik.com/teststudio (дата доступа: 14.09.2015)

[32] TestComplete Platform: Testing for Desktop, Mobile, & Web Applications [электронный ресурс] // smartbear.com. URL: http://smartbear.com/product/testcomplete/overview/ (дата доступа: 21.09.2015)

[33] Automated Testing of Desktop. Web. Mobile. [электронный ресурс] // ranorex.com. URL: http://www.ranorex.com/ (дата доступа: 25.09.2015)

[34] Use Jenkins [электронный ресурс] // jenkins-ci.org. URL: https://wiki.jenkins-ci.org/display/JENKINS/Use+Jenkins (дата доступа: 21.09.2015)

[35] Skybot Job Scheduler [электронный ресурс] // helpsystems.com. URL: http://www.helpsystems.com/skybot/products/skybot-scheduler (дата доступа: 08.09.2015)

[36] cURL [электронный ресурс] // haxx.se. URL: http://curl.haxx.se/ (дата доступа: 28.09.2015)