Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Балашов Илья Вадимович

Облачное тестирование программ TRIK Studio в среде Travis CI на примере задачи локализации в лабиринте

Курсовая работа

Научный руководитель:
ст. преп. Кириленко Я.А.

Санкт-Петербург
2019

Оглавление

Введение

В 2018-2019 гг. уже в четвёртый раз состоялась Олимпиада Национальной Технологической Инициативы. Это соревнование проводится по различным инженерным профилям, в числе которых участникам доступно направление "Интеллектуальные робототехнические системы", в рамках которого командам предлагаются к решению задачи, связанные с робототехникой. В частности, в заключительном этапе финалистам предстоит создать программу для конструктора ТРИК с тем, чтобы запрограммированный робот ТРИК решал определённую задачу. В сезоне 2017-2018 гг. в заключительном этапе участники решали задачу локализации робота в ортогональном лабиринте по известной карте. Жюри считало, что запрограммированный конструктор успешно и полностью выполнил эту задачу, если он, будучи расположенным в неизвестной клетке лабиринта с ортогональными стенами и в неизвестном ортогональном направлении, смог по мере своего движения по карте вычислить свои абсолютные координаты.

По результатам Олимпиады, как участники, так и члены жюри нередко высказывают свои предложения по совершенствованию процесса проведения соревнования. Например, в последние годы ввиду сложности предлагаемых задач проверка решений заключительного этапа осуществляется фактически в ручном режиме, что растягивает и усложняет и без того нелегкий финальный этап. Более того, не имея автоматических средств, команды вынуждены вручную тестировать свои программы на большом количестве тестовых данных, что для них крайне неудобно.

Кроме того, научный руководитель данной работы не доволен тем, что опубликованное в официальном сборнике решение задачи локализации в ортогональном лабиринте по известной карте является крайне плохо читаемым, и, возможно, работает не самым оптимальным образом. Соответственно, такое решение не является хорошим примером для подготовки будущих участников Олимпиады.

В рамках представленной работы будет предложен и реализован

способ автоматизации тестирования программ для набора ТРИК на базе сервиса для организации облачного тестирования Travis CI, который может быть использован и жюри, и участниками. Для наглядной демонстрации его работы будет разработан целый комплекс средств, который мог бы использоваться в 2017-2018 гг. организаторами олимпиады для интеграции описанного способа, включая "эталонное" решения задачи локализации. Также, будут показаны преимущества применения автоматической системы по сравнению с ручным тестированием решений, использовавшимся участниками в условиях олимпиады. Ввиду вышесказанного такой программный комплекс будет обладать высокой практической значимостью.

1. Постановка задачи

Учитывая сформулированные во введении проблемы были сформулированы следующие задачи:

1. Решить задачу локализации в ортогональном лабиринте по известной карте
    - Обеспечить возможность использования полученного кода в качестве ”эталона” для подготовки участников Олимпиады НТИ
2. Реализовать систему облачного тестирования программ для конструктора ТРИК на наборе тестов
3. Осуществить интеграцию данной системы с задачей локализации
4. Провести анализ эффективности использования реализованной системы по сравнению с ручным тестированием

2. Обзор

2.1. Обзор существующих решений

2.1.1. Система тестирования задач, используемая на платформе Stepik

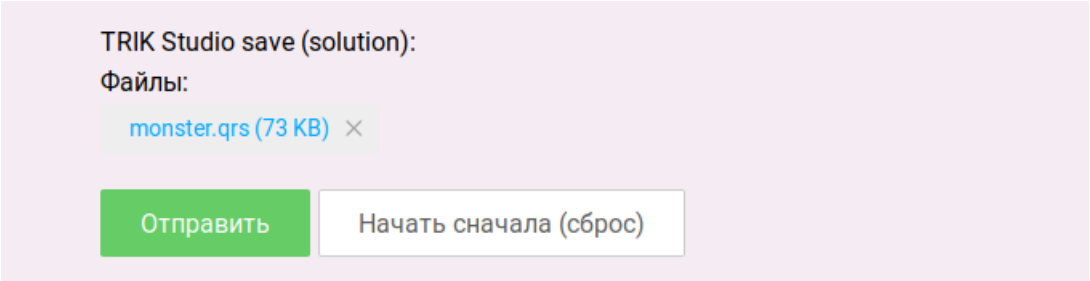

Рис. 1: Форма отправки решения на платформе Stepik

На платформе Stepik[1] реализована система, похожая на требуемую. Однако, данная реализация обладает некоторыми существенными недостатками, которые не позволяют её использовать для организации Олимпиады НТИ:

- Невозможность тестирования решения на произвольном, заданном пользователем наборе тестовых полей
- Нестабильность, иногда не проходят правильные решения, иногда зависает

2.1.2. Система олимпиадного тестирования EJudge

Для организации олимпиад по программированию различных типов существуют специальные профессиональные программные комплексы, одним из ярких представителей которых является система EJudge[2]. Однако, несмотря на гибкость, безопасность и применимость почти к любым задачам, она тоже не полностью подходит для решения поставленной задачи, поскольку:

[1]Stepik - образовательная платформа. URL: https://stepik.org/ (дата обращения: 9.05.2019)
[2]EJudge - система менеджмента контестов. URL: https://ejudge.ru/ (дата обращения 9.05.2019)

- Не позволяет пользователям использовать собственные наборы тестов для решаемой задачи, что ограничивает возможность использовать систему для самопроверки участников Олимпиады
- Тяжеловесна; потребуется проводить специальный инструктаж по работе с системой для соревнующихся

Таким образом, решение поставленной задачи, которое будет представлено в данной работе, высокоактуально, поскольку сочетает в себе, с одной стороны, гибкость и простоту настройки для пользователей, а также стабильность и возможность использовать собственные тестовые наборы, с другой.

2.2. Обзор использованных технологий и инструментов

2.2.1. Консольный чекер решений на базе TRIK Studio

Разработчиками ТРИК был предоставлен консольный чекер проектов для конструктора ТРИК, разработанный на основе TRIK Studio. Данная программа распространяется в виде Docker-образа и позволяет протестировать заданное решение на фиксированном наборе тестовых полей. Решение может быть задано как формате проекта TRIK Studio (.qrs), так и в виде исходного кода на языке JavaScript. Результат выдаётся в виде JSON-файлов, содержащих исход решения на конкретном тестовом полей. Чекер предполагается использовать в качестве ядра программного комплекса.

2.3. Travis CI

Travis CI представляет собой сервис облачного тестирования. Фактически, он предоставляет виртуальную машину, в которой будет выполнен необходимый пользователю сценарий тестирования. Отличительной особенностью сервиса является его тесная интеграция с хостингом IT-проектов Github. Это делает возможным автоматический запуск

тестирования по факту изменения файла решения задачи или по факту загрузки или изменения тестовых данных, что даёт необходимую в условиях проведения олимпиады для участников и жюри легковесность и простоту использования.

3. Описание решения

3.1. Сценарий использования

Общую схему работы реализованной системы и взаимодействия с ней пользователей рассмотрены на схемах 2 и 3:

Рис. 2: Прецедентная диаграмма системы

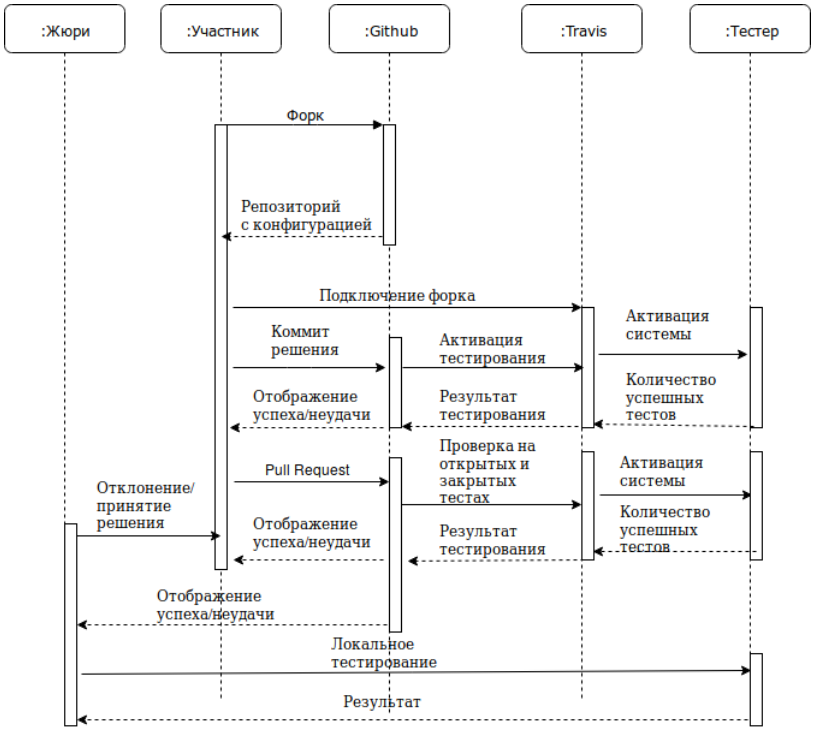

Рис. 3: Диаграмма взаимодействия

3.2. Решение задачи локализации в ортогональном лабиринте

Для реализации решения задачи локализации в ортогональном прямоугольном лабиринте по известной карте было принято решение использовать язык JavaScript, поскольку TRIK Studio, среда разработки для конструктора ТРИК, имеет отличную поддержку этого языка, что отражено в официальных материалах, распространяемых командой ТРИК [3]. Кроме того, этот язык хорошо поддерживается и консольным чекером на базе TRIK Studio. На вход алгоритму подаётся закодированная в формате ASCII art карта поля размером $n \times m$. Пример карты можно наблюдать на Рисунке 4.

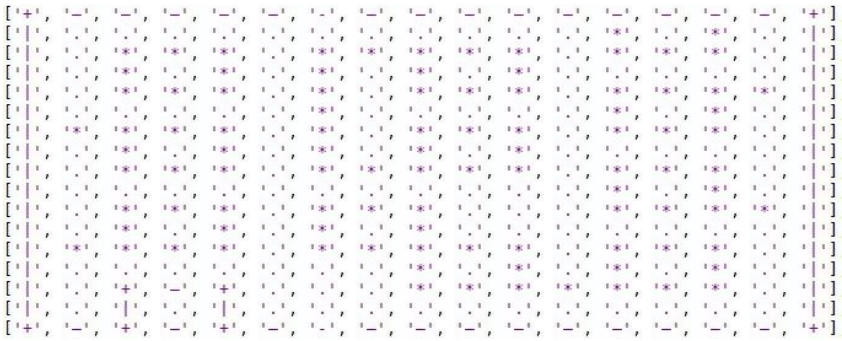

Рис. 4: Пример входного поля

Программа управляет роботом из конструктора ТРИК и имеет доступ, помимо двигателей и гироскопа, к двум инфракрасным сенсорам (по бокам робота) и ультразвуковому датчику расстояния в передней части робота. Для успешного решения задачи программа должна вывести на экран управляемого робота его текущие абсолютные координаты (начало координат - левый верхний угол). Это можно сделать, оценивая пройденный роботом путь по карте, а также сверяя замеченную конфигурацию стен с известной картой.

Говоря подробнее, для успешной локализации робота был использован следующий алгоритм. Сначала, генерируются *гипотезы* - множества различных возможных стартовых координат и направлений робота. Затем платформа ТРИК под управлением программы начинает

движения по так называемому правилу левой руки: при наличии возможности робот совершает поворот налево, иначе, если можно, едет прямо, иначе поворачивает направо. При перемещении в новую клетку по данному правилу робот запоминает свой новый сдвиг относительно положения в нулевой момент времени (то есть считаем начальную позицию началом координат и направлением 0 градусов). Кроме того, при перемещении в новое положение запоминается и наличие или отсутствие стен слева, справа и спереди от робота.

Вся эта информация используется для того, чтобы осуществить "симуляцию" каждой гипотезы. А именно, перебирается каждая из оставшихся сгенерированных в начале работы алгоритма гипотез, и проверяется возможность движения робота по пути, совершённому им в действительности, в предположении, что его абсолютное положение на момент старта совпадало с координатами и направлением из гипотезы.

По мере работы также проверяется совпадение конфигурации пройденных симулятором стен с конфигурацией, полученной реальным роботом. В результате подобной проверки отсекаются неверные гипотезы до тех пор, пока не останется ровно одно предположение, которое и окажется искомым решением задачи. Стоит также отметить, что в процессе движения робота, программа отслеживает попадание себя в цикл и предпринимает меры по выходу из него.

Решение использовать именно алгоритм перебора гипотез было принято в связи с тем, что он является, с одной стороны, достаточно простым для понимания, чтобы использовать полученный код в качестве наглядного примера для участников Олимпиады НТИ, а с другой, весьма эффективным по сравнению со многими тривиальными алгоритмами, некоторые из которых имеют асимптотику вплоть до экспоненциальной. Предложенный же алгоритм имеет полиномиальную сложность. Успешность работы рассмотренного алгоритма можно увидеть на примере его исполнения в симуляторе TRIK Studio, показанного на Рисунке 5.

Отдельного внимания заслуживает реализованный алгоритм передвижения робота. В рамках этой задачи потребовалось разрешить сразу

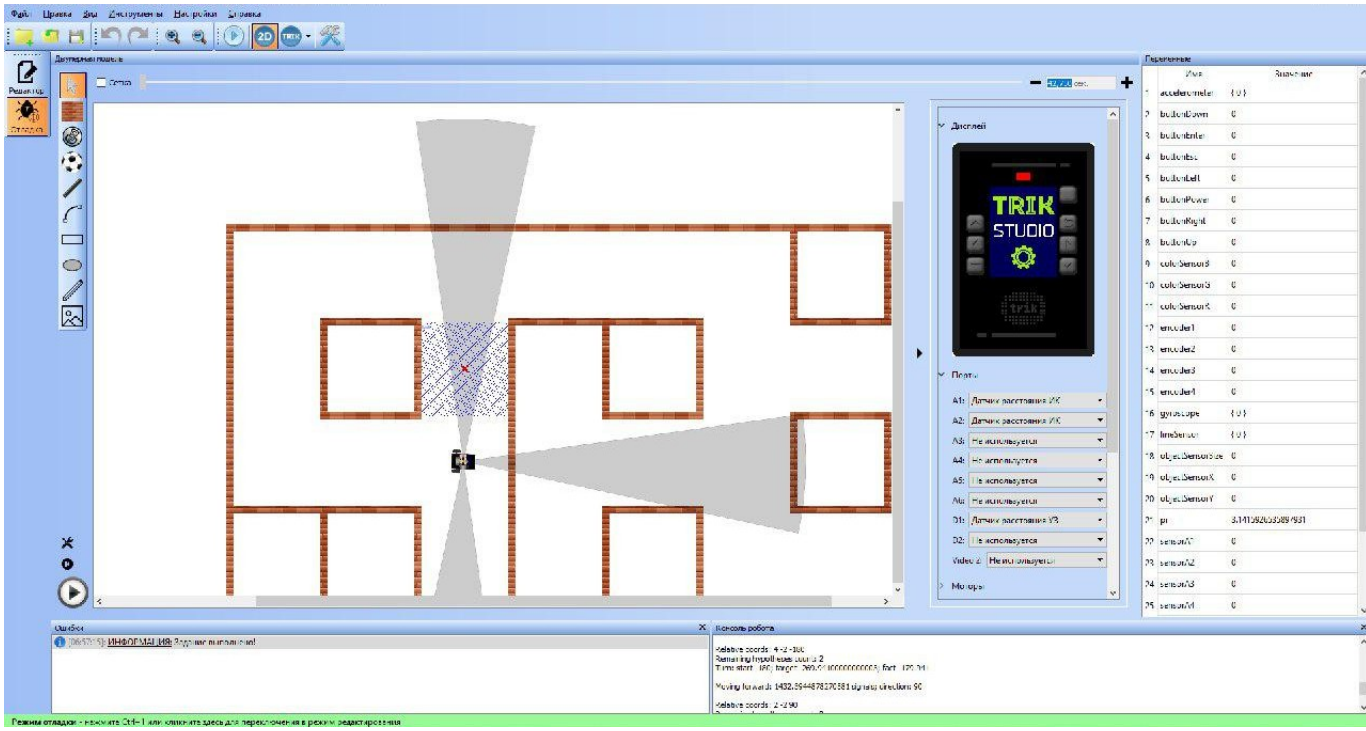

Рис. 5: Успешный процесс симуляции работы алгоритма на реальном роботе в среде TRIK Studio

несколько подзадач. Во-первых, нужно было ”научить” робота двигаться прямолинейно. Для этого был выбран алгоритм ”виртуального колеса” [2], который заключается в использовании виртуального энкодера в виде увеличивающейся с определённой частотой переменной. Реальные же моторы, сравнивая значения на своих реальных энкодерах, синхронизируются с виртуальным энкодером. Применение такой схемы, по исследованиям, проведенных в [2], является оптимальным при необходимых для решения задачи локализации средних и низких скоростях передвижения робота, что выражается в меньшем отклонении робота от прямолинейной траектории.

Также, для стабилизации робота на прямой используются так называемые пропорциональные регуляторы (П-регуляторы), опирающиеся на данные гироскопа и показания боковых инфракрасных датчиков расстояния. Общая формула, описывающая действие пропорционального регулятора представлена на Рисунке 6.

П-регуляторы отслеживают опасное приближение роботехнической платформы к стенам и его отклонение от прямолинейной траектории

$$
\begin{aligned}
power\_left(t+\delta t) &= power\_left(t) + K_p\, e(t) \\
power\_right(t+\delta t) &= power\_right(t) - K_p\, e(t) \\
e(t) &= \text{невязка (по гироскопу или ИК-датчикам)} \\
K_p &= \text{эмпирическая константа}
\end{aligned}
$$

Рис. 6: Формулы для мощности двигателей с использование П-регулятора

и прибавляют или убавляют мощность левого или правого мотора с тем, что вернуть тележку к корректному положению. Использование пропорционального регулятора для такого рода задач является практически стандартом в теории управления и робототехнике, поэтому его применения позволит обучающимся ознакомится с этим классическим механизмом, что соответствует поставленной задаче.

Далее, в рамках подзадачи движения робота, реализован продвинутый алгоритм поворота робота. Суть оптимизации заключается в том, что направление робота, получаемое с гироскопа, тождественно отображается со шкалы -180..180 градусов на шкалу $-\inf..+\inf$, то есть при осуществлении более чем одного оборота отсчёт угла продолжается далее за отметку 180, а не резко сменяется на -180. Это позволяет изящно решить "проблему 180/-180": при повороте вокруг своей оси на угол, близкий к 180, реальный робот, в силу инерции, может повернуться больше, чем надо, и тогда показания гироскопа резко сменятся со 180 на -180, что заставит робота осуществить еще один полный поворот, тележка начнёт крутиться. В предложенном же способе прокручивание исключено ввиду отсутствия резких перепадов угла. Такое решение тоже имеет методическую ценность для обучения юных робототехников, поэтому его применение в рамках поставленных целей оправдано.

Подводя итог, был реализован алгоритм локализации робота-тележки в ортогональном лабиринте по известной карте. Полученный код был вычитан автором, а также профессиональными программистами и разработчиками ТРИК, были написаны подробные комментарии, обеспечен достойный уровень качества кода. Программа была вручную, а так-

же с использованием автоматических средств протестирована на 200 тестовых полях с полным успехом. Таким образом, в полной мере выполнены все поставленные в рамках обсуждаемой подзадачи цели.

3.3. Облачная система тестирования на базе Travis CI

Облачная система, которую предстояло разработать и интегрировать с задачей локализации в контексте поставленных целей, представляет собой целый программный комплекс и состоит из:

- Автоматического генератора тестовых наборов для задачи локализации
- Средств интеграции чекера на базе TRIK Studio с платформой Travis CI
- Инструментов для запуска описываемой системы на локальной машине

3.3.1. Генератор тестовых наборов для задачи локализации в ортогональном лабиринте

Для подробного и всеобъемлющего тестирования решений участников олимпиад уровня заключительного этапа Олимпиады НТИ требуются качественные и продуманные тестовые наборы в достаточно большом количестве. В рамках разрабатываемого комплекса для этого будет крайне полезен генератор полей, который был бы пригоден для интеграции с автоматической тестовой системой, что позволило бы обеспечить должное качество проверки (и самопроверки) программ участников.

Для выполнения этих целей в состав разрабатываемого пакета программ был включен консольный компонент MapGenerator, который генерирует тестовые поля для задачи локализации в ортогональном лабиринте по известной карте. Генератор реализован на языке Python как

наиболее популярном скриптовом языке программирования для решения подобного рода задач на момент написания работы. Кроме того, интерпретатор Python входит в стандартную поставку ОС семейства Ubuntu, с которыми работает сервис Travis CI.

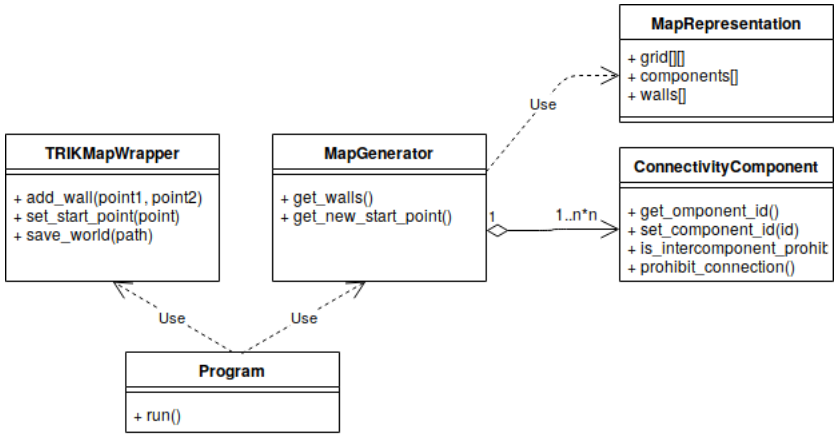

Рис. 7: Архитектура генератора

Выбор архитектуры, представленной на Рисунке 7, где реализация основной логики MapGenerator отделена от обёртки над собственным форматом полей TRIK Studio TRIKMapWrapper, обусловлен тем, что эти компоненты могут использоваться как в составе отдельной консольной утилиты, которая будет полезна для использования участниками и жюри на локальных машинах, так и в виде программного модуля, который может быть интегрирован в систему автоматической проверки.

Общий алгоритм работы инструмента таков: сначала псевдослучайным образом генерируется количество стен, недоступных областей (”контейнеров”), а также компонент связности в графе стен, который будет построен. Дальше, на карте расставляются контейнеры, и они распределяются по нужному количеству компонент связности. Потом для каждой компоненты выбирается, может ли она касаться других компонент. Этот шаг обусловлен необходимостью генерировать циклические структуры на поле, что прописано в условиях заключительного этапа Олимпиады НТИ сезона 2017-2018 гг. В конце, генерируются стены в соответствии с описанными выше ограничениями и, наконец, класс-обертка генерирует готовое поле.

Таким образом, был реализован генератор полей для задачи лока-

лизации в ортогональном лабиринте по известной карте. В дальнейшем этот генератор был использован для тестирования реализованной в рамках поставленных целей облачной системы на основе Travis CI, а также генерированные программой поля были включены в начальную поставку системы тестирования. Более того, консольная утилита на базе MapGenerator может быть использована готовящимися к Олимпиаде студентами для самопроверки. Значит, код обладает высокой практической ценностью и полностью соответствует поставленным задам по разработке программного комплекса.

3.3.2. Реализация системы тестирования TRIK Studio с платформой Travis CI

Одной из основных поставленных задач является разработка стратегии по интеграции существующего чекера на основе TRIK Studio, разработанного командой ТРИК, с облачной системой. В качестве облачной системы была выбрана платформа Travis CI, поскольку она обладает простой и понятной схемой работы как для разработчиков, так и для участников Олимпиады, которые, возможно, захотят использовать систему тестирования. Также она отлично документирована, и, в отличие от конкурентов, обладает хорошей поддержкой ОС семейства GNU/Linux, которые используются на Олимпиаде НТИ.

Для осуществления интеграции был написан обширный набор скриптов на языках Bash и Python, интерпретаторы которых включены в поставку используемой системы Ubunutu, а также собственной модификации сервисом Travis CI языка YML. Основные скрипты данного набора следующие:

- Управляющий скрипт Travis CI, реализован на YML/Bash.

  Выполняет автоматическую сборку и полную настройку Docker-образа с чекером от команды ТРИК. Также выполняет загрузку решения участника и его тестовые поля из подаваемого на вход Github-репозитория, специальным образом настраивает чекер так, чтобы он корректно распознавал и обработал входные

данные (код и тесты). В конце, запускает тестирование, а затем дополнительную программу, интерпретирующую результаты.

- Интерпретатор результатов работы чекера, реализован на языке Python.

  Выполняет извлечение результатов тестирования из внутренней памяти Docker-контейнера, их разбор из формата JSON. Также осуществляется проверка статуса тестирования каждого теста и подсчёт успешных и неуспешных тестов, число возвращается системе тестирования.

  Выбор языка Python обусловлен необходимостью осуществлять парсинг JSON-файлов, с чем с трудом справляется уже используемый в проекте Bash.

- Скрипт, который позволяет дополнительно протестировать решение на "закрытом" наборе полей. Написан на YML/Bash.

  Программа предназначена для использования жюри и реализована как в виде отдельной утилиты, так и в составе управляющего скрипта. Скрипт загружает скрытые от участников Олимпиады наборы тестов из судейского репозитория Github и осуществляет тестирование решения участника дополнительно и на этом наборе. Для сокрытия расположения судейского репозитория и конфигурации закрытых тестов используется технология "encrypted values", предоставляемая сервисом Travis CI [1].

  Скрипт позволяет оптимизировать процесс сдачи готового решения на проверку судьям. На Рисунке 8 представлена схема этого процесса.

Таким образом, реализованная система полностью удовлетворяет поставленным задачам, а именно осуществляет автоматизированное тестирование решений задач в облачной среде.

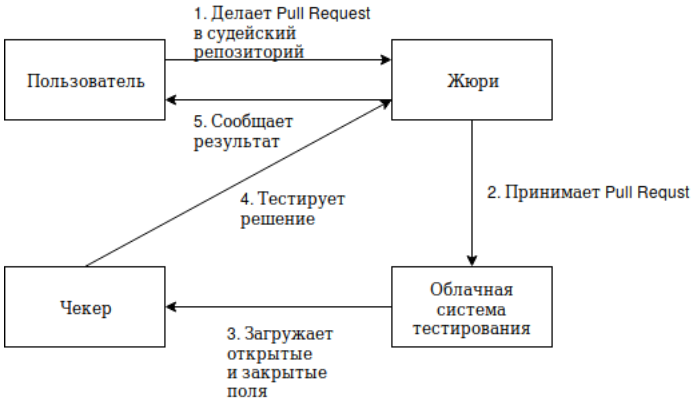

Рис. 8: Последовательная схема сдачи решения и его проверки жюри

3.3.3. Интеграция системы с задачей локализации

В качестве доказательства работоспособности и функциональности программного комплекса для тестирования, он был проинтегрирован с описанной выше реализацией задачи локализации в ортогональном лабиринте. В частности, была проведена успешная проверка системы сдачи участником решения на проверку, результат которой можно наблюдать на Рисунке 9.

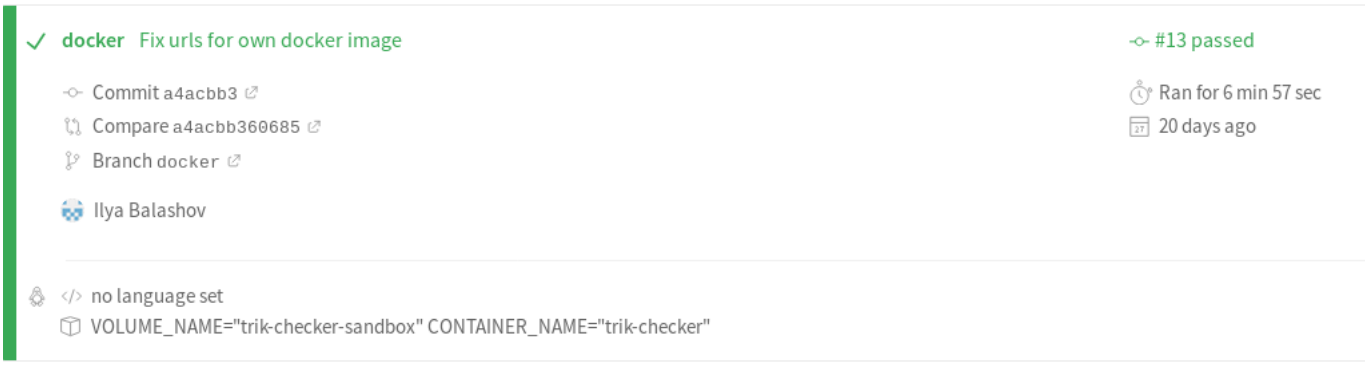

Рис. 9: Отчёт системы Travis CI об успешном тестировании решения

В процессе интеграции была осуществлена генерация тестовых полей посредством генератора, настройка управляющих скриптов на загрузку закрытых полей, а также подготовка для участников Олимпиады и её организаторов репозитория-примера, на основе которого они могли бы в последующих сезонах настроить представленную в данной работе систему для других задач.

Проверка, проведённая опрошенными студентами матмеха СПбГУ,

по критериям функциональности, удобства и интуитивности использования системы на примере задачи локализации показала положительные результаты, что говорит о высокой значимости решения на практике.

4. Проверка эффективности применения системы

Поскольку основным назначением разрабатываемого автоматического комплекса тестирования является сокращение времени, которое участники Олимпиады НТИ затрачивают на проверку своих решений, было проведено сравнение времени проверки эталонного решения задачи локализации в ортогональном лабиринте на 30 тестовых полях с временем запуска той программы и на тех же полях вручную в среде TRIK Studio с ускорением времени в симуляторе 300%.

В рамках проверки, участники тестирования, 3 студента матмеха СПбГУ, в среднем затратили на ручное тестирование 17 минут, в то время как среднее время работы автоматической системы по результатам 3х запусков составило 7 минут. Таким образом, решение на базе Travis CI обладает неоспоримым преимуществом по времени работы и, соответственно, полностью удовлетворяет поставленной задаче.

Заключение

Подводя итог всему вышесказанному, были выполнены все задачи, поставленные в рамках курсовой работы:

- Решена задача локализации в ортогональном лабиринте по известной карте
  - Код подготовлен для использования в качестве примера для будущих участников Олимпиады НТИ
- Реализована система облачного тестирования программ для конструктора ТРИК на наборе тестов с использованием сервиса Travis CI
- Осуществлена интеграция системы тестирования с задачей локализации, а именно:
  - Разработан генератор карт для задачи локализации
  - Подготовлен образцовый репозиторий с примеров использования системы
  - Проведено функциональное и юзабилити тестирование системы
- Проведено сравнение эффективности использования полученной автоматической системы и ручного тестирования решений задач.
  - Показаны существенные преимущества разработанной системы.

Список литературы

[1] Encryption keys - Travis CI // Travis CI User Documentation. — Режим доступа: https://docs.travis-ci.com/user/encryption-keys/ (дата обращения: 9.05.2019).

[2] Киселев О.М., Киселев И.О., Я.А.Кириленко. Управление моторами тележки с контроллером Трик на JavaScript // Олимпиада НТИ, профиль "ИРС". Материалы для участников. — 2017.

[3] Широколобов И.Ю. Знакомство со средой программирования TRIK Studio [Презентация] // Олимпиада НТИ, профиль "Интеллектуальные робототехнические системы", материалы. — 2017. — Режим доступа: https://drive.google.com/file/d/0B_ZzDCvzeLSQVTlRbVFRc3lYWmc (дата обращения: 9.05.2019).