УДК 004.054 DOI: 10.26102/2310-6018/2020.28.1.032

Решение задачи оптимизации регрессионного тестирования с использованием нейросетевого подхода

А.Д. Данилов, В.М. Мугатина
Воронежский Государственный Технический Университет, Воронеж, Россия

Резюме: Жизненный цикл разработки любого программного обеспечения в обязательном порядке сопровождается активностями по верификации и тестированию продукта. Среди всех процессов, лежащих в основе тестирования ПО, наиболее ресурсозатратным, но вместе с тем обязательным для исполнения, является регрессионное тестирование. Главная задача регрессионного тестирования заключается в проверке, что внесенные в код продукта изменения не повлияют на уже реализованную и протестированную функциональность разрабатываемой системы. Указанная задача определяет необходимость в запуске регрессионных проверок после выпуска каждой новой версии программного обеспечения. Однако в условиях ограниченности ресурсов при регрессионном тестировании из полного набора проверок, как правило, запускаются лишь выборочные тестовые кейсы. Этот факт обуславливает задачу оптимального выбора максимально приоритетных для запуска регрессионных проверок. Решением поставленной задачи может выступать использование системы управления тестированиям, позволяющей определять приоритеты тест-кейсов в зависимости от внесенных в код продукта правок, основанной на нейросетевой модели. Функционирование такой системы возможно благодаря сбору и анализу данных об изменениях в коде из системы контроля версий и дальнейшему использованию этой информации в качестве входных данных для искусственной нейронной сети. Выходными данными в такой модели являются результаты выполнения регрессионных проверок, то есть факт обнаружения ошибки или удостоверения, что продукт работает ожидаемым образом. Таким образом, обучение нейронной сети проходит на основе реальных данных, полученных на основании результатов запуска тестов на этапе разработки программного обеспечения. Обученная нейронная сеть способна приоритизировать тестовые случаи и оптимизировать ресурсы на проведение регрессионного тестирования.

Ключевые слова: тестирование программного обеспечения, верификация, искусственные нейронные сети, регрессионное тестирование

Для цитирования: Данилов А.Д., Мугатина В.М. Решение задачи оптимизации регрессионного тестирования с использованием нейросетевого подхода. *Моделирование, оптимизация и информационные технологии.* 2020;8(1). Доступно по: https://moit.vivt.ru/wp-content/uploads/2020/02/DanilovMugatina_1_20_1.pdf DOI: 10.26102/2310-6018/2020.28.1.032

Neural network based solution for regression testing optimization

A.D. Danilov, V.M. Mugatina
Voronezh State Technical University, Voronezh, Russia

Abstract: Regression testing is important task of retesting software systems after changes in the code of product to ensure that changes do not influence previously implemented functionality. Regression testing is run after a new version of software has been developed. Usually only limited subset of test cases is executed for a new version of software through restricted resources. This shows the problem of selection the most important regression test cases. To cope with limited resources, different regression

testing techniques was developed to reduce the number of test cases to be executed. One of these techniques is test case prioritization based on neural network model. Such mechanism can collect data about code changes from Version Control System and use it as inputs for neural network. The outputs for such neural network model are regression tests' execution results. Groups of regression tests can be united by functionality under the test. Neural network model can be trained on real results during the phase of software developing. Trained neural network can detect the most important test cases for execution after each change in product code. Such technique can be used to guide the focus of the testing efforts.

Keywords: software quality assurance, software verification, artificial neural network, regression testing

For citation: Danilov A.D., Mugatina V.M. Neural network based solution for regression testing optimization. *Modeling, Optimization and Information Technology*. 2020;8(1). Available from: https://moit.vivt.ru/wp-content/uploads/2020/02/DanilovMugatina_1_20_1.pdf DOI: 10.26102/2310-6018/2020.28.1.032 (In Russ).

Введение

Тестирование программного обеспечения – сложный процесс, который, как правило, требует привлечения специалистов, располагающих экспертными знаниями о разрабатываемой системе. Но в связи с тем, что данный процесс требует большого числа ресурсов, в том числе временных, становится актуальным вопрос о возможности автоматизации процесса тестирования. Наиболее подходящим для автоматизации является вид тестирования, называемый регрессионным. Регрессионное тестирования проводится для того, чтобы убедиться, что новые правки в коде продукта не повлияют на уже разработанные и протестированные модули системы. Регламент проекта обычно требует запуска регрессионных проверок после каждого обновления программного обеспечения, связанного с изменениями в коде или инфраструктуре продукта.

Автоматизация регрессионного тестирования может осуществляться при помощи специальных скриптов – сценариев выполнения предварительно заданных действий в тестируемой системе, которые исполняются в автоматическом режиме и заканчиваются составлением отчета (также в автоматическом режиме), на основании которого можно сделать выводы о качестве программного продукта. Автоматизированные тесты разрабатываются с помощью языков программирования и позволяют при помощи специального драйвера, инструмента взаимодействия с браузером, управлять последовательностью действий в системе. При успешном выполнении сценария тест помечается как пройденный, а в случае обнаружения ошибки (отклонения от ожидаемого поведения) – неуспешным, с указанием, какая именно проверка в тесте определила, что фактическое поведение программного продукта отличается от заложенного в ходе разработки. Проверки, осуществляемые в рамках разработанных тестов, могут затрагивать функциональные аспекты (такие как проверка логики работы системы в соответствии с бизнес-требованиями, проверка корректности отображения интерфейса, тестирование безопасности) и нефункциональные характеристики (нагрузочное тестирование, тестирование на отказ и восстановление).

Примером функционального теста, входящего в регрессионное тестирование большинства программных продуктов, является проверка авторизации в системе. Действиями в рамках такого теста является последовательность: открыть браузер – перейти на страницу авторизации – ввести учетные данные. Ожидаемым результатом является сообщение об успешной авторизации. Если в ходе теста скрипт не обнаружил сообщения, что авторизация произошла успешно, он будет помечен как неуспешный, и в отчете можно будет увидеть, что в системе наблюдаются какие-то проблемы с

авторизацией. Отчет также может содержать скриншот интерфейса системы в момент ошибки и файл лога для более удобного анализа ошибки.

Однако многие команды по разработке ПО сталкиваются с ограниченностью ресурсов на проведение тестовых активностей, даже с учетом автоматизации, в связи с чем встает вопрос об оптимизации проведения регрессионного тестирования. Оптимизация возможна путем приоритизации тестовых кейсов: наиболее важные и потенциально содержащие ошибки модули программного обеспечения тестируются в первую очередь и более тщательно, позволяя сократить время нахождения критичных ошибок, а значит, увеличить скорость выпуска новых версий продукта [1].

В качестве примера, показывающего необходимость приоритизации регрессионных тестовых проверок, можно рассмотреть абстрактный программный продукт, полностью протестированный перед началом использования. Однако, уже после начала использования может появиться необходимость в срочной доработке определенной части функционала, которую необходимо запустить в очень сжатые сроки. Таким образом, для оптимального использования имеющихся временных ресурсов на проведение тестирования, необходимо в первую очередь проверить ту часть функционала, в которую были внесены изменения, а также функционал, который глобально влияет на всю систему (такой как авторизация), все остальные кейсы могут быть проверены позже. Инструмент, позволяющий определять очередность запуска тестов в автоматическом режиме, позволяет в таких ситуациях избегать привлечения специалистов-экспертов со знаниями о разрабатываемой системе.

Таким образом, для решения задачи оптимизации процесса регрессионного тестирования программного обеспечения в условиях ограниченности ресурсов, необходимо, с одной стороны, разработать тесты, выполняющиеся в автоматическом режиме, а с другой стороны, использовать специальный инструмент, позволяющий приоритизировать тестовые кейсы и запускать их оптимальным образом.

Регрессионное тестирование программного обеспечения

Современным программным продуктам предъявляется обширный набор требований ввиду их технической сложности, ожидаемой длительной продолжительности использования и закладываемой комплексной бизнес-логики. Основные активности процесса обеспечения качества программного обеспечения (ПО) связаны с оценкой соответствия поведения программы предъявляемым к ней требованиям. При этом, чем сложнее тестируемая система, тем выше нагрузка на тестирование, которое может занимать более 50% всех ресурсов проекта в процессе разработки и поддержки программных систем.

Являясь важной и неотъемлемой частью большинства проектов по разработке ПО, тестирование при этом имеет ряд значительных ограничений. В первую очередь ограничения связаны с тем, что требуемые усилия на проведение исчерпывающего тестирования превышают доступные ресурсы, и поэтому тестирование должно быть сосредоточено главным образом на наиболее важных аспектах ПО. Функционал, подлежащий первостепенной проверке, как правило, обусловлен либо приоритетным значением для общей работоспособности системы, либо имеющимися предпосылками считать, что внутри данного модуля программы может быть обнаружена ошибка. Таким образом, становится актуальной проблема приоритизации тестирования программного обеспечения.

На данном этапе следует выделить две основных стратегии тестирования программных продуктов: тестирование черного и белого ящика [2] (представлены на

Рисунке 1). Тестирование черного ящика предполагает проверку функционального поведения системы с точки зрения конечного пользователя, при котором не используется знание инженера по тестированию о внутреннем устройстве тестируемого кода проекта. В стратегии белого ящика во время тестирования рассматривается внутренняя логика, структура и реализация кода.

Большинство команд разработки программных продуктов придерживаются практики использования тестирования ПО методом черного ящика, что в значительной мере снижает требования к квалификации инженеров по тестированию, но при этом обеспечивает приемлемый уровень оценки качества системы. Однако в таком случае выявление важных тестовых сценариев нетривиально из-за отсутствия знаний об исходном коде. Методы тестирования белого ящика способны идентифицировать изменения в ПО на уровне кода, и анализ этих изменений способен в значительной мере помочь в составлении сценариев проверки продукта и тем самым обеспечить возможность избегать избыточных проверок, характерных стратегии тестирования черного ящика.

Среди различных видов тестирования, в той или иной мере применяемых в проектах по разработке ПО (функциональное, нагрузочное, тестирование безопасности, тестирование установки и т.д.) наиболее ресурсозатратным, но вместе с тем обязательным к исполнению является регрессионное тестирование. Такой вид тестирования может быть как функциональным, так и нефункциональным, и направлен на проверку изменений, сделанных в коде продукта или инфраструктуре, для подтверждения факта, что существующая ранее функциональность системы работает как и прежде. Под регрессионными ошибками понимаются неисправности, которые появляются при добавлении новых методов кода или при исправлении допущенных ранее ошибок. Как было сказано ранее, регрессионные проверки требуют максимальной концентрации ресурсов, так как необходимы после всех изменений в коде продукта и должны охватывать полную функциональность разрабатываемой системы, что обуславливает острую необходимость в оптимизации данного вида тестирования.

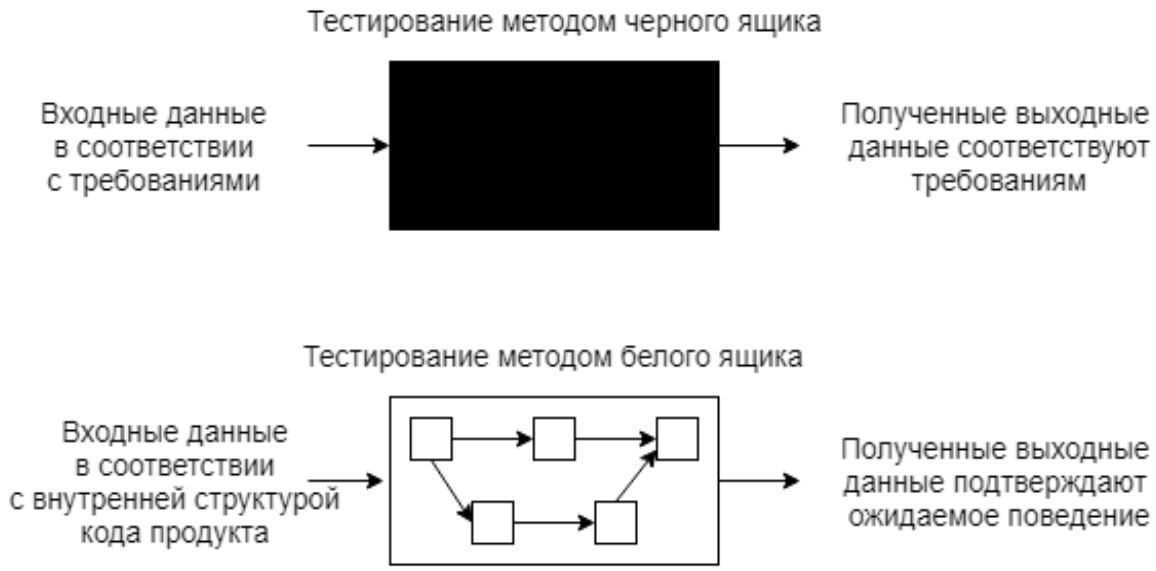

Рисунок 1 - Основные характеристики стратегий тестирования белого и черного ящика
Figure 1 - Main features of White and Black Box testing strategies

Системы контроля версий программного кода в аспектах разработки и тестирования ПО

Так как набор регрессионных проверок требуется проводить после любых правок, вносимых в код программы, необходим механизм отслеживания таких изменений. Для этой цели командам разработки программного обеспечения чрезвычайно важно использовать систему контроля версий.

Система контроля версий представляет собой программное обеспечение, записывающее все изменения в специальный файл или набор файлов в течение всего времени работы с кодом и предоставляющее возможность вернуться к определённой версии ПО в любой момент времени. Помимо поддержки версионности системы контроля версий решают не менее важную задачу взаимодействия с единым кодом продукта большого числа пользователей, то есть программистов, работающих над проектом. Общий механизм работы с распределенной системой контроля версий представлен на Рисунке 2.

Каждое обновление кода продукта должно быть протестировано регрессионными проверками. При этом, если приближаться к концепции тестирования белого ящика, каждый измененный или добавленный участок кода может быть проанализирован, и на основе такого анализа набор регрессионных тестов может быть сокращен, если правки затрагивают только определенную функциональность продукта и никаким образом не влияют на остальной функционал [3].

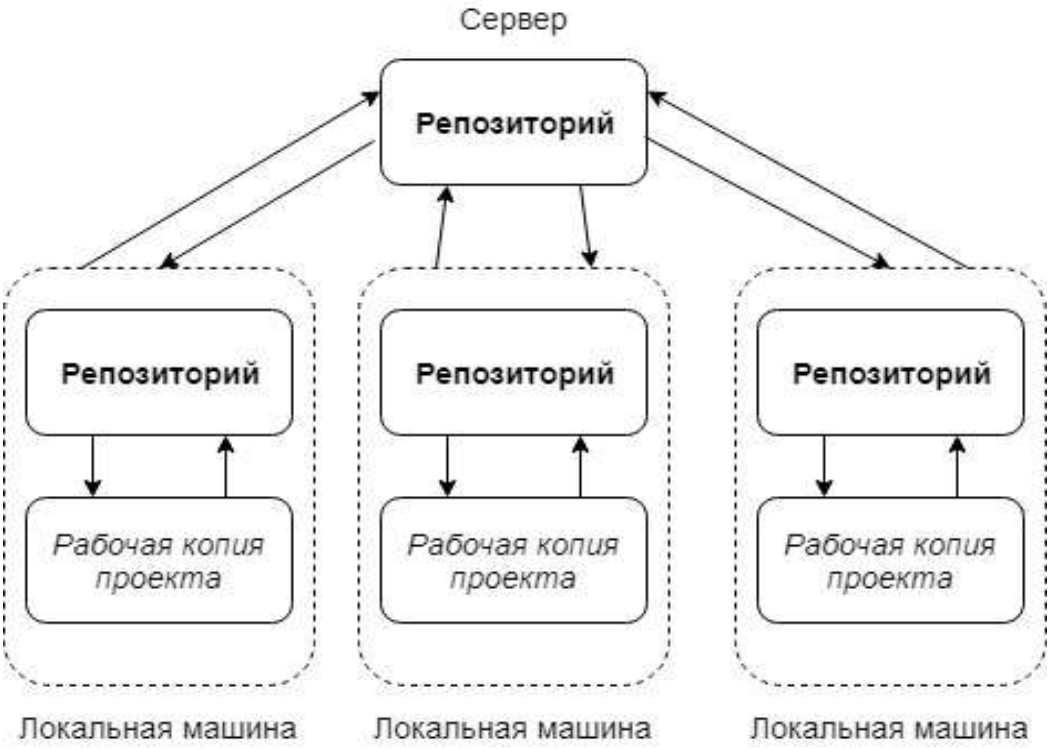

Рисунок 2 - Общий механизм работы с распределенной системой контроля версий
Figure 2 - General mechanism of working with a distributed version control system

Нейросетевая модель приоритизации регрессионного тестирования

Инструмент искусственных нейронных сетей успешно применяется для решения различного рода задач, в том числе для задачи принятия решений и управления в различных технических системах, близкой к задаче классификации. Область применения нейросетевых моделей достаточно обширна, так как классификации подлежат любые ситуации, характеристики которых могут быть поданы на вход нейронной сети. На выходе нейросети при этом рассчитывается признак решения, которое сеть приняла, и это решение может быть принято в расчет для определения требуемого поведения рассматриваемой системы.

Система управления регрессионным тестированием на проекте по разработке сложных программных продуктов также может быть рассмотрена в качестве системы принятия решений [4]. В роли входных данных могут быть использованы характеристики изменений, внесенные в код каждого из модулей программного проекта, такими характеристиками могут служить число добавленных/измененных строк кода, число добавленных методов, цикломатическая сложность функций, модулей, методов и т.д. Разделение на модули программного проекта может быть проведено различными способами, например, в соответствии с определенной функциональностью системы или назначением каждого из классов кода продукта. Выходными параметрами является спрогнозированный результат, является ли каждый рассматриваемый модуль потенциально содержащим ошибку, которая может привести к поведению системы, отличному от ожидаемого. Если результаты прогноза сводятся к тому, что вероятность наличия ошибки в модуле больше порогового значения, регрессионные тесты для этого модуля запускаются в первую очередь и в полном объеме. Если, основываясь на информации от нейросети, модуль потенциально не содержит ошибки, набор регрессионных тестов для такого модуля может быть минимальным и может быть запущен после наиболее приоритетных тестов. В наиболее общем виде модель нейронной сети представлена на Рисунке 3.

Обучение нейронной сети может быть проведено на основе реальных данных непосредственно в процессе разработки проекта. Каждое внесенное изменение в код инициирует запуск полного набора регрессионных тестов, и, если выполнение теста заканчивается обнаружением ошибки, данные об этом при помощи алгоритма обратного распространения ошибки передаются на вход нейросети и корректируют веса нейронов скрытых слоев, улучшая тем самым способность нейросети адекватно рассчитывать вероятность появления ошибки.

Таким образом, на основе данных, рассчитанных при помощи нейронной сети, может быть принято решение в задаче приоритизации набора регрессионных проверок. Системы контроля версий используются при разработке продукта повсеместно, а потому входные данные могут быть получены без специальных ресурсных и временных затрат. Запуск регрессионного тестирования на начальном этапе разработки проекта также является стандартным решением, предоставляя данные для обучения сети без постороннего привлечения источников экспертных знаний. При этом, наличие системы управления регрессионным тестированием на основе предложенного подхода делает возможным определение наиболее приоритетных для первостепенного запуска тестов, что позволяет в значительной мере сократить время нахождения критичных ошибок и предоставляет возможность избежать избыточных проверок.

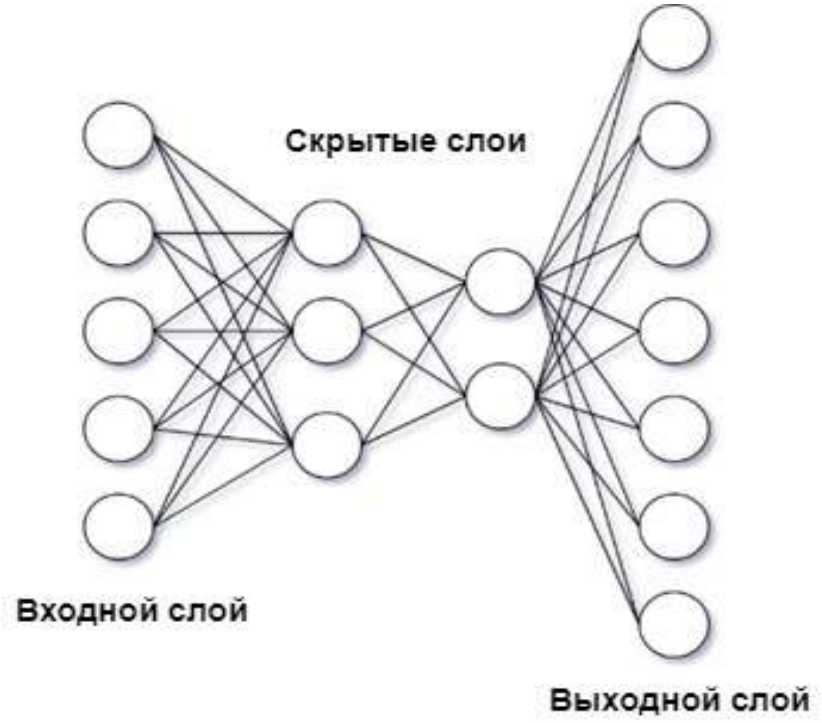

Рисунок 3 - Модель нейронной сети
Figure 3 - Neural Network model

ЛИТЕРАТУРА

1. Данилов А.Д., Мугатина В.М. Верификация и тестирование сложных программных продуктов на основе нейросетевых моделей. *Вестник Воронежского государственного технического университета.* 2016;12(6):62-67.
2. Данилов А.Д., Фёдоров А.И. Иерархическая структура процесса тестирования сложного программного обеспечения. *Вестник Воронежского государственного технического университета.* 2014;10(3-1):18-21.
3. Danilov A.D., Samotsvet D.A., Mugatina V.M. Using neural network models in the quality management system for the software defect prediction. *IOP conference series: Materials science and engineering. International Workshop "Advanced Technologies in Material Science, Mechanical and Automation Engineering".* MIP: Engineering. 2019
4. Данилов А.Д., Мугатина В.М. Применение аппарата искусственных нейронных сетей в задаче оптимизации процесса тестирования программного обеспечения// *Вестник Воронежского государственного технического университета*. 2018;14 (2):7-14.

REFERENCES

1. Danilov A.D., Mugatina V.M. Verification and software testing of complex products based on neural network models, *The Bulletin of Voronezh State Technical University*, 2016;12(6):62-67.
2. Danilov A.D., Fedorov A.I. Hierarchical structure of software testing process, *The Bulletin of Voronezh State Technical University*, 2014;10(3-1):18-21.
3. Danilov A.D., Samotsvet D.A., Mugatina V.M. Using neural network models in the quality management system for the software defect prediction, *IOP conference series: Materials*

science and engineering. International Workshop "Advanced Technologies in Material Science, Mechanical and Automation Engineering" .MIP: Engineering .2019

4. Danilov A.D., Mugatina V.M. Application of artificial neural networks for software testing optimization, *The Bulletin of Voronezh State Technical University*, 2018;14 (2):7-14.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Данилов Александр Дмитриевич, доктор технических наук, профессор, Воронежский Государственный Технический Университет, профессор кафедры электропривода, автоматики и управления в технических системах, Воронеж, Российская Федерация.
e-mail: ivanilovatn@gmail.com

Мугатина Варвара Михайловна, аспирант, кафедры электропривода, автоматики и управления в технических системах, Воронежский Государственный Технический Университет, Воронеж, Российская Федерация.
e-mail: varvaramugatina@gmail.com

Aleksandr D. Danilov, Doctor Of Technical Science, Professor Of The Department Of Electric Drive, Automation And Control In Technical Systems, Voronezh State Technical University, Voronezh, Russian Federation.

Varvara M. Mugatina, Graduate Student Of The Department Of Electric Drive, Automation And Control In Technical Systems, Voronezh State Technical University, Voronezh, Russian Federation