Titova Vera, Khmelnitsky national university,
Assistant Professor, Ph.D of Technical Science, the Dept. of System Programming
sobaka2032@rambler.ru

Neuronetwork method for analysis results of critical software testing

Abstract: The problem of determining matching of results of critical software testing to functional and non-functional requirements has been considered in this article. This problem was solved by using fuzzy neural network.

Keywords: Functional and non-functional requirements, critical software, fuzzy neural network.

Титова Вера, Хмельницкий национальный университет,
Доцент, кандидат технических наук, каф. системного программирования
sobaka2032@rambler.ru

Нейросетевой метод для анализа результатов тестирования программного обеспечения критического применения

Аннотация: В статье рассмотрена задача определения соответствия результатов тестирования программного обеспечения критического применения функциональным и нефункциональным требованиям с помощью нечеткой нейронной сети.

Ключевые слова: функциональные и нефункциональные требования, программное обеспечение критического применения, нечеткая нейронная сеть.

Вступление. Одной из важных и актуальных задач разработки программного обеспечения (ПО) является повышение уровня его качества. Решение этой задачи особенно важно для ПО критического применения,

использование которого связано с безопасностью жизнедеятельности. Качество разработанного ПО в первую очередь зависит от качества проведенного тестирования. Поэтому на сегодняшний день важным условием повышения качества ПО критического применения является разработка нормативной базы, определяющей требования к качеству указанного ПО, и методы оценивания реализации этих требований.

Целью данной статьи является разработка метода оценивания соответствия результатов тестирования ПО его функциональным и нефункциональным требованиям.

Характеристика предметной области. Тестирование (software testing) - деятельность, выполняемая для оценивания и совершенствования ПО. Эта деятельность, в общем случае, базируется на обнаружении дефектов и проблем в программных системах [1]. Тестирование ПО состоит из динамической верификации поведения программ на конечном (ограниченном) наборе тестов (set of test cases), выбранных соответствующим образом с обычно выполняемых действий в прикладной области, которые обеспечивают проверку соответствия ожидаемого поведения ПО [1]. Выполнение тестов должно содержать основные принципы ведения научного эксперимента:

- должны фиксироваться все работы и результаты процесса тестирования;

- форма записи таких работ и их результатов должна быть такой, чтобы соответствующий смысл их был понятным;

- тестирование должно проводиться в соответствии с заданными и документированными процедурами;

- тестирование должно проводиться над идентифицированной версией и конфигурацией ПО.

Для определения успешности результаты тестов должны оцениваться и анализироваться. В большинстве случаев, "успешность" тестирования подразумевает, что тестируемое ПО функционирует так, как ожидалось, не

определяя насколько именно близко к желаемому результату оно функционирует [1].

Постановка задачи. Из характеристики предметной области можно сделать вывод, что фактически результаты, полученные в процессе тестирования, сводятся к выводу: пройдены тесты (pass) или не пройдены (fail) [1]. Подобное оценивание является относительно поверхностным и не дает точного определения насколько разработанное ПО соответствует требованиям к нему. А потому, признанное качественным ПО может стать источником сбоев при изменении конфигурации или условий функционирования.

Разработка метода определения уровня соответствия ПО критического применения функциональным и нефункциональным требованиям позволило бы в дальнейшем обеспечить большую глубину измерений и, соответственно, повысить эффективность тестирования.

Входные данные задачи оценивания соответствия ПО требованиям по результатам тестирования не являются числовыми, поэтому использование для ее решения математических методов не представляется целесообразным. Формализация входных данных вышеуказанной задачи позволила бы упростить ее решение и, как результат, повысить качество используемого ПО. Для проведения формализации целесообразным является использование методов нечеткого логического вывода, которые позволяют оперировать вместо нечисловых значений числовыми значениями их принадлежности к соответствующим множествам.

Будем считать, что каждому требованию отвечает отдельный тест. Для результата выполнения каждого из тестов определим его принадлежности к двум нечетким множествам «выполнение» и «невыполнение».

В зависимости от целей тестирования, тест считается выполненным, если все или более половины его запусков на выполнение завершились успешно, тест считается невыполненным, если все или более половины его запусков на выполнение завершились неуспешно [1].

Поэтому, нечеткое множество «выполнение» будет состоять из трех подмножеств «выполнен полностью», «частично не выполнен» и «больше выполнен, чем не выполнен». Нечеткое множество «невыполнение» будет состоять из следующих подмножеств «полностью не выполнен», «частично выполнен» и «больше не выполнен, чем выполнен».

μ_j^i - степень принадлежности теста каждому из подмножеств множества «выполнение», *i* – порядковый номер теста, *j* – номер нечеткого подмножества (1 – «выполнен полностью», 2 - «частично не выполнен», 3 - «больше выполнен, чем не выполнен»), $\mu_j^i \in [0,1]$. Эти степени определяются экспертным путем, по результатам оценивания результатов тестов разработчиками ПО.

μ'^i_j - степень принадлежности теста каждому из подмножеств множества «невыполнение», *i* – порядковый номер теста, *j* – номер нечеткого подмножества (1 – «полностью не выполнен», 2 - «частично выполнен», 3 - «больше не выполнен, чем выполнен»), $\mu'^i_j \in [0,1]$. Эти степени также определяются экспертным путем, по результатам оценки результатов тестов разработчиками ПО.

Для определения соответствия каждого теста соответствующему требованию μ^i воспользуемся правилом разницы нечетких множеств [2]:

$$\mu^i = \mu_1^i \wedge (1 - (\mu_2^i \wedge (1 - \mu_3^i))), \mu^i \in [0,1] \quad (1)$$

Аналогичным образом найдем значение общего несоответствия каждого теста конкретному требованию:

$$\mu'^i = \mu'^i_1 \wedge (1 - (\mu'^i_2 \wedge (1 - \mu'^i_3))), \mu'^i \in [0,1] \quad (2)$$

Результаты тестирования не являются взаимозависимыми и не могут компенсировать худшие значения одних тестов лучшими значениями других. Поэтому общее соответствие разработанного ПО требованиям *M* найдем по

формуле аддитивного критерия и разницы множеств «выполнение» и «невыполнение»:

$$M = \frac{\mu^1 + \mu^2 + \ldots + \ldots + \mu^N}{N} \wedge \left(1 - \frac{\mu'^1 + \mu'^2 + \ldots + \ldots + \mu'^N}{N}\right) \qquad (3)$$

Предложенный метод позволяет определить соответствие разработанного ПО критического применения функциональным и нефункциональным требованиям, учитывая, насколько именно удачно или неудачно реализовано то или иное требование. Однако ему присущ ряд существенных недостатков.

Во-первых, при его использовании могут игнорироваться единичные, но важные факты, которые не вписываются в предложенные формулы.

Во-вторых, математический аппарат предоставляет лишь приближенные расчеты соответствия, не учитывая нелинейные зависимости между входными данными задачи оценивания соответствия результатов тестирования ПО требованиям и ее исходным результатом. Поэтому, для повышения качества оценивания соответствия результатов тестирования ПО критического применения требованиям целесообразно будет на основе приведенного метода построить нечеткую нейронную сеть для решения указанной задачи.

Структура нечеткой нейросети для оценивания соответствия ПО критического применения требованиям к нему изображена на рис.1

Она состоит из трех слоев. Количество входов равно числу одновременно оцениваемых тестов к ПО, максимально возможным количеством одновременно оцениваемых тестов было выбрано 50. Количество выходов - один.

Для каждого из входных параметров сформирована шкала соответствия результатов тестирования ПО критического применения требованиям (рис. 2) с использованием подмножеств из нечетких множеств «выполнение» и «невыполнение».

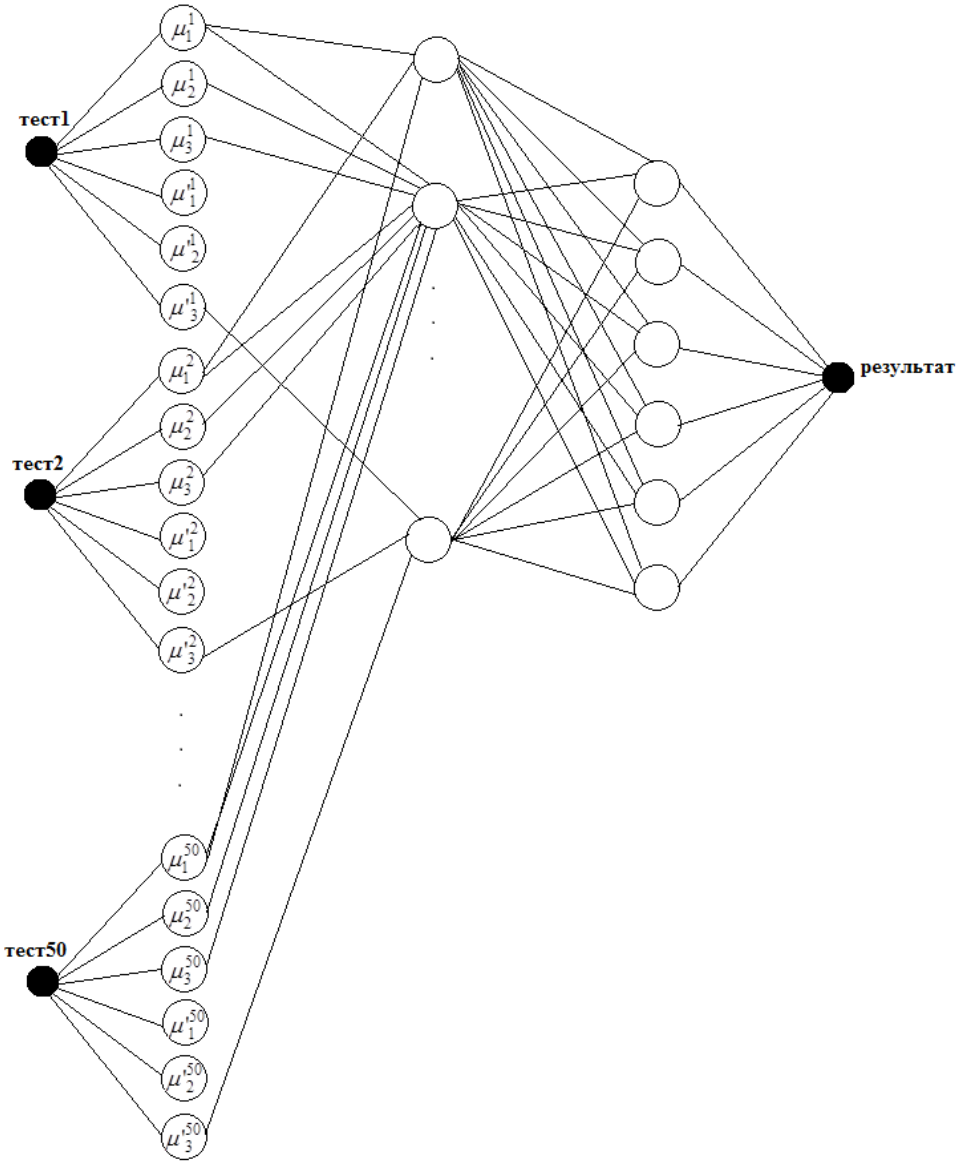

Рисунок 1 - Структура нечеткой нейросети для оценивания соответствия ПО критического применения требованиям

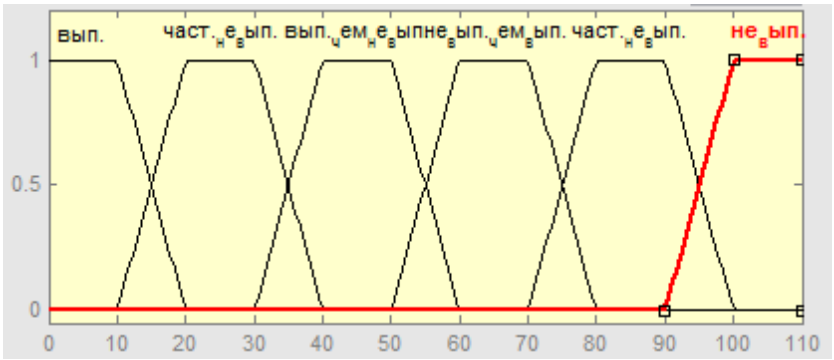

Рисунок 2 - Функции принадлежности результатов каждого теста

Нейроны второго слоя определяют степени истинности для каждого из последующих 1800 правил:

1. If (*тест1* is *вып.*) and (*тест2* is *вып.*) and (*тест3* is *вып.*) ... and (*тест50* is *вып.*) then (*результат* is *вып*.).
2. If (*тест1* is *част_не_вып.*) and (*тест2* is *вып.*) and (*тест3* is *вып.*) ... and (*тест50* is *вып*.) then (*результат* is *част_не_вып*).
3. If (*тест1* is *вып.*) and (*тест2* is *част_не_вып.*) and (*тест3* is *вып.*) ... and (*тест50* is *вып*.) then (*результат* is *част_не_вып*).
4. If (*тест1* is *вып.*) and (*тест2* is *вып.*) and (*тест3* is *част_не_вып.*) ... and (*тест50* is *вып*.) then (*результат* is *част_не_вып.*).

.

.

1800. If (*тест1* is *не_вып*.) and (*тест2* is *не_вып*) and (*тест3* is *не_вып*) ... and (*тест50* is *не_вып*) then (*результат* is *не_вып*).

Нейрон третьего слоя определяет принадлежность общего результата тестирования к подмножествам нечетких множеств «выполнение» и «невыполнение» (рис. 3):

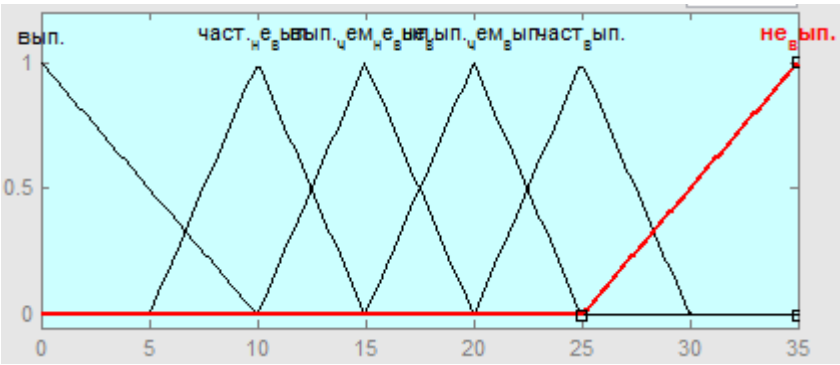

Рисунок 3 - Функции принадлежности результата общего тестирования

Нечеткая нейронная сеть для оценивания соответствия результатов тестирования ПО критического применения требованиям была построена при использовании прикладного пакета Fuzzy Logic Toolbox программы Matlab. Результаты работы построенной нейросети представлены в виде поверхности отклика (рис. 4).

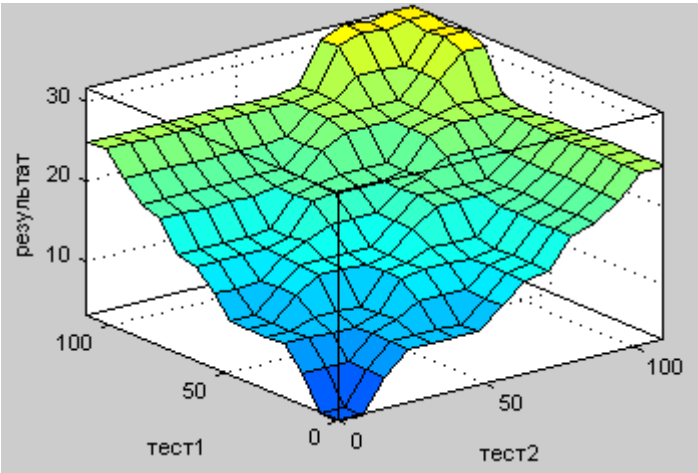

Рисунок 4 - Результаты работы нейросети

Выводы. Известные средства оценивания соответствия результатов тестирования ПО критического применения функциональным и нефункциональным требованиям фактически сводятся к выводам о том, удачно или неудачно были выполнены тесты на соответствие, не учитывая насколько именно удачно или неудачно.

Предложенный в статье метод на основе нечеткой нейронной сети позволяет устранить этот недостаток путем замены нечисловых результатов прохождения тестов числовыми значениями их принадлежности к соответствующим нечетким множествам «выполнение» и «невыполнение» и определить насколько именно разработанное ПО не соответствует требованиям к нему.

Литература

1. IEEE - 2004 IEEE Guide to the Software Engineering Body of Knowledge (Руководство к перечню знаний по программной инженерии).
2. Круглов Владимир Васильевич и др. нечеткая логика и искусственные нейронные сети. / / Круглов В.В., Дли М.И., Голунов Р. Ю. - Москва: Физматлит. - 2001. - 224с.