УДК 004.415.538

Е.В. ТИХОНОВА, О.М. ЯВОРСКАЯ, В.В. БЕРЕЗОВСКИЙ
Одесская национальная академия связи им. А.С. Попова

ВЕРИФИКАЦИЯ АЛГОРИТМА МАКСИМАЛЬНОГО ПОТОКА ДЛЯ ТРЕХПОЛЮСНОГО ГРАФА СЕТИ

В работе рассматриваются некоторые вопросы верификации программного алгоритма для нахождения максимального потока в открытом трехполюсном взвешенном графе телекоммуникационной сети со свободно ориентированными ребрами. Данный алгоритм впервые предложен на кафедре сетей связи Одесской национальной академии связи им. А.С. Попова. Особенностью данного алгоритма вычисления максимального потока является его многоуровневая иерархическая структура, которая начинается от простейшего трехполюсного примитива с одним узлом и тремя внешними связями, получившим название «триджет». Для тестирования алгоритма максимального потока в трехполюсной сети построено древовидное пространство состояний на основе классов фон Неймана. Каждый класс состояний алгоритма определяет неограниченное подмножество реализаций полно-связного графа общего типа с заданным количеством вершин. Для каждого класса состояний выбран эталонный представитель класса, для которого задача о максимальном потоке имеет аналитическое решение. На основании классов состояний алгоритма построена тестовая таблица, по которой был проверен программный алгоритм расчета.

Ключевые слова: трехполюсный граф, максимальный поток, верификация алгоритма, классы состояний.

O.V. TYKHONOVA, O.M. YAVORSCKAYA, V.V. BEREZOVSKIY
O.S. Popov Odessa National Academy of Telecommunications

VERIFICATION OF THE MAXFLOW ALGORITHM FOR A THREE-POLE NETWORK GRAPH

The paper discusses some issues of verifying a software algorithm for finding the maximum flow in an open three-pole weighted graph of a telecommunications network with free-oriented edges. This algorithm developed at the Telecommunication Networks Department of A.S. Popov Odessa National Academy of Telecommunications. The purpose of this paper is constructing a methodology for verifying the algorithm of maximum flow problem solving on a three-pole network graph with an arbitrary topology and six nodes. In this paper, we analyzed the features of various methods for verifying algorithms and testing applied software. A general scheme of the algorithm for solving the maximum flow problem on a three-pole weighted fully connected graph with six vertices is considered. The peculiarity of this algorithm for calculating the maximum flow is its multilevel hierarchical structure, which starts from the simplest three-pole primitive with one node and three external links, called the "triget". For the algorithm under consideration, a three-level state space is constructed on the basis of the von Neumann classes. Each class of states of the algorithm defines an unlimited subset of realizations on a graph of general type with a given number of vertices. For each class of states, a representative of the class is chosen for which the maximum flow task has an analytic solution. Based on the state classes of the algorithm, a test table is constructed, according to which the software algorithm of calculation has been verified. Verification and testing of complex software algorithms is a non-trivial scientific and engineering task that is difficult to automate, and in many cases requires an individual approach. In this paper, a methodology constructed for verifying the algorithm of maximum flow problem solving for a telecommunication network, which is represented by an open three-pole weighted graph with free- oriented edges.

Keywords: Maxflow problem, free-oriented graph, multi pole telecom network, dynamic channel configuration.

Введение. Постановка задачи. Верификация программного обеспечения является актуальной и достаточно сложной задачей в области разработки интерфейсов прикладного программирования. В данной работе рассматривается задача тестирования программного алгоритма для нахождения максимального потока на трехполюсном открытом взвешенном графе сети со свободно ориентированными ребрами; этот алгоритм разработан на кафедре сетей связи Одесской национальной академии связи, [1]. В отличие от известных алгоритмов для нахождения максимального потока на графе сети, данный алгоритм находит максимально возможный суммарный поток между тремя парами открытых полюсов на графе сети. При этом, данный поток не зависит от направления отдельных составляющих любого потока, а также от распределения весов каждого ребра графа в прямом и обратном направлении. В основе алгоритма лежат три положения:

а) аналитически полученное решение задачи о максимальном потоке для простейшего трехполюсного графа сети с одним узлом и тремя открытыми свободно ориентированными ребрами (этот граф назван «триджет») [2];

б) принцип минимального блокирующего потока между всеми парами полюсов; согласно этому принципу, в первую очередь, анализируются независимые потоки между всеми парами полюсов, которые протекают по кратчайшим путям в метрике переходов по ребрам (hops);

в) принцип сведения сложной задачи на общем графе к решению нескольких простых задач на примитивных графах, для которых ране получены т.н. элементарные алгоритмы (cellular algorithms).

На основе этих положений, в работе [3] получен алгоритм нахождения максимального потока на полно-связном графе из трех узлов с тремя открытыми ребрами (названного «триплет»). Алгоритмы решения задачи о максимальном потоке для двух примитивных трехполюсных графов (типа «триджет» и «триплет») лежат в основе более общего алгоритма для отыскания максимального потока на графе общего вида из шести вершин.

В силу того, что два из трех названных выше базовых положений алгоритма носят эвристический характер, в целом, алгоритм отыскания максимального потока на трехполюсном графе пока еще не имеет исчерпывающе строгого доказательства для общего случая. Такая ситуация характерна для многих эмпирически полученных решений сложных задач (например, компьютерные программы для игры в шахматы). Поэтому методика тестирования и обоснования корректности таких алгоритмов является важной составляющей для метода решения содержательной задачи.

Целью данной работы является построение методики для верификации алгоритма решения задачи о максимальном потоке на трехполюсном графе сети с произвольной топологией и шестью узлами.

Особенности известных методов верификации программного обеспечения

Ключевым вопросом в области верификации алгоритмов и их программных реализаций является степень полноты для процедур тестирования [4]. Опыт показывает, что сложные программные продукты, как правило, не могут быть исчерпывающе протестированы в силу огромного числа возможных состояний входных данных [5]. Существуют различные подходы и методики верификации программных алгоритмов, например, статистический анализ области определения алгоритма методом Монте-Карло [6], построение множества возможных сценариев для входных данных (scenario based software architecture evaluation, [7]), динамические способы мониторинга и тестирования [8] и др. В области тестирования программного обеспечения используются понятия «альфа-тестирование» и «бета-тестирование». Альфа-тестирование является первым этапом предварительного тестирования программного продукта силами самих разработчиков. Бета-тестирование является вторым этапом, и осуществляется пользователями программного продукта.

Достаточно подробное изложение различных подходов к верификации программного обеспечения дано в работах [9, 10]. В частности, в [9] рассматривается методика на основе специально разработанных сценариев и критериев адекватности тестирования (test adequacy criteria). Различные сценарии включают в себя классы эквивалентности состояний алгоритма. Известны различные критерии полноты, однако абсолютно «полное» тестирование невозможно для практически значимых систем. При подготовке тестирования формируется некоторый набор тестов, а отладка программного кода с помощью тестов служит для устранения различного рода ошибок в текстах программ (как синтаксических, так и семантических). По результатам отладки может потребоваться расширение набора тестов.

В работе [10] отмечается, что разработка плана тестирования программного продукта начинается с построения набора тестовых примеров, которые отражают наиболее характерные сценарии реальных условий функционирования алгоритма. Основными приемами такого тестирования являются методы «черного ящика» (Black box testing), «белого ящика» (White Box testing) и серого ящика» (Grey Box testing), [11]. Генерация набора сценариев на практике является одним из наиболее эффективных методов верификации программ; но в то же время, он труднее всего поддается автоматизации и формализации, поскольку предполагает сложный логический анализ ситуативного поведения алгоритмов. Такой анализ, по-прежнему, является исключительной прерогативой человеческого интеллекта. Иными словами, создание методики тестирования для достаточно сложных программных алгоритмов является нетривиальной интеллектуальной задачей (искусством инженера-математика), которую необходимо решать в каждом конкретном случае [12].

Общая схема решения задачи о максимальном потоке для трехполюсного графа

Среди открытых публикаций о максимальном потоке в сети авторам неизвестны исследования открытых графов с числом полюсов больше чем два и со свободно ориентированными ребрами (кроме указанных выше работ [1–3]). Большинство алгоритмов о максимальном потоке рассматриваются на ориентированном графе с двумя специальными вершинами (исток S и сток T). Известны также обобщения задачи о максимальном потоке на ориентированном двудольном графе со многими истоками и многими стоками. Последний случай легко сводится к графу с одним истоком и одним стоком. Такого рода модели сетей достаточно адекватно описывают традиционные логистические системы и транспортные потоки, в которых конкретный продукт имеет вполне определенное направление перемещения. Например, едва ли имел бы практическое применение газопровод, в котором газ одновременно движется во встречных направлениях.

Вместе с тем, современные телекоммуникационные транспортные сети имеют двунаправленные информационные потоки в каждом канале. При этом емкость канала может динамически изменяться в прямом и обратном направлении в зависимости от характера нагрузки, а число открытых полюсов может быть больше двух. Ввиду отсутствия накопленного опыта исследования подобного рода задач, общая схема нахождения максимального потока в многополюсной сети сформулирована по принципу постепенного перехода от сравнительно простых моделей сетей к все более сложным сетевым структурам. На первом этапе была поставлена задача исследования трехполюсной открытой сети произвольной топологии с ограниченным количеством узлов (до шести). Решение данной задачи позволяет добавить еще один примитивный граф («фрактал») к полученному ранее набору из двух типовых примитивов («триджет» и «триплет»).

Общее количество ребер на трехполюсном графе с шестью вершинами равно 18 (15 внутренних ребер и три открытых внешних ребра на полюсах графа); граф такого типа обозначен как фрактал, рис.1. Общий поток на трехполюсном графе определен как сумма всех возможных потоков, протекающих между тремя парами полюсов (вершины графа с номерами 1, 2, 3) через открытые (внешние) ребра a_1, a_2, a_3.

Полные потоки между тремя парами полюсов обозначены на рис.1 как f_1, f_2, f_3. При этом суммарный поток равен $f_\Sigma = f_1 + f_2 + f_3$.

Согласно приведенному выше принципу минимального блокирующего потока, все потоки фрактала разделены на классы иерархии (классы фон Неймана, [13]). Потоки нулевого уровня иерархии протекают по

путям, в которых количество внутренних ребер равно нулю (т.е. потоки через два внешних ребра простейшего открытого трехполюсного графа с одной вершиной, названного «триджет», рис. 2).

Несмотря на столь простую структуру графа типа «триджет», решение задачи о максимальном потоке для этого графа не является тривиальным и очевидным. Например, для значений $a_1 = 4$, $a_2 = 5$, $a_3 = 6$, максимально возможный суммарный поток равен 7.5 (точный алгоритм расчета максимального потока для такого графа доказан дедуктивно в [2]). Получить это значение (вместе с конкретизацией трех составляющих этого потока) достаточно непросто. Данный факт неоднократно проверен на контрольных задачах для студентов ОНАС им. А.С. Попова в курсе «Моделирование сетей и систем». В данном случае, составляющие максимального потока равны: $f_1 = 3.5$, $f_2 = 2.5$, $f_3 = 1.5$. При этом

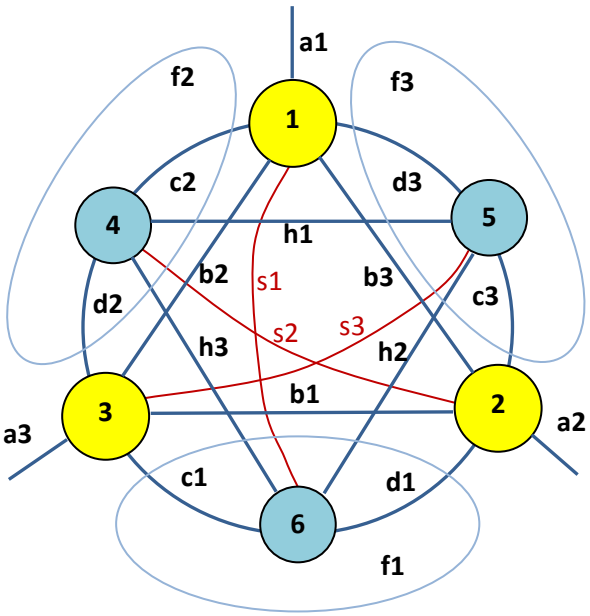

Рис. 1. Открытый полно-связный трехполюсный граф из шести вершин (фрактал)

по каждому из трех открытых ребер протекает суммарный поток, в точности равный весу ребра:

$$f_{a1} = f_2 + f_3 = 2.5 + 1.5 = 4;\ f_{a2} = f_1 + f_3 = 3.5 + 1.5 = 5;\ f_{a3} = f_1 + f_2 = 3.5 + 2.5 = 6.$$

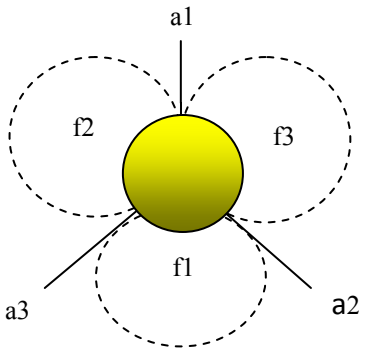

Рис. 2. Открытый простейший трехполюсный граф (триджет)

Отметим, что в традиционной постановке (для графа с одним истоком и одним стоком) данная задача не может быть решена известными алгоритмами.

Потоки первого уровня протекают по путям, содержащим одно внутренне ребро. Типовой трехполюсный граф такого типа (триплет) показан на рис.3. Для графа типа «триплет» также получено достаточно строгое алгоритмическое решение задачи о максимальном потоке, в основе которого лежит промежуточное вспомогательное преобразование триплета к схеме, эквивалентной графу «триджет» ([2, 3]).

Алгоритм решения задачи о максимальном потоке на графе типа «фрактал» (рис. 1) разделяется на две основных части.

Первая часть алгоритма. Фрактал приводится к промежуточной форме внутреннего треугольника в составе триплета на рис. 3. Для этого веса открытых ребер фрактала на рис. 1 принимаются сколь угодно большими (т.е. такими, которые не ограничивают потоки во фрактале). Поэтому потоки между всеми парами полюсов зависят только от внутренних ребер фрактала. После расчёта всех возможных потоков, фрактал легко приводится к схеме треугольника, ребра которого имеют веса, равные соответствующим промежуточным потокам фрактала $f1, f2, f3$.

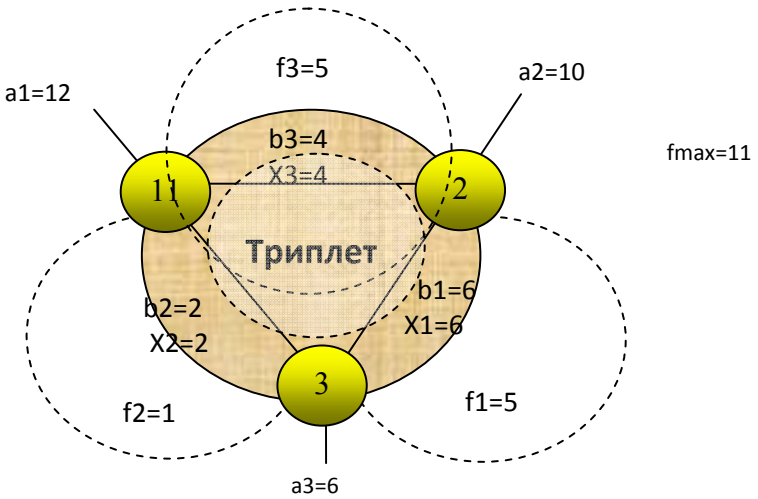

Рис. 3. Открытый трехполюсный граф типа "триплет"

Вторая часть алгоритма. Из внешних ребер фрактала и эквивалентного треугольника внутренней проводимости между парами полюсов (с весами $f1, f2, f3$) строится триплет, для которого имеется процедура вычисления максимального потока; данный поток и есть окончательный максимальный поток всего фрактала.

Таким образом, основные усилия по созданию и верификации алгоритма максимального потока для фрактала касаются вопроса построения эквивалентного треугольника проводимости фрактала между парами

полюсов (т.е. к расчету трех промежуточных потоков *f*1, *f*2, *f*3). Данный алгоритм строится для усеченного фрактала с 15 внутренними ребрами (внешние ребра не принимаются во внимание). Эта часть программного алгоритма названа **Triangle**.

Построение дерева сценариев. Классы состояний алгоритма. Множество различных сценариев работы алгоритма для расчета максимального потока на трехполюсном графе типа «усеченный фрактал» (без учета влияния открытых внешних ребер) разбивается на иерархические классы (известные как «классы фон Неймана», [13]). Структура этих классов является строго древовидной, и подчинена упомянутому выше эвристическому принципу минимального блокирующего потока на каждом шаге алгоритма.

Сценарий первого уровня: вычисление потоков типа «*x*» по кратчайшим (одношаговым) путям между полюсами; таких путей три (по ребрам *b*1, *b*2, *b*3 на рис. 1). Эти составляющие потоков в алгоритме обозначены индексированной переменной *x*1, *x*2, *x*3, т.е. *x*1=*b*1; . *x*2=*b*2; . *x*3=*b*3. После реализации данного сценария три ребра *b*1, *b*2, *b*3 на рис.1 получают нулевые остаточные веса и в дальнейших сценариях не участвуют. В остатке от усеченного фрактала на рис.1 остаются 15−3=12 ребер.

Сценарий второго уровня: вычисление потоков типа «*y*», каждый из которых протекает по двум внутренним ребрам. Если в качестве таких путей взять пути по кольцу (между каждой парой полюсов есть пары независимых ребер на кольце), то после вычисления потоков из шести ребер как минимум три получат нулевые остаточные веса, и возможно три остальных будут иметь ненулевые веса. Такой вариант не удовлетворяет принципу минимального блокирующего потока. Действительно, если в качестве двушаговых путей рассмотреть три независимых элементарных графа типа «триджет (с центрами в вершинах 4, 5, 6 на рис.1), то из 9 ребер в остатке могут быть не более 3 ребер; степень блокировки ребер во втором случае меньше: 3/6>3/9.

Сценарии первых двух уровней являются линейными (в них не используются операторы *if*).

В сценарии второго уровня для каждой пары полюсов вычисляются три составляющие из общего максимального потока:

$$\begin{cases} Y1 = y11 + y12 + y13, \\ Y2 = y21 + y22 + y23, \\ Y3 = y31 + y32 + y33, \end{cases} \quad (1)$$

где y11 – составляющая потока *f1* между полюсами 2 и 3, которая протекает по ребрам c1, d1 первого триджета (вершина 6, ребра c1, d1, s1);

y12 – составляющая потока *f1* между полюсами 2 и 3, которая протекает по ребрам s2, d2 второго триджета (вершина 4, ребра c2, d2, s2);

y13 – составляющая потока *f1* между полюсами 2 и 3, которая протекает по ребрам c3, s3 третьего триджета (вершина 5, ребра c3, d3, s3);

y21 – составляющая потока *f2* между полюсами 1 и 3, которая протекает по ребрам c1, s1 первого триджета (вершина 6, ребра c1, d1, s1);

y22 – составляющая потока *f2* между полюсами 1 и 3, которая протекает по ребрам c2, d2 второго триджета (вершина 4, ребра c2, d2, s2);

y23 – составляющая потока *f2* между полюсами 1 и 3, которая протекает по ребрам d3, s3 третьего триджета (вершина 5, ребра c3, d3, s3);

y31 – составляющая потока *f3* между полюсами 1 и 2, которая протекает по ребрам s1, d1 первого триджета (вершина 6, ребра c1, d1, s1);

y32 – составляющая потока *f3* между полюсами 1 и 2, которая протекает по ребрам s2, c2 второго триджета (вершина 4, ребра c2, d2, s2);

y33 – составляющая потока *f3* между полюсами 1 и 2, которая протекает по ребрам c3, d3 третьего триджета (вершина 5, ребра c3, d3, s3).

Сценарии третьего уровня: вычисление потоков, которые проходят по трем внутренним ребрам. Общее количество ребер фрактала на рис.1, которые могут остаться с ненулевыми весами после рассмотрения сценариев первого и второго уровней, равно 15−3−6=6. Однако точное расположение ненулевых остаточных ребер заранее неизвестно, поскольку оно зависит от конкретных значений исходных данных (начальных весов всех 15 ребер). В программном алгоритме Triangle выделены следующие четыре класса сценариев:

3.1. На главном кольце фрактала (рис.1) остались три ребра. Внутри этого сценария могут быть следующие восемь случаев:

(c1, c2, c3); (c1, c2, d3); (c1, d2, c3); (c1, d2, d3); (d1, c2, c3); (d1, c2, d3); (d1, d2, c3); (d1, d2, d3).

3.2. На главном кольце фрактала (рис.1) остались два ребра. Внутри этого сценария могут быть следующие восемь случаев:

(c1, c2); (c1, c3); (c1, d2); (c1, d3); (d1, c2); (d1, c3); (d1, d2); (d1, d3).

3.3. На главном кольце фрактала (рис. 1) осталось одно ребро. Внутри этого сценария могут быть следующие шесть случаев: (c1); (c2); (c3); (d1); (d2); (d3).

3.4. На главном кольце фрактала (рис. 1) не осталось ни одного ребра. Это означает, что могли остаться ребра s1, s2, s3, а также h1, h2, h3. Нетрудно увидеть, что этот сценарий приводится к типовому структурному примитиву «триплет» (см. рис. 3). Треугольник этого триплета образован вершинами 4, 5, 6 и

ребрами h1, h2, h3; роль открытых ребер триплета играют ребра s1, s2, s3.

Таблица 1

Таблица тестирования для вариантов сценария третьего уровня

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
c1	5	5	5	5	1	1	1	1	6	6	6	6	1	1	1	1	1
c2	5	5	1	1	5	5	1	1	6	1	1	1	6	1	1	1	1
c3	5	1	5	1	5	1	5	1	1	6	1	1	1	6	1	1	1
d1	1	1	1	1	5	5	5	5	1	1	1	1	6	6	6	6	1
d2	1	1	5	5	1	1	5	5	1	1	6	1	1	1	6	1	1
d3	1	5	1	5	1	5	1	5	1	1	1	6	1	1	1	6	1
h1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7
h2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7
h3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7
s1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7
s2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7
s3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7
Расчетные (тестовые) приращения потоков																	
z1	1																
z2	1																
z3	0																
Фактические приращения потоков																	
z1	1																
z2	1																
z3	0																

Таким образом, общее число различных случаев по сценарию 3-го уровня составляет 8+8+1=17. Сценарии четвертого уровня (потоки, протекающие через 4 внутренние ребра фрактала на рис.1) строятся аналогичным образом. В алгоритме Triangle доказано логически, что в результате работы данного алгоритма не могут остаться никакие сценарии с путями длиной в пять ребер. Исходя из приведенной выше классификации сценариев работы алгоритма Triangle, построена таблица тестирования (табл. 1). В этой таблице веса тех ребер, которые должны остаться в сценарии третьего уровня, приняты значительно больше всех остальных (равных единице). Каждый из данных вариантов приводится либо к триджету, либо к последовательно-параллельному двухполюсному графу, для которого несложно посчитать вручную приращение потоков между полюсами. Так, например, первый вариант приводится к триджету, в котором приращения потоков равны: z1=1; z2=1; z3=0. Описанная выше методика была использована для верификации программного кода в алгоритме вычисления максимального потока в открытой трехполюсной телекоммуникационной сети.

Выводы. Верификация и тестирование сложных программных алгоритмов является нетривиальной научно-инженерной задачей, которая трудно поддается автоматизации, и во многих случаях требует индивидуального подхода. В данной работе построена методика верификации программного кода в алгоритме вычисления максимального потока в телекоммуникационной сети, которая представлена открытым трехполюсным взвешенным графом со свободно ориентированными ребрами.

Литература

1. Tykhonova O.V. The principles of maxflow task study for multi-pole software defined network / O.V. Tykhonova // Вимірювальна та обчислювальна техніка в технологічних процесах. – 2017. – № 2. – С. 159–163.

2. Tykhonova O.V. The maxflow problem analysis on free-oriented network graph / O.V. Tykhonova // Вимірювальна та обчислювальна техніка в технологічних процесах. – 2018. – № 1. – С. 139–143.

3. Тихонова О.В. Оптимізація потоків у багатополюсній мережі / О.В. Тихонова // Матеріали XI міжнарожної науково-практичної конференції “Інтернет, Освіта, Наука” (Вінниця, 22–25 травня, 2018). – В. : ВНТУ, 2018. – С. 70–72.

4. Куликов С.С. Тестирование программного обеспечения. Базовый курс / С.С. Куликов. – 2017. – 2-е изд. – 296 с.

5. Software systems verification. Volume 4. – 2016. – Retrieved from : https://ww2.eagle.org/content/dam/eagle/rules-and-guides/current/other/253_cybersafetyV4/CyberSafety_V4_SSV_Guide_e.pdf.

6. Singh H. Software Reliability Testing using Monte Carlo Methods / H. Singh, P. Pal // International Journal of Computer Applications. – Nr. 4. – 2013. – P. 41–44.

7. Scenario-Based Software Architecture Evaluation Methods: An Overview / M.T. Ionita, D.K. Hammer, H.Ob bink. – Retrieved from: http://www.win.tue.nl/oas/architecting/aimes/papers/Scenario-Based%20SWA%20Evaluation%20Methods.pdf.

8. Hedaoo A.H. Study of Dynamic Testing Techniques / A.H. Hedaoo, A. Khandelwal // International Journal of Advanced Research in Computer Science and Software Engineering. – Vol. 7, Issue 4. – 2017. – P. 322–330.

9. Кулямин В.В. Методы верификации программного обеспечения [Электронный ресурс] / Кулямин В.В. – 2008. – Режим доступа : http://www.ict.edu.ru/ft/005645/62322e1-st09.pdf .

10. Jamil M. Software Testing Techniques: A Literature Review / M. Jamil, M. Arif, N. Abubakar, A. Ahmad // 2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M). – 2016. – p. 177–182.

11. Babbar H. Software testing: techniques and test cases / H. Babbar // International journal of research in computer applications and robotics. – Vol.5, Issue 3. – 2017. – p. 44–53.

12. Paulasaari M.Tools for Code Quality in Front-end Software Development. – 2018. – Retrieved from : https://www.theseus.fi/bitstream/handle/10024/143272/Mika%20Paulasaari%20-%20IT%20Masters%20Thesis%20-%202018.pdf?sequence=1.

13. Von Neumann J. Die Axiomatisierung der Mengenlehre / J. von Neumann // Mathematische Zeitschrift. – 1928. – № 27. – s. 669–752.

References

1. Tykhonova O.V. The principles of maxflow task study for multi-pole software defined network / O.V. Tykhonova // Measuring and Computing Devices in Technological Processes. – 2017. – N. 2. – P. 159-163.

2. Tykhonova O.V. The maxflow problem analysis on free-oriented network graph / O.V. Tykhonova // Measuring and Computing Devices in Technological Processes. – 2018. – N. 1. – P. 139-143.

3. Tykhonova O.V. Optymizacija potokiv u baghatopoljusnij merezhi / O.V. Tykhonova // proceedings of the XI international scientific-practical conference "Internet-Education-Science" (Vinnytsia, May 22-25, 2018). – V.: VNTU, 2018. – p. 70-72.

4. Kulikov S.S. Testirovanie programmnogo obespecheniya. Bazovyiy kurs / S.S. Kulikov. –2017. – 2nd ed.. – 296 p.

5. Software systems verification. Volume 4. – 2016. – Retrieved from : https://ww2.eagle.org/content/dam/eagle/rules-and-guides/current/other/253_cybersafetyV4/CyberSafety_V4_SSV_Guide_e.pdf.

6. Singh H. Software Reliability Testing using Monte Carlo Methods / H. Singh, P. Pal // International Journal of Computer Applications. – Nr. 4. – 2013. – P. 41–44.

7. Scenario-Based Software Architecture Evaluation Methods: An Overview / M.T. Ionita, D.K. Hammer, H.Ob bink. – Retrieved from: http://www.win.tue.nl/oas/architecting/aimes/papers/Scenario-Based%20SWA%20Evaluation%20Methods.pdf.

8. Hedaoo A.H. Study of Dynamic Testing Techniques / A.H. Hedaoo, A. Khandelwal // International Journal of Advanced Research in Computer Science and Software Engineering. – Vol. 7, Issue 4. – 2017. – P. 322–330.

9. Kulyamin V.V. Metodyi verifikatsii programmnogo obespecheniya. – 2008. – Retrieved from: http://www.ict.edu.ru/ft/005645/62322e1-st09.pdf .

10. Jamil M. Software Testing Techniques: A Literature Review / M. Jamil, M. Arif, N. Abubakar, A. Ahmad // 2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M). – 2016. – p. 177–182.

11. Babbar H. Software testing: techniques and test cases / H. Babbar // International journal of research in computer applications and robotics. – Vol.5, Issue 3. – 2017. – p. 44–53.

12. Paulasaari M.Tools for Code Quality in Front-end Software Development. – 2018. – Retrieved from : https://www.theseus.fi/bitstream/handle/10024/143272/Mika%20Paulasaari%20-%20IT%20Masters%20Thesis%20-%202018.pdf?sequence=1.

13. Von Neumann J. Die Axiomatisierung der Mengenlehre / J. von Neumann // Mathematische Zeitschrift. – 1928. – № 27. – s. 669–752.

Рецензія/Peer review : 9.9.2018 р. Надрукована/Printed : 19.9.2018 р.
Рецензент: д.т.н., проф. Бондаренко О.В.