ISSN 1992-6502 (Print) Вестник УГАТУ ISSN 2225-2789 (Online)
2026. Т. 30, № 2 (112). С. 49-63 http://journal.uust.ru/index.php/Vestnik

УДК: 004.41 doi 10.54708/19926502_2026_30211249
Код ГРНТИ: 55.05.19

Методика селективного тестирования CLI-интерфейсов на основе графовых моделей и минимизации информационной энтропии

Э.М. Чембарисов*, О.Н. Сметанина, Е.Ю. Сазонова

ФГБОУ ВО «Уфимский университет науки и технологий» (УУНиТ), г. Уфа, Россия

Аннотация. В статье рассматриваются вопросы оптимизации тестового покрытия интерфейсов командной строки (CLI) с учетом сложных спецификаций и ограниченных ресурсов. Применение традиционных методов тестирования часто не учитывает реальные сценарии эксплуатации, что может приводить к неэффективному использованию ресурсов для покрытия тестами редко встречаемого функционала.
Авторами рассмотрено современное состояние проблемы и приведена предложенная теоретико-множественная модель, учитывающая топологические свойства графа спецификации, алгоритмические и информационные компоненты. Множество тестовых сценариев формируется методами топологического поиска в ширину и глубину (BFS и DFS). Важной особенностью модели является включение в нее параметра осведомленности об использовании (Usage Awareness), который на основе динамического трекинга вызовов сопоставляет узлам графа веса осведомленности. Процесс выбора оптимального набора тестов формализован через функционал информационной энтропии Шеннона, где селективная стратегия на каждой итерации максимизирует снятие неопределенности относительно эксплуатационного состояния системы. Введено понятие среднего взвешенного покрытия и метрика относительного прироста осведомленности. Предложенная модель обеспечивает переход от структурного покрытия кода к покрытию «ценности» для пользователя.

Ключевые слова: *CLI, жадный алгоритм, граф спецификации, BFS, DFS, Usage Awareness, средневзвешенное покрытие, динамический трекинг.*

***lawluker@gmail.com**

Введение

Проблема неполного покрытия CLI-интерфейсов и использование динамического анализа для ее решения активно исследуются в области системного программирования и информационной безопасности. Среди наиболее популярных направлений следует отметить тестирование на основе обратной связи, символьное исполнение, анализ графа зависимостей опций и другие. Применение динамического анализа превращает CLI-тестирование в управляемый процесс исследования кода, где целью является покрытие не только синтаксиса (строки ввода), но и семантики (логики обработки этих данных внутри программы).

Вопросам CLI-тестирования посвятили свои исследования многие российские и зарубежные специалисты: Wang T., Wei T., Gu G., Ganesh V. [1], Shelke S., Patil S. [2], Вартанов С.П., Герасимов А.Ю. [3], Вишняков А.В., Кобрин И.А., Федотов А.Н. [4], Плаксин С.И. [5], Ерохина Н.С. [6], Грибалев Н.Г. [7], Godefroid P., Levin M., Molnar D. [8], Fioraldi A., D'Elia D.C., Coppa E. [9], Fioraldi A., Maier D., Zhang D., Crump A. [10], Cadar Cr., Dunbar D., Engler D. [11], Владимиров М.А. [12], Inozemtseva L., Holmes R. [13], Arcuri A., Briand L. [14] и др.

В работах перечисленных специалистов рассматривается множество аспектов от чисто теоретических до практической реализации механизмов: создание теоретической базы для динамического отслеживания меток [1, 3], позволяющей точно связать конкретную опцию командной строки с ветвлением в коде; решение задачи генерации векторов флагов с использованием генетических алгоритмов для максимизации покрытия путей [8, 10, 12]; создание механизмов, позволяющих инструментам анализа «понимать» структуру входных аргументов [1, 6, 8]; извлечение грамматики опций из справочных руководств с последующим динамическим тестированием [3, 6, 9]; генерации значений для флагов через динамическую

символьную интерпретацию [4, 11] и др. Обоснование того, что просто оценить степень покрытия недостаточно, рассмотрены авторами в статьях [13, 14].

Однако вопросам оценки средневзвешенного покрытия графа спецификаций CLI на основе данных динамического мониторинга не уделено достаточного внимания. Анализ CLI через призму графовой модели позволяет формализовать связи между командами, подкомандами и флагами, а также применять методы теории графов для вычисления метрик покрытия. Классический подсчет пройденных узлов графа не учитывает реальную ценность различных путей.

В работе исследуется проблема обеспечения полноты тестирования интерфейсов командной строки (CLI) методами динамического трекинга путей исполнения. Проведенный анализ современного состояния отрасли выявил существенный разрыв между формальными метриками покрытия кода и фактической надежностью критических бизнес-сценариев. Авторами предложена и обоснована методика средневзвешенного анализа графовых моделей CLI, в рамках которой узлам (командам и опциям) присваиваются весовые коэффициенты релевантности. Это позволяет переориентировать процесс верификации с экстенсивного перебора параметров на приоритизированную проверку функциональных доминантов системы. Теоретическую значимость работы составляет введение метрики на основе функционала информационной энтропии Шеннона, количественно описывающей стохастическую неопределенность эксплуатационного состояния интерфейса. Разработанная селективная стратегия генерации тестовых сценариев базируется на принципе максимизации информационного выигрыша на каждой итерации. Данный подход обеспечивает ускоренную минимизацию остаточной энтропии, позволяя эффективно верифицировать наиболее значимые пути исполнения при ограниченном объеме тестовой выборки.

Современное состояние проблемы

Современное программное обеспечение все чаще использует интерфейсы командной строки (CLI) для автоматизации задач, администрирования систем, обработки данных и интеграции с внешними сервисами. CLI-инструменты применяются как в промышленной разработке, так и в научных и исследовательских проектах. Каждая команда CLI, как правило, сопровождается множеством опций, параметров и флагов, выполняющих тонкую настройку поведения команды. Однако несмотря на важность корректного функционирования таких интерфейсов, в практике тестирования акцент часто делается исключительно на факте успешного вызова команды, в то время как проверка покрытия всех допустимых параметров и флагов остается вне фокуса. В аспекте анализа покрытия опций может быть применен метод, направленный на проверку использования всех доступных параметров, аргументов и их комбинаций (Рис. 1).

Базовые аспекты анализа покрытия опций	Назначение
• Проверка активации хотя бы в одном тесте каждого флага и аргумента; • Анализ взаимодействия опций друг с другом; • Тестирование опций с экстремальными значениями.	• Обнаружение аргументов, имеющихся в коде, но не влияющих на логику программмы; • Предотвращение ошибок некорректного ввода, способных, например, привести к переполнению буфера; • Уверенность в том, что программа корректно интерпретирует сложные структуры команд.

Рисунок 1. Основные аспекты метода тестирования программ, управляемых через интерфейс командной строки для анализа покрытия опций.

Недостаточное внимание к анализу используемых флагов может привести к ряду проблем. Во-первых, часть параметров остается не охваченной, от того их корректная оценка процессом тестирования не гарантируется. Во-вторых, в кодовой базе скапливаются устаревшие флаги, которые более не используются, но продолжают поддерживаться, повышая

когнитивную сложность интерфейса. В-третьих, неполнота покрытия возможных значений перечисляемых опций создает риск появления неочевидных ошибок при использовании редко встречающихся вариантов.

Для повышения прозрачности и объективности оценки покрытия параметров CLI-интерфейса в процессе тестирования, необходимо разработать подход, позволяющий автоматически определять, какие команды и флаги действительно используются, какие проверяются через assert-механизмы тестов, а какие остаются нетронутыми. Учитывая потенциал таких решений для повышения качества тестового покрытия и выявления устаревших или забытых опций, особую актуальность приобретает задача анализа CLI-покрытия с использованием автоматизированных средств.

Проблема автоматизации тестирования интерфейсов командной строки (CLI) и анализа покрытия опций рассматривается в мировой и отечественной литературе с различных позиций – от фундаментальной теории до прикладных инструментов (табл. 1).

Механизмы динамического отслеживания меток для установления прямой связи между входной опцией и логикой ветвления в коде заложены в работах Wang T., Wei T., Gu G. и Ganesh V. В российском сегменте развитие этого направления представлено в исследованиях Вартанова С.П. и Герасимова А.Ю., где динамический анализ используется для целенаправленной генерации векторов данных, способных достигать уязвимых участков кода.

Таблица 1. Аспекты и механизмы CLI-тестирования.

Аспект / Механизм	**Ключевые авторы**	**Реализация / Инструмент**
Динамическое отслеживание меток (Taint Tracking)	Wang T., Wei T., Gu G., Ganesh V. [1], Вартанов С. П., Герасимов А. Ю. [3]	TaintScope; связь конкретной опции с логикой ветвления в коде
Генетические алгоритмы и максимизация покрытия	Fioraldi A., Maier D. [10], Godefroid P., Levin M., Molnar D. [8]	AFL++/LibAFL/SAGE; эволюционная мутация векторов флагов
Символьное исполнение (вычисление аргументов)	Cadar C., Dunbar D. [11], Вишняков А. В., Кобрин И. А., Федотов А. Н. [4]	KLEE/Sydr; математический расчет значений для прохождения *if*-условий
«Понимание» структуры и грамматики ввода	Fioraldi A., Coppa E. [9], Ерохина Н. С. [6], Герасимов А. Ю. [3]	WEIZZ; извлечение структуры аргументов и мутация сложных форматов
Формализация и метрики успеха	Плаксин С. И. [5], Грибалев Н. Г. [7], Shelke S., Patil S. [2]	теоретическая база описания кода и KPI для оценки тестов

Задача максимизации покрытия путей исполнения при переборе конфигураций флагов решается с помощью эволюционных и генетических алгоритмов. Здесь ключевую роль играют работы Fioraldi A., Maier D. и коллектива разработчиков AFL++, предложивших механизмы мутации входных векторов на основе обратной связи. Параллельно с этим Godefroid P., Levin M. и Molnar D. обосновали эффективность фаззинга для глубокого исследования логики программ.

Для автоматического вычисления значений аргументов, необходимых для прохождения сложных условий в парсерах, применяются методы символьного исполнения. Фундаментальные основы этого подхода описаны Cadar C. и Dunbar D. (инструмент KLEE), а их современная адаптация для анализа бинарного кода представлена в работах Вишнякова А.В., Кобрина И.А. и Федотова А.Н.

Вопрос «понимания» структуры и грамматики аргументов решается через механизмы анализа структурированных бинарных форматов (Fioraldi A., Coppa E.) и методы мутации

сложно-структурированных входных данных, предложенные Ерохиной Н.С. Теоретический базис для формального описания поведения программ и оценки полноты тестирования через метрики покрытия дополняется работами Плаксина С.И., Грибалева Н.Г., а также обзорными исследованиями Shelke S. и Patil S.

Несмотря на успехи в автоматизации тестирования интерфейсов командной строки (CLI) и анализа покрытия опций, существующие подходы ориентированы в большей части на покрытие структуры программы и выявление ошибок, не учитывая при этом бизнес-значимость функциональных узлов (веса опций) и реальные паттерны эксплуатации, что приводит к избыточности тестовых выборок и сохранению высокой информационной неопределенности в наиболее критичных для пользователя сценариях.

Необходимость перехода от формального перебора опций к анализу наиболее значимых сценариев приводит к необходимости рассмотрения следующей краткой постановки задачи: разработать методику динамического трекинга CLI, способную на основе вычисления информационной энтропии формировать минимально достаточную выборку тестовых сценариев.

Формальная постановка задачи и концептуальные положения ее решения

Проведенный анализ современного состояния проблемы выявил ряд факторов, которые недостаточно рассмотрены при проведении CLI-тестирования, но могут повлиять на качество и эффективность процесса. Анализ CLI на основе графовой модели позволяет формализовать связи между командами [15], подкомандами и флагами. В дополнение к топологическим (структурным) свойствам графа спецификации и принципа полноты покрытия путей, следует добавить принцип валидации бизнес-логики переходов, позволяющий экспертно осуществлять проверку.

Принцип приоритизации путей может быть обеспечен динамическим трекингом и соответствующим вектором параметров динамического трекинга – $w_v = \phi(f_v, c_v, t_v)$, где f_v – частота вызовов, c_v – критичность, t_v – новизна опции. При этом акцент тестирования с формального перебора узлов смещается на прохождение ключевых функциональных узлов.

Принцип «прозрачности реализации» или учет фактора осведомленности об использовании (usage awareness), с одной стороны, вносит дополнительную сложность в процесс оценки покрытия тестами, с другой стороны, он ориентирован на выявление функционала, который либо избыточен, либо не протестирован должным образом, несмотря на высокий спрос со стороны пользователей. Без учета контекста реальной эксплуатации приложения граф остается теоретической абстракцией.

Для формализации задачи анализа покрытия опций командной строки через динамический трекинг необходимо представить процесс как задачу достижимости состояний программы, зависящих от входных параметров:

Дано: граф спецификации $G = (V, E)$, где V – множество узлов (команд, флагов, аргументов), E – допустимые переходы между ними, $E \subseteq V \times V$ – множество ориентированных ребер, задающих иерархию и зависимости; веса осведомленности $W = \{w_1, w_2, \dots, w_n\}$, где $w_i \geq 0$ – показатель интенсивности использования узла $v_i \in V$, полученный в результате динамического трекинга; L – множество допустимых путей (линейные последовательности команд) $L = \{l_1, l_2, \dots, l_m\}$, где каждый путь l_j – сгенерированная методами DFS или BFS корректная последовательность опций (сценарий). Каждый путь l_j покрывает подмножество узлов $V_j \subseteq V$.

Найти: минимальный набор путей (сценариев) $S' \subseteq L$, который максимизирует средневзвешенное покрытие C_w: $\max C_w = \frac{\sum_{v_i \in \bigcup_{l_j \in S'} V_j} w(v_i)}{\sum_{v_k \in V} w(v_k)}$.

При ограничении на количество тестов (бюджет ресурсов): $|S'| \leq K$, где: K – максимально допустимое число тестовых запусков; $\sum_{v_i \in \bigcup_{l_j \in S'} V_j} w(v_i)$ – сумма весов всех уникальных узлов, задействованных в выбранных путях; $\sum_{v_k \in V} w(v_k)$ – полная сумма весов всех узлов в спецификации.

Предложено применить жадную стратегию, поскольку задача максимизации покрытия на графе является NP-трудной. На каждом шаге t алгоритм выбирает путь l^*, который обеспечивает максимальный удельный прирост взвешенного покрытия:

$$l^* = \arg\max_{l_j \in L \setminus S'} \left(\sum_{v_i \in (V_j \setminus V_{\text{покр}})} w(v_i)\right),$$

где $V_{\text{покр}}$ – множество узлов, уже покрытых на предыдущих шагах.

Дополнительно предложено ввести функционал информационной энтропии, оценивающий неопределенность тестового покрытия. Тогда постановка задачи позволяет перевести обсуждение из области «интуитивного тестирования» в область комбинаторной оптимизации. Если данные трекинга меняются, решение задачи пересчитывается автоматически, адаптируя тесты под пользователя.

Поскольку перебор 2^n комбинаций невозможен, задача сводится к поиску минимального набора тестовых запусков, обеспечивающего максимальное покрытие.

Предлагаемое решение можно сформулировать с помощью теоретико-множественного описания как кортеж (система), объединяющий структурные (G), алгоритмические (A, S) и информационные (W, H) компоненты.

Предложенная система M есть упорядоченная пятерка

$$M = \langle G, A, W, S, H \rangle,$$

где G – *топологический базис* (структура, граф спецификации CLI) или *графовая модель спецификации* ($V = \{v_{ctd}, v_{flag}, v_{arg}\}$ – конечное множество гетерогенных узлов);

A – *множество алгоритмов (методов) обхода графа* для формирования пространства потенциальных тестов – $A = \{f_{BFS}, f_{DFS}\}$ или *генераторы пространства состояний*, результат работы алгоритмов представляется семейством путей (сценариев) $L = \{l_j | l_j$ – допустимый путь в G от корня к листу$\}$; $l_j = \{v_{j1}, v_{j2}, \dots, v_{jk}\}$, где $v_{ji} \in V$;

W – *характеристика осведомленности об использовании*, функция весов, которая интегрирована в топологию графа $W: V \to R^+$ (R^+ – положительные числа), для каждого узла $v \in V$ вес определяется через вектор параметров динамического трекинга или при наличие вероятностного распределения весов осведомленности $W: V \to [0, 1]$, где w_i есть нормализованная вероятность $p(v_i)$ обращения пользователя к узлу;

$H(P,W)$ – *функционал информационной энтропии*, определяющий неопределенность тестового покрытия, где учитываются относительная частота использования узла (команды/флага) в реальных сценариях и вес опции: $H(S') = -\sum_{v \in V \setminus US'} w_v \log w_v$, (сумма ведется по еще не покрытым узлам – мера «остаточного незнания» о системе).

S – *селективная стратегия*, минимизирующая энтропию: $l_t = \arg\min_{l_j \in L} H(S'_{t-1} \cup \{l\})$.

Эффективность предложенной модели M выражается в ее способности определять состояния высоконагруженных узлов графа спецификации. В то время как случайный поиск снижает энтропию линейно, система M за счет селективной стратегии по критерию осведомленности об использовании (Usage Awareness) обеспечивает экспоненциальное снятие неопределенности. Это позволяет интерпретировать средневзвешенное покрытие не просто как статистический факт, а как меру информационной достоверности функционирования CLI-интерфейса:

Коэффициент информационной детерминации (η). Эффективность системы M при фиксированном количестве тестов K определяется как отношение снятой энтропии к максимально возможной: $\eta(M,K) = \frac{H_{apr} - H(S'_k)}{H_{apr}}$, где H_{apr} – априорная энтропия системы (полная неопределенность до начала тестов); $H(S'_k)$ – остаточная энтропия после выполнения сценариев S', отобранных жадным алгоритмом. Чем ближе η к 1, тем выше эффективность системы в «извлечении знаний» о поведении CLI.

Информационный выход итерации. Эффективность стратегии выбора S выражается через максимизацию прироста информации на каждом шаге t:

$$\triangle I_t = H(S_{t-1}) - H(S_t) \to max.$$

В системе M высокая эффективность достигается за счет того, что Usage Awareness (W) направляет поиск BFS/DFS в те зоны графа, где плотность «неопределенности» (веса неисследованных узлов) максимальна.

Снятие неопределенности. Эффективность системы M также можно описать через функцию выигрыша осведомленности:

$$G_{\text{адапт}} \approx \frac{1}{\triangle H}.$$

Это означает, что система считается эффективной, если она минимизирует энтропию H за минимальное число переходов в графе G.

Такое описание позволяет представить решение в виде математической модели оптимизации тестового покрытия, где граф отвечает за синтаксическую корректность, алгоритмы DFS/BFS – за полноту поиска, осведомленность об использовании – за релевантность данных, селективная стратегия – за вычислительную эффективность.

В отличие от классических моделей тестирования предложенный кортеж рассматривает процесс покрытия не как количественное накопление пройденных узлов, а как качественное снятие неопределенности относительно эксплуатационной надежности CLI. Введение энтропийного функционала позволяет жадному алгоритму максимизировать информационный выход каждой тестовой итерации. Таким образом, прирост осведомленности интерпретируется как детерминация наиболее вероятных путей исполнения в условиях ограниченных вычислительных ресурсов.

К жадному алгоритму и методам BFS/DFS (стратегии выбора) на графе спецификации, следует добавить средневзвешенное покрытие, а также оценку осведомленности пользователя, что позволит осуществить выбор наиболее значимых путей и повысить оценку осведомленности пользователя при тестировании. Предлагаемая модель отличается от ранее предложенной авторами [15] графовой модели тем, что обеспечивает оценку в динамике.

Таблица 2. Компоненты предлагаемого решения.

Компонент подхода	**Назначение**	**Эффективность**
Селективная стратегия	выбор набора узлов графа (команд), активирующего как можно больше неисследованных компонентов спецификации	быстрый поиск субоптимального набора тестов, который покрывает большинство узлов графа
BFS (Breadth-First Search) – поиск в ширину	для тестирования коротких путей	нахождение кратчайшего пути от корня до нужной опции
DFS (Depth-First Search) – поиск в глубину	имитирует поведение пользователя, который изучает конкретную ветку до конца	требует меньше памяти, чем BFS, на широких графах; быстрое нахождение сценариев с глубокой вложенностью подкоманд
Графовая модель	описание структуры команд (вершины (команды, флаги, параметры)) и переходов между ними (ребра)	переход к структурному пониманию программы
Средневзвешенное покрытие	характеризует, что не все опции равноценны.	акцент на критически важных путях
Оценка осведомленности пользователя	связь теоретической модели графа с реальным поведением пользователей	повышение бизнес-ценности тестирования за счет первоочередного «закрытия» частей графа, которые наиболее востребованы пользователями
Функционал информационной энтропии	неопределенность тестового покрытия	интерпретация средневзвешенного покрытия как информационной составляющей предлагаемого решения.

Алгоритм выбора оптимального покрытия S. В рамках предложенной теоретико-множественной модели $M = \langle G, A, W, S, H \rangle$ селективная стратегия трансформируется из простого сумматора весов в механизм минимизации энтропии (неопределенности). Следовательно, цель – выбор последовательности путей, обеспечивающей максимально быстрое «снятие» неопределенности относительно эксплуатационного состояния CLI.

На каждой итерации t алгоритм выбирает путь $l' \in L$, который максимизирует информационный выигрыш, то есть максимально снижает текущую энтропию системы:

$$l' = \arg\max_{l \in L}[H(V_{t-1}) - H(V_{t-1} \cup \text{узлы}(l))],$$

где $H(V)$ – информационная энтропия множества непокрытых узлов, рассчитанная на основе распределения осведомленности об использовании.

Входные данные: $L = \{l_1, l_2, \dots, l_m\}$ – множество допустимых путей (сценариев), сгенерированных методами A (BFS/DFS); $W = \{w_1, w_2, \dots, w_n\}$ – веса осведомленности об использовании для узлов $v \in V$; K – лимит количества тестовых сценариев (бюджет).

Шаг 1. *Вероятностная формализация*: вероятность обращения к узлу определяется как: $p(v_i) = \frac{w(v_i)}{\sum_{k=1}^{n} w(v_k)}$, то есть осуществляется преобразование весов осведомленности в вероятностное распределение $p(v_i)$.

Вычисляется начальная энтропия системы (полная неопределенность):

$$H_0 = -\sum p(v_i) \log p(v_i).$$

Шаг 2. *Инициализация параметров*: на начальном этапе множество выбранных сценариев $S = \emptyset$; множество детерминированных (покрытых) узлов $V_{\text{покр}} = \emptyset$; текущая остаточная энтропия $H_{res} = H_0$.

Шаг 3. *Итерационный жадный поиск*: пока множество сценариев $|S| < K$ и текущая остаточная энтропия $H_{res} > \varepsilon$ (ε – порог допустимой неопределенности):

– рассчитывается вклад в снятие неопределенности – оценка информационного выигрыша ($\triangle I$) для каждого пути $l_j \in L \backslash S$:

$$\triangle I(l_j) = \sum_{v \in (\text{узлы}(l_j) \backslash V_{\text{покр}})} -p(v) \log p(v);$$

– выбирается путь l', обеспечивающий максимальный информационный выигрыш:

$$l' = \arg\max_{l_j} \triangle I(l_j);$$

– осуществляется обновление состояния системы: пополняется множество сценариев $S = S \cup \{l'\}$; учитываются добавленные покрытые узлы $V_{\text{покр}} = V_{\text{покр}} \cup \text{узлы}(l')$; пересчитывается текущая остаточная энтропия $H_{res} = H_{res} - \triangle I(l')$.

Шаг 4. *Расчет эффективности модели*: Вычисление коэффициента информационной детерминации: $\eta = \frac{H_0 - H_{res}}{H_0}$.

В описании алгоритмов на основе предложенных для выявления путей методов BFS и DFS сделаем акцент на топологический и информационный компоненты (Рис. 2 и Рис. 3). Информационные компоненты модифицируют методы в рамках модели *M*.

Модифицированный внесением информационного компонента метод BFS обеспечивает в первую очередь покрытие верхних уровней функционала (команд) спецификации.

За исследование и оценку неопределенности более глубоких уровней при использовании предложенной модели *M* отвечает метод DFS (Рис. 3).

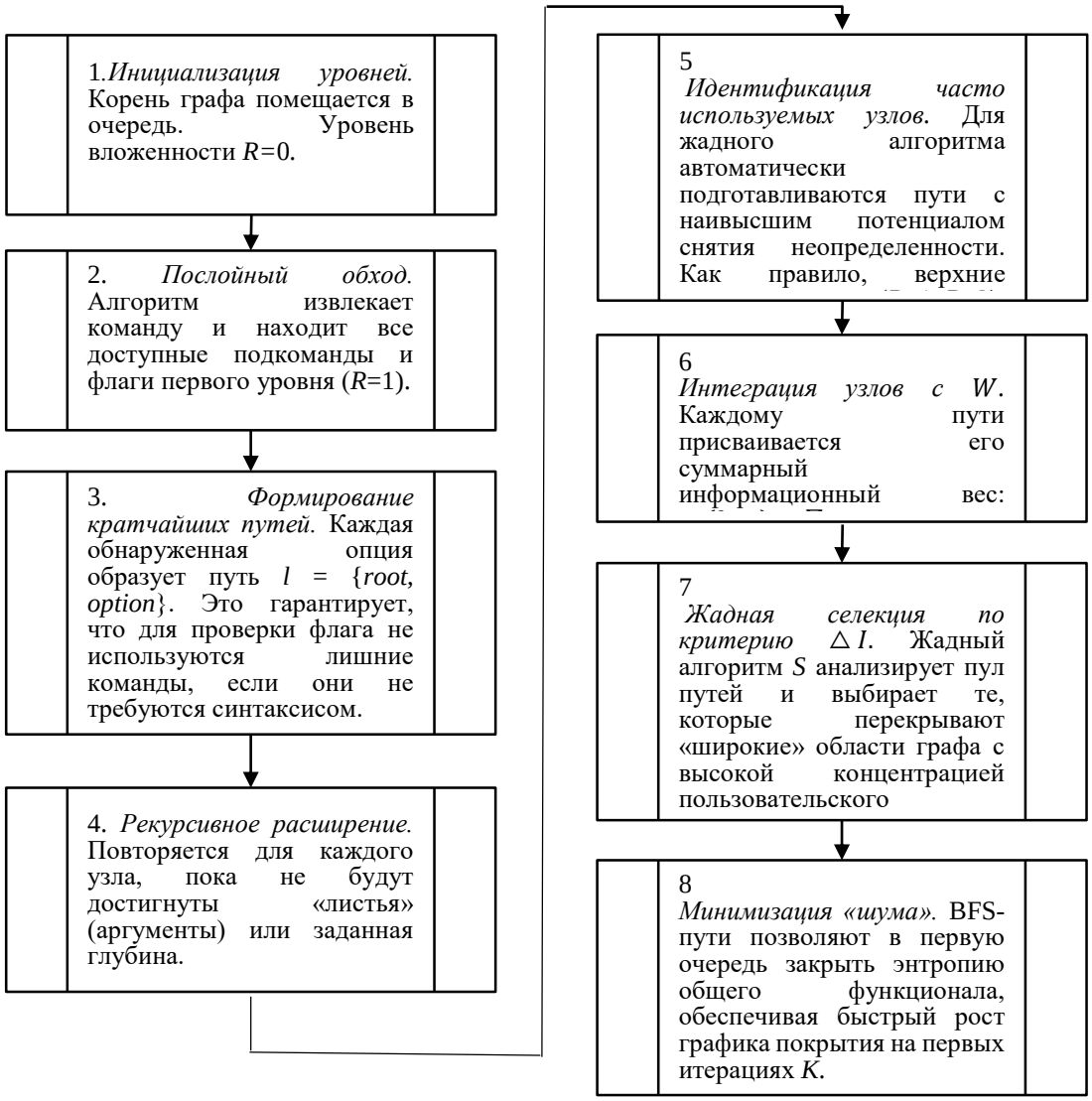

Рисунок 2. Этапы топологического и информационного компонентов метода BFS.

Одной из оценок информационного компонента метода DFS выступает локальное покрытие узла.

Локальное покрытие узла $C(v)$. Локальное покрытие узла $v_i \in V$ определяется как коэффициент детерминации функционального состояния команды и вычисляется как отношение мощности множества активированных в трассах A (BFS/DFS) опций к общей мощности множества доступных флагов узла:

$$C(v_i) = \left|F_{i,active}\right|/|F_i|.$$

С позиции теории информации рост $C(v_i)$ эквивалентен снижению условной энтропии конкретной команды графа спецификации.

Оценка осведомленности $w(v_i)$. Вес осведомленности $w(v_i) = W(v_i)$ интерпретируется как априорная плотность информационной значимости узла. В вероятностном пространстве модели *M* нормированный вес $p(v_i) = w(v_i)/\sum w(v_k)$ задает

распределение вероятностей использования функций CLI. Таким образом, узлы с высоким показателем Usage Awareness обладают наибольшим потенциалом снятия неопределенности при их включении в тестовую выборку *S*.

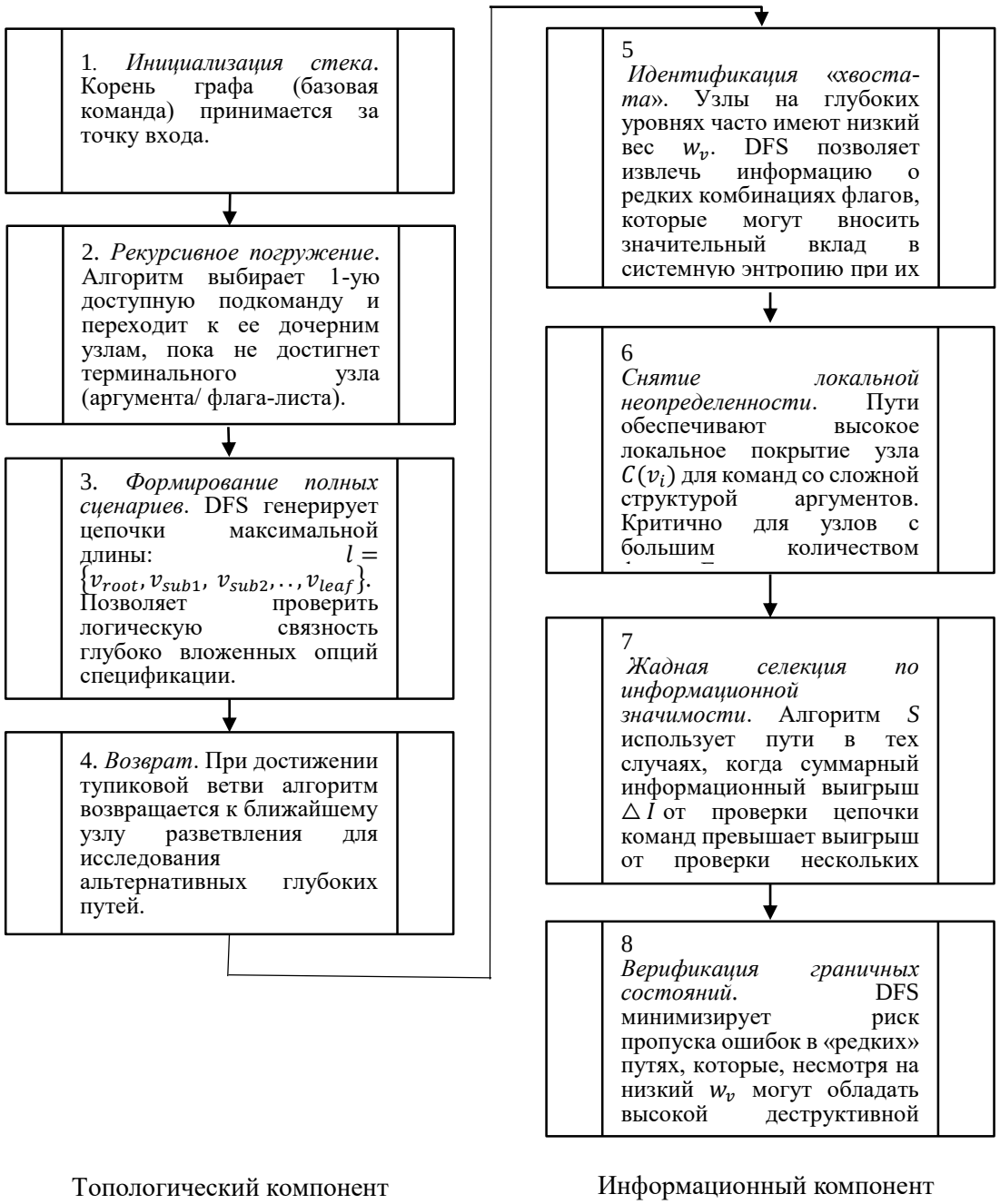

Рисунок 3. Этапы топологического и информационного компонентов метода DFS.

Интегральный показатель ($C_{\text{глоб}}$) представляет собой коэффициент глобальной детерминации системы η, выражающий долю устраненной энтропии графа CLI:

$$C_{\text{глоб}} = \eta = \frac{\sum_{v_i \in V}(C(v_i)w(v_i)}{\sum_{v_k \in V} w(v_k)}.$$

Эффективность модели *M* считается доказанной, если скорость роста $C_{\text{глоб}}$ относительно объема выборки *K* коррелирует со скоростью минимизации остаточной энтропии *H*(*S*). Это позволяет утверждать, что система в первую очередь верифицирует сценарии, вносящие максимальный вклад в общую неопределенность пользовательского опыта.

Интерпретация в контексте топологического поиска. Такая формализация позволяет математически обосновать эффективность методов обхода:

– BFS-трассы минимизируют знаменатель в локальном покрытии, быстро закрывая высокоуровневые узлы v_i с максимальными весами w_i;

– DFS-трассы максимизируют числитель $F_{i,active}$, обеспечивая глубокую проверку аргументов для узлов, имеющих сложную внутреннюю структуру, но, возможно, меньший суммарный вес в общей популяции вызовов.

Итоговая метрика $C_{\text{глоб}}$ является объективным индикатором готовности CLI-интерфейса к эксплуатации, так как она приоритизирует проверку тех «веток» графа, которые имеют наибольшее влияние на пользовательский опыт и стабильность системы.

Вес $w(v_i)$ отражает значимость команды (например, частоту ее встречаемости или критичность кода).

Для завершения описания метрики в теоретико-множественном представлении системы $M = \langle G, A, W, S, H \rangle$, необходимо формализовать понятие «прироста знания» через изменение функции весов W и селектора S.

Оценка динамики осведомленности. В рамках предложенной модели M относительный прирост знания $G_{\text{адапт}}$ интерпретируется как мера адаптивности системы тестирования к реальному пользовательскому опыту. Пусть $C_{\text{апр}}$ – показатель покрытия, полученный при допущении равнозначности всех узлов графа (априорное состояние до внедрения трекинга):

$$C_{\text{апр}} = \frac{1}{|V|} \sum_{v_i \in V} C(v_i),$$

где $w_i = 1$ для всех i. Пусть $C_{\text{свп}}$ – средневзвешенное покрытие, рассчитанное после внедрения системы динамического трекинга и пересчета весов W в стратегии поиска S:

$$C_{\text{свп}} = \frac{\sum_{v_i \in V} (C(v_i) w(v_i)}{\sum_{v_k \in V} w(v_k)},$$

где $w_i = W(v_i)$.

Тогда относительный прирост осведомленности $G_{\text{адапт}}$ выражается через оператор эффективности:

$$G_{\text{адапт}} = \frac{C_{\text{свп}} - C_{\text{апр}}}{C_{\text{апр}}} \times 100\%.$$

Теоретическая интерпретация параметров представлена в Табл. 3.

Таблица 3. Интерпретация параметров.

Обозначение	**Наименование параметра**	**Интерпретация**
$C_{\text{свп}}$	информационная ценность/ средневзвешенное покрытие	отражает «попадание» тестового набора в наиболее критические и часто используемые узлы графа; высокое значение $C_{\text{свп}}$ при низком количестве тестов, как правило, свидетельствует об успешной приоритизации путей (BFS/DFS)
$C_{\text{апр}}$	эффект «слепого» покрытия/ априорное состояние до внедрения трекинга	демонстрирует избыточность или недостаточность классических методов обхода, которые могут покрывать 100% кода, но 0% реально используемых пользовательских сценариев
$G_{\text{адапт}}$	показатель адаптации	позволяет количественно оценить пользу от внедрения модуля, в основе которого лежит модель осведомленности; положительный прирост $G_{\text{адапт}} > 0$ показывает, что жадный алгоритм перераспределил ресурсы тестирования в сторону узлов с максимальным весом w_v

Прирост знания в рамках модели *M* интерпретируется как снятие неопределенности относительно поведения CLI в реальной среде. Использование энтропийного функционала *H* позволяет количественно оценить переход от «структурного» тестирования к «интеллектуальному». Рост показателя $G_{\text{адапт}}$ свидетельствует о том, что система *M* успешно детерминирует состояния наиболее значимых узлов графа *G*, превращая априорную неопределенность в апостериорную осведомленность о надежности интерфейса. При выполнении жадного алгоритма выбираются такие пути, которые обеспечивают максимальное снижение энтропии на каждом шаге.

При сравнении «Слепого» и «Осведомленного» покрытия следует отметить, что в первом случае снимается неопределенность со структуры кода, но оставляется неопределенность в сценариях использования. Во втором случае снимается неопределенность именно в зоне пересечения кода и пользовательского опыта.

Введение метрики прирост осведомленности через призму неопределенности интерпретируется не просто как накопление тестовых логов, а как целенаправленное извлечение информации о наиболее критичных путях исполнения. Это объясняет высокую эффективность метода при малых значениях *K*: алгоритм интуитивно «снимает» основную массу неопределенности в первые итерации, оставляя на периферии узлы с низкой вероятностью активации (информационным шумом).

Проведение эксперимента и интерпретация результатов

Рассмотрим реализацию предлагаемого подхода на примере CLI-интерфейса системы управления онлайн-магазином.

Модель графа (узлы и веса). Пусть имеются три ветки команд с разной значимостью (весом w_i) и частотой использования (p_i): кластер A (заказы (*order create*, *pay*) – критическая, вес w = 0,9; проблема узла – означает остановку бизнеса); кластер B (каталог (*item list*, *search*) – высокая, вес w = 0,5; проблема узла – мешает продажам); кластер C (настройки профиля (*config set-theme*) – низкая, вес w = 0,5; проблема узла – неприятна, но не критична).

Процесс селективного тестирования (минимизация энтропии) представлен в Табл. 4.

Таблица 4. Процесс селективного тестирования.

Шаг (*K*)	**Выбранный сценарий**	**Извлеченная информация ($\triangle H$)**	**Остаточная неопределенность *H*(*S*)**
0	Нет тестов	–	100% (max)
1	*order create → id1 → pay*	высокая (−60%), проверен самый тяжелый узел	40%
2	*item search → query “phone”*	средняя (−25%), проверен основной путь пользователя	15%
3	*order create → promo “SALE”*	низкая (−10%), проверка вариации уже известного пути	5%
4	*config set-theme → dark*	минимальная (−4%), проверка низкоприоритетной ветки	1% ($\approx \varepsilon$)

Эффективность модели *M* продемонстрирована тем, что за 2 итерации можно устранить 85% неопределенности пользовательского опыта. Обычный случайный перебор мог бы

отратить эти 2 итерации на тесты *set-theme* и *help*, оставив критическую ветку *pay* неисследованной (сохранив высокую энтропию и риски).

Для возможности обобщения результатов рассмотрен еще один пример.

Предположим, что граф содержит 3 ключевых кластера команд: кластер Transaction (T) (создание заказов, оплата, возврат (вес узла w = 0,9, высокая вероятность использования p = 0,7)); кластер Inventory (I) (складской учет, обновление цен, поиск остатков (вес 0,6, p = 0,25)); кластер UI-Config (U): настройка цвета чеков, локализация интерфейса (вес узла w = 0,1, p = 0,05)).

Динамика тестирования представлена в Табл. 5.

Таблица 5. Динамика процесса тестирования.

Шаг (*K*)	**Выбранный сценарий**	**Причина выбора (селекция)**	**Вклад в C_{total}**	**Остаточная неопределенность $H(S)$**	**Состояние системы**
0	Нет тестов	-	0%	**12,0** бит (max)	полная неопределенность.
1	*shop order →pay → cc*	максимальный вес *w* и вероятность *p*	+35%	5,8 бит (−6,2)	снятие неопределенности: верифицирован критический путь оплаты
2	*shop inv → update → price*	высокий риск расхождения данных	+20% (55%)	3,1 бит (−2,7)	рост осведомленности: проверена синхронизация цен
3	*shop order → refund*	редкий, но юридически критичный путь	+15% (70%)	1,4 бит (−1,7)	уточнение модели: закрыта зона высокого риска
4	*shop inv → search → fast*	оптимизация поиска (популярно у пользователей)	+10% (80%)	**0,6** бит (−0,8)	насыщение: основные паттерны пользователей покрыты
5	*shop config → lang ru*	низкий вес, малый вклад в стабильность	+3% (83%)	0,45 бит (−0,15)	зона шума: прирост информации ниже порога ε

Модель *M* не использует множество комбинаций в config на первых шагах, а ресурсы уходят в order и inventory. В итоге за 4 теста устранено 95% неопределенности (12 → 0,6 бит), хотя формально покрыто 15% всех возможных флагов.

Выводы

Сформулирована теоретико-множественная модель, которая объединяет топологические свойства графа спецификации с вероятностным распределением пользовательских предпочтений. Обосновано применение энтропийного подхода для оценки качества тестирования.

Интерпретация процесса покрытия как последовательного снятия неопределенности позволила математически доказать субоптимальность жадной стратегии при выборе путей в графе. Установлена роль методов поиска: определено, что BFS эффективен для быстрой детерминации высокоуровневой энтропии (общих команд), тогда как DFS необходим для снятия неопределенности в глубоких, специфических ветках спецификации.

Практическая значимость подтверждается способностью алгоритма автоматически перестраивать приоритеты тестирования при обновлении данных динамического трекинга. Это минимизирует «информационный шум» в отчетах о покрытии и фокусирует внимание разработчиков на участках кода, наиболее подверженных риску с точки зрения реальной эксплуатации.

Разработанный подход закладывает фундамент для создания «умных» систем автоматизированного тестирования, способных самостоятельно обучаться на основе логов использования программных продуктов.

Экспериментальные данные показали, что учет осведомленности об использовании позволяет устранить 95% неопределенности, хотя формально покрыто лишь 15% всех возможных флагов.

Литература:

1. Wang T., Wei T., Gu G., Ganesh V. TaintScope: A checksum-aware directed fuzzing tool for automatic software vulnerability detection. In: Proceedings of the 31st IEEE Symposium on Security and Privacy (S&P '10). IEEE Computer Society, 2010. P. 497–512.

2. Shelke S., Patil S. A survey of automated test data generation techniques // International Journal of Advance Engineering and Research Development. 2018. Vol. 5. No. 8. P. 115–122.

3. Вартанов С.П., Герасимов А.Ю. Динамический анализ программ с целью поиска ошибок и уязвимостей при помощи целенаправленной генерации входных данных // Труды Института системного программирования РАН. Математика, Компьютерные и информационные науки. 2014. Т. 26. № 1. С. 375–393. [Vartanov S.P., Gerasimov A.Yu. Dynamic analysis of programs for finding errors and vulnerabilities using targeted input data generation // Proceedings of the Institute for System Programming of the RASciences. Mathematics, Computer and Information Sciences. 2014. Vol. 26. No. 1. P. 375–393 (in Russian)].

4. Вишняков А.В., Кобрин И.А., Федотов А.Н. Поиск ошибок в бинарном коде методами динамической символьной интерпретации // Труды ИСП РАН. 2022. Т. 34. № 2. С. 25–42. [Vishnyakov A.V., Kobrin I.A., Fedotov A.N. Error detection in binary code using dynamic symbolic interpretation methods // Proceedings of the ISP RAS. 2022. Vol. 34. No. 2. P. 25–42 (in Russian)].

5. Плаксин С.И., Методы формального описания кода программного обеспечения в задачах автоматического анализа поведения программ // Известия ТулГУ. Технические науки. 2023. № 12. С. 514–519. [Plaksin S. I., Methods of formal description of software code in problems of automatic analysis of program behavior // Bulletin of Tula State University. Technical Sciences. 2023. No. 12. P. 514–519 (in Russian)].

6. Ерохина Н.С. Метод мутации сложно-структурированных входных данных при фаззинг-тестировании JavaScript интерпретаторов // Труды ИСП РАН. 2023. Т. 35. № 5. С. 55–66. [Erokhina N.S. A Method for mutating complexly structured input data in fuzz testing JavaScript interpreters // Proceedings of ISP RAS. 2023. Vol. 35. No. 5. P. 55–66 (in Russian)].

7. Грибалев Н.Г.Code coverage и стабильность – ключевые метрики успеха технологической команды // Символ науки. 2025. № 4-2. С. 34–42. [Gribalev N.G. Code coverage and stability are key metrics for the success of a technology team // Symbol of Science. 2025. No. 4-2. P. 34–42 (in Russian)].

8. Godefroid P., Levin M., Molnar D. Automated whitebox fuzz testing. In: Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS’08), San Diego, CA, February 2008.
9. Fioraldi A., D'Elia D.C., Coppa E. WEIZZ: Automatic Grey–box Fuzzing for Structured Binary Formats. In: ISSTA 2020: Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. ACM Digital Library, 2020. P. 1–13. DOI: 10.1145/3395363.3397372.
10. Fioraldi A., Maier D., Zhang D., Crump A. AFLrustrust: A LibAFL–based AFL++ prototype. In: 2023 IEEE/ACM International Workshop on Search-Based and Fuzz Testing (SBFT). IEEE, 2023. DOI: 10.1109/SBFT59156.2023.00024.
11. Cadar C., Dunbar D., Engler D. KLEE: Unassisted and Automatic Generation of High–Coverage Tests for Complex Systems Programs.In: OSDI'08: Proceedings of the 8th USENIX conference on Operating systems design and implementation. ACM Digital Library, 2008. P. 209–224.
12. Владимиров М.А. Критерии полноты тестового покрытия в генетических алгоритмах генерации тестов // Труды Института системного программирования РАН. 2006. Т. 9. С. 57–66. [Vladimirov M.A. Test coverage criteria in genetic test generation algorithms // Proceedings of the ISP RAS. 2006. Vol.9. P. 57–66 (in Russian)].
13. Inozemtseva L. Holmes R. Coverage is not strongly correlated with test suite effectiveness. Proceedings of the 36th ICSE. ACM Digital Library, 2014. P. 435–445.
14. Arcuri A., Briand L. Adaptive random testing: An illusion of effectiveness? In: Proceedings of the 2011 International Symposium on Software Testing and Analysis (ISSTA '11). ACM Digital Library, 2011. P. 265–275.
15. Чембарисов Э.М., Сметанина О.Н., Сазонова Е.Ю. Графовые модели и операции при оценке микросервисных решений // Вестник УГАТУ. 2026. Т. 30. № 1(111). С. 39–47. [Chembarisov E.M., Smetanina O.N., Sasonova Yu.Ye. Graph models and operations in evaluating microservice solutions // Vestnik UGATU. 2026. Vol. 30. No. 1(111). P. 39–47 (in Russian)].

Об авторах:

ЧЕМБАРИСОВ Эмиль Марсович, аспирант третьего года обучения Уфимского университета науки и технологий, 32, ул. Заки Валиди, 450076 г. Уфа, Республика Башкортостан, Россия, lawluker@gmail.com.
СМЕТАНИНА Ольга Николаевна, доктор технических наук, доцент, профессор кафедры ВМиК Уфимского университета науки и технологий, 32, ул. Заки Валиди, 450076 г. Уфа, Республика Башкортостан, Россия, smoljushka@mail.ru.
САЗОНОВА Екатерина Юрьевна, кандидат технических наук, доцент, доцент кафедры ВМиК Уфимского университета науки и технологий, 32, ул. Заки Валиди, 450076 г. Уфа, Республика Башкортостан, Россия, rassadnikova_ekaterina@mail.ru.

Metadata:

Title: A Methodology for Selective Testing of CLI Interfaces Based on Graph Models and Information Entropy Minimization.

Author 1: Emil Marsovich Chembarisov, third year graduate at the Ufa University of Science and Technology, 32 Zaki Validi st., 450076 Ufa, Republic of Bashkortostan, Russia, lawluker@gmail.com.
Author 2: Olga Nikolaevna Smetanina, Doctor of Technical Sciences, Associate Professor, Professor of the CMaC Department at the Ufa University of Science and Technology, 32 Zaki Validi st., 450076 Ufa, Republic of Bashkortostan, Russia, smoljushka@mail.ru.
Author 3: Yekaterina Yuryevna *Sasonova,* Candidate of Technical Sciences, Associate Professor, Associate Professor of the CMaC Department at the Ufa University of Science and Technology, 32 Zaki Validi st., 450076 Ufa, Republic of Bashkortostan, Russia, rassadnikova_ekaterina@mail.ru.

Abstract: This article discusses how to optimize command-line interface (CLI) test coverage in conditions of complex specifications and limited resources. Traditional testing methods often fail to consider real-world scenarios, which can lead to inefficient use of resources to cover rarely used functionality. The authors review the current state of the problem and present a proposed set-theoretic model that takes into account the topological properties of the specification graph, as well as algorithmic and informational components. The set of test scenarios is generated using topological breadth-first and depth-first search (BFS and DFS) methods. A key feature of the model is the inclusion of a usage awareness parameter, which, based on dynamic call tracking, assigns awareness weights to graph nodes. The process of selecting the optimal test set is formalized using Shannon's information entropy, where the selection strategy at each iteration maximizes the removal of uncertainty regarding the system's operational state. The concept of average weighted coverage and a metric for relative awareness gain are introduced. The proposed model provides a transition from structural code coverage to user-value coverage.

Keywords: CLI, greedy algorithm, specification graph, BFS, DFS, usage awareness, weighted average coverage, dynamic tracking.