2.3.2.14. Классификация по моменту выполнения (хронологии)

Несмотря на многочисленные попытки создать единую хронологию тестирования, предпринятые многими авторами, по-прежнему можно смело утверждать, что общепринятого решения, которое в равной степени подходило бы для любой методологии управления проектами, любого отдельного проекта и любой его стадии, просто не существует.

Если попытаться описать хронологию тестирования одной общей фразой, то можно сказать, что происходит постепенное наращивание сложности самих тест-кейсов и сложности логики их выбора.

- Общая универсальная логика последовательности тестирования состоит в том, чтобы начинать исследование каждой задачи с простых позитивных тест-кейсов, к которым постепенно добавлять негативные (но тоже достаточно простые). Лишь после того, как наиболее типичные ситуации покрыты простыми тест-кейсами, следует переходить к более сложным (опять же, начиная с позитивных). Такой подход — не догма, но к нему стоит прислушаться, т.к. углубление на начальных этапах в негативные (к тому же — сложные) тест-кейсы может привести к ситуации, в которой приложение отлично справляется с кучей неприятностей, но не работает на элементарных повседневных задачах. Ещё раз суть универсальной последовательности:
    1) простое позитивное тестирование;
    2) простое негативное тестирование;
    3) сложное позитивное тестирование;
    4) сложное негативное тестирование.
- Последовательность тестирования, построенная по иерархии компонентов:
    - **Восходящее тестирование** (bottom-up testing[264]) — инкрементальный подход к интеграционному тестированию{72}, в котором в первую очередь тестируются низкоуровневые компоненты, после чего процесс переходит на всё более и более высокоуровневые компоненты.
    - **Нисходящее тестирование** (top-down testing[265]) — инкрементальный подход к интеграционному тестированию{72}, в котором в первую очередь тестируются высокоуровневые компоненты, после чего процесс переходит на всё более и более низкоуровневые компоненты.
    - **Гибридное тестирование** (hybrid testing[266]) — комбинация восходящего и нисходящего тестирования, позволяющая упростить и ускорить получение результатов оценки приложения.

Поскольку термин «гибридное» является синонимом «комбинированное», под «гибридным тестированием» может пониматься практически любое сочетание двух и более видов, техник или подходов к тестированию. Всегда уточняйте, о гибриде чего именно идёт речь.

[264] **Bottom-up testing.** An incremental approach to integration testing where the lowest level components are tested first, and then used to facilitate the testing of higher level components. This process is repeated until the component at the top of the hierarchy is tested. [ISTQB Glossary]

[265] **Top-down testing.** An incremental approach to integration testing where the component at the top of the component hierarchy is tested first, with lower level components being simulated by stubs. Tested components are then used to test lower level components. The process is repeated until the lowest level components have been tested. [ISTQB Glossary]

[266] **Hybrid testing, Sandwich testing.** First, the inputs for functions are integrated in the bottom-up pattern discussed above. The outputs for each function are then integrated in the top-down manner. The primary advantage of this approach is the degree of support for early release of limited functionality. [«Integration testing techniques», Kratika Parmar]

- Последовательность тестирования, построенная по концентрации внимания на требованиях и их составляющих:
    1) Тестирование требований, которое может варьироваться от беглой оценки в стиле «всё ли нам понятно» до весьма формальных подходов, в любом случае первично по отношению к тестированию того, как эти требования реализованы.
    2) Тестирование реализации функциональных составляющих требований логично проводить до тестирования реализации нефункциональных составляющих, т.к. если что-то просто не работает, то проверять производительность, безопасность, удобство и прочие нефункциональные составляющие бессмысленно, а чаще всего и вовсе невозможно.
    3) Тестирование реализации нефункциональных составляющих требований часто становится логическим завершением проверки того, как реализовано то или иное требование.

- Типичные общие сценарии используются в том случае, когда не существует явных предпосылок к реализации иной стратегии. Такие сценарии могут видоизменяться и комбинироваться (например, весь «типичный общий сценарий 1» можно повторять на всех шагах «типичного общего сценария 2»).
    - Типичный общий сценарий 1:
        1) Дымовое тестирование{74}.
        2) Тестирование критического пути{75}.
        3) Расширенное тестирование{76}.
    - Типичный общий сценарий 2:
        1) Модульное тестирование{72}.
        2) Интеграционное тестирование{72}.
        3) Системное тестирование{73}.
    - Типичный общий сценарий 3:
        1) Альфа-тестирование{79}.
        2) Бета-тестирование{79}.
        3) Гамма-тестирование{79}.

В завершение ещё раз подчеркнём, что рассмотренные здесь классификации тестирования не являются чем-то каноническим и незыблемым. Они лишь призваны упорядочить огромный объём информации о различных видах деятельности тестировщиков и упростить запоминание соответствующих фактов.

2.3.3. Альтернативные и дополнительные классификации тестирования

Для полноты картины остаётся лишь показать альтернативные взгляды на классификацию тестирования. Одна из них (рисунки 2.3.i и 2.3.j) представляет не более чем иную комбинацию ранее рассмотренных видов и техник. Вторая (рисунки 2.3.k и 2.3.l) содержит много новых определений, но их подробное изучение выходит за рамки данной книги, и потому будут даны лишь краткие пояснения (при необходимости вы можете ознакомиться с первоисточниками, которые указаны для каждого определения в сноске).

Ещё раз подчеркнём: здесь приведены лишь определения. Соответствующим видам и техникам тестирования в первоисточниках посвящены десятки и сотни страниц. Пожалуйста, не ожидайте от этого раздела подробных пояснений, их не будет, т.к. это — «очень дополнительный» материал.

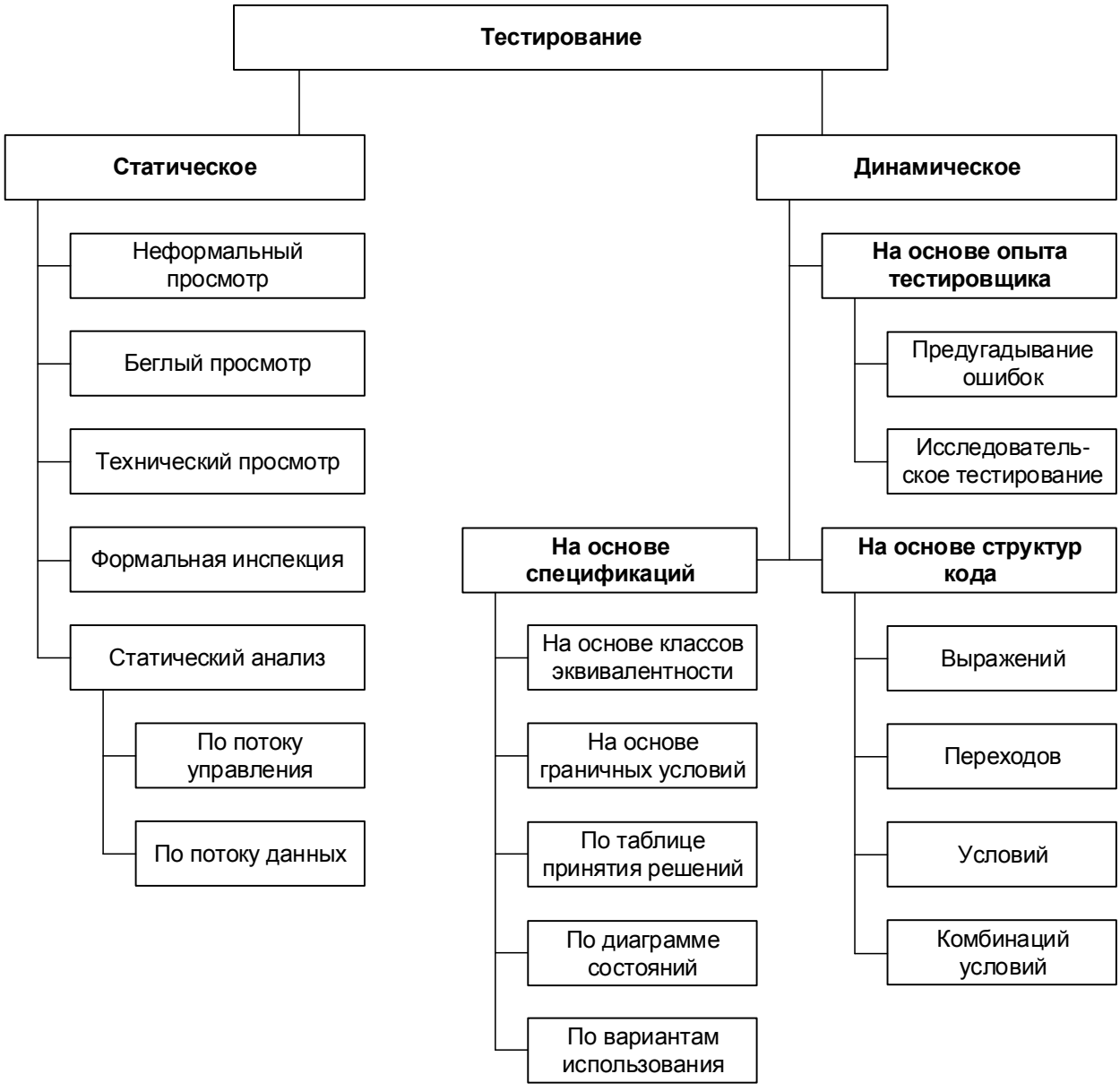

Рисунок 2.3.i — Классификация тестирования согласно книге «Foundations of Software Testing: ISTQB Certification» (Erik Van Veenendaal, Isabel Evans) (русскоязычный вариант)

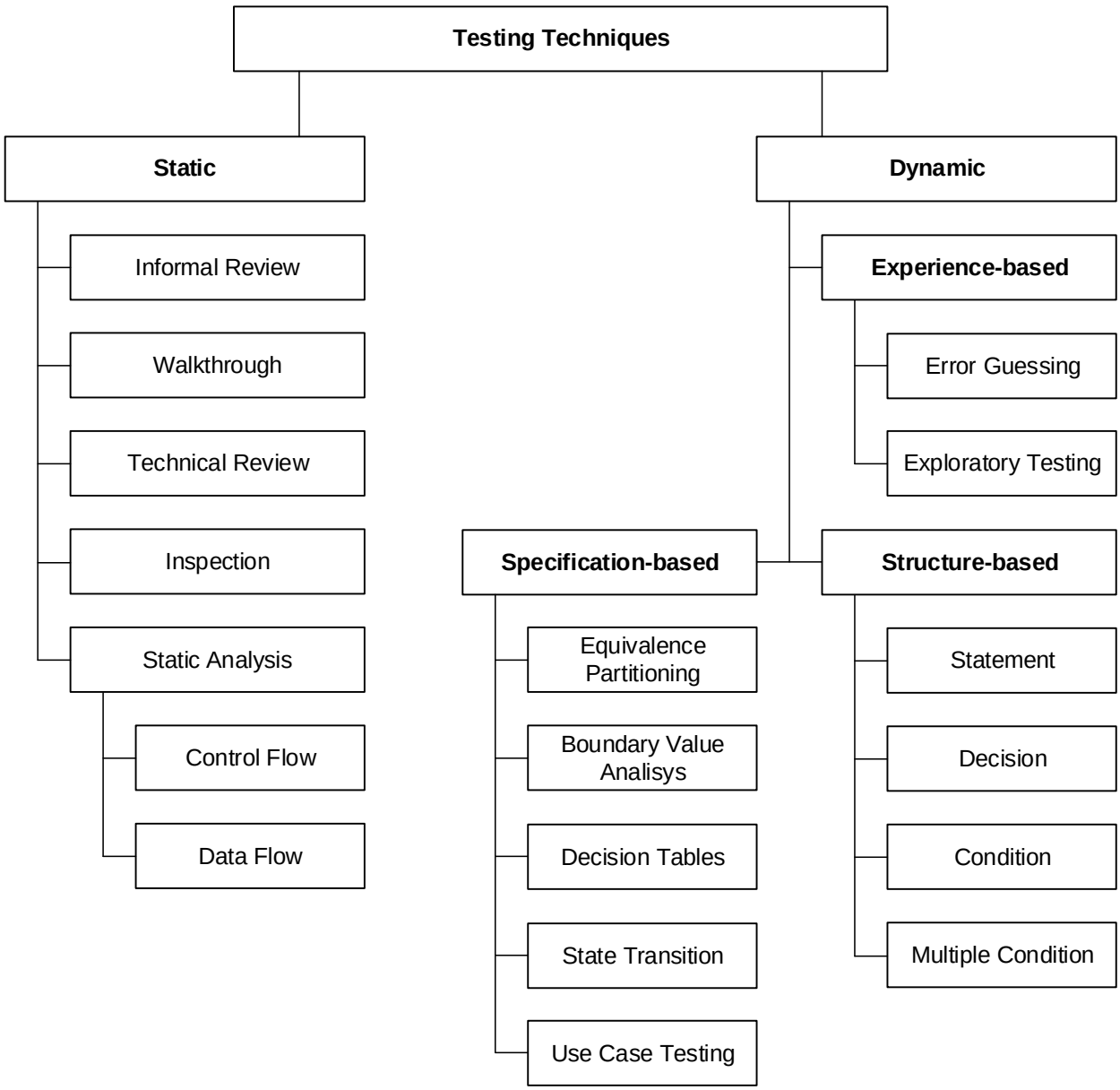

Рисунок 2.3.j — Классификация тестирования согласно книге «Foundations of Software Testing: ISTQB Certification» (Erik Van Veenendaal, Isabel Evans) (англоязычный вариант)

В следующей классификации встречаются как уже рассмотренные пункты, так и ранее не рассмотренные (отмечены пунктирной линией). Краткие определения не рассмотренных ранее видов тестирования представлены после рисунков 2.3.k и 2.3.l.

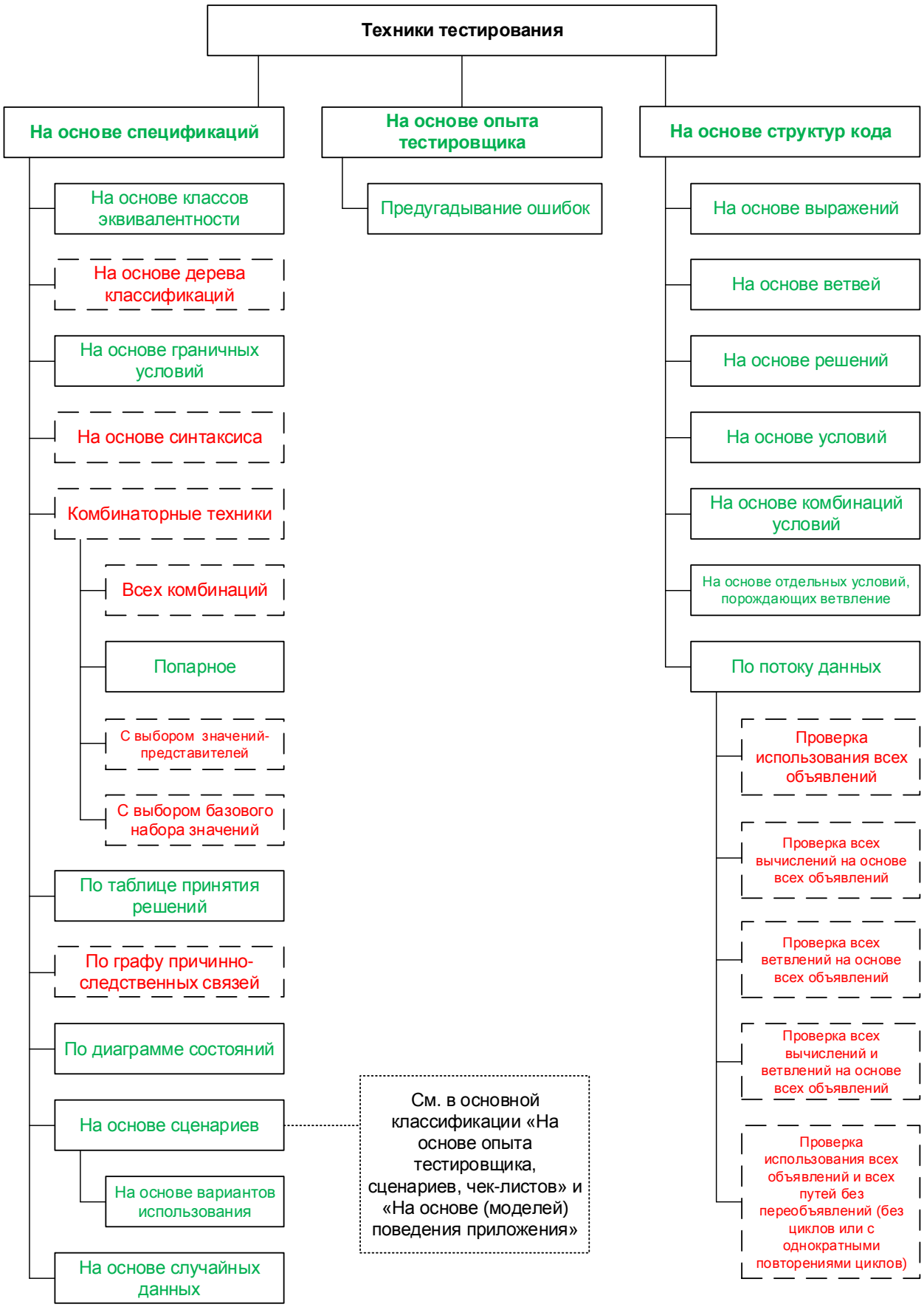

Рисунок 2.3.k — Классификация тестирования согласно ISO/IEC/IEEE 29119-4 (русскоязычный вариант)

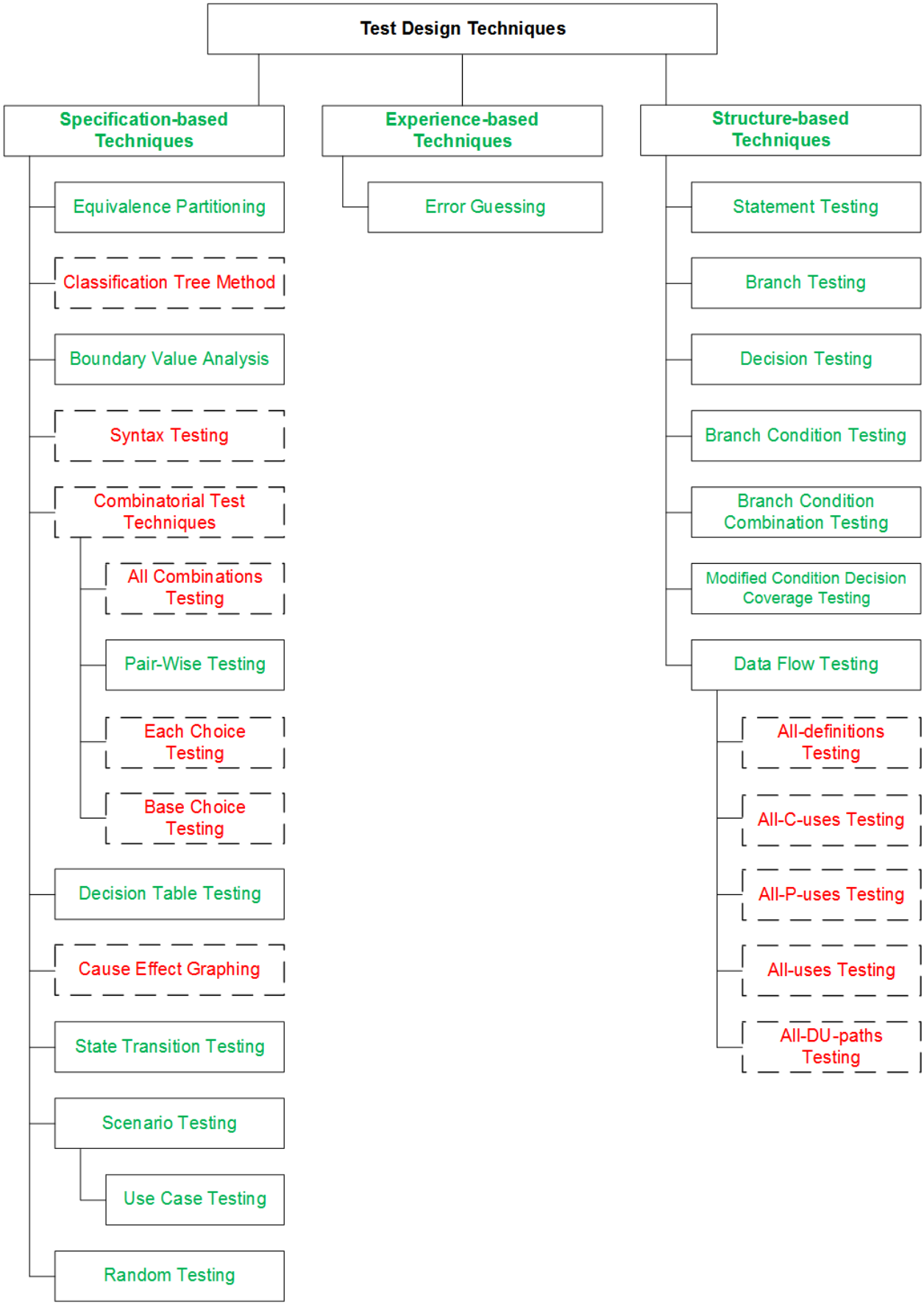

Рисунок 2.3.I — Классификация тестирования согласно ISO/IEC/IEEE 29119-4 (англоязычный вариант)

- **Тестирование на основе дерева классификаций** (classification tree[267] method[268]) — техника тестирования (по методу чёрного ящика), в которой тест-кейсы создаются на основе иерархически организованных наборов эквивалентных входных и выходных данных.
- **Тестирование на основе синтаксиса** (syntax testing[269]) — техника тестирования (по методу чёрного ящика), в которой тест-кейсы создаются на основе определения наборов входных и выходных данных.
- **Комбинаторные техники или комбинаторное тестирование** (combinatorial testing[270]) — способ выбрать подходящий набор комбинаций тестовых данных для достижения установленного уровня тестового покрытия в случае, когда проверка всех возможных наборов значений тестовых данных невозможна за имеющееся время. Существуют следующие комбинаторные техники:
  - **Тестирование всех комбинаций** (all combinations testing[271]) — тестирование всех возможных комбинаций всех значений всех тестовых данных (например, всех параметров функции).
  - **Попарное тестирование** (рассмотрено ранее{90}).
  - **Тестирование с выбором значений-представителей** (each choice testing[272]) — тестирование, при котором по одному значению из каждого набора тестовых данных должно быть использовано хотя бы в одном тест-кейсе.
  - **Тестирование с выбором базового набора значений** (base choice testing[273]) — тестирование, при котором выделяется набор значений (базовый набор), который используется для проведения тестирования в первую очередь, а далее тест-кейсы строятся на основе выбора всех базовых значений, кроме одного, которое заменяется значением, не входящим в базовый набор.

    Также см. классификацию тестирования на основе выбора входных данных{89}, которая расширяет и дополняет данный список.
- **Тестирование по графу причинно-следственных связей** (cause-effect graphing[274]) — техника тестирования (по методу чёрного ящика), в которой тест-кейсы разрабатываются на основе графа причинно-следственных связей (графического представления входных данных и воздействий со связанными с ними выходными данными и эффектами).

[267] **Classification tree.** A tree showing equivalence partitions hierarchically ordered, which is used to design test cases in the classification tree method. [ISTQB Glossary]

[268] **Classification tree method.** A black box test design technique in which test cases, described by means of a classification tree, are designed to execute combinations of representatives of input and/or output domains. [ISTQB Glossary]

[269] **Syntax testing.** A black box test design technique in which test cases are designed based upon the definition of the input domain and/or output domain. [ISTQB Glossary]

[270] **Combinatorial testing.** A means to identify a suitable subset of test combinations to achieve a predetermined level of coverage when testing an object with multiple parameters and where those parameters themselves each have several values, which gives rise to more combinations than are feasible to test in the time allowed. [ISTQB Glossary]

[271] **All combinations testing.** Testing of all possible combinations of all values for all parameters. [«Guide to advanced software testing, 2nd edition», Anne Matte Hass].

[272] **Each choice testing.** One value from each block for each partition must be used in at least one test case. [«Introduction to Software Testing. Chapter 4. Input Space Partition Testing», Paul Ammann & Jeff Offutt]

[273] **Base choice testing.** A base choice block is chosen for each partition, and a base test is formed by using the base choice for each partition. Subsequent tests are chosen by holding all but one base choice constant and using each non-base choice in each other parameter. [«Introduction to Software Testing. Chapter 4. Input Space Partition Testing», Paul Ammann & Jeff Offutt]

[274] **Cause-effect graphing.** A black box test design technique in which test cases are designed from cause-effect graphs (a graphical representation of inputs and/or stimuli (causes) with their associated outputs (effects), which can be used to design test cases). [ISTQB Glossary]

- **Тестирование по потоку данных** (data-flow testing[275]) — семейство техник тестирования, основанных на выборе отдельных путей из потока управления с целью исследования событий, связанных с изменением состояния переменных. Эти техники позволяют обнаружить такие ситуации, как: переменная определена, но нигде не используется; переменная используется, но не определена; переменная определена несколько раз до того, как она используется; переменная удалена до последнего случая использования.

  Здесь придётся немного погрузиться в теорию. Над переменной в общем случае может выполняться несколько действий (покажем на примере переменной **x**):
  - объявление (declaration): int **x**;
  - определение (definition, d-use): **x** = 99;
  - использование в вычислениях (computation use, c-use): z = **x** + 1;
  - использование в условиях (predicate use, p-use): if (**x** > 17) { … };
  - удаление (kill, k-use): **x** = null;

  Теперь можно рассмотреть техники тестирования на основе потока данных. Они крайне подробно описаны в разделе 3.3 главы 5 книги Бориса Бейзера «Техники тестирования ПО» («Software Testing Techniques, Second Edition», Boris Beizer), мы же приведём очень краткие пояснения:
  - **Проверка использования всех объявлений** (all-definitions testing[276]) — тестовым набором проверяется, что для каждой переменной существует путь от её определения к её использованию в вычислениях или условиях.
  - **Проверка всех вычислений на основе всех объявлений** (all-c-uses testing[277]) — тестовым набором проверяется, что для каждой переменной существует путь от каждого её определения к её использованию в вычислениях.
  - **Проверка всех ветвлений на основе всех объявлений** (all-p-uses testing[278]) — тестовым набором проверяется, что для каждой переменной существует путь от каждого её определения к её использованию в условиях.
  - **Проверка всех вычислений и ветвлений на основе всех объявлений** (all-uses testing[279]) — тестовым набором проверяется, что для каждой переменной существует хотя бы один путь от каждого её определения к каждому её использованию в вычислениях и в условиях.
  - **Проверка использования всех объявлений и всех путей без переобъявлений (без циклов или с однократными повторениями циклов)** (all-du-paths testing[280]) — тестовым набором для каждой переменной проверяются все пути от каждого её определения к каждому её использованию в вычислениях и в условиях (самая мощная стратегия, которая в то же время требует наибольшего количества тест-кейсов).

[275] **Data flow testing.** A white box test design technique in which test cases are designed to execute definition-use pairs of variables. [ISTQB Glossary]

[276] **All-definitions strategy.** Test set requires that every definition of every variable is covered by at least one use of that variable (c-use or p-use). [«Software Testing Techniques, Second Edition», Boris Beizer]

[277] **All-computation-uses strategy.** For every variable and every definition of that variable, include at least one definition-free path from the definition to every computation use. [«Software Testing Techniques, Second Edition», Boris Beizer]

[278] **All-predicate-uses strategy.** For every variable and every definition of that variable, include at least one definition-free path from the definition to every predicate use. [«Software Testing Techniques, Second Edition», Boris Beizer]

[279] **All-uses strategy.** Test set includes at least one path segment from every definition to every use that can be reached by that definition. [«Software Testing Techniques, Second Edition», Boris Beizer]

[280] **All-DU-path strategy.** Test set includes every du path from every definition of every variable to every use of that definition. [«Software Testing Techniques, Second Edition», Boris Beizer]

Для лучшего понимания и запоминания приведём оригинальную схему из книги Бориса Бейзера (там она фигурирует под именем «Figure 5.7. Relative Strength of Structural Test Strategies»), показывающую взаимосвязь стратегий тестирования на основе потока данных (рисунок 2.3.m).

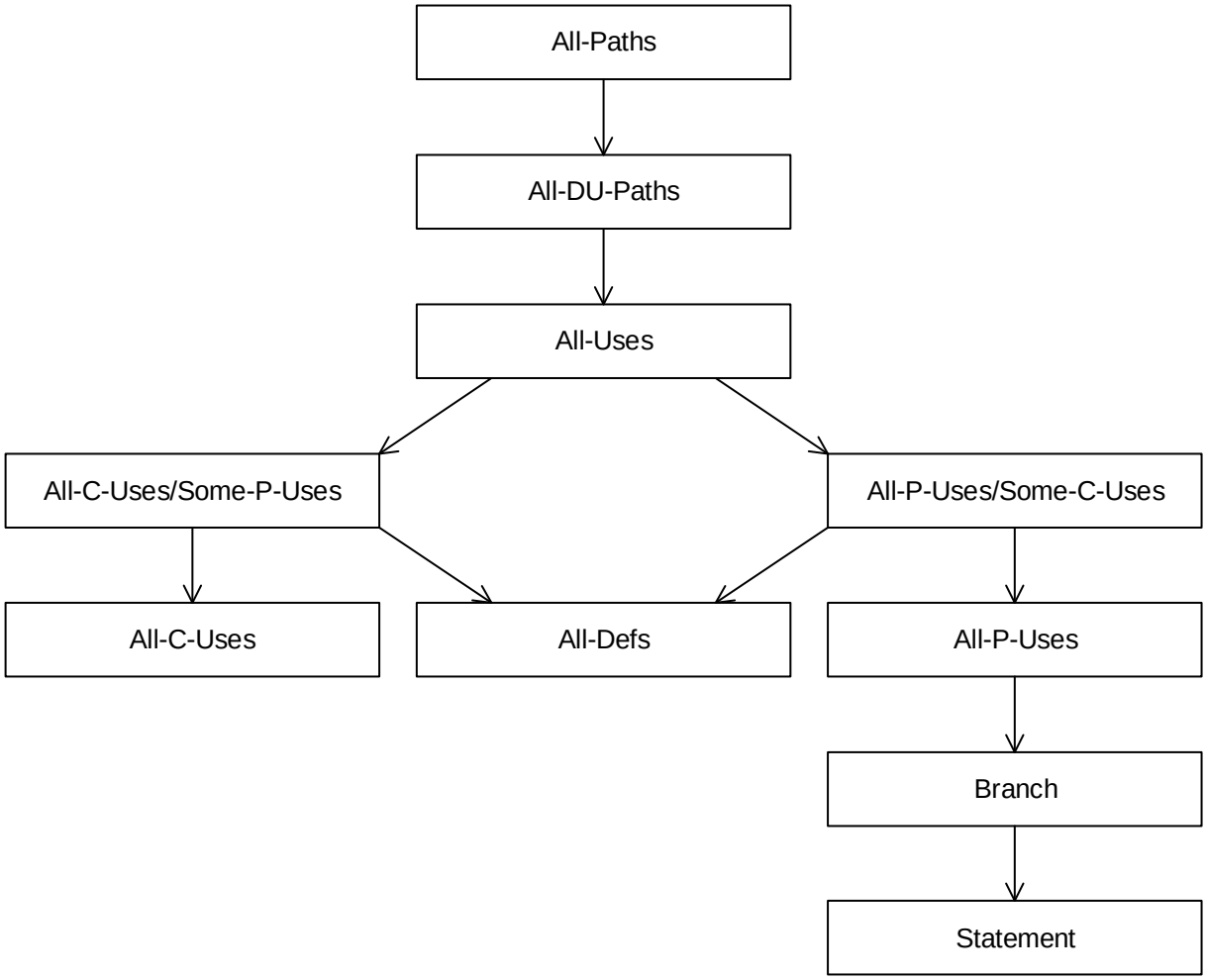

Рисунок 2.3.m — Взаимосвязь и относительная мощность стратегий тестирования на основе потока данных (по книге Бориса Бейзера «Техники тестирования ПО»)

2.3.4. Классификация по принадлежности к тестированию по методу белого и чёрного ящиков

Типичнейшим вопросом на собеседовании для начинающих тестировщиков является просьба перечислить техники тестирования по методу белого и чёрного ящиков. Ниже представлена таблица 2.3.d, в которой все вышерассмотренные виды тестирования соотнесены с соответствующим методом. Эту таблицу можно использовать также как справочник по видам тестирования (они представлены в той же последовательности, в какой описаны в данной главе).

Важно! В источниках наподобие ISTQB-глоссария многие виды и техники тестирования жёстко соотнесены с методами белого или чёрного ящика. Но это не значит, что их невозможно применить в другом, не отмеченном методе. Так, например, тестирование на основе классов эквивалентности отнесено к методу чёрного ящика, но оно прекрасно подходит и для написания модульных тест-кейсов, являющихся ярчайшими представителями тестирования по методу белого ящика.

Воспринимайте данные из представленной ниже таблицы не как «этот вид тестирования может применяться только для…», а как «чаще всего этот вид тестирования применяется для…»

Таблица 2.3.d — Виды и техники тестирования в контексте методов белого и чёрного ящиков

Вид тестирования (русскоязычное название)	Вид тестирования (англоязычное название)	Белый ящик	Чёрный ящик
Статическое тестирование{68}	Static testing	Да	Нет
Динамическое тестирование{68}	Dynamic testing	Изредка	Да
Ручное тестирование{70}	Manual testing	Мало	Да
Автоматизированное тестирование{71}	Automated testing	Да	Да
Модульное (компонентное) тестирование{72}	Unit testing, Module testing, Component testing	Да	Нет
Интеграционное тестирование{72}	Integration testing	Да	Да
Системное тестирование{73}	System testing	Мало	Да
Дымовое тестирование{74}	Smoke test, Intake test, Build verification test	Мало	Да
Тестирование критического пути{75}	Critical path test	Мало	Да
Расширенное тестирование{76}	Extended test	Мало	Да
Позитивное тестирование{77}	Positive testing	Да	Да
Негативное тестирование{77}	Negative testing, Invalid testing	Да	Да
Тестирование веб-приложений{78}	Web-applications testing	Да	Да
Тестирование мобильных приложений{78}	Mobile applications testing	Да	Да
Тестирование настольных приложений{78}	Desktop applications testing	Да	Да

Тестирование уровня представления[78]	Presentation tier testing	Мало	Да
Тестирование уровня бизнес-логики[78]	Business logic tier testing	Да	Да
Тестирование уровня данных[78]	Data tier testing	Да	Мало
Альфа-тестирование[79]	Alpha testing	Мало	Да
Бета-тестирование[79]	Beta testing	Почти никогда	Да
Гамма-тестирование[79]	Gamma testing	Почти никогда	Да
Тестирование на основе тест-кейсов[79]	Scripted testing, Test case based testing	Да	Да
Исследовательское тестирование[80]	Exploratory testing	Нет	Да
Свободное (интуитивное) тестирование[80]	Ad hoc testing	Нет	Да
Функциональное тестирование[80]	Functional testing	Да	Да
Нефункциональное тестирование[81]	Non-functional testing	Да	Да
Инсталляционное тестирование[81]	Installation testing	Изредка	Да
Регрессионное тестирование[82]	Regression testing	Да	Да
Повторное тестирование[82]	Re-testing, Confirmation testing	Да	Да
Приёмочное тестирование[82]	Acceptance testing	Крайне редко	Да
Операционное тестирование[83]	Operational testing	Крайне редко	Да
Тестирование удобства использования[83]	Usability testing	Крайне редко	Да
Тестирование доступности[83]	Accessibility testing	Крайне редко	Да
Тестирование интерфейса[83]	Interface testing	Да	Да
Тестирование безопасности[84]	Security testing	Да	Да
Тестирование интернационализации[84]	Internationalization testing	Мало	Да
Тестирование локализации[84]	Localization testing	Мало	Да
Тестирование совместимости[84]	Compatibility testing	Мало	Да
Конфигурационное тестирование[84]	Configuration testing	Мало	Да
Кросс-браузерное тестирование[85]	Cross-browser testing	Мало	Да
Тестирование данных и баз данных[85]	Data quality testing and Database integrity testing	Да	Мало
Тестирование использования ресурсов[85]	Resource utilization testing	Крайне редко	Да
Сравнительное тестирование[86]	Comparison testing	Нет	Да

Демонстрационное тестирование[86]	Qualification testing	Нет	Да
Избыточное тестирование[86]	Exhaustive testing	Крайне редко	Нет
Тестирование надёжности[86]	Reliability testing	Крайне редко	Да
Тестирование восстанавливаемости[86]	Recoverability testing	Крайне редко	Да
Тестирование отказоустойчивости[86]	Failover testing	Крайне редко	Да
Тестирование производительности[86]	Performance testing	Крайне редко	Да
Нагрузочное тестирование[86]	Load testing, Capacity testing	Крайне редко	Да
Тестирование масштабируемости[87]	Scalability testing	Крайне редко	Да
Объёмное тестирование[87]	Volume testing	Крайне редко	Да
Стрессовое тестирование[87]	Stress testing	Крайне редко	Да
Конкурентное тестирование[87]	Concurrency testing	Крайне редко	Да
Инвазивное тестирование[88]	Intrusive testing	Да	Да
Неинвазивное тестирование[88]	Nonintrusive testing	Да	Да
Тестирование под управлением данными[88]	Data-driven testing	Да	Да
Тестирование под управлением ключевыми словами[88]	Keyword-driven testing	Да	Да
Тестирование предугадыванием ошибок[89]	Error guessing	Крайне редко	Да
Эвристическая оценка[89]	Heuristic evaluation	Нет	Да
Мутационное тестирование[89]	Mutation testing	Да	Да
Тестирование добавлением ошибок[89]	Error seeding	Да	Да
Тестирование на основе классов эквивалентности[89]	Equivalence partitioning	Да	Да
Тестирование на основе граничных условий[90]	Boundary value analysis	Да	Да
Доменное тестирование[90]	Domain testing, Domain analysis	Да	Да
Попарное тестирование[90]	Pairwise testing	Да	Да
Тестирование на основе ортогональных массивов[90]	Orthogonal array testing	Да	Да
Тестирование в процессе разработки[91]	Development testing	Да	Да
Тестирование по потоку управления[91]	Control flow testing	Да	Нет
Тестирование по потоку данных[91]	Data flow testing	Да	Нет
Тестирование по диаграмме или таблице состояний[92]	State transition testing	Изредка	Да
Инспекция (аудит) кода[92]	Code review, code inspection	Да	Нет

Тестирование на основе выражений[92]	Statement testing	Да	Нет
Тестирование на основе ветвей[92]	Branch testing	Да	Нет
Тестирование на основе условий[93]	Condition testing	Да	Нет
Тестирование на основе комбинаций условий[93]	Multiple condition testing	Да	Нет
Тестирование на основе отдельных условий, порождающих ветвление[93] («решающих условий»)	Modified condition decision coverage testing	Да	Нет
Тестирование на основе решений[93]	Decision testing	Да	Нет
Тестирование на основе путей[93]	Path testing	Да	Нет
Тестирование по таблице принятия решений[94]	Decision table testing	Да	Да
Тестирование по моделям поведения приложения[94]	Model-based testing	Да	Да
Тестирование на основе вариантов использования[94]	Use case testing	Да	Да
Параллельное тестирование[95]	Parallel testing	Да	Да
Тестирование на основе случайных данных[95]	Random testing	Да	Да
A/B-тестирование[95]	A/B testing, Split testing	Нет	Да
Восходящее тестирование[96]	Bottom-up testing	Да	Да
Нисходящее тестирование[96]	Top-down testing	Да	Да
Гибридное тестирование[96]	Hybrid testing	Да	Да
Тестирование на основе дерева классификаций[102]	Classification tree method	Да	Да
Тестирование на основе синтаксиса[102]	Syntax testing	Да	Да
Комбинаторные техники[102] (комбинаторное тестирование)	Combinatorial testing	Да	Да
Тестирование всех комбинаций[102]	All combinations testing	Да	Нет
Тестирование с выбором значений-представителей[102]	Each choice testing	Да	Нет
Тестирование с выбором базового набора значений[102]	Base choice testing	Да	Нет
Тестирование по графу причинно-следственных связей[102]	Cause-effect graphing	Мало	Да
Проверка использования всех объявлений[103]	All-definitions testing	Да	Нет

Проверка всех вычислений на основе всех объявлений[103]	All-c-uses testing	Да	Нет
Проверка всех ветвлений на основе всех объявлений[103]	All-p-uses testing	Да	Нет
Проверка всех вычислений и ветвлений на основе всех объявлений[103]	All-uses testing	Да	Нет
Проверка использования всех объявлений и всех путей без переобъявлений[103] (без циклов или с однократными повторениями циклов)	All-du-paths testing	Да	Нет