ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

ПРЕДВАРИТЕЛЬНЫЙ НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПНСТ 965— 2024

Системная и программная инженерия

ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Часть 11

Тестирование систем искусственного интеллекта

(ISO/IEC TR 29119-11:2020, NEQ)

Издание официальное

Москва
Российский институт стандартизации
2024

Предисловие

1 РАЗРАБОТАН Федеральным государственным автономным образовательным учреждением высшего образования «Национальный исследовательский университет «Высшая школа экономики» (НИУ ВШЭ)

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 164 «Искусственный интеллект»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 28 октября 2024 г. № 73-пнст

4 Настоящий стандарт разработан с учетом основных нормативных положений международного документа ISO/IEC TR 29119-11:2020 «Системная и программная инженерия. Тестирование программного обеспечения. Часть 11. Руководство по тестированию систем на основе искусственного интеллекта» (ISO/IEC TR 29119-11:2020 «Software and systems engineering — Software testing — Part 11: Guidelines on the testing of AI-based systems», NEQ)

Правила применения настоящего стандарта и проведения мониторинга его использования установлены в ГОСТ Р 1.16—2011 (разделы 5 и 6).

Федеральное агентство по техническому регулированию и метрологии собирает сведения о практическом применении настоящего стандарта. Данные сведения, а также замечания и предложения по содержанию стандарта можно направить не позднее чем за 4 мес до истечения срока его действия разработчику настоящего стандарта по адресу: info@tc164.ru и/или в Федеральное агентство по техническому регулированию и метрологии по адресу: 123112 Москва, Пресненская набережная, д. 10, стр. 2.

В случае отмены настоящего стандарта соответствующая информация будет опубликована в ежемесячном информационном указателе «Национальные стандарты» и также будет размещена на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (www.rst.gov.ru)

Содержание

1 Область применения 1
2 Нормативные ссылки 1
3 Термины и определения 2
4 Введение в ИИ и тестирование ПО 5
5 Тестирование систем машинного обучения 7
6 Тестирование экспертных систем 25
7 Тестирование логических программ 29
Приложение А (справочное) Краткая характеристика экспертных систем 32
Приложение Б (справочное) Введение в тестирование ПО 33
Библиография 38

Введение

Цель серии стандартов на системную и программную инженерию заключается в определении согласованных на международном уровне стандартов для тестирования программного обеспечения (ПО), которые могут использоваться любой организацией при проведении тестирования ПО.

В настоящем стандарте приведены разъяснения о применимости серии стандартов на системную и программную инженерию для тестирования систем на основе искусственного интеллекта (ИИ), содержащих один или несколько компонентов ИИ.

В серии стандартов на системную и программную инженерию описаны основные подходы к тестированию ПО. Основу настоящего стандарта составляют подходы, установленные в ГОСТ Р 56920.

Положения ГОСТ Р 56921 содержат описание процессов, включая тестирование ПО на организационном уровне, управление тестированием и динамическое тестирование. ГОСТ Р 56921 применяют при динамическом тестировании, функциональном и нефункциональном тестировании, ручном и автоматизированном тестировании, а также тестировании по сценарию и без него и используют для тестирования систем с любым ПО, включая системы ИИ.

Положения ГОСТ Р 56922 определяют перечень документов по тестированию ПО. Требования к шаблонам и примерам тестовой документации, представленные в ГОСТ Р 56922, применяют при разработке тестовой документации для любой системы ИИ.

Положения [1] определяют подходы к разработке тестов, которые применяют для тестирования систем и компонентов ИИ.

В [2] установлены положения в отношении автоматизированного тестирования на основе ключевых слов.

В настоящем стандарте описано применение ГОСТ Р 56921 для тестирования систем ИИ с использованием конкретных технологий ИИ и показано, как шаблоны тестовой документации, установленные в ГОСТ Р 56922, могут быть реализованы при тестировании систем или компонентов ИИ.

ПРЕДВАРИТЕЛЬНЫЙ НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системная и программная инженерия

ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Часть 11

Тестирование систем искусственного интеллекта

Software and systems engineering. Software testing. Part 11. Guidelines on the testing of AI-based systems

Срок действия — с 2025—01—01
до 2026—01—01

1 Область применения

Настоящий стандарт содержит требования и руководство по применению серии стандартов на системную и программную инженерию для тестирования систем на основе искусственного интеллекта и их компонентов. В стандарте применяется подход, основанный на оценке рисков при разработке и обслуживании систем ИИ для определения подходящих методов тестирования.

Стандарт определяет методы в виде практик, подходов и методик испытаний, которые могут быть применены к системам ИИ и компонентам. В тех случаях, когда подходы и методы испытаний указаны в серии стандартов на системную и программную инженерию, данный стандарт определяет дополнительные подробности и описывает их применение в отношении систем ИИ.

Настоящий стандарт предназначен для применения при тестировании различных систем ИИ, включая отдельные компоненты системы ИИ, и, в случае необходимости, при проверке взаимодействия компонентов, не связанных с ИИ, с компонентами с использованием ИИ.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты:

ГОСТ Р 56921/ISO/IEC/IEEE 29119-2:2013 Системная и программная инженерия. Тестирование программного обеспечения. Часть 2. Процессы тестирования

ГОСТ Р 56922/ISO/IEC/IEEE 29119-3:2013 Системная и программная инженерия. Тестирование программного обеспечения. Часть 3. Документация тестирования

ГОСТ Р 71476 (ИСО/МЭК 22989:2022) Искусственный интеллект. Концепции и терминология искусственного интеллекта

ГОСТ Р ИСО/МЭК 25010 Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов

Примечание — При пользовании настоящим стандартом целесообразно проверить действие ссылочных стандартов в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет или по ежегодному информационному указателю «Национальные стандарты», который опубликован по состоянию на 1 января текущего года, и по выпускам ежемесячного информационного указателя «Национальные стандарты» за текущий год. Если заменен ссылочный стандарт, на который дана недатированная ссылка, то рекомендуется использовать действующую версию этого стандарта с учетом всех внесенных в данную версию изменений. Если заменен ссылочный стандарт, на который дана датированная ссылка, то рекомендуется использовать версию этого стандарта с указанным выше годом

Издание официальное

утверждения (принятия). Если после утверждения настоящего стандарта в ссылочный стандарт, на который дана датированная ссылка, внесено изменение, затрагивающее положение, на которое дана ссылка, то это положение рекомендуется применять без учета данного изменения. Если ссылочный стандарт отменен без замены, то положение, в котором дана ссылка на него, рекомендуется применять в части, не затрагивающей эту ссылку.

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

3.1

подотчетность (accountability): Состояние подотчетности (3.5.1).

Примечание 1 — Подотчетность связана с установленной ответственностью. Ответственность может быть основана на регламенте, соглашении или посредством поручения в рамках делегирования.

Примечание 2 — Подотчетность подразумевает, что физическое или юридическое лицо несет ответственность за что-либо перед другим физическим или юридическим лицом с помощью определенных средств и в соответствии с определенными критериями.

[ГОСТ Р 71476—2024, статья 3.5.2]

3.2

искусственный интеллект <дисциплина> (artificial intelligence): Исследование и разработка механизмов и приложений ИИ-систем (3.1.4).

Примечание — Исследования и разработки могут проводиться в одной или нескольких областях, таких как информатика, наука о данных, гуманитарные науки, математика и естественные науки.

[ГОСТ Р 71476—2024, статья 3.1.3]

3.3

система искусственного интеллекта; система ИИ (artificial intelligence system, AI system): Техническая система, которая порождает такие конечные результаты, как контент, прогнозы, рекомендации или решения для заданного набора определенных человеком целей.

Примечание 1 — В технической системе могут применяться различные связанные с искусственным интеллектом (3.1.3) методы и подходы для разработки модели (3.1.23) для представления данных, знаний (3.1.21), процессов и т. д., которая может быть использована для решения задач (3.1.35).

Примечание 2 — ИИ-системы проектируются для эксплуатации с различными уровнями автоматизации (3.1.7).

[ГОСТ Р 71476—2024, статья 3.1.4]

3.4

автономность (autonomy): Характеристика системы, которая показывает способность системы изменять свою предполагаемую область использования или цель без внешнего вмешательства, контроля или надзора.

[ГОСТ Р 71476—2024, статья 3.1.5]

3.5

доступность (availability): Свойство быть доступным и пригодным для использования по требованию уполномоченного субъекта.

[ГОСТ Р 71476—2024, статья 3.5.3]

3.6

смещенность (bias): Систематическое различие в обработке определенных объектов, людей или групп по сравнению с другими.

Примечание 1 — Обработка — это любой вид действия, включая восприятие, наблюдение, представление, предсказание (3.1.27) или решение.

[ГОСТ Р 71476—2024, статья 3.5.4]

3.7

контроль (control): Целенаправленное действие над процессом или в процессе для достижения определенных целей.

[ГОСТ Р 71476—2024, статья 3.5.5]

3.8

контролируемость (controllability): Свойство системы ИИ (3.1.4), позволяющее человеку или другому внешнему агенту вмешиваться в функционирование системы.

[ГОСТ Р 71476—2024, статья 3.5.6]

3.9

объяснимость (explainability): Свойство системы ИИ (3.1.4) выражать важные факторы, влияющие на результаты работы системы ИИ (3.1.4) способом, понятным человеку.

Примечание 1 — Предполагается ответить на вопрос «почему?», не пытаясь утверждать, что принятый курс действий обязательно был оптимальным.

[ГОСТ Р 71476—2024, статья 3.5.7]

3.10

машинное обучение (machine learning): Процесс оптимизации параметров модели (3.3.8) с помощью вычислительных методов таким образом, чтобы поведение модели (3.1.23) отражало данные и/или опыт.

[ГОСТ Р 71476—2024, статья 3.3.5]

3.11

предсказуемость (predictability): Свойство системы ИИ (3.1.4), которое позволяет заинтересованным сторонам (3.5.13) делать надежные предположения о выходных результатах работы.

[ГОСТ Р 71476—2024, статья 3.5.8]

3.12

надежность (reliability): Свойство согласованного предполагаемого поведения и результатов.

[ГОСТ Р 71476—2024, статья 3.5.9]

3.13

отказоустойчивость (resilience): Способность системы быстро восстанавливать рабочее состояние после инцидента.

[ГОСТ Р 71476—2024, статья 3.5.10]

3.14

риск (risk): Влияние неопределенности на цели.

Примечание 1 — Эффект — это отклонение от ожидаемого. Оно может быть положительным, отрицательным или и тем и другим и может устранять, создавать или приводить к возникновению возможностей и угроз (3.39).

Примечание 2 — Цели могут иметь различные аспекты и категории и могут применяться на разных уровнях.

Примечание 3 — Риск обычно выражается через источники риска, потенциальные события, их последствия и их вероятности.

[ГОСТ Р 71476—2024, статья 3.5.11]

3.15

робастность (robustness): Способность системы сохранять свой уровень производительности при любых обстоятельствах.

[ГОСТ Р 71476—2024, статья 3.5.12]

3.16

тестирование (testing): Набор операций, проводимых для обеспечения выявления и/или оценки свойств одного или более элементов тестирования.

Примечание — Действия тестирования могут включать в себя планирование, подготовку, выполнение, создание отчетов и менеджмент, поскольку все они направлены на тестирование.

[ГОСТ Р 56921—2016, статья 4.70]

3.17

прозрачность <организация> (transparency): Свойство организации, заключающееся в том, что соответствующие действия и решения доводятся до сведения соответствующим заинтересованным сторонам (3.5.13) всеобъемлющим, доступным и понятным образом.

Примечание — Ненадлежащее информирование о деятельности и решениях может нарушить требования безопасности, приватности или требования конфиденциальности.

[ГОСТ Р 71476—2024, статья 3.5.14]

3.18

прозрачность <система> (transparency): Свойство системы, заключающееся в том, что соответствующая информация о системе становится доступной для соответствующих заинтересованных сторон (3.5.13).

Примечание 1 — Соответствующая информация для обеспечения прозрачности системы может включать такие аспекты, как характеристики, ограничения, компоненты, процедуры, меры, цели проектирования, варианты проектирования и допущения, источники данных и протоколы маркировки.

Примечание 2 — Ненадлежащее раскрытие некоторых аспектов системы может нарушить требования безопасности, приватности или требования конфиденциальности.

[ГОСТ Р 71476—2024, статья 3.5.15]

3.19

доверенность (trustworthiness): Способность удовлетворять ожиданиям заинтересованных сторон (3.5.13) проверяемым способом.

Примечание 1 — В зависимости от контекста или сектора, а также от конкретного продукта или услуги, используемых данных и технологий, применяются различные характеристики, которые необходимо проверить, чтобы обеспечить соответствие ожиданиям заинтересованных сторон (3.5.13).

Примечание 2 — Характеристики доверенности включают, например, надежность, доступность, устойчивость, конфиденциальность, безопасность, подотчетность, прозрачность, целостность, подлинность, качество и удобство использования.

Примечание 3 — Доверенность — это атрибут, который может быть применен к услугам, продуктам, технологиям, данным и информации, а также, в контексте управления, к организациям.

[ГОСТ Р 71476—2024, статья 3.5.16]

3.20

валидация (validation): Подтверждение посредством предоставления объективных доказательств того, что требования для конкретного предполагаемого использования или применения были выполнены.

[ГОСТ Р 71476—2024, статья 3.5.18]

3.21

верификация (verification): Подтверждение посредством предоставления объективных доказательств того, что установленные требования были выполнены.

Примечание — Верификация обеспечивает только уверенность в том, что продукт соответствует своей спецификации.

[ГОСТ Р 71476—2024, статья 3.5.17]

4 Введение в ИИ и тестирование ПО

4.1 Общие положения

В настоящем разделе приведены общие сведения об ИИ и тестировании ПО.

В приложении А изложено описание экспертных систем как разновидности систем ИИ, в приложении Б — описание тестирования ПО.

4.2 Жизненный цикл системы ИИ

На рисунке 1 показан жизненный цикл системы ИИ в соответствии с ГОСТ Р 71476, а также его связь с тестированием ПО.

Рисунок 1 — Пример этапов жизненного цикла ИИ и тестирования ПО

4.3 Технологии ИИ

4.4 Тестирование на основе рисков

Тестирование на основе рисков положено в основу серии стандартов на системную и программную инженерию. Предполагают, что риски являются основным фактором для определения содержания и стратегии теста, а также последующего тестирования ПО. Все испытания компонентов систем ИИ должны выполняться в рамках подхода, основанного на оценке рисков, в соответствии с требованиями ГОСТ Р 56921.

Для проведения тестирования на основе рисков необходимо определить риски. Определение рисков для поддержки систем и компонентов ИИ должно проводиться в соответствии с [3]. Для выполнения тестирования необходимо идентифицировать риски. Идентификация рисков для поддержки систем и компонентов ИИ должна проводиться в соответствии с [3].

Тестирование на основе рисков описано в Б.6.

4.5 Процесс тестирования

4.5.1 Общие положения

Процесс тестирования представляет собой совокупность взаимосвязанных действий, которые преобразуют входные данные в выходные. Процесс тестирования следует анализировать отдельно от процесса разработки (и других процессов), т. к. они могут быть описаны в процессе тестирования.

Тестирование систем и компонентов ИИ должно выполняться в соответствии с ГОСТ Р 56921. Организации могут дополнить процесс тестирования по ГОСТ Р 56921 дополнительными действиями, процедурами и практиками при необходимости.

4.5.2 Тестирование в контексте жизненного цикла системы ИИ

На рисунке 2 показан пример модели жизненного цикла системы ИИ в соответствии с ГОСТ Р 71476 в процессе тестирования в соответствии с ГОСТ Р 56921.

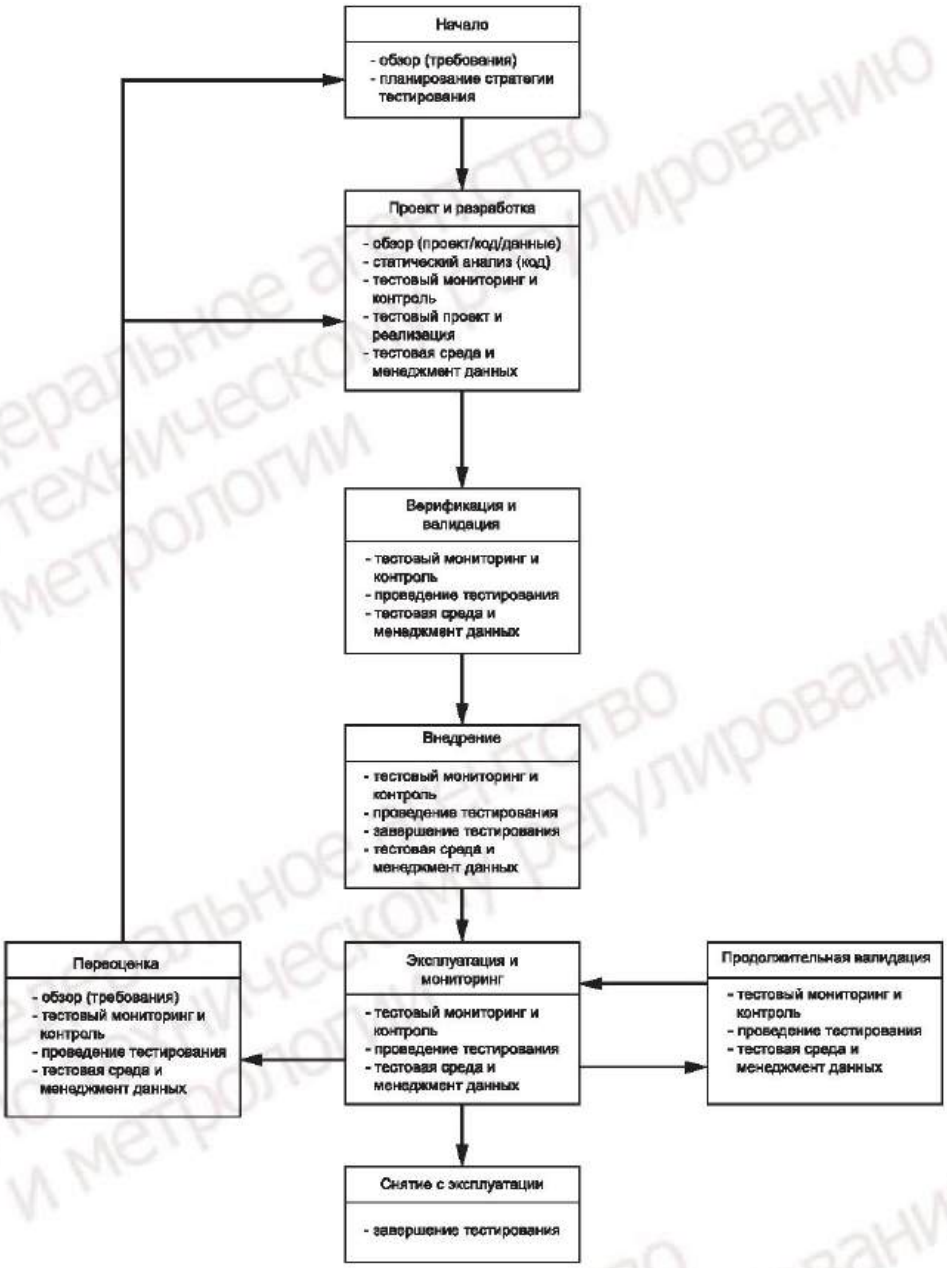

Рисунок 2 — Пример модели жизненного цикла системы ИИ в процессе тестирования

4.6 Документация по тестированию

Документацию по тестированию создают в результате процесса тестирования в соответствии с ГОСТ Р 56922. Полученные выходные данные могут быть переданы заинтересованной стороне (например, отчеты о состоянии тестирования могут быть интересны для менеджера проекта) или использованы для организации других процессов (например, план тестирования применяют для управления динамическим тестированием). Поскольку документация по тестированию является результатом процесса, выходные данные могут быть классифицированы на трех уровнях процесса тестирования.

ГОСТ Р 56922 не устанавливает обязательных требований к документации.

Положения ГОСТ Р 56922 определяют информацию, подлежащую записи, но не содержат требований по использованию специальных наименований и терминологии или созданию конкретных

документов (информация может храниться в нескольких документах или быть объединена в один документ). Документы могут быть не только на бумажном носителе, а информация может храниться в электронном виде или на одном из инструментов автоматизации тестирования.

Документация по тестированию систем и компонентов ИИ должна быть оформлена в соответствии с ГОСТ Р 56922.

5 Тестирование систем машинного обучения

5.1 Введение в тестирование систем машинного обучения

В настоящем разделе изложена информация об обеспечении качества, проблемах тестирования и возможностях машинного обучения (МО).

5.1.1 Уровни тестирования СМО

Тестирование СМО выполняют на следующих трех уровнях:

- тестирование входных данных (см. 5.2). Тестирование связано с предоставлением данных как для обучения модели, так и для генерации моделью результата в операционной среде;
- тестирование модели (см. 5.3). Тестирование связано с проверкой моделей МО, полученных в результате операционного процесса;
- тестирование среды разработки (см. 5.4). Тестирование связано с алгоритмами и средой МО, а также поддержкой оценки, настройки, проверки деятельности по МО в условиях разработки среды для МО.

Входные данные, модель и среда подлежат тестированию.

5.1.2 Тестирование моделей МО на основе рисков

Тестирование СМО должно быть основано на рисках. Риски подлежат выявлению и анализу с последующим присвоением уровня. Идентификацию рисков выполняют в соответствии с [3]. Примеры рисков на каждом из уровней тестирования приведены в 5.2.2 (входные данные), 5.3.2 (модель МО) и 5.4.2 (среда разработки). Риски могут быть классифицированы в зависимости от продукта или проекта (см. Б.6.2).

В процессе выявления и классификации рисков должны быть определены приемлемые способы их устранения. При проведении тестирования устранение рисков включает применение разных типов и методов тестирования. Типы и методы тестирования для устранения рисков на каждом из уровней тестирования приведены в 5.2.3 (входные данные), 5.3.3 (модель МО) и 5.4.3 (среда разработки). В ряде случаев типы и методы тестирования применимы, тогда как в некоторых случаях наиболее эффективные подходы к устранению рисков не включают тестирование (например, такие как принятие риска, передача риска или составление плана действий в чрезвычайной ситуации).

5.2 Тестирование входных данных

5.2.1 Введение в тестирование входных данных

Тестирование входных данных проводят для обеспечения репрезентативности данных, их соответствия алгоритму и моделям МО и использования в пригодном формате (данные должны быть точными и корректными). Тестирование также включает анализ производственного процесса и процесса разработки. Проверка, проводимая в ходе тестирования входных данных, включает как статическое тестирование (например, анализ наборов данных), так и динамическое тестирование (например, тестирование конвейера аналитики данных).

5.2.2 Риски, связанные с входными данными

Риски, связанные с входными данными, разделены на следующие категории для удобства документирования (допустимы другие категории):

а) риски необъективности обучающих данных — см. 5.2.3.9 для устранения таких рисков;

б) риски при сборе данных — см. 5.2.3.4 для устранения таких рисков:

1) данные, полученные из ненадежных источников;
2) небезопасные каналы ввода данных;

в) риски управления данными — см. 5.2.3.2 для устранения таких рисков;

г) риски конвейера данных — см. 5.2.3.3 для устранения таких рисков:

1) ошибка управления конфигурацией;
2) конструктивный дефект;
3) ошибка реализации;

д) риски наборов данных:

1) несбалансированность данных из-за недостаточного охвата всех целевых классов — см. 5.2.3.6 для устранения таких рисков;

2) противоречивые данные — см. 5.2.3.7 для устранения таких рисков;

3) искажения при аугментации данных — см. 5.2.3.5 для устранения таких рисков;

4) неоптимальный выбор признаков — см. 5.2.3.8 для устранения таких рисков;

е) примеры рисков — см. 5.2.3.5 для устранения таких рисков:

1) неполные данные;

2) неверный тип данных;

3) данные вне диапазона;

4) выпадение данных;

5) некорректно маркированные данные — см. 5.2.3.9 для устранения таких рисков;

ж) нерепрезентативные обучающие данные — см. 5.2.3.5 для устранения таких рисков:

1) анализ вариантов использования;

2) наборы данных не покрывают все пространство данных.

Не все риски могут быть отнесены к каждой системе или компоненту ИИ. Некоторые риски (не перечислены выше) могут возникать ввиду особенностей отдельных систем и/или компонентов.

5.2.3 Типы и методы тестирования входных данных

5.2.3.1 Общие положения

Указанные типы и методы тестирования используют для устранения рисков, перечисленных в 5.2.2.

Перечисленные типы и методы тестирования не являются исчерпывающими, другие типы и методы применимы в зависимости от ситуации.

5.2.3.2 Тестирование вопросов управления данными

Статическое тестирование, обычно в форме обзора, позволяет определить нарушения внутренней политики, стандартов, правил или положений (например, выявить нарушение запрета на использование внутренних конфиденциальных данных). Необходимо принимать во внимание положения о конфиденциальности и защите данных, коммерческой тайны и прав интеллектуальной собственности.

Подробная информация приведена в [4].

5.2.3.3 Тестирование конвейеров данных

Конвейер данных можно рассматривать как самостоятельную подсистему СМО. Для большинства СМО существует два варианта конвейера данных: конвейер разработки, который предоставляет данные для обучения модели, и конвейер производства, который предоставляет данные для развернутой модели (например, для прогнозирования). Производственный конвейер часто является оптимизированным вариантом, полученным из обучающего конвейера.

Конвейер данных должен быть разработан с использованием лучших практик программной инженерии, следуя полному жизненному циклу разработки. С точки зрения тестирования, это обычно требует следующих уровней тестирования:

Модульное тестирование каждого компонента конвейера данных, включая:

- статическое тестирование, такое как обзоры кода (подробное определение обзоров рабочих продуктов приведено в [4]) и статический анализ;

- тестирование аппаратных источников, таких как датчики.

Интеграционное тестирование для проверки корректности взаимодействия и связи компонентов:

- компонентов, составляющих конвейер;

- аппаратного и программного обеспечения (например, проверка правильности интерпретации входных сигналов датчиков программным обеспечением).

Системное тестирование интегрированного конвейера, включая:

- дымовое тестирование начального конвейера с простой базовой моделью (например, линейной регрессией) и несколькими функциями;

- тестирование как функциональных, так и нефункциональных характеристик конвейера, таких как время, необходимое для преобразования исходных данных в учебные или производственные данные, а также безопасность и конфиденциальность данных;

- тестирование «спина к спине» (дифференцированное) вариантов разработки и производства конвейера, чтобы убедиться, что оба варианта предоставляют согласованные данные для модели. Более подробная информация о тестировании «спина к спине» приведена в 5.3.3.7.

Эксплуатационное тестирование, включая:

- тестирование текущей и обновленной версий конвейера в режиме «спина к спине» (дифференцированное тестирование), чтобы убедиться, что новая версия предоставляет модели ожидаемые данные после обновления конвейера (для сравнения могут использоваться статистические показатели данных);

- A/B-тестирование, чтобы убедиться, что обновленные версии конвейера работают не хуже, чем текущая версия (может включать сравнение как функциональных, так и нефункциональных характеристик).

Тестирование конвейера данных также должно учитывать:

- обзор процессов управления конфигурацией и результирующих конфигураций, чтобы убедиться, что используются правильные версии компонентов и наборов данных (подробное определение обзора продуктов приведено в [4]);

- тестирование на введение ошибок для определения устойчивости конвейера данных к дефектным входным данным, которые могут быть вызваны использованием данных, собранных неправильно, данных с неисправных датчиков или данных, которые были преднамеренно повреждены.

5.2.3.4 Проверка происхождения данных

Статическое тестирование, обычно в форме обзора, которое позволяет проверить источники данных, составляющих обучающий набор данных. Эти проверки позволяют определить, насколько надежны источники, предоставляющие данные для набора данных, и безопасны каналы передачи данных. Этот тип тестирования направлен на предотвращение проблем с набором данных, таких как отравление данных (см. [4]).

5.2.3.5 Проверка репрезентативности и выборки данных

Проверка репрезентативности данных позволяет определить, являются ли наборы данных, используемые для обучения и тестирования, достоверными и будут ли они, как ожидается, обнаружены операционной моделью.

Существует несколько подходов, которые позволяют выполнять различные этапы этого тестирования, например:

- сравнивать статистические характеристики наборов данных для обучения и тестирования с характеристиками производственных данных и выявлять существенные различия. Например, индекс стабильности совокупности является мерой для оценки смещения в распределении между двумя разными выборками. Такое сравнение между обучающими и эксплуатационными данными может быть выполнено только после развертывания системы МО или при наличии статистических данных, полученных из аналогичной системы;

- привлечение узких специалистов для проверки обучающих данных позволяет убедиться, что данные будут адекватно обработаны моделью в процессе эксплуатации. Проверка может быть выполнена до развертывания системы МО, в ходе которой применимы такие подходы, как разведочный анализ данных для анализа обучающих данных. Проверка позволяет убедиться, что обучающий набор данных не включает атрибутов, которые являются излишними и, вероятно, не представляют интереса как операционные данные (например, идентификаторы пациентов, которые включают индикаторы болезни) (см. [4]);

- проверка позволяет выявить действия по аугментации данных, которые могут создавать нерепрезентативные данные;

- проверка процедур по сбору данных позволяет убедиться, что такие процедуры не приводят к искажению данных (например, опрос только части населения) (см. [4]).

5.2.3.6 Проверка полноты данных

Проверка полноты данных позволяет определить, достаточен ли объем данных для обучения и тестирования.

Существует несколько подходов, которые позволяют выполнять различные этапы этого тестирования, например:

- проверка позволяет убедиться, что набор данных сбалансирован с точки зрения всех целевых классов и целевые классы представлены в достаточной степени. Если обучающие или тестовые данные кажутся несбалансированными, необходимо убедиться, что в качестве одного из критериев приемлемости выбрана подходящая метрика для МО. Например, вместо метрики точности могут быть использованы метрики, в меньшей мере подверженные влиянию диспропорции, такие как F1-мера. Кроме того, рассматривают возможность расширения набора данных путем создания синтезированных

выборок для увеличения числа примеров в классах меньшинств, используя такие методы, как метод передискретизации синтезированных меньшинств (см. [4]);

- разведочный анализ данных и привлечение экспертов применяют для идентификации областей пространства входа, которые не представлены примерами. По возможности включают в обучающий набор данных новые примеры для заполнения и, как минимум, дополняют тестовые примеры, которые позволяют проверить такие области с целью определения, как модель МО справляется с такими случаями.

5.2.3.7 Проверка ограниченности набора данных

Данные, используемые для обучения, оценки, тестирования, оперативного прогнозирования, классификации или подготовки рекомендаций, могут быть проверены несколькими способами. В частности, можно проверить согласованность значений в наборе данных (например, убедиться, что возраст детей не установлен в значениях выше, чем возраст их родителей).

Схема базы данных часто представляет собой набор формул, определяющих ограничения для базы данных. Аналогичным образом для наборов данных МО можно определить ограничения, действующие как логическая модель данных в наборе данных, которая должна отвечать требованиям, если данные верны. Если ограничение данных нарушено, вполне вероятно, что такие данные содержат отклонение. Как правило, анализ отклонений позволяет определить дефекты данных. Эти дефекты могут быть вызваны разными причинами, такими как неисправность датчика или дефект кода конвейера данных, используемых алгоритмом или моделью МО.

Проверка данных на соответствие ограничениям схемы базы данных может выполняться вручную, но, как правило, масштаб проверки требует, чтобы этот процесс был автоматизирован как часть конвейера данных. В этом случае инструмент, внедренный как часть конвейера и реализующий тестирование ограничений данных, может предоставлять отчеты специалистам (для обучения работе с отклонениями) или операционному персоналу (для решения операционных проблем).

Ограниченность набора данных

Существует несколько способов категоризации ограничений данных. Ограничение может быть применено к одному значению атрибута, который находится в одном экземпляре (ограничение с единственным значением). В качестве альтернативы ограничение может применяться к нескольким значениям, обычно с учетом значений одного атрибута в нескольких экземплярах (ограничение с несколькими значениями). Специальная форма многозначного ограничения позволяет сравнивать разные значения (ограничение сравнения). Также могут быть определены ограничения для сравнения полных наборов данных. Они известны как пакетные ограничения, их используют, например, для определения допустимости отличий нового набора данных от существующего или возможностей определения дрейфа данных с течением времени.

Для каждой из категорий существует несколько типов ограничений. На практике выбор соответствующего подмножества категорий и типов зависит от решаемой задачи. Необходимо обратить внимание, что для большинства ограничений можно установить порог, который должен быть превышен для обнаружения отклонения. Например, для типа ограничения «отсутствие» отклонение может быть определено только в том случае, если в наборе данных из 10 000 записей обнаружено отсутствие более 100 значений атрибутов.

Ограничения с единственным значением

Примеры типов ограничений данных с единственным значением:

- отсутствие — проверка отсутствующих значений и атрибутов;
- уникальность — проверка уникальности каждого значения атрибута (например, позволяет убедиться в отсутствии дублирования рейтингов для данных по регионам страны);
- диапазон — позволяет определить, находится ли значение в заданном диапазоне (например, возраст находится в диапазоне от нуля до 120 лет);
- тип — позволяет убедиться, что каждое значение атрибута соответствует указанному типу (например, если атрибут указан как целое число, представленное значение не может быть выражено строкой или действительным числом);
- положительное значение — позволяет убедиться, что каждое значение атрибута является положительным;
- отрицательное значение — позволяет убедиться, что каждое значение атрибута является отрицательным;
- минимальное значение — позволяет убедиться, что каждое значение атрибута является равным или не превышает заданного значения (например, возраст всех водителей от 16 лет и старше);

- максимальное значение — позволяет убедиться, что каждое значение атрибута меньше заданного значения (например, все отметки на карте ниже 9000 м);

- ограничение пользователем — это ограничение, специфичное для данного атрибута данных (например, позволяет убедиться, что каждая запись является допустимым почтовым индексом).

Ограничения с несколькими значениями

Примеры ограничений данных с несколькими значениями:

- сумма — позволяет убедиться, что сумма всех значений равна, превышает или не превышает заданное значение (например, общая сумма всех очков Формулы-1 не должна превышать 102 или быть ниже 50,5 очков);

- среднее значение — позволяет убедиться, что среднее арифметическое всех значений атрибута равно, превышает или не превышает заданное значение (например, среднее значение IQ находится в диапазоне от 95 до 105);

- стандартное отклонение — позволяет убедиться, что стандартное отклонение значений атрибута равно, выше или ниже заданного значения (например, стандартное отклонение для производства болтов не превышает 0,001 м);

- количество — позволяет убедиться, что количество всех ненулевых значений/атрибутов/экземпляров равно, выше или ниже заданного значения;

- неоднородность — позволяет убедиться, что набор значений атрибута выше или ниже заданной степени разнообразия (например, значение P критерия однородности хи-квадрата для медицинских результатов превышает значение 0,1);

- дублирование — позволяет убедиться, что в наборе данных нет идентичных (или близких) значений атрибутов или экземпляров;

- распределение — позволяет убедиться, что распределение значений обучающих данных соответствует ожидаемому;

- баланс классов — позволяет убедиться, что между значениями в целевом классе отсутствует дисбаланс (например, преобладание ложных значений над истинными значениями в обучающих данных для прогнозирования заболеваний). Несбалансированные данные затрудняют оценку производительности (например, точности), что может быть устранено путем дополнения данных или организации целенаправленного сбора данных;

- баланс характеристик — позволяет убедиться, что между разными значениями определенного атрибута отсутствует дисбаланс (например, много ответов от молодых людей и мало ответов от пожилых людей, что делает возрастную характеристику несбалансированной). Несбалансированность данных может быть устранена с помощью дополнения данных или организации целенаправленного сбора данных.

Ограничение сравнения

Примеры типов ограничений данных при сравнении:

- больше, чем — позволяет убедиться, что одно значение атрибута больше, чем второе значение атрибута (например, метрика строк кода для программы превышает ее цикломатическую сложность);

- совпадение — позволяет убедиться, что значение одного атрибута совпадает со значениями второго атрибута (например, все учащиеся со значением оценок не менее чем на 1,33 стандартного отклонения выше среднего также имеют значение «А» в качестве оценки);

- ограничение пользователем — ограничение, заданное пользователем для данных (например, чтобы убедиться, что каждое значение для одного атрибута в десять раз превышает значение второго атрибута).

Пакетные ограничения

Пример пакетного ограничения для набора данных:

- распределение — позволяет установить, как отличаются распределения значений атрибута в одном пакете данных от распределения во втором пакете данных (например, что распределения обучающих и данных не сильно разнятся, или что распределение данных на сегодняшний день не отличается от распределения данных за вчерашний день). Например, индекс стабильности популяции является показателем для оценки смещения в распределении между двумя разными выборками популяции. Обратите внимание, что различия между наборами обучающих и операционных данных для некоторых систем могут наблюдаться ввиду использования разных конвейеров данных. Это может быть связано с тем, что обучающие данные создаются в автономном режиме, а операционные данные создаются в режиме реального времени из-за необходимости более высокой пропускной способности при меньшей задержке.

Инструменты проверки ограничений данных

Инструменты проверки ограничений данных могут работать как в автономном режиме (что применяют в случае конвейера разработки), так и в режиме реального времени (применяют в случае производственного конвейера).

5.2.3.8 Тестирование факторов

Введение в тестирование факторов

Проверка факторов позволяет убедиться, что для обучающих данных был выбран оптимальный набор факторов (атрибутов).

Проверка ограничений факторов

Проверка выбранных факторов позволяет определить необходимость включения отдельных факторов для обучающих данных.

Существует несколько подходов, которые можно использовать для тестирования вклада факторов, например:

а) анализ экспертами обоснования отбора факторов при условии, что результаты должны быть задокументированы (см. [4]);

б) измерение корреляции между отдельными факторами и прогнозом, классификацией или рекомендацией:

1) в том случае, если коэффициент корреляции превышает ожидаемый, это может указывать на использование фактора моделью для определения класса результатов, или что эксперт анализирует отбор факторов;

2) в том случае, если коэффициент корреляции ниже ожидаемого, это может указывать на несущественность функции, которая подлежит исключению;

в) исключение отдельных функций и измерение точности модели. В том случае, если модели с исключенными функциями не снижают показатели (например, точности), это может указывать на несущественность исключенных факторов.

Проверка эффективности факторов

Проверка эффективности факторов позволяет определить важность отбора факторов на данных для МО. Как правило, это подразумевает оценку затрат на включение фактора (например, затраты на обучение, временные затраты и предполагаемые объемы памяти), сравнение затрат с улучшением требуемых критериев производительности для модели МО, таких как точность, что обеспечивается включением такого фактора.

Тестирование пары «фактор-значение»

Тестирование пары «фактор-значение» направлено на проверку того, что наиболее важные значения фактора правильно обрабатываются моделью МО и их ожидаемое влияние на результаты не маскируется другими характеристиками в наборе данных. Например, подход, основанный на оценке рисков, может определить, что при установке атрибута страны в значение «США» модель МО реагировала правильно. Тестирование пары «фактор-значение» — это динамический тип тестирования, используемый для проверки того, что установка фактора на определенное значение правильно обрабатывается моделью МО.

5.2.3.9 Проверка правильности разметки

При тестировании систем МО, в которых используется обучение с учителем, каждый из образцов обучающих данных должен быть помечен как целевой класс. Проверка позволяет убедиться в правильности разметки.

Существует несколько подходов, которые могут применяться для проверки правильности разметки, например:

- проверка процедур разметки. Анализ задокументированных процедур и наблюдений экспертов позволяет выявить системные проблемы (см. [4]);

- динамическое тестирование путем создания тестовых образцов с известными целевыми классами в качестве тестовых примеров. Этот подход можно использовать для тестирования как ручной разметки образцов, так и автоматической разметки с помощью инструментов. Такой подход позволяет использовать различные методы тестирования ПО для создания тестовых выборок, таких как разделение по эквивалентности (чтобы обеспечить охват всех регионов в пространстве данных) и анализ граничных значений (что позволяет корректно обрабатывать предельные значения);

- проверка результатов разметки данных, которая, как правило, осуществляется через отбор примеров образцов независимыми испытателями. Аналогичным образом для обнаружения конфликтующих меток можно использовать метод последовательного тестирования путем назначения

подмножества всех образцов второму устройству, используемому для разметки, и сравнения результатов двух устройств.

5.2.3.10 Проверка нежелательной смещенности данных

Необъективность системы МО может возникнуть из-за данных (по причине смещенности выборки) или алгоритмов. Данные обучения, которые содержат защищенные характеристики в качестве признаков (например, расу, пол, сексуальную ориентацию, этническую принадлежность и возраст), могут привести к появлению необъективных моделей МО, которые по-разному относятся к разным группам пользователей. Группы, на которые распространяется необъективность, известны как группы риска.

Статическое тестирование может выполняться на обучающих данных, предпочтительно узкими специалистами. В ходе тестирования проверке подлежат те факторы, которые могут привести к необъективности, что позволит убедиться, что защищенные от рисков данные представлены равноправно, а исторически сложившаяся необъективность не влияет на работу системы. Кроме того, проверка позволяет убедиться, что факторы прокси, которые коррелируют с защищенными факторами, используются с осторожностью (см. [4]).

В случае аугментации обучающих данных проверка позволяет убедиться, что модификация данных не привела к непропорциональному увеличению выборок, что, ожидаемо, может стать причиной необъективности модели МО.

Положения [5] определяют требование, чтобы оценка следующих характеристик данных для каждой группы риска проводилась в соответствии с положениями [6] (6.3) (с последующим сравнением с результатами для групп, не подверженных риску, в целях обеспечения отсутствия необъективности):

- проверяемость данных, в том числе наличие возможности определить источник данных;
- сбалансированность;
- актуальность и релевантность данных;
- полнота, включая число недостающих факторов;
- отсутствие погрешности, включая число неточных обучающих данных;
- согласованность, включая метки (если применимо);
- разнообразие;
- эффективность;
- точность;
- релевантность;
- репрезентативность;
- подобие;
- своевременность.

Некоторые из вышеперечисленных характеристик довольно сложно оценить объективно, поэтому применяют субъективные оценки.

5.3 Тестирование модели

5.3.1 Введение в тестирование модели

Тестирование моделей связано с тестированием моделей МО, созданных специалистами по обработке данных, и предполагает выявление соответствия как функциональным, так и нефункциональным критериям. Тестирование модели включает как статическое тестирование (например, проверку пригодности модели), так и динамическое тестирование моделей МО. Динамическое тестирование включает в себя тестирование сгенерированных моделей МО на основе как «черного», так и «белого ящика».

5.3.1.1 A/B-тестирование

A/B-тестирование — это статистический метод тестирования, который позволяет определить, какая из двух систем работает лучше. Его также называют раздельным тестированием. Его часто используют в цифровом маркетинге (например, чтобы определить, какой e-mail получит лучший отклик от клиентов).

A/B-тестирование часто используется для оптимизации дизайна пользовательского интерфейса. Например, разработчик интерфейса предполагает, что, изменив цвет кнопки «Купить» с используемого сейчас красного на синий, продажи увеличатся. Новый вариант интерфейса создается с синей кнопкой, и оба интерфейса предлагаются разным пользователям. При сравнении темпов продаж и, приняв во внимание статистически значимое число пользователей, можно определить, верна ли гипотеза. Если синяя кнопка позволяет увеличить продажи, то новый интерфейс с синей кнопкой заменяет текущий интерфейс с красной кнопкой. Ключевое требование A/B-тестирования — статистически значимое число

пользователей, что затратно по времени, хотя проведение тестирования предполагает использование инструментов (часто на основе ИИ).

A/B-тестирование не является методом, применяемым для создания тестовых примеров, поскольку тестовые входные данные не генерируются. A/B-тестирование — это средство решения проблемы тестового оракула путем использования существующей системы в качестве частичного оракула. Сравнивая новую систему с существующей, можно определить, является ли новая система лучше. В цифровом маркетинге показателем эффективности может быть увеличение продаж, но для системы МО, такой как, например, классификатор, можно использовать такие показатели эффективности, как точность, чувствительность и отклик.

A/B-тестирование применяется при обновлении модели МО, при условии, что критерии (например, «указанные показатели производительности должны улучшиться или остаться прежними») определены и согласованы. Если A/B-тестирование автоматизировано, его можно использовать для онлайн-тестирования самообучающихся систем МО путем сравнения эффективности новой системы с ее предыдущей версией и возврата к предыдущей версии, если самообучение не позволило улучшить производительность системы. По этой причине необходимо установить действительные критерии.

Поддержка A/B-тестирования осуществляется при помощи специальных инструментов (некоторые из которых основаны на ИИ), а также среды разработки МО.

5.3.1.2 Состязательная проверка

Примером состязательной проверки является внесение незначительных изменений во входные данные нейронной сети, что может привести к неожиданному (часто ошибочному) изменению выходных данных (т. е. совершенно иному результату, чем до изменения входных данных). Состязательные примеры впервые были замечены в классификаторах изображений. Изменив всего несколько пикселей (невидимых человеческому глазу), мы можем убедить нейронную сеть изменить классификацию изображения на совершенно другой объект (с высокой степенью уверенности). Обратите внимание, что состязательные примеры не ограничиваются классификаторами изображений, но также являются атрибутом нейронных сетей в целом, и поэтому применимы к нейронным сетям в целом (а также некоторым моделям МО).

Поскольку состязательные примеры основаны на небольших изменениях или искажениях входных данных, состязательное тестирование также иногда называют тестированием на возмущение.

Состязательные примеры, как правило, переносимы. Это означает, что состязательный пример, вызывающий сбой одной нейронной сети, часто приводит к сбою других нейронных сетей, обученных выполнять ту же задачу. Обратите внимание, что другие нейронные сети, возможно, были обучены на других данных и на основе других архитектур, но они по-прежнему склонны к сбоям в тех же состязательных примерах.

Состязательное тестирование часто называют состязательными атаками. Выполнение таких атак и выявление уязвимости в процессе тестирования позволяет принимать меры для защиты от будущих сбоев и, следовательно, повысить надежность и/или безопасность нейронной сети.

Атаки могут осуществляться при обучении модели, а затем и на самой обученной модели (нейронной сети). Атаки во время обучения могут включать в себя повреждение обучающих данных (например, изменение меток), добавление неверных данных в обучающие данные (например, нежелательных функций) и повреждение алгоритма обучения. Атаки на обученную модель могут осуществляться по схеме «белый ящик» или «черный ящик» и включать в себя выявление состязательных примеров, которые могут заставить модель давать неудовлетворительные результаты.

При атаках методом «белого ящика» злоумышленник знает алгоритм, используемый для обучения модели, а также используемые настройки и гиперпараметры. Злоумышленник использует эти знания для создания состязательных примеров, например, через внесение небольших изменений во входные данные и отслеживание, какие из них вызывают изменения в модели.

При атаках методом «черного ящика» злоумышленник не располагает информацией о внутренней работе модели или знанием о том, как она обучалась. В этой ситуации злоумышленник сначала использует модель для определения ее функциональности, а затем создает «дубликат» модели, который обеспечивает ту же функциональность. Затем злоумышленник использует подход «белого ящика», что позволяет выявить состязательные примеры для этой модели. Поскольку состязательные примеры, как правило, можно переносить, те же состязательные примеры обычно работают и в основной модели («черного ящика»).

При выявлении состязательных примеров в классификаторах они могут быть добавлены к обучающим данным с правильной меткой и использованы для обучения модели правильным действиям в аналогичных условиях.

5.3.1.3 Тестирование альтернативных моделей

Убедитесь, что альтернативная модель сложно обучаема (т.е. нельзя использовать уже готовую модель или применять трансферное обучение) и позволит обеспечить значительное увеличение производительности модели по сравнению с тестируемой моделью.

5.3.1.4 Тестирование программного интерфейса приложения

Программный интерфейс приложения (API, Application Programming Interface) — это интерфейс, который позволяет программе вызывать другую систему, которая предоставляет ей сервис. Эти услуги могут предоставляться удаленно через Интернет, как, например, доступ к удаленному ресурсу. В контексте МО предоставляемый сервис может быть моделью МО, например моделью, предоставляющей сервис перевода.

Тестирование API позволяет оценить входные и возвращаемые значения.

При работе с API важны как положительные, так и отрицательные результаты тестирования. Положительные тесты позволяют проверить предоставляемые сервисы, а подход, основанный на оценке рисков, обычно обеспечивает более тщательное тестирование наиболее важных сервисов.

Программисты, использующие API для получения доступа к внешним службам, часто используют API способы, для которых они не предназначены, отрицательное тестирование позволяет проверить, что обработка исключений протекает должным образом.

Комбинаторное тестирование может дополнять тестирование API для анализа потенциальных комбинаций параметров и значений интерфейса.

API часто почти не связаны, что может привести к потере передачи и проблемам синхронизации, что требует тщательного тестирования механизмов восстановления и повторных попыток.

Услуги, предоставляемые через API, обычно широко доступны, что требует тестирования надежности поставщиков сервисов.

Тестирование сервиса через API может быть затруднено из-за отсутствия пользовательского интерфейса (доступ к услуге осуществляется непосредственно через API), а для доступа обычно требуются инструменты тестирования.

5.3.1.5 Атаки

Общие положения

Тестовая атака — это вид тестирования, основанный на распространенных режимах сбоя в программном обеспечении (ПО) для поддержки сбора информации. Атака является шаблоном, поскольку она сочетает в себе классические методы тестирования, концепции тестирования, тесты, предусмотренные положениями настоящего стандарта, и мнение тестировщика. Шаблон атаки модифицируется для проведения тестирования. Атаки целенаправленны и определены в рамках исследовательского подхода к тестированию. Атаки основаны на предположении о том, что опытные тестировщики часто используют шаблоны ментальных проверок, разрабатываемые годами, которые тестовые атаки пытаются воспроизвести.

Для проведения тестовых атак на ИИ тестируемое ПО должно включать «систему» (т. е. аппаратное обеспечение, ПО, операции, данные и пользователей). Устранение рисков ПО, выявленных на этапе планирования тестирования, и распространенных видов сбоев, нацеленных на атаки тестирования ПО, становится частью плана тестирования.

Планирование тестирования атак начинается с самого ИИ, системы ИИ, а затем включает периферийные и облачные соединения, если они имеются. Использование анализа данных для оценки тестируемого ИИ позволяет узнать, какие атаки применимы в тестировании и в какой степени их следует применять. Ниже приведены примеры тестовых атак на ИИ.

Атаки на ИИ с использованием социальной инженерии

Тестировщики должны думать и действовать как злоумышленники при организации атаки. Тестировщики играют роль злоумышленника (например, хакера) с целью предотвращения возможных проблем, которые могут вызвать атаки. Они задаются вопросом: «Как мы можем провести пользователей и систему ИИ, чтобы заставить их действовать так, как они не должны?» С этой целью они могут использовать информацию о конструктивных особенностях, обучающих данных, рисках ИИ, аналитику данных и информацию, доступную разработчикам, такую как, например, логины пользователей, пароли, связанное ПО.

Вредоносное ПО и атаки на готовое ПО

Вредоносное ПО позволяет выявить риски продукта. Многие системы ИИ используют готовое ПО, находящееся в открытом доступе, или приобретают ПО у третьих лиц. ПО, готовое к использованию, может содержать ошибочный код, несертифицированный код, вредоносное ПО или проблемы с данными.

Если вы не доверяете готовому ПО, такие атаки помогают проверить и проанализировать его. В этом случае тестировщику потребуется доступ к исходному или двоичному коду, чтобы организовать атаку. Использование инструментов и анализа данных для поддержки тестировщиков при проведении этой атаки является хорошей практикой. Кроме того, код ПО должен быть визуально проверяться во время инспекционных совещаний команды. Однако люди склонны упускать что-то из виду, поэтому проведение инструментальных проверок также рекомендуется.

Атака с искажением данных

Тестировщики пытаются «взломать» аппаратное и ПО, чтобы получить доступ. Действия связаны со взломом пароля, но требуют и других подходов, таких как внедрение вредоносного кода, повреждение данных обучения/тестирования или повреждение конфигураций оборудования или ПО с использованием инструментов и процессов управления конфигурацией (CM, configuration management) или управления конфигурацией ПО (SCM, software configuration management). Атака с искажением данных может изменить данные или ПО, используемые в системе ИИ. Цель состоит в том, чтобы манипулировать информацией ИИ, такой как код, оборудование, пароли, обучающие наборы, данные проверки, выходные значения, разрешения и т. д. При этом тестировщик пытается понять следующее:

- может ли внешняя система CM/SCM или внутренняя система проверки обнаружить вмешательство в ИИ?
- отслеживает ли система ИИ атаку и информирует заинтересованные стороны о ней?
- могут ли неправильные данные ИИ (обучающие данные, тестовые данные, ввод данных пользователем и т. д.) быть введены, выведены и использованы без обнаружения, когда предполагается, что они должны быть обнаружены?

Атака на смещенность ИИ

Настоящий стандарт определяет тесты, которые позволяют решить проблему смещенности ИИ. Этот подход объединяет несколько тестов в набор выполняемых и повторяющихся атак на систему ИИ. Использование этой атаки на начальном этапе разработки важно, но еще важнее проводить атаки на смещенность ИИ регулярно во время эксплуатации и обновления ИИ. Атаки на смещенность стали обычным явлением, что позволяет избегать проблем, связанных со слабостью регрессионного теста.

Атака на интерфейс ИИ

Этот тип атаки включает серию дополнительных атак. Выбор таких дополнительных атак зависит от особенностей интерфейса. Большинство систем ИИ имеют вэб-интерфейс или мобильный интерфейс.

Атака с проникновением

Пентестирование — это попытка тестировщика проникнуть в безопасность системы ИИ. Пентестирование предполагает тесную работу с командой разработчиков и нацелено на обеспечение удобства для пользователей ПО. Результаты пентестов включают тестирование ПО, выявление проблем, подготовку отчета о выполненных работах, а также подготовку рекомендаций. Если результаты пентестирования указывают на проблемы, разработчики могут обратиться к команде тестировщиков за получением разъяснений и рекомендаций по устранению выявленной проблемы. Этапы пентестирования включают:

- получение доступа и взлом пароля. Взлом пароля или получение доступа к управлению системой позволяет выявить проблему. Для этого можно использовать инструменты, которые позволяют идентифицировать пароли и получить доступ к системе;
- получение привилегий доступа. После того, как злоумышленнику удастся взломать пароль и получить доступ, он может получить дополнительные привилегии доступа (чтобы попытаться нанести еще больший ущерб).

DoS-атака (отказ в обслуживании) на систему ИИ

Злоумышленники могут попытаться взломать систему, поэтому необходимо, чтобы система реагировала как можно более «безопасно». Хакеры атакуют все системы, включая системы ИИ. DoS-атака позволяет заблокировать работу системы ИИ с помощью отправки большого количества запросов, которые превышают пропускные способности сети и/или способности системы ИИ по обработке запро-

сов, таким образом снижая способность системы ИИ к реагированию. Такие действия в рамках тестирования помогают выявить уязвимости, которые позволят взломать систему или проникнуть в нее.

Атака с целью анализа вредоносных данных

Анализ вредоносных программ с помощью ИИ в некоторой степени связан с вредоносным ПО, ошибками кода и DoS. Это процесс, когда разработчики/тестировщики проверяют поток пользовательских входных данных в ходе обработки ИИ, чтобы определить, может ли непредвиденный ввод отрицательно повлиять на работу программы на каждом функциональном этапе. В число специалистов, выполняющих этот анализ, могут входить разработчики, сотрудники службы поддержки или тестировщики, которым предстоит выполнять разные роли с целью обнаружения нормальных и аномальных процессов обработки входных данных. Анализ вредоносных данных позволяет определить поверхность атаки на ИИ для последующего тестирования. Специалисты, проводящие тестирование, должны иметь доступ к следующим видам информации, которая может быть повреждена: входные данные, значения обучающих данных, тестовые данные, последовательность входных данных, выходные сообщения, выходные данные файлов, ответные действия и связи с другими элементами системы ИИ, такими как периферийные, распределенные и облачные вычисления. В ходе этого анализа «пользователь» пытается испортить входные данные, а затем проверяет выходные данные на наличие некорректных действий. Результаты доступны к просмотру в режиме реального времени, но позже можно изучить их более углубленно в процессе постобработки для выявления проблем. Обработка может быть улучшена с помощью динамического мониторинга вредоносных данных, анализа данных или проверки работоспособности системы ИИ.

5.3.1.6 Параллельное тестирование

При параллельном тестировании альтернативная версия системы (например, уже существующая, разработанная другой командой или реализованная с использованием другого языка программирования) используется в качестве «псевдооракула» для генерации ожидаемых результатов и их сравнения на основе одних и тех же тестовых входных данных. Иногда такое тестирование называют дифференциальным.

Параллельное тестирование не применяют для создания тестовых примеров, поскольку тестовые входные данные не генерируются. Функционально эквивалентная система («псевдооракул») автоматически генерирует только ожидаемые результаты. Тестирование в сочетании с инструментами для генерации тестовых входных данных (случайных или данных другого типа) является эффективным для проведения автоматизированного тестирования в больших объемах.

При параллельном тестировании в рамках функционального тестирования «псевдооракул» необязательно должен соответствовать тем же нефункциональным ограничениям, что и тестируемая система. Например, «псевдооракул» может работать гораздо медленнее, чем требуется для тестируемой системы. Также необязательно, чтобы «псевдооракул» был полностью функционально эквивалентной системой, поскольку параллельное тестирование может выполняться с использованием «псевдооракула», который эквивалентен только части тестируемой системы.

МО позволяет использовать разные структуры, алгоритмы и настройки для создания «псевдооракулов» (в некоторых ситуациях возможно создание «псевдооракула» с использованием обычного ПО, не связанного с ИИ). Основная проблема с использованием «псевдооракулов» заключается в том, что для бесперебойной работы они должны быть полностью независимы от тестируемого ПО. Поскольку для разработки систем на основе ИИ используется разное ПО с открытым исходным кодом, это требование может быть нарушено.

5.3.1.7 Анализ граничных значений

Анализ позволяет проверить обработку граничных условий данных.

Подробное описание приведено в [1].

5.3.1.8 Комбинаторное тестирование

Динамическое тестирования такого типа позволяет убедиться, что конкретный элемент тестирования соответствует всем требованиям. Все возможные комбинации входных значений во всех возможных состояниях подлежат тестированию. Такой подход называется «исчерпывающим тестированием». На практике при тестировании ПО наборы тестов создаются путем выборки из (чрезвычайно большого) набора возможных входных значений и состояний. Комбинаторное тестирование — это один из систематических (и эффективных) подходов к получению полезного подмножества комбинации из входного пространства.

Необходимые комбинации определяются на основе параметров (т. е. входных данных и условий среды) и значений, которые эти параметры могут принимать. При объединении множества параме-

тров (каждый из которых имеет множество дискретных значений) этот метод позволяет сократить количество необходимых тестовых примеров без ущерба для способности тестовых данных обнаруживать дефекты.

Положения [1] определяют несколько методов комбинаторного тестирования, таких как тестирование всех комбинаций, тестирование каждого варианта, тестирование базовой выборки и парное тестирование. На практике чаще используется парное тестирование, в основном из-за простоты интерпретации результатов, широкой инструментальной поддержки и наличия исследований, доказывающих, что большинство дефектов бывают вызваны взаимодействиями с участием нескольких параметров.

Число параметров, представляющих интерес для системы МО, может быть чрезвычайно большим, особенно если система использует большие данные или взаимодействует с внешним миром. В качестве примера можно привести беспилотные автомобили. Систематическое сокращение по сути бесконечного числа возможных комбинаций до управляемого подмножества может быть реализовано с помощью комбинаторного тестирования, например парного тестирования. На практике использование парного тестирования может привести к созданию обширных наборов тестов для таких систем. По этой причине приходится прибегать к использованию средств автоматизации и виртуальных тестовых сред.

Если в качестве примера рассмотреть беспилотные автомобили, то сценарии тестирования систем должны учитывать как функции транспортных средств, так и условия, в которых они будут работать. Таким образом, параметры должны будут включать в себя функции самостоятельного вождения (например, круиз-контроль, адаптивный круиз-контроль, помощь в удержании полосы движения, помощь при смене полосы движения, помощь со светофором и т. д.), а также ограничения окружающей среды (например, типы и поверхности дорог, географические условия), местность, время суток, погодные условия, условия движения, видимость и т. д.). В дополнение к этим параметрам следует учитывать входные данные, полученные с датчиков разного уровня эффективности (например, входные данные, качество сигнала видеокамеры будет ухудшаться по мере движения ввиду загрязнения или точность GPS-устройства будет меняться по мере того, как спутники входят в зону прямой видимости и выходят из нее). Результаты проводимых исследований в настоящее время не позволяют установить строгие требования для проведения комбинаторного тестирования критически важных с точки зрения безопасности систем МО беспилотных автомобилей (например, попарного тестирования может быть недостаточно). Однако мы можем утверждать, что этот подход эффективен для обнаружения дефектов, а также может использоваться для оценки остаточного уровня риска.

Подробное описание комбинаторного тестирования и связанных с ним мер тестового покрытия приведено в [1].

5.3.1.9 Тестирование этических аспектов

При тестировании этических аспектов могут быть использованы принципы, методы и подходы к решению этических проблем при разработке и эксплуатации систем ИИ, изложенные в [7].

5.3.1.10 Исследовательское тестирование

Разработка и выполнение тестов могут осуществляться с использованием разных подходов, в зависимости от потребностей каждого проекта, но, как правило, всегда с участием узкоспециализированных специалистов. Оно может быть сценарным или исследовательским. На практике обычно используется комбинация сценарного и исследовательского тестирования, поскольку тестирование по сценариям обеспечивает достижение требуемых уровней тестового покрытия и лучше поддерживает автоматическое тестирование, в то время как исследовательское тестирование позволяет проявлять творческий подход и быстрее выполнять тесты. При тестировании систем ИИ исследовательское тестирование часто применяется в отсутствие полных и подробных спецификаций (например, при гибкой разработке).

При исследовательском тестировании тесты разрабатываются и выполняются по ходу работы, когда тестировщик взаимодействует с элементом тестирования и изучает его. Таблицы сеансов позволяют структурировать этапы исследовательского тестирования (например, с помощью установки ограничений по времени для каждого сеанса тестирования). Также таблицы сеансов используются для сбора информации о ходе тестирования и наблюдаемом аномальном поведении. Исследовательские тесты часто не имеют полного описания, поскольку сценарии тестирования высокого уровня (иногда называемые «идеями тестирования») часто документируются в таблицах сеансов, чтобы сосредоточить внимание на конкретном сеансе исследовательского тестирования.

5.3.1.11 Фазз-тестирование

Фазз-тестирование (или фаззинг) включает в себя автоматическую генерацию больших объемов псевдослучайных тестовых данных на основе известных ограничений входных данных. Данный вид

тестирования можно использовать для проверки надежности алгоритма сгенерированной модели, создавая тестовые наборы данных на основе ограничений, определенных в схеме данных.

Фаззинг на основе покрытия — это средство определения критерия для завершения данного вида тестирования. Однако ввиду недостаточно четкого определения мер этот подход еще только разрабатывается.

5.3.1.12 Метаморфическое тестирование или тестирование на соответствие бизнес-логике

Метаморфическое тестирование — это подход к созданию тестовых примеров, который частично решает проблему тестового оракула, часто возникающую в системах на основе ИИ, где генерация ожидаемых результатов может быть затруднена (например, из-за сложности, недетерминированного выполнения и вероятностных систем). Основное различие между тестовыми примерами, созданными с использованием метаморфического тестирования, и традиционными методами разработки тестовых сценариев заключается в том, что ожидаемые результаты метаморфических испытаний могут быть не фиксированным значением, а определяться взаимосвязью с другим ожидаемым результатом.

Метаморфическое тестирование основано на анализе метаморфических отношений для создания тестовых примеров на основе исходного тестового примера, который, как известно, является правильным. Метаморфическое ПО позволяет описать, как изменение входных тестовых данных исходного тестового примера и последующего тестового сценария влияет на изменение (или отсутствие изменений) ожидаемых выходных данных исходного тестового примера.

Примеры

1 Элемент теста измеряет расстояние между начальной и конечной точками. Исходный тестовый пример имеет тестовые входные данные A (начальная точка) и B (конечная точка), а также ожидаемый результат C (расстояние). Метаморфическое отношение позволяет утверждать, что если поменять местами начальную и конечную точки, то ожидаемый результат останется неизменным. Таким образом, последующий тестовый пример может использовать B в качестве начальной точки, A в качестве конечной точки и C в качестве расстояния.

2 Элемент теста позволяет спрогнозировать возраст смерти человека на основе параметров образа жизни. Исходный тестовый пример имеет входные данные, включая установленный параметр в 10 сигарет, выкуриваемых в день, и ожидаемый результат — возраст 58 лет. Метаморфическое соотношение позволяет предположить, что если человек выкуривает больше сигарет, то ожидаемый возраст смерти, вероятно, уменьшится (а не увеличится). Таким образом, тестовый пример может быть создан с теми же входными параметрами образа жизни, за исключением увеличения количества выкуриваемых сигарет до 20 в день. Ожидаемый результат (прогнозируемый возраст смерти) для этого тестового примера может быть равным или меньшим 58 лет.

Ожидаемый результат не всегда выражен точным значением и часто описывается как функция фактического результата, достигнутого при выполнении исходного тестового примера (например, результат тестового примера больше, чем ожидаемый фактический результат для исходного тестового примера).

Одно и то же метаморфическое отношение часто можно использовать для получения нескольких тестовых примеров (например, метаморфическое отношение для функции, которая переводит речь в текст, можно использовать для создания нескольких тестовых примеров с использованием одного и того же входного речевого файла с разной входной громкостью). Если метаморфические отношения сформулированы формально (или полуформально), а исходные тестовые примеры доступны, необходимо обеспечить возможность автоматизации генерации тестовых примеров. Автоматическая генерация метаморфических отношений невозможна и требует дополнительных специализированных знаний о предметной области.

Этапы метаморфического тестирования:

1) строят метаморфические отношения:

- выявляют характеристики тестируемой программы и представляют их в виде метаморфических отношений между входными данными и ожидаемыми выходными данными, а также создают тестовый пример на основе исходного тестового примера;

2) анализируют метаморфические отношения:

- проводят анализ и согласовывают результаты с заказчиком/пользователями;

3) создают исходные тестовые примеры:

- создают исходные тестовые примеры (используя любой метод, включая случайное тестирование);

4) создают тестовые примеры:

- используя метаморфические отношения, генерируют новые тестовые примеры;

5) выполняют метаморфическое тестирование:

- выполняют тестирование на исходных и сгенерированных тестовых примерах. Убеждаются, что метаморфические отношения не нарушены. В противном случае, результат тестирование будет неудовлетворительным, что является свидетельством ошибки.

Метаморфическое тестирование успешно используют в самых разных областях применения ИИ, таких как биоинформатика, веб-сервисы, классификаторы МО, поисковые системы и безопасность.

Подробное описание метаморфического тестирования и связанных с ним требований к тестовому покрытию приведено в [1].

5.3.1.13 Тестирование модели на смещенность

Тестирование модели МО на необъективность может выполняться на двух уровнях.

Первоначально можно использовать независимый набор тестов без выявленных систематических ошибок, чтобы определить, работает ли модель одинаково с набором тестов без систематических ошибок и с обучающими данными. Если результаты динамического тестирования схожи, это говорит о том, что обучающие данные также не содержат систематических ошибок. Однако существует и другое объяснение. Например, причина может заключаться в том, что тестовые данные содержат систематические ошибки, но при этом являются необъективными, как и обучающие данные, хотя вероятность такого сценария мала. Или же тестовые данные не содержат систематических ошибок, но такая ошибка имеется в обучающих данных, но возникает она только в группах риска. Поэтому общие результаты, как нам кажется, не содержат систематических ошибок. Опять же, вероятность такого сценария мала. В случае, когда результаты тестовых и обучающих данных отличаются, мы можем сделать выводы о наличии ошибки. Если мы можем с уверенностью говорить о корректности тестовых данных, то это означает, что ошибка связана с обучающими данными. Однако без дальнейшего анализа мы не вправе утверждать, что расхождение возникло из-за необъективности данных или по другой причине. Тем не менее при выявлении ошибки необходимо внимательнее анализировать обучающие данные, независимо от причин.

Альтернативным решением для выявления нежелательной систематической ошибки может стать проведение динамического тестирования модели для определения подгрупп риска из общей совокупности пользователей. Далее результаты для каждой из этих групп можно сравнить, чтобы определить, как модель взаимодействует с каждой из групп риска, что позволит выявить систематическую ошибку.

При реализации обоих подходов сравниваемые результаты необходимо определять для каждого конкретного случая. Точность прогноза является одним из показателей эффективности модели и может применяться для сравнения результатов модели.

5.3.1.14 Анализ документации

В настоящее время не существует стандартного формата документирования данных систем МО. Поскольку системы МО все более распространены и используются активнее, предпринимаются согласованные усилия по улучшению как внутренней, так и внешней документации путем определения той информации, которая должна регистрироваться и предоставляться в отношении системы МО.

Документация системы МО должна включать следующие разделы:

- общую информацию: идентификационный номер, описание, сведения о разработчике, требования к оборудованию, сведения о лицензии, версию, дату и контактное лицо;
- применение — основной вариант использования, пользователи, вторичные варианты использования, методы самообучения, выявленные случаи необъективности, этические проблемы, проблемы безопасности, прозрачность, пороговые значения принятия решений, отклонение платформы и нарушения производительности;
- данные — сбор, доступность, предварительная обработка, использование, содержание, маркировка, размер, конфиденциальность, безопасность, смещенность и ограничения;
- обучение и производительность — алгоритм МО, структура МО, набор тестовых данных, выбор показателей производительности, пороговые значения для показателей производительности, фактические показатели производительности;
- тестирование — набор тестовых данных (описание и доступность), независимость, результаты, надежность, дрейф и переносимость.

Это позволит составлять контрольные списки и использовать их для проверки документации МО.

Подробное описание приведено в [4].

5.3.1.15 Тестирование производительности модели

Производительность оценивают в соответствии с определенными установленными для конкретной модели критериями. Оценку выполняют с помощью среды разработки МО.

Более подробная информация о различных показателях эффективности МО, на основе которых будет проводиться тестирование модели, приведена в приложении.

5.3.1.16 Анализ соответствия модели

Данный анализ заключается в том, чтобы убедиться, что модель соответствует обозначенной проблеме. Например, что выбранная модель отвечает требованиям текущей ситуации и не будет использоваться в ряде других ситуаций, где она не может быть применена.

Также убеждаются в соответствии показателей производительности МО установленным требованиям.

5.3.1.17 Валидация модели

Проверяют, отвечает ли модель (и достигнутые ею показатели производительности) запросу пользователей.

A/B-тестирование на преднамеренно ухудшенных моделях позволяет определить, какие критерии производительности важны для пользователей.

5.3.1.18 Эксплуатационные испытания

Дрейф-тестирование

Дрейф-тестирование — это вид регрессионного тестирования, основанного на измерении показателей производительности операционной модели для выявления отклонений концепции выше пороговых значений.

Дрейф-тестирование применимо в тех случаях, когда существует вероятность того, что операционная модель могла измениться с течением времени из-за изменения операционной среды или при развертывании обновленной модели.

Тестирование на дрейф можно использовать для прогнозирования необходимости повторного обучения модели, когда показатели производительности модели записываются, а затем используются для определения тенденции к ухудшению.

Дрейф-тестирование — довольно сложная задача, если применяется к моделям, которые не имеют автоматического тестового оракула. В некоторых случаях эталонные данные становятся доступны после того, как модель предлагает прогноз, классификацию или рекомендацию (например, модель, предлагающая прогноз исхода выборов или вероятность продаж розничной сети).

Для решения проблемы дрейфа может потребоваться обновление модели или повторное обучение с использованием новых данных, что позволит создать более точную и надежную модель. В этом случае новая модель может быть создана и обучена с использованием обновленных обучающих данных. Сравнение новой модели с существующей моделью может быть проведено с помощью A/B-тестирования (см. 5.3.1.1), которое позволит убедиться, что производительность не ухудшилась.

Данный вид также известен как тестирование на устаревание модели.

Повторное тестирование и регрессионное тестирование

После устранения дефекта разработчиком необходимо убедиться, что исправление было выполнено успешно. Для этого проводится проверка, которая известна как повторное тестирование или подтверждающее тестирование. В большинстве случаев для повторного тестирования используются исходные тестовые примеры, связанные с фиксированным кодом, которые иногда дополняют новыми тестовыми примерами.

Когда разработчик вносит изменения в существующее ПО (например, для устранения дефекта, добавления функциональности, изменения нефункциональных характеристик), также проводится тестирование, чтобы убедиться, что исходное поведение ПО, которое, как ожидается, должно измениться в результате модификации, не подверглось негативному влиянию таких изменений. Такая проверка известна как регрессионное тестирование.

Регрессионное тестирование выполняется при обновлении модели, что также позволяет убедиться, что модель ведет себя корректно. Такие регрессионные тесты часто основаны на множестве результатов тестов на основе сценария, проведенных до выпуска модели.

Регрессионное тестирование может также включать тестирование производительности, что позволяет убедиться в том, что время, необходимое модели для получения результата, и/или использование памяти не превышают требуемые значения.

5.3.1.19 Тестирование производительности

Сравнение времени прогнозирования и использования памяти с требуемыми значениями

5.3.1.20 Проверка переобучения

Переобучение необходимо в тех случаях, когда модель изучает некорректные взаимосвязи на основе посторонней информации, такой как незначительные детали, случайные колебания и шум в

обучающих данных (т. е. обучающие данные содержат слишком много функций). Это значит, что модель запомнила обучающие данные и в рабочем режиме способна работать с данными, схожими с обучающими данными, но модель испытывает сложности с обобщением и обработкой новых данных. Один из способов выявить случаи переобучения — использовать независимые тестовые данные, которые отличаются от обучающих данных.

5.3.1.21 Тестирование нарушения функции вознаграждения

Независимые тесты могут выявить нарушения функции вознаграждения, поскольку в этих тестах используются иные средства измерения результата по сравнению с тестируемым интеллектуальным агентом.

5.3.1.22 Сценарное тестирование

Общие положения

Сценарное тестирование основано на отработке модели последовательности возможных взаимодействий между объектом тестирования и другими системами (в данном контексте пользователи часто рассматриваются как другие системы). Тестовая модель для сценарного тестирования обычно включает набор основных и альтернативных сценариев (например, аномальное использование, экстремальные или стрессовые условия, исключения и обработка ошибок), связанных с объектом тестирования, где каждый сценарий охватывает одну последовательность взаимодействий. Тестирование по сценариям использования является одной из распространенных форм сценарного тестирования.

Подробное определение сценарного тестирования и соответствующие показатели тестового покрытия приведены в [1].

Получение тестового сценария СМО

Для тестирования системы МО необходимо сформировать тестовые сценарии для проверки отдельных компонентов системы МО, взаимодействия этих компонентов с остальными компонентами системы, а также всей системы взаимодействующих компонентов и взаимодействия системы с окружающей средой.

Тестовые сценарии могут быть получены из нескольких источников:

- требования к системе;
- проблемы пользователей;
- автоматические сообщения о проблемах (например, для автономных систем);
- отчеты об авариях (например, для физических систем);
- данные о страховании (например, для застрахованных систем, таких как беспилотные автомобили);
- данные контрольно-надзорных органов (например, собранные в рамках законодательства);
- тестирование на различных уровнях (например, неисправности/аномалии на полигоне или реальных дорогах могут породить примечательные сценарии тестирования беспилотного автомобиля на других уровнях тестирования; кроме того, выборка тестовых сценариев на виртуальной среде должна быть также проведена на реальных дорогах для подтверждения репрезентативности виртуальной среды).

Для генерации тестовых сценариев могут быть использованы комбинаторное тестирование (см. 5.3.1.8), метаморфическое тестирование (см. 5.3.1.12) и фазз-тестирование (см. 5.3.1.11).

5.3.1.23 Тестирование на побочные эффекты

Определяют потенциально опасные побочные эффекты и генерируют тесты, которые приводят к проявлению этих побочных эффектов в системе.

5.3.1.24 Дымовое тестирование

Убеждаются, что полученная модель пригодна для дальнейшего, более детального и целевого тестирования путем проведения обратного (дифференциального) тестирования в сравнении с простой базовой моделью (например, моделью линейной регрессии с небольшим количеством признаков) и простым тестовым набором данных с известной истинностью/бинарной классификацией.

5.3.1.25 Тестирование нейронных сетей по методу «белого ящика»

Общие положения

Тестирование нейронных сетей по методу «белого ящика» существенно отличается от тестирования процедурного кода, например, с использованием тестирования операторов и ветвей. Традиционные показатели покрытия не очень подходят для нейронных сетей, поскольку 100 %-ное покрытие операторов обычно достигается с помощью одного тест-кейса. Как правило, дефекты скрываются в самой нейронной сети. В связи с этим были предложены различные показатели покрытия, основанные

на значениях активации нейронов (или пар нейронов) в нейронной сети, в процессе тестирования нейронной сети.

Наличие показателей покрытия нейронной сети позволяет тестировщикам максимизировать покрытие, что, как было показано, позволяет выявлять некорректное поведение в системах, основанных на ИИ, например в системах беспилотных автомобилей.

Покрытие нейронов

Покрытие нейронов для набора тестов определяется как доля активированных нейронов, деленная на общее количество нейронов в нейронной сети (обычно выражается в процентах). При определении охвата нейронов нейрон считается активированным, если его значение активации больше нуля.

Покрытие насыщения

Покрытие насыщения для набора тестов определяется как доля нейронов, превышающих значение активации насыщения, деленная на общее количество нейронов в нейронной сети (обычно выражается в процентах). Для покрытия насыщения в качестве значения активации насыщения должно быть выбрано значение между 0 и 1.

Покрытие изменения знака

Покрытие изменения знака для набора тестов определяется как доля нейронов, активированных как положительными, так и отрицательными значениями активации, деленная на общее количество нейронов в нейронной сети (обычно выражается в процентах). Значение активации, равное нулю, считается отрицательным значением активации.

Покрытие изменения значения

Покрытие изменения значений для набора тестов определяется как доля активированных нейронов, значения активации которых отличаются более чем на величину изменения, деленная на общее количество нейронов в нейронной сети (обычно выражается в процентах). Для определения покрытия изменения значения в качестве величины изменения следует выбирать значение от 0 до 1.

Покрытие по знаку

Покрытие по знаку для набора тестов достигается в случае, если можно показать, что каждый нейрон, меняя знак, по отдельности вызывает изменение знака у другого нейрона в следующем слое, в то время как все остальные нейроны в следующем слое остаются неизменными (т. е. не меняют знак). По своей концепции этот уровень охвата нейронов аналогичен охвату модифицированных условий/решений.

Покрытие слоя

Показатели покрытия также могут быть определены на основе целых слоев нейронной сети и того, как изменяются значения активации для набора нейронов в целом слое (например, абсолютно или относительно друг друга).

Эффективность тестирования по методу «белого ящика»

В настоящее время имеется мало данных об эффективности тестирования нейронных сетей различными мерами покрытия по методу «белого ящика». В целом можно сделать вывод, что критерии, требующие большего количества тестов, обнаруживают больше дефектов, чем те, которые требуют меньшего количества тестов, что позволяет сделать вывод об относительной эффективности мер. Из вышеописанных мер покрытия можно выделить несколько взаимосвязей. Все остальные меры являются заместителями покрытия нейронов, а покрытие по знаку также является заместителем покрытия изменения знака. Полная иерархия заместителей показана на рисунке 3. Если стрелка указывает от одной меры к другой, то это означает, что если первая мера полностью достигнута, то автоматически достигается и вторая. Например, если достигнуто покрытие насыщения, то автоматически достигается покрытие нейронов.

Рисунок 3 — Иерархия в нейронной сети по методу «белого ящика»

Несмотря на удобную для понимания форму, высокий уровень покрытия нейронов может быть достигнут лишь при использовании нескольких тестовых случаев, что ограничивает эффективность тестирования. Первые результаты по покрытию насыщения показывают, что данная мера может быть полезной для создания тестов, охватывающих угловые случаи, приводящие к дефектам, однако может потребоваться установить индивидуальное пороговое значение для каждой нейронной сети. Естественно, что более высокие значения величины изменения потребуют большего количества тестовых случаев для покрытия изменений значений. Покрытие по знаку обычно является наиболее строгим из указанных здесь критериев покрытия.

5.4 Тестирование среды разработки

5.4.1 Введение в тестирование среды разработки

Тестирование среды разработки непосредственно связано с выбором алгоритма и среды разработки МО. Оно также охватывает любое тестирование, связанное с поддержкой деятельности по оценке, настройке и тестированию рабочего процесса МО, обеспечиваемой средой разработки МО.

5.4.2 Риски в среде разработки

Примеры рисков среды разработки разделены на несколько категорий для удобства документирования (другие категории допустимы):

а) риски, связанные с системой разработки:
- неоптимальный выбор;
- дефект проектирования;
- дефект реализации;
- дефект пользовательского интерфейса;
- дефект библиотеки разработки;
- дефект API;
- дефект развертывания;

б) риски алгоритмов МО:
- неоптимальный выбор алгоритма;
- дефект проектирования;
- дефект реализации;
- недостаточная объяснимость;
- дефект документации;

в) риски, связанные с обучением, оценкой и настройкой:
- неправильное распределение данных на обучающий, валидационный и тестовый наборы данных;
- неправильный выбор подхода к оценке (например, n-кратная перекрестная валидация);
- неоптимальный выбор гиперпараметров.

Не все перечисленные риски, связанные с входными данными, применимы к каждой системе или компоненту ИИ, и другие (не перечисленные) риски также могут быть применены в зависимости от конкретной тестируемой системы или компонента ИИ.

5.4.3 Виды и методы тестирования среды разработки

Общие положения

Виды и методы тестирования, используемые для устранения соответствующих рисков, перечисленных в 5.4.2. Представленный список типов и методов тестирования не является исчерпывающим, и в различных ситуациях будут уместны другие типы и методы тестирования:

- тестирование на алгоритмическую предвзятость;

- тестирование API;
- анализ кода;
- тестирование оптимизации внедрения;
- тестирование динамического элемента;
- тестирование конфигурации среды разработки;
- тестирование безопасности среды разработки;
- анализ области применения среды разработки;
- тестирование реализации библиотек;
- тестирование алгоритмов МО;
- тестирование развертывания модели МО;
- тестирование объяснимости модели МО;
- тестирование воспроизводимости модели МО;
- тестирование отката модели МО;
- тестирование структуры модели МО;
- статический анализ;
- тестирование восстанавливаемости тренировочных данных;
- тестирование качества производительности.

5.5 Тестовая среда МО

Требования к тестовой среде должны быть рассмотрены в рамках планирования тестирования, чтобы обеспечить возможность приобретения, установки, настройки и валидации всех необходимых компонентов до начала тестирования, а также возможность их очистки после его завершения. Требования к тестовой среде могут включать:

- ПО (операционные системы, программы и другие приложения);
- сервисы (виртуальные или облачные сервисы);
- аппаратное обеспечение (серверы, настольные компьютеры, ноутбуки и мобильные устройства);
- сеть (коммутаторы, маршрутизаторы, сетевые соединения на определенных скоростях);
- интерфейсы (для подключения к внутренним, виртуальным или сторонним системам);
- периферийные устройства (принтеры, сканеры и сканеры карт);
- данные (включая данные, хранящиеся в базах данных);
- средства тестирования (средства управления и выполнения тестов);
- доступность (часы/день и время установки).

Требования к тестовой среде должны быть сформулированы таким образом, чтобы их можно было объяснить и понять. В случае если настройкой тестовой среды занимается отдельная команда, то она должна подготовить отчет о готовности тестовой среды и подтвердить, что среда была разработана и настроена в соответствии с требованиями.

Кроме того, в случае привлечения отдельной команды должно быть заключено соглашение об уровне услуг, определяющее, когда и в каком порядке будут решаться проблемы с тестовой средой.

6 Тестирование экспертных систем

6.1 Введение в тестирование экспертных систем

Экспертные системы (ЭС) или системы, основанные на знаниях (KBS, Knowledge-based systems) описаны в приложении А. В настоящем разделе кратко описаны методы тестирования и оценки качества, относящиеся к ЭС. Для ЭС проводят два вида тестирования: структурное тестирование и тестирование на уровне системы.

6.2 Структурное тестирование

При структурном тестировании ЭС оценивается на предмет выявления структурных аномалий, таких как непротиворечивость правил или знаний и краткость закодированных знаний. Также могут быть проведены дополнительные тесты на структурную полноту и нарушение семантических ограничений.

6.2.1 Тестирование на уровне правил

Данное тестирование проводится на каждом уровне правил или элементов знаний. Иногда набор связанных правил или связанных элементов знаний рассматривается совместно и тестируется вместе.

6.2.1.1 Тестирование на последовательность

ЭС является последовательной, если набор ее правил или элементов знаний не приводит к противоречивым решениям, двусмысленностям или запускает циклические цепочки выводов из корректных входных данных. Данное тестирование можно проводить различными методами.

Тестирование на противоречивость знаний

Данное тестирование позволяет определить, не приводят ли два правила или элемента знаний с одинаковыми начальными условиями к противоречащим друг другу решениям или выводам.

Циклические цепочки выводов

Данное тестирование проводится для определения любого набора правил или элементов знаний, которые ссылаются друг на друга в своих условиях таким образом, что в процессе вывода формируется цикл.

Неоднозначность знаний

Данное тестирование выполняется для определения наличия различных правил или элементов знаний, которые приводят к возникновению неопределенных результатов в процессе рассуждений или выводов.

6.2.1.2 Проверка на лаконичность

ЭС считается лаконичной, если она не содержит ни одного лишнего или бесполезного правила в какой-либо форме. Данное тестирование можно проводить различными методами.

Избыточность правил

Данное тестирование проводится для того, чтобы найти любой набор правил или элементов знаний, которые имеют одинаковый набор начальных условий и одинаковый набор конечных решений или выводов. Если это так, то в ЭС имеются избыточные правила или элементы знаний, которые необходимо удалить.

Вложенные знания

Данное тестирование выполняется для поиска любого набора правил или элементов знаний, которые являются полным подмножеством другого набора правил или элементов знаний.

Излишние и бесполезные знания

Данное тестирование проводится для поиска всех правил и элементов знаний, содержащих нереалистичные условия, которые никогда не могут возникнуть в домене, или правила, которые никогда не будут достигнуты в процессе рассуждений (также называемые недостижимыми правилами), а также просто правила, недействительные в силу ограничений, накладываемых доменом, или имеющие избыточные условия в описании правила.

6.2.1.3 Проверка на полноту и корректность

ЭС считается полной и корректной, если она содержит все релевантные правила или элементы знаний, кодирующие все необходимые условия и ситуации, релевантные для данной области. Также выявляются недостающие знания и правила.

Данное тестирование можно проводить различными методами.

Тестирование на уровне правил

При данном тестировании каждое правило или элемент знаний тестируется или оценивается отдельно с моделированием входных условий, а выходные решения или выводы оцениваются на предмет их корректности для заданных входных условий. Данное тестирование может быть выполнено с использованием различных инструментов тестирования, позволяющих моделировать различные сценарии.

Тестирование ограничений

При данном тестировании проверяются ограничения, наложенные на систему, основанную на правилах или знаниях, с целью определить возможное нарушение данных ограничений во время выполнения ЭС.

Тестирование выходных решений

При данном тестировании набор всех возможных выходных решений, которые должны быть выведены системой KBS, специально проверяется на предмет того, способна ли система ЭС вывести все правильные решения, которые ожидаются от системы KBS. Кроме того, система не должна выдавать недействительные и незапланированные решения.

Тестирование недостающих знаний

При данном тестировании правила рассматриваются и проверяются на предмет того, все ли возможные диапазоны значений каждого входного параметра учтены в ЭС.

Тестирование квантификации правил

При данном тестировании проверяется универсальная квантификация правил в ЭС на предмет того, охватывает ли она соответствующие объекты входной области в процессе рассуждения.

6.2.1.4 Тестирование онтологии

При данном тестировании онтология, используемая в ЭС, проверяется на предмет корректности таксономии и связей внутри нее.

Тестирование структуры онтологии

При тестировании структуры онтологии проверяется соответствие структуры онтологии той области, для которой была разработана ЭС. Это может быть сделано путем ручной проверки или с помощью инструментов сравнения онтологий для выявления различий между закодированной и истинной онтологией.

Тестирование логики описания

При данном тестировании используются методы онтологических рассуждений для определения корректности и достоверности производных, выполненных в онтологии.

6.3 Тестирование на системном уровне

При тестировании на уровне системы ЭС оценивается на предмет выявления аномалий на уровне системы.

6.3.1 Тестирование выводов

Данное тестирование включает в себя тестирование механизма вывода ЭС на предмет правильности применения правил базы знаний для получения точных результатов. Это может быть сделано путем создания тестовых примеров, охватывающих различные сценарии, и последующей оценки результатов, генерируемых механизмом вывода.

6.3.1.1 Тестирование прямого вывода

В данном тестировании оцениваются правила, закодированные для рассуждений на основе данных или на основе прямого вывода (т. е. правила, которые запускаются при наличии данных), с целью определить, являются ли они валидными частями процесса рассуждений. Алгоритмы прямого вывода, такие как алгоритм Рете, могут быть визуализированы и проверены на наличие соответствующих частей процесса рассуждения.

6.3.1.2 Тестирование обратного вывода

При данном тестировании правила, в которых реализовано рассуждение по целям или обратному выводу, оцениваются на предмет того, являются ли они валидными частями процесса рассуждения. Рассуждения, основанные на целях и подцелях, являются основной составляющей обратного вывода, и в данном тестировании этот процесс проверяется.

6.3.1.3 Тестирование по перебору с возвратом

При данном тестировании правила или элементы знаний, предназначенные для множественных решений, подвергаются проверке на отказ и возврату назад, чтобы определить, найдены ли правильные множественные решения в процессе вывода.

6.3.1.4 Тестирование вероятностного рассуждения

При данном тестировании выполняются правила, использующие вероятности или оценку неопределенности как часть процесса принятия решений, для того чтобы определить, являются ли выведенные ими значения вероятности действительными для решений, принимаемых на выходе при заданном наборе входных данных. Некоторые ЭС могут использовать байесовские рассуждения, и в таком случае все правила или элементы знаний, использующие байесовские рассуждения, проверяются на предмет достоверности выводов.

6.3.1.5 Тестирование нечеткой логики

При данном тестировании проверяются диапазоны и производные выводов, основанных на нечеткой логике.

6.3.1.6 Немонотонное тестирование и обновление представлений

При данном тестировании общие выведенные факты могут подвергаться пересмотру в процессе вывода. Динамическая база фактов проверяется на предмет того, что соответствующие факты («представления») отменяются, пересматриваются и снова утверждаются.

6.3.1.7 Тестирование на эвристику

При данном тестировании эвристика, закодированная в ЭС, проверяется на предмет того, являются ли корректными результаты, получаемые в результате выполнения эвристики во время выполнения ЭС. Также проверяются значения эвристик на предмет их допустимости и последовательности.

6.3.1.8 Тестирование методов поиска

При данном тестировании выполняется проверка методов поиска, используемых в ЭС, на предмет того, приводят ли они к правильному набору решений за нужное количество времени.

6.3.1.9 Тестирование избыточной специализации

При данном тестировании правила или элементы знаний в ЭС проверяются, чтобы исключить ситуации, когда в каком-либо из правил задано больше условий, чем предполагалось в данном элементе знаний.

6.3.1.10 Тестирование на чрезмерную обобщенность

При данном тестировании правила или элементы знаний в ЭС проверяются, чтобы исключить ситуации, когда в данном элементе знаний задано меньше условий, чем предполагалось.

6.3.2 Тестирование практичности

Данное тестирование подразумевает тестирование удобства использования системы ЭС, а именно ее удобность для пользователей и простоту применения. Это может быть сделано путем выборки различных групп потенциальных пользователей системы, привлечения этих пользователей к бета-тестированию, а также путем анализа поведения пользователей и моделей их взаимодействия.

6.3.3 Тестирование производительности

Данное тестирование подразумевает тестирование производительности ЭС на предмет ее способности обрабатывать различные наборы входных данных и быстро отвечать на запросы пользователей. Для этого можно использовать различные средства нагрузочного тестирования, имитирующие высокий уровень пользовательского трафика.

6.3.4 Регрессионное тестирование

Данное тестирование подразумевает тестирование ЭС после внесения изменений, чтобы убедиться в том, что она по-прежнему работает корректно. Для этого можно использовать автоматизированные средства тестирования, которые моделируют различные сценарии и оценивают выходные данные, выдаваемые системой.

6.3.5 Тестирование с внедрением неисправностей

Данное тестирование ЭС заключается в преднамеренном внесении неисправностей в систему для проверки ее устойчивости. Для этого можно использовать различные средства внедрения неисправностей, моделирующие различные типы сбоев и ошибок.

6.3.6 Тестирование на граничные условия

Данное тестирование заключается в проверке реакции ЭС на входные сигналы, находящиеся вблизи границы системы.

6.3.7 Негативное тестирование

Данное тестирование включает в себя проверку реакции ЭС при подаче недопустимых входных сигналов или сигналов, выходящих за пределы ожидаемой области.

6.3.8 Тестирование домена

Данное тестирование включает в себя проверку реакции ЭС при поступлении входных данных из новой области, не включенной в базу знаний.

6.3.9 Тестирование на устойчивость

Данное тестирование включает в себя тестирование ЭС в экстремальных условиях и при наличии ошибок, таких как поврежденные данные и отсутствующие правила.

6.3.10 Тестирование модулей ЭС

В данном тестировании весь модуль ЭС тестируется отдельно, чтобы проверить, обеспечивает ли этот модуль правильный набор выходных решений.

6.3.11 Тестирование на основе исторических данных и решений

При этом тестировании ЭС выполняется с использованием исторических данных, чтобы проверить, соответствуют ли ее выходные решения историческим решениям, принятым в домене. Такой процесс требует наличия исторических входных данных и исторических данных о решениях за период, которые принимались человеком или другими альтернативными способами принятия решений до появления системы.

6.3.12 Тестирование точности и полноты системы

При этом тестировании точность системы рассчитывается с использованием различных показателей, которые соотносятся с реальным процессом принятия решений, который пытается автоматизировать ЭС. При этом также проверяется полнота ЭС.

6.3.13 Пробное тестирование с экспертами

При этом тестировании правила и элементы знаний, закодированные в системе, повторно объясняются экспертам в данной области, чтобы они могли подтвердить закодированные в ЭС знания.

6.3.14 Тестирование по случайным выборкам

При этом тестировании случайные входные выборки передаются ЭС для определения ее выходных решений, а затем эти решения сверяются с истинными решениями.

6.3.15 Тестирование методом полного перебора

При этом тестировании ЭС передаются различные входные образцы таким образом, что выполняются все правила, а затем выходные решения сверяются с истинными.

6.3.16 Систематическое тестирование

При этом тестировании сложность входных образцов планомерно увеличивается, они передаются в ЭС для определения ее выходных решений, а затем выходные решения сверяются с истинными.

6.3.17 Параллельное тестирование

Данное тестирование заключается в одновременном принятии решений человеком и ЭС. ЭС и группе экспертов передаются различные входные образцы для определения их выходных решений, а затем выходные решения ЭС сверяются с эталонными решениями, принятыми человеческими экспертами.

6.3.18 Система работает самостоятельно, выходные данные проверяются экспертами

При этом тестировании ЭС получает реальные входные образцы для определения своих выходных решений, а затем эти решения проверяются человеческими экспертами.

7 Тестирование логических программ

Логические программы (ЛП) — это программы, которые создаются с помощью языка логического программирования, например языка Prolog, для построения систем ИИ (см. приложение А). В данном разделе кратко описаны методы тестирования и методы оценки качества, относящиеся к ЛП.

7.1 Введение в тестирование ЛП

ЛП состоит из клауз первого порядка, содержащих логические формулы и выражения, кодирующие некоторые знания об области. Для ЛП проводится два типа тестирования: структурное тестирование и тестирование на уровне системы. Оба типа тестирования позволяют определить, правильно ли ЛП представляет заданные знания и обосновывает их.

7.2 Структурное тестирование

При структурном тестировании ЛП оценивается на предмет выявления структурных аномалий в логических положениях на предмет их согласованности и краткости. Также могут проводиться дополнительные тесты на структурную полноту и нарушение семантических ограничений.

7.2.1 Тестирование на уровне клауз

Тестирование проводится на каждом уровне логических клауз. Иногда набор связанных клауз тестируется вместе.

7.2.1.1 Тестирование унификации

Это тестирование проверяет, правильно ли и в соответствии с ожиданиями происходит объединение клаузы или набора связанных клауз с заданным запросом или подцелью. Термины и переменные логической клаузы должны унифицироваться с терминами и переменными данного запроса или подцели, и эта унификация не должна создавать невыполнимых ситуаций.

7.2.1.2 Тестирование связывания

Это тестирование проверяет, происходит ли связывание неизвестных переменных/литералов клаузы или набора связанных клауз с заданным запросом или подцелью с их связанными переменными или известными литералами правильно, т. е. как это ожидается в данном случае.

7.2.1.3 Проверка квантификации

Это тестирование проверяет, правильна ли экзистенциальная и/или универсальная квантификация для каждой из переменных, используемых в логических клаузах, правильно ли они отражают данное знание. Экзистенциальная квантификация проверяет, существует ли один экземпляр переменной, который успешно связывается, чтобы удовлетворить логическую клаузу. Универсальная квантифика-

ция проверяет, все ли возможные экземпляры переменной успешно связываются для удовлетворения логической клаузы.

В ЛП переменные в логических клаузах по умолчанию будут универсально квантифицированы. Однако, если считается, что какая-либо переменная имеет экзистенциальное квантование, то этот аспект должен быть проверен, чтобы убедиться, что любой экземпляр значения этой переменной удовлетворяет привязке этой переменной в клаузе. Аналогично, если переменная универсально квантифицирована, то должна быть проведена проверка, чтобы убедиться, что все экземпляры значений этой переменной могут удовлетворять привязке этой переменной в клаузе.

7.2.1.4 Тестирование запросов/тестирование целей

Это тестирование проверяет, возвращают ли запросы (также называемые целями), задаваемые ЛП, корректные результаты. Либо результатом должно быть удовлетворение логических положений с допустимыми связями для неизвестных переменных в запросе (или цели), либо результатом должен быть сбой запроса и его подцелей с отсутствием допустимых привязок для переменных в запросе (или цели).

7.2.1.5 Тестирование динамической базы фактов

Это тестирование проверяет, правильно ли создаются и удаляются факты в динамической базе фактов. Существует два типа тестирования: тестирование создания фактов и тестирование извлечения фактов.

Тестирование создания фактов

Это тестирование проверяет, действительно ли факты, которые должны быть получены из существующих фактов и логических клауз, получены и утверждены в динамической базе фактов.

Тестирование на извлечение фактов

Это тестирование проверяет, действительно ли факты, которые должны быть извлечены (или удалены) из существующей динамической базы фактов, удалены или не удалены из базы фактов.

7.2.1.6 Тестирование обработки списков

В логических языках, поддерживающих обработку списков, таких как Prolog, это тестирование проводится для того, чтобы убедиться, что переменные списков в предложениях обрабатываются так, как было задумано при кодировании знаний. Начало и конец списка должны объединяться и связываться с соответствующими переменными, значениями или списками во время выполнения.

7.2.1.7 Проверка краткости

ЛП считается краткой, если она не содержит ни одного лишнего или бесполезного пункта в любой форме. Существуют различные специфические тесты, которые необходимо выполнить.

Подчиненная клауза

Этот тест выполняется для поиска любой клаузы, которая является полным подмножеством другой клаузы. Это происходит, когда две разные клаузы имеют одинаковое логическое заключение, но условия, указанные в одной клаузе, являются полным подмножеством условий, указанных во второй клаузе.

7.2.1.8 Проверка полноты и корректности

ЛП считается полной и корректной, если она содержит все и только соответствующие логические клаузы, кодирующие все и только необходимые условия, и ситуации, которые относятся к данной области.

7.3 Тестирование на уровне системы

При тестировании на уровне системы ЛП оценивается на предмет обнаружения аномалий на уровне системы.

7.3.1 Тестирование выводов

Тестирование выводов включает в себя тестирование ЛП на предмет того, правильно ли она применяет правила логического вывода для получения корректных результатов. Это может быть сделано путем создания тестовых примеров, охватывающих различные сценарии, а затем путем оценки результатов, полученных логическим механизмом программы.

7.3.1.1 Проверка рекурсии

Некоторые логические клаузы могут содержать рекурсию в своем теле, когда часть клаузы рекурсивно обращается к заголовку той же клаузы. Это должно быть тщательно проверено, чтобы определить, выполняет ли логическая клауза корректную рекурсию для всех и только предполагаемых значений переменных.

7.3.1.2 Проверка вложенных положений

Некоторые логические клаузы могут ссылаться на другие логические клаузы в своем теле вложенным образом. Это должно быть тщательно протестировано, чтобы проверить, вызываются ли корректные вложенные клаузы и правильно ли обрабатывается возвращаемое значение (удовлетворительно или отказ).

7.3.1.3 Тестирование с обратной цепочкой

При этом тестировании логические клаузы проверяются на предмет того, правильно ли оценена цель, поставленная перед ЛП. Рассуждения о целях и подцелях являются основной частью обратной цепочки, и в данном тестировании этот процесс проверяется.

7.3.1.4 Тестирование обратным отслеживанием

При этом тестировании логические клаузы, которые должны найти несколько решений, подвергаются сбоям и обратному отслеживанию чтобы определить, найдены ли в процессе вывода правильные множественные решения.

7.3.1.5 Тестирование отказа

При этом тестировании логические клаузы проверяются на предмет того, что произойдет, если в процессе привязки значения этой клаузы произойдет сбой связывания. Результатом может быть завершение клаузы через отказ, откат назад для другого значения или успешное выполнение другой родственной клаузы с тем же заголовком.

7.3.1.6 Разрез для управления обратным отслеживанием

В некоторых языках логического программирования, таких как Prolog, для управления обратным отслеживанием используется функция разреза. Если эта функция используется, то необходимо проверить, контролируется ли автоматический обратный путь при выполнении необходимых условий.

7.3.1.7 Тестирование избыточной специализации

В этом тесте логические клаузы проверяются на предмет наличия в них большего количества условий, чем предполагалось.

7.3.1.8 Тестирование на чрезмерную обобщенность

В этом тесте логические клаузы проверяются на предмет того, не содержат ли они меньше условий, чем предполагалось.

7.3.1.9 Циклические цепочки умозаключений

Это тестирование проводится для выявления любого набора логических клауз, которые ссылаются друг на друга в своих условиях таким образом, что во время выполнения процесса вывода образуется цикл.

7.3.1.10 Проверка на непротиворечивость

ЛП является непротиворечивой, если набор ее клауз не выводит противоречивых решений и не вызывает бесконечной рекурсии из корректных входных данных или фактов. Существуют различные специальные тесты.

7.3.1.11 Противоречивые знания

Этот тест помогает определить, не приводят ли две логические клаузы с одинаковым телом к противоречивым решениям или выводам.

Приложение А
(справочное)

Краткая характеристика экспертных систем

А.1 Системы, основанные на правилах

Системы, основанные на правилах, представляют собой разновидность экспертных систем, в которых знания кодируются в виде набора правил для решения реальных задач. В системе, основанной на правилах, знания о проблемной области представляются в виде набора правил «если-тогда», которые затем используются для вывода новых фактов или принятия решений на основе имеющихся данных. Каждое правило в системе, основанной на правилах, состоит из двух частей: условия и действия. Условие определяет обстоятельства, при которых правило применимо, а действие — порядок действий, которые должны быть предприняты при выполнении условия. Системы, основанные на правилах, часто используются в приложениях, где проблемная область хорошо определена, а правила могут быть легко сформулированы экспертами в данной области. По сути, система на основе правил — это особый тип системы, основанной на знаниях, в которой знания представлены в виде набора правил «если-тогда».

А.2 Системы, основанные на знаниях

Системы, основанные на знаниях, — это экспертные системы, использующие для решения задач схему представления знаний. В системах, основанных на знаниях, знания о проблемной области представляются в виде набора взаимосвязанных понятий и правил (иногда с использованием онтологии), которые затем используются для рассуждений о проблеме и генерации новых знаний и фактов.

Одним из ключевых преимуществ систем, основанных на знаниях, является их способность рассуждать о сложных связях и зависимостях между понятиями. Системы, основанные на знаниях, используют различные схемы представления знаний, такие как представление на основе прецедентов, представление на основе шаблонов, логика описания, семантические сети, и онтологии, для представления знаний в структурированном и организованном виде. Такие схемы представления позволяют системе рассуждать о связях между понятиями и генерировать новые знания и факты на основе имеющихся знаний.

Приложение Б
(справочное)

Введение в тестирование ПО

Б.1 Введение

Тестирование ПО является одним из видов контроля качества, который вместе с обеспечением качества составляет менеджмент качества. Верификация и валидация — это концепции контроля качества, поддерживаемые тестированием ПО. Верификация ориентирована на соответствие тестового элемента спецификациям, заданным требованиям или другим документам, а валидация — на ценность тестового элемента в отношении предполагаемого использования заинтересованными сторонами.

Б.2 История тестирования ПО

Тестирование ПО стало неотъемлемой частью разработки ПО задолго до появления моделей жизненного цикла, причем упоминания об отдельной деятельности по тестированию ПО появились еще в 1954 г. Сегодня доля затрат на тестирование в жизненном цикле варьируется в значениях от 20 % до 80 % для критически важных систем.

Б.3 Статическое и динамическое тестирование

Тестирование ПО может проводиться в двух формах: статической и динамической.

Статическое тестирование представляет собой оценку тестового элемента, при которой выполнение кода не происходит и может выполняться вручную (например, обзоры) или с помощью инструментов (например, статический анализ).

Обзоры, как определено в [8], различаются по формальности и включают в себя инспекции, технические обзоры, ознакомительные и неформальные обзоры. Статический анализ предполагает использование инструментов для обнаружения аномалий в коде или документах без их выполнения (например, компилятор, анализатор цикломатической сложности или анализатор безопасности кода).

Динамическое тестирование предполагает выполнение кода и запуск тестовых примеров и может выполняться вручную или с помощью тестовых инструментов. Тестовые случаи создаются с использованием методов проектирования тестов, определенных в [1], представленных «черным ящиком» (на основе спецификации), «белым ящиком» (на основе исходного кода) или их сочетанием («серым ящиком»).

Требование создания тестовых примеров, как правило, делает затраты на динамическое тестирование гораздо выше, по сравнению со стоимостью статического анализа.

Б.4 Систематическое тестирование ПО

Чтобы доказать, что конкретный тест удовлетворяет всем требованиям при любых обстоятельствах, необходимо провести динамическое тестирование всех возможных комбинаций входных значений во всех возможных состояниях. Такая деятельность называется «исчерпывающим тестированием», однако на практике тестовые элементы оказываются настолько сложными, что применение исчерпывающего тестирования оказывается невозможным. Поэтому на практике при тестировании ПО наборы тестов формируются путем выборки из (чрезвычайно большого) множества возможных комбинаций входных данных и состояний. Выбор подмножества возможных тестов, которые с наибольшей вероятностью обнаружат интересующие проблемы, является одной из самых сложных задач тестировщика, но его помогает сделать использование методов разработки тестовых примеров, которые обеспечивают систематические средства для получения этого подмножества.

Б.5 Цель тестирования

Обычно тестирование выполняют для решения сразу нескольких задач. Типовые задачи включают (но не ограничиваются):

а) обнаружение дефектов, что позволяет впоследствии устранить их, повысив тем самым качество ПО;

б) сбор информации об объекте тестирования. Тестирование генерирует информацию, которая может служить различным целям, например:

1) разработчики могут использовать информацию для устранения дефектов, повышения качества кода или создания более качественного кода в будущем;

2) тестировщики могут использовать информацию для создания лучших тестовых примеров;

3) менеджеры могут использовать информацию для оценки ситуации в проекте;

в) создание уверенности и принятие решений. Предоставляя доказательства того, что тестовый элемент работает правильно в определенных условиях, повышается уверенность заинтересованных сторон в том, что тестовый элемент будет работать правильно в эксплуатации. При достаточной уверенности заинтересованные стороны могут принять решение о выпуске тестового образца.

Тестирование проводят для достижения некоторых или всех вышеуказанных целей. Также могут существовать дополнительные цели, не указанные в перечне. Эти цели должны быть определены и согласованы в качестве отправной точки любой деятельности по тестированию.

Б.6 Тестирование на базе рисков в основе тестирования ПО

Б.6.1 Общие положения

Процесс управления рисками путем тестирования аналогичен большинству других процессов управления рисками. Вначале выявляются потенциальные риски, иногда с использованием контрольных списков, основанных на характеристиках качества, например, определенных в ГОСТ Р ИСО/МЭК 25010 и в [9]. Затем их анализируют с целью определения потенциального воздействия (серьезности), которое они могут оказать (на поставляемый продукт или проект) в случае их возникновения. Определяют вероятность возникновения каждого риска, которая может быть основана на таких факторах, как качество требований, возможности персонала, сложность системы и историческая информация. Затем согласовывают уровень подверженности риску, основанный на сочетании влияния и вероятности каждого риска. Затем риски расставляют по приоритетам и при необходимости (или возможности) принимают решение об их устранении, принимая во внимание тот факт, что устранение одного риска может стать причиной возникновения или увеличения подверженности другому риску.

Примеры рисков, связанных с тремя уровнями тестирования МО, перечислены в 5.2—5.4.

Б.6.2 Категории риска

Риски разделяют на продуктовые и проектные. Продуктовые риски связаны с поставляемым продуктом и включают в себя возможные потери или ущерб для пользователей или других заинтересованных сторон из-за того, что продукт не соответствует требованиям. Проектные риски связаны с тем, как разрабатывается продукт, и включают угрозу того, что разработчики, тестировщики и другие участники проекта не будут обладать необходимыми навыками или временем для выполнения требуемых действий, а также того, что продукт будет сдан с опозданием или превысит бюджет.

Б.7 Процессы тестирования (см. ГОСТ Р 56921)

Б.7.1 Общие положения

В ГОСТ Р 56921 используется трехуровневая модель процесса тестирования — на уровне организационного управления, уровне менеджмента тестирования и уровне динамического тестирования. Процессы тестирования обладают общими характеристиками, которые могут быть использованы в различных ситуациях.

Б.7.2 Тестирование организационного процесса

Тестирование на организационном уровне связано с определением и поддержанием политики и практики тестирования, которые применяют в рамках всей организации, а не для конкретных проектов.

Б.7.3 Тестирование проектов

Серия стандартов на системную и программную инженерию определяет процессы тестирования проекта на двух уровнях: для поддержки менеджмента тестирования и для поддержки динамического тестирования. Техники статического тестирования, такие как обзоры и статический анализ, описаны в [4]. В целом, выделяют три уровня процессов тестирования, как показано на рисунке Б.1.

Рисунок Б.1 — Три уровня процессов тестирования

Б.7.4 Процессы менеджмента тестирования

Менеджмент тестирования описывается процессами менеджмента тестирования, показанными на рисунке Б.2.

Рисунок Б.2 — Процессы менеджмента тестирования

Б.7.5 Процессы динамического тестирования

Процессы динамического тестирования описывают разработку тестовых примеров, настройку тестовой среды, выполнение тестовых примеров и отчет о возникших проблемах. Взаимодействие процессов динамического тестирования показано на рисунке Б.3.

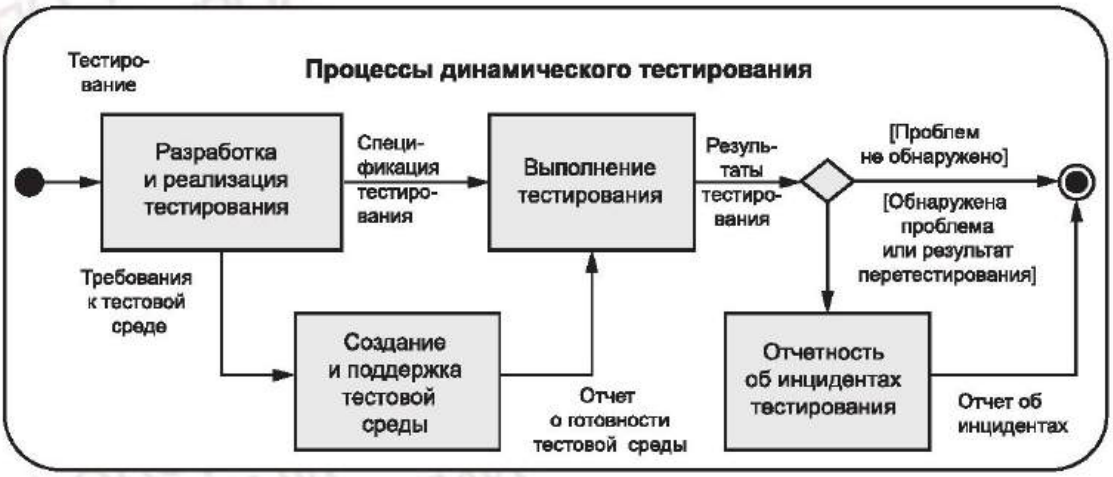

Рисунок Б.3 — Процессы динамического тестирования

Б.8 Тестовая документация (см. ГОСТ Р 56922)

Б.8.1 Общие положения

Тестовую документацию создают в результате выполнения процессов тестирования в соответствии с ГОСТ Р 56922. Обзор тестовой документации показан на рисунке Б.4.

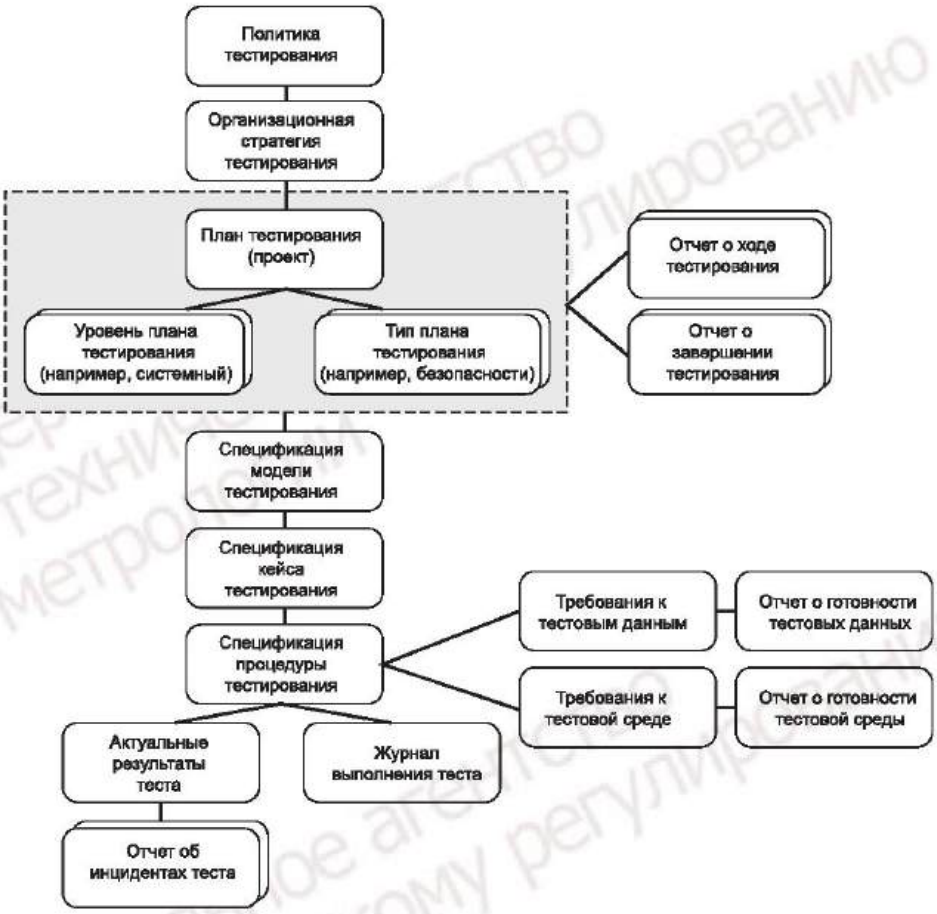

Рисунок Б.4 — Обзор тестовой документации

Б.8.2 Документация организационного уровня

Б.8.2.1 Общие положения

В процессе организационного тестирования разрабатывают и поддерживают спецификации организационного тестирования. Как правило, данные спецификации могут иметь форму:

а) политики тестирования;

б) документа о практике организационного тестирования.

Б.8.2.2 Политика тестирования

Политика тестирования выражает ожидания и подход организации к тестированию ПО в бизнес-терминах. Хотя она будет полезна всем, кто связан с тестированием, она ориентирована на руководителей и менеджеров высшего звена. Подход, описанный в политике тестирования (то, чего организация хочет достичь), определяет содержание документа по организационной практике тестирования, который описывает, как организация будет этого добиваться.

Б.8.2.3 Организационные практики тестирования

Организационный документ по практике тестирования согласуется с политикой тестирования и выражает требования и ограничения по проведению тестирования во всех проектах, выполняемых в организации (если они не являются слишком разнородными по характеру, в случае чего может быть создано несколько организационных документов по практике тестирования).

Б.8.3 Документация на уровне менеджмента тестирования

Б.8.3.1 Общие положения

Каждый из трех процессов менеджмента тестирования формирует одну основную часть тестовой документации. План тестирования формируют в результате разработки стратегии и планирования тестирования, отчеты о состоянии тестирования — для документирования хода тестирования в процессе мониторинга и контроля тестирования, отчет о завершении тестирования — для подведения итогов тестирования.

Б.8.3.2 План тестирования

План тестирования содержит подробное описание того, как должно проводиться тестирование для соответствующих процессов управления тестированием. Поскольку процессы управления тестированием могут быть реализованы для достижения различных целей (например, управление тестированием проектов, систем, производительности), соответствующие планы тестирования также обязательно имеют различную направленность в зависимости от целей проведения тестирования.

Однако типы информации, содержащейся в плане тестирования, как правило, остаются неизменными вне зависимости от формы планируемого тестирования.

План тестирования используют в процессе мониторинга и управления тестированием как основу для управления тестированием, а также в процессе динамического тестирования, поскольку в нем определено, как должно выполняться динамическое тестирование (например, необходимые методы тестирования, критерии завершения тестирования, кем и когда выполняется тестирование). План тестирования также используют для определения того, как должно выполняться статическое тестирование (например, обзор и статический анализ). План тестирования может быть обновлен в процессе мониторинга и управления тестированием, например, в связи с изменением требований (например, переносом даты завершения тестирования) или в случае, когда указанное тестирование не может быть выполнено тестировщиками.

Б.8.3.3 Отчет о ходе тестирования

Отчеты о ходе тестирования формируют в процессе мониторинга и управления тестированием для предоставления информации о текущем ходе тестирования в соответствии с планом тестирования. Эти отчеты используют менеджеры проекта для оценки прогресса или, если управление тестированием осуществляется на более низком уровне, их могут использовать для информирования менеджера по тестированию проекта о прогрессе на этом более низком уровне тестирования. Отчеты о состоянии тестирования обычно составляют на регулярной основе (как указано в плане тестирования), хотя к концу тестирования частота отчетов может увеличиться.

Б.8.3.4 Отчет о завершении тестирования

Отчет о завершении тестирования содержит краткое описание тестирования, подробную информацию об архивированном тестовом оборудовании и состоянии тестовой среды, а также выводы, полученные в ходе тестирования. Если отчет о завершении тестирования относится к более высокому уровню тестирования, то в него также должны быть включены сведения о тестировании, выполненном на более низких уровнях (например, если это отчет о завершении тестирования проекта, но при этом существовал отдельный план тестирования системы, то в него должны быть включены сведения о тестировании системы, которые должны содержаться в отчете о завершении тестирования системы). Для тестирования, связанного с каждым соответствующим планом тестирования, обычно формируют один отчет о завершении тестирования.

Б.8.4 Документация на уровне динамического тестирования

Б.8.4.1 Общие положения

Объем тестовой документации, создаваемой в процессе динамического тестирования, может сильно различаться, поскольку требования к документированию разработки тестовых примеров, тестовых сред и результатов тестирования могут быть совершенно разными для различных форм тестирования. Требования к тестовой документации должны быть указаны в плане тестирования, и, в зависимости от уровня автоматизации тестирования, часть документации может храниться в инструменте управления тестированием.

Б.8.4.2 Спецификация тестирования

Спецификации тестов должны содержать достаточно подробную информацию, позволяющую определить, как тестовые случаи в тестовых процедурах были получены из тестовой базы и что они достигают требуемого уровня тестового покрытия. Спецификации тестов могут иметь форму тестовой процедуры (поддерживающей ручное выполнение тестов) или автоматизированного тестового сценария (поддерживающего автоматизированное выполнение тестов). Для подтверждения покрытия и выбора тестов для повторного выполнения в случае внесения изменений необходима прослеживаемость от тестовых примеров к тестовой базе.

Б.8.4.3 Требования к тестовой среде и тестовым данным

Если они не включены в план испытаний, то указывают любые требования к тестовой среде и тестовым данным.

Б.8.4.4 Отчет по тестовой среде и готовности тестовых данных

Состояние тестовой среды (например, доступность) и тестовые данные доводят до сведения соответствующих заинтересованных сторон.

Б.8.4.5 Журнал выполнения теста

Выполнение тестов, включая фактические результаты, фиксируют в журналах выполнения теста.

Б.8.4.6 Отчет об инцидентах тестирования

При необходимости составляют отчеты о происшествиях (отчеты о дефектах), позволяющие, если это будет экономически целесообразным, устранить дефекты.

Библиография

[1]	ISO/IEC/IEEE 29119-4:2021	Системная и программная инженерия. Тестирование программного обеспечения. Часть 4. Методы тестирования (Software and systems engineering — Software testing — Part 4: Test techniques)
[2]	ISO/IEC/IEEE 29119-5:2016	Системная и программная инженерия. Тестирование программного обеспечения. Часть 5. Тестирование на основе ключевых слов (Soft-ware and systems engineering — Software testing — Part 5: Keyword-Driven Testing)
[3]	ИСО/МЭК 23894:2023	Информационные технологии. Искусственный интеллект. Руководство по менеджменту риска (Information technology — Artificial intelligence — Risk management)
[4]	ИСО/МЭК 20246:2017	Системная и программная инженерия. Обзор рабочих продуктов (Software and systems engineering — Work product reviews)
[5]	ISO/IEC TS 12791	Информационные технологии. Искусственный интеллект. Обработка нежелательных смещений в задачах машинного обучения классификации и регрессии (Information technology — Artificial intelligence — Treatment of unwanted bias in classification and regression machine learning tasks)
[6]	ИСО/МЭК 5259-4:2024	Искусственный интеллект. Качество данных для аналитики и машинного обучения (МО). Часть 4. Система процессов обеспечения качества данных (Artificial intelligence — Data quality for analytics and machine learning (ML) — Part 4: Data quality process framework)
[7]	ISO/IEC TR 24368:2022	Информационные технологии. Искусственный интеллект. Обзор этических и социальных проблем (Information technology — Artificial intelligence — Overview of ethical and societal concerns)
[8]	ИСО/МЭК 23053:2022	Экосистема разработки систем искусственного интеллекта (ИИ) с использованием машинного обучения (МО) [Framework for Artificial Intelligence (AI) Systems using Machine Learning]
[9]	ИСО/МЭК 25059:2023	Программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модель качества для систем на основе искусственного интеллекта [Software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality model for AI systems]

УДК 004.01:006.354 ОКС 35.020

Ключевые слова: искусственный интеллект, руководство по применению искусственного интеллекта

Технический редактор *В.Н. Прусакова*
Корректор *Е.Д. Дульнева*
Компьютерная верстка *И.Ю. Литовкиной*

Сдано в набор 30.10.2024. Подписано в печать 13.11.2024. Формат 60×84⅛. Гарнитура Ариал.
Усл. печ. л. 5,12. Уч-изд. л. 4,40.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

Создано в единичном исполнении в ФГБУ «Институт стандартизации»
для комплектования Федерального информационного фонда стандартов,
117418 Москва, Нахимовский пр-т, д. 31, к. 2.
www.gostinfo.ru info@gostinfo.ru