Гибридная проверка сценариев использования: rule-based анализ, семантический NLP и онтологическое заземление LLM

А.Р. Карпова, Л.С. Ераносян

***Аннотация* — Проверка сценариев использования (Use Cases) играет важную роль в разработке ПО. Одна из главных задач системных аналитиков заключается в том, чтобы предотвратить ошибки в сценариях использования до того, как они приведут к огромному увеличению стоимости исправлений. Однако ручная проверка часто страдает от неполноты, двусмысленности, логических противоречий, субъективности оценок, непоследовательности процесса и высоких временных затрат. В данной работе рассматривается гибридный подход к автоматизированной проверке сценариев использования, который сочетает rule-based анализ и семантический анализ с использованием модели Sentence-BERT, а также онтологическое заземление LLM.**

Разработанный программный модуль поддерживает три способа ввода: структурированная форма, JSON, загрузка файлов и экспорт структурированного отчёта. Верификация на 28 тест-кейсах подтвердила 100% корректность работы модуля, а валидация показала 100% обнаружение. Внедрение модуля позволяет сократить время проверки одного сценария с 50 минут до 30 секунд.

***Ключевые слова* — Use Case, LLM, NLP, онтология, программный модуль, rule-based анализ, семантический анализ.**

I. Введение

Сценарии использования (Use Case) позволяют нам описывать взаимодействие между пользователем (актором) и системой для достижения конкретной цели. Однако на практике, сценарии, написанные на естественном языке, содержат дефекты, которые снижают качество итогового продукта [3]. Стоимость исправления дефектов, обнаруженных на этапе реализации, как показывают исследования Боэма и Базили [1], в 10 раз выше, чем на этапе требований. Это значит, что ошибки, которые допускают при написании сценариев, приводят к превышению бюджетов и срыву сроков проекта.

Подходы к автоматизированной проверке сценариев использования, которые существуют на сегодняшний день имеют существенные ограничения. Чистый rule-based [6] не способен выявлять семантические дефекты. Методы машинного обучения требуют больших размеченных датасетов, которые трудно найти в открытом доступе. Большие языковые модели (LLM) [7] сами по себе не обеспечивают интерпретируемости и воспроизводимости результатов, что неприемлемо для проверки сценариев использования.

Таким образом, проблема исследования заключается в отсутствии доступного инструмента для автоматизированной проверки сценариев использования, который обеспечивал бы структурный, лексический и семантический анализ.

II. Необходимая теоретическая информация

A. *Сценарии использования и их свойства*

Сценарий использования — это способ описания функциональных требований, который показывает взаимодействие между пользователем и системой. Важная мысль: use case не обязан быть идеальным, просто он должен быть «достаточно хорош» для своей задачи [3]. Если сценарий использования написан хорошо, то он имеет 3-9 шагов в основном потоке. Если шагов больше, то скорее всего, вы описали слишком низкоуровневые действия. Качественный сценарий использования должен соответствовать основным свойствам. Из них можно выделить: полноту, непротиворечивость, ясность, атомарность, проверяемость, терминологическую согласованность.

B. *Критерии качества и стандарты*

Формальные критерии качества сценариев использования определены в ряде стандартов и руководств. В нашей работе используются следующие источники:

1. INCOSE Guide to Writing Requirements(GFWR).
2. ISO/IEC/IEEE 29148:2018.
3. Pohl K. Requirements Engineering: Fundamentals, Principles, and Techniques.
4. Cockburn A. Writing Effective Use Cases.

На основе этих источников была разработана система из 25 формальных правил проверки: полнота (структурная полнота сценария), качество

Статья получена 17 мая 2026.

Ераносян Лилия Сергеевна, старший преподаватель кафедры системной инженерии - Российский технологический университет (РТУ МИРЭА), Институт искусственного интеллекта, кафедра системной инженерии, (e-mail: eranosyan@mirea.ru).

Карпова Алина Романовна, студент - Российский технологический университет (РТУ МИРЭА), Институт искусственного интеллекта, кафедра системной инженерии, направление подготовки 27.03.03 «Системный анализ и управление», (e-mail: alin4car@yandex.ru).

формулировок (ясность и атомарность), логическая согласованность (непротиворечивость), проверяемость и согласованность терминов (терминологическая согласованность).

C. Методы анализа естественного языка

Для автоматизированной проверки сценариев используются следующие методы обработки естественного языка (NLP) [8]:

1. Rule based анализ — основан на формальных правилах и словарях. Применяется для проверки активного залога, модальных глаголов, атомарности, наличия явного актора. Преимущества этого анализа: детерминированность и полная интерпретируемость результатов.
2. Семантический анализ — для выявления дублирующихся шагов и проверки связности с целью сценария используется модель Sentence BERT [7]. Эффективность данного подхода подтверждена в работе [11], где на основе эмбеддингов требований строится граф семантической близости. Модель преобразует текст в векторное представление, после чего семантическое сходство вычисляется как косинусное расстояние:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \times \|B\|}, \quad (1)$$

где:

- cos(θ) — косинус угла между двумя векторами.
- A·B — скалярное произведение векторов.
- ‖A‖ — длина вектора A.
- ‖B‖ — длина вектора B.
- ‖A‖ × ‖B‖ — произведение длин векторов.

Пороги срабатывания: ≥0.85 — семантическое дублирование шагов; <0.40 — шаг не связан с целью.

D. Большие языковые модели и их проблема заземления

В работе используется онтологический подход к формализации предметной области [9]. Разработанная онтология сценариев использования служит основой для заземления больших языковых моделей (LLM) [7], что позволяет преодолеть ограничения LLM, сохраняя их генеративные способности для вспомогательных задач.

III. Процесс разработки сценариев: проблема достижения свойств

Качественные сценарии использования должны обладать рядом свойств, которые на практике трудно обеспечить вручную. Ниже кратко описаны основные из них с указанием типичных нарушений (см. таблицу 1).

Таблица 1. Свойства качественных сценариев использования

Свойство	Описание	Типичные нарушения
Полнота	Наличие всех обязательных компонентов (актор, цель, предусловия, постусловия, шаги)	Отсутствие предусловий или постусловий
Непротиворечивость	Отсутствие логических конфликтов между шагами	Один актор назван по-разному в разных шагах
Проверяемость	Возможность объективного тестирования каждого шага	Формулировка «система работает корректно» (нет измеримого критерия)
Ясность	Однозначность понимания текста	Модальные глаголы («должен», «необходимо»)
Атомарность	Один шаг описывает одно действие	«Система проверяет и сохраняет данные»
Согласованность терминов	Единая терминология во всех сценариях	«Клиент» и «пользователь» об одном и том же

Проблема заключается в том, что ручная проверка субъективна, трудоёмка и не масштабируется. Системный аналитик физически не может одновременно удерживать в голове все критерии качества и применять их к десяткам сценариев, не допуская пропусков. Автоматизация становится необходимой, но традиционные подходы, к примеру rule-based анализ, не справляются с семантическими аспектами.

IV. Гипотеза: улучшение качества с помощью LLM через онтологию

Большие языковые модели демонстрируют хорошие способности в понимании и генерации естественного языка. Однако их стохастическая природа приводит к недетерминированности результатов и галлюцинациям, что неприемлемо для проверки сценариев использования.

Данная работа выдвигает следующую гипотезу: если формализовать предметную область сценариев использования в виде онтологии (классы, свойства, аксиомы) и использовать данную онтологию для заземления LLM, то можно повысить качество автоматической проверки сценариев за счёт:

- обеспечения трассируемости от требований к правилам проверки — каждое нарушение ссылается на аксиому онтологии;
- детерминированной интерпретации результатов через аксиомы онтологии — результат проверки становится предсказуемым и объяснимым.

V. Онтология сценариев использования

Онтология является формальной основой разработанного подхода. Она определяет понятия предметной области, их свойства и ограничения.

На рисунке 2 представлена онтология сценариев использования (разработана авторами). Цветовая дифференциация классов позволяет визуально различать сущности, отношения и аксиомы.

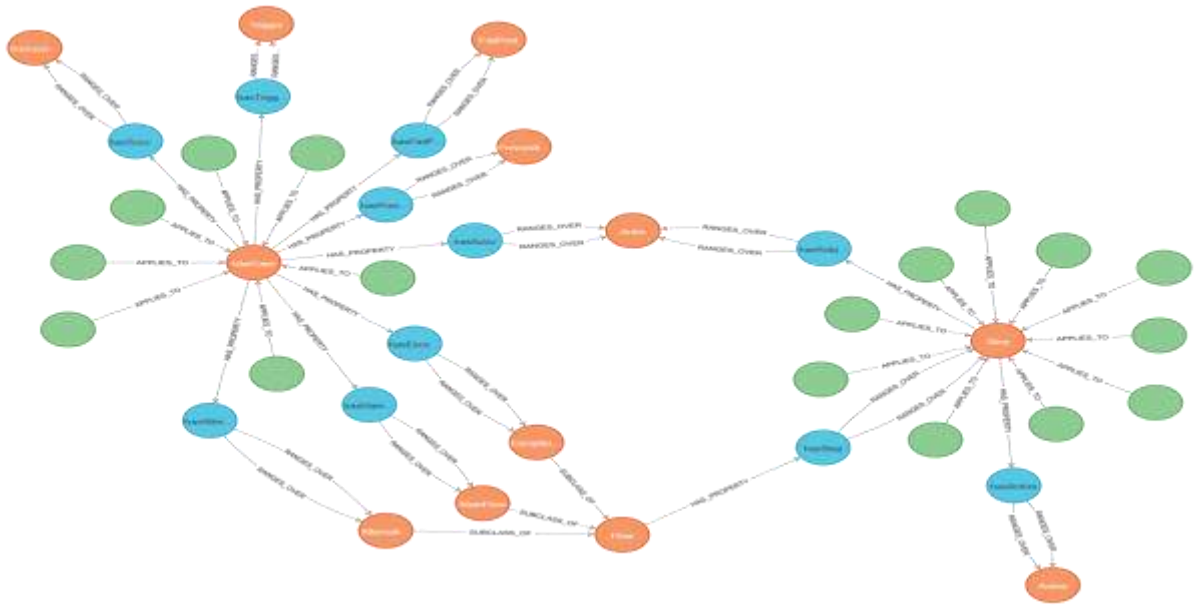

Рисунок 1 — Онтология сценариев использования

Онтология включает шесть основных классов (см. таблицу 2).

Таблица 2. Классы онтологии сценариев использования

Класс	**Описание**
Use Case	Сценарий использования
Actor	Участник взаимодействия
Step	Атомарный шаг сценария
Condition	Условие (предусловие/постусловие)
Extension	Альтернативный или исключительный поток
Glossary	Глоссарий предметной области

Аксиомы онтологии задают ограничения, нарушение которых означает дефект сценария. Именно аксиомы являются формальной основой для правил проверки модуля. В таблице 3 приведены ключевые аксиомы.

Таблица 3. Аксиомы онтологии

Аксиома	Формулировка	Правило	Тип
A1	UseCase.goal не пусто	CMP-01	Структурная полнота
A2	Ровно один первиный актор	CMP-02	Структурная полнота
A3	Шаг содержит явный актор	CNS-01	Логическая согласованность
A5	Глагол в активном залоге	CLR-02	Качество формулировок
A9	Семантическое сходство шагов < 0.85	SEM-S1	Логическая согласованность
A10	Семантическое сходство шага с goal ≥ 0.40	SEM-S2	Связность

A. *Машиночитаемое представление онтологии*

Онтология хранится в формате JSON со следующей структурой:

Рисунок 2 — Фрагмент онтологии в формате JSON

VI. Реализация гибридного модуля

5.1. Система формальных правил проверки

Модуль реализует 25 формальных правил, разделённых на 5 групп:

1. Структурная полнота (R.01–R.08): наличие названия (R.01), цели (R.02), первичного актора (R.03), предусловий (R.04), постусловий успеха (R.05) и неудачи (R.06), основного потока (R.07), альтернативных и исключительных потоков (R.08).

2. Качество формулировок (R.09–R.14): активный залог (R.09), атомарность шага (R.10), явное указание актора (R.11), отсутствие размытых формулировок

(R.12), отсутствие технических деталей (R.13), нумерация шагов (R.14).

3. Логическая согласованность (R.15–R.18): согласованность имён акторов (R.15), согласованность имён объектов (R.16), корректность ссылок на потоки (R.17), непротиворечивость предусловий и постусловий (R.18)

4. Верифицируемость (R.19–R.20): наличие глагола действия в шаге (R.19), конкретность результата в постусловии (R.20).

5. Правила глоссария (R.21–R.23) и конфигурации (R.24–R.25).

5.2. Онтологическое заземление LLM в процессе проверки

В отличие от подходов, использующих LLM как «чёрный ящик» для прямой классификации, в данной работе LLM интегрирована в конвейер проверки как генератор структурированных утверждений. Этот принцип применяется в нейросимволических системах и позволяет сочетать генеративные способности LLM с формальными гарантиями символьной валидации [9].

Процесс проверки с участием LLM включает следующие шаги:

1) Модуль получает шаг сценария и онтологию (классы, свойства, аксиомы).
2) Формируется запрос к LLM с инструкцией: «Извлеки из шага структурированные утверждения в формате JSON, используя только классы и свойства из онтологии. Не добавляй фактов, которых нет в тексте».
3) LLM возвращает JSON с утверждениями (например: {"subject": "Клиент", "predicate": "выбирает", "object": "марку бетона"}).
4) Символьный валидатор проверяет утверждения по аксиомам онтологии (A1–A10).
5) Если утверждение нарушает аксиому — фиксируется нарушение. Если LLM сгенерировала сущность, отсутствующую в онтологии — утверждение отклоняется.

Результат применения подхода:

- LLM выполняет «интерпретационную работу» — переводит естественный язык в формальные утверждения.
- Символьный валидатор обеспечивает детерминизм и интерпретируемость результатов.
- Галлюцинации LLM отсекаются на этапе символьной валидации.

Таким образом, LLM используется не как самостоятельный классификатор, а как вспомогательный компонент, дополняющий rule-based и семантический модули.

Разработан программный модуль, который реализует гибридный подход.

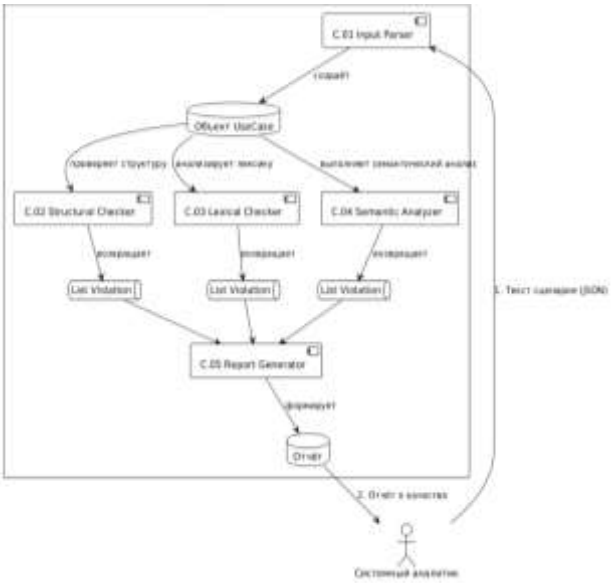

Рисунок 3 — Диаграмма компонентов модуля

Модуль реализован на языке Python с использованием веб-фреймворка FastAPI (бэкенд) и HTML/CSS/JavaScript (фронтенд). Для семантического анализа используется библиотека Sentence-BERT, для лексического анализа — spaCy и встроенные словари. Модуль поддерживает три способа ввода: структурированная форма, JSON-редактор и загрузка внешних файлов (glossary.json). Для семантического анализа используется модель paraphrase-multilingual-MiniLM-L12-v2 (версия Sentence-BERT 2.2.0). Выбор обусловлен поддержкой русского языка и оптимальным соотношением точности и производительности. Токенизация выполняется стандартным токенизатором модели (bert-base-multilingual-cased) без дополнительной настройки. Результат проверки можно экспортировать в формате JSON.

Фрагмент реализации проверки семантических дублей:

```
from sentence_transformers import SentenceTransformer, util

model = SentenceTransformer('paraphrase-multilingual-MiniLM-L12-v2')

def check_duplicate_steps(step1: str, step2: str, threshold: float = 0.85) -> dict:
    emb1 = model.encode(step1, convert_to_tensor=True)
    emb2 = model.encode(step2, convert_to_tensor=True)
    similarity = util.pytorch_cos_sim(emb1, emb2).item()
    return {
        "step1": step1,
        "step2": step2,
        "similarity": similarity,
        "is_duplicate": similarity >= threshold
    }
```

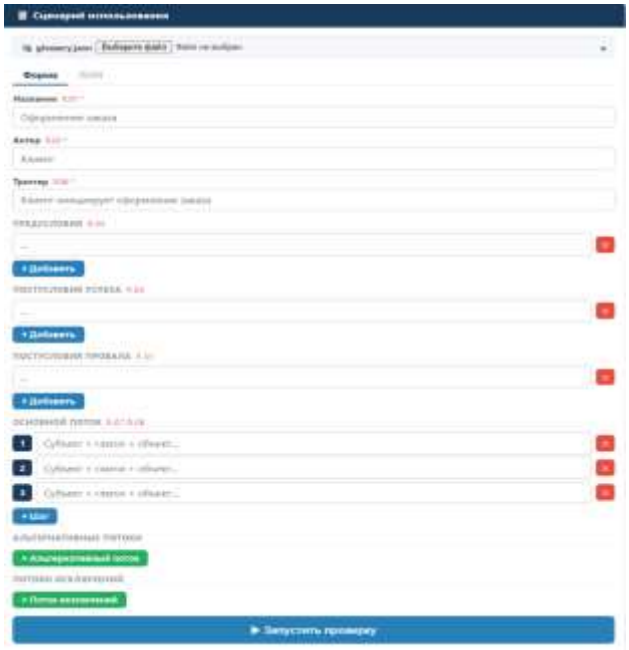

Рисунок 4 — Интерфейс модуля

VII. Результаты и плюсы внедрения

A. *Верификация и валидация*

Тест-кейсы формировались методом покрытия: для каждого из 25 правил создавалось по два теста — позитивный (без нарушения) и негативный (с нарушением). Дополнительно были добавлены 6 тестов для семантического анализа (3 на дублирование шагов, 3 на связность с целью). Отбор тестов осуществлялся не случайным образом, а целенаправленно — для проверки каждого формального правила. Такой подход не позволяет экстраполировать результаты на все возможные сценарии, но достаточен для верификации корректности реализации правил.

В ходе тестирования зафиксировано отсутствие ложноположительных и ложноотрицательных срабатываний. Однако это объясняется ограниченным набором тестов; при расширении выборки на реальные проекты возможны отклонения.

Валидация на одном сценарии носит пилотный характер и не является статистически значимой. Для получения достоверных выводов требуется тестирование на репрезентативной выборке из 50–100 сценариев различных предметных областей, что запланировано, как дальнейшее исследование.

B. *Количественные метрики*

В целях получения объективных данных о текущем состоянии процесса проверки сценариев использования было проведено предпроектное обследование методом анкетирования. Затем были определены базовые значения метрик. В таблице 4 приведено сравнение «до» и «после» внедрения модуля.

Таблица 4. Метрики

Метрика	**До внедрения**	**После внедрения**
Время проверки одного сценария(мин)	50	0.5
Количество итераций согласования(кол-во)	5	2
Уточняющие вопросы(кол-во)	10	3
Процент переделок на этапе реализации (%)	30	10

C. *Качественные результаты и плюсы от LLM*

Внедрение разработанного модуля обеспечивает:

1. Объективную оценку — вместо субъективного «кажется» используются формальные правила и аксиомы онтологии.
2. Полную трассируемость — каждое обнаруженное нарушение ссылается на аксиому онтологии, правило проверки и тест-кейс.
3. Детерминированность — в отличие от «чистых» LLM, результат не зависит от случайных факторов.
4. Поддержку русского языка — через многоязычную модель Sentence-BERT и настройку лексических словарей.

Использование LLM, заземлённой на разработанную онтологию, позволило автоматически генерировать начальную онтологию для новых предметных областей, сокращая время настройки модуля с 2–3 дней до 10–15 минут. При этом детерминированность проверки сохраняется за счёт rule-based модуля.

D. *Сравнение с существующими подходами*

Для оценки эффективности разработанного модуля рассмотрим четыре альтернативных подхода к проверке сценариев использования.

1) Чистый rule-based подход наиболее распространённый метод в существующих инструментах. Его преимущества: высокая скорость работы, полная детерминированность и интерпретируемость результатов. Каждое найденное нарушение напрямую трассируется к конкретному правилу. Однако данный подход принципиально ограничен: он не способен выявлять семантические дефекты — дублирующиеся шаги с разными формулировками, шаги, не связанные с целью сценария, а также смысловые противоречия между предусловиями и постусловиями.

2) Методы машинного обучения обеспечивают высокое качество семантического анализа и способны выявлять сложные смысловые паттерны. Однако они требуют больших размеченных датасетов для обучения, которые в области сценариев использования практически отсутствуют в открытом доступе. Предлагаемый подход использует

предобученную модель Sentence-BERT без необходимости разметки данных, и сохраняет полную интерпретируемость через rule-based модуль.

3) Большие языковые модели без онтологического заземления демонстрируют впечатляющие способности в понимании естественного языка и могут быть использованы для анализа сценариев. Однако их стохастическая природа приводит к недетерминированности результатов: одинаковый запрос может давать разные ответы при повторном обращении. Кроме того, LLM не предоставляют объяснений своих решений, что критически важно для верификации требований.

4) Ручная проверка традиционно считается наиболее надёжным методом, поскольку опытный аналитик способен учесть контекст и нюансы предметной области. Однако ручная проверка не масштабируется: по результатам проведённого опроса, время проверки одного сценария составляет в среднем 50 минут, а количество итераций согласования достигает 4–5. При этом субъективность оценки оценивается в 3,8 из 5 баллов, а 55% аналитиков не используют формализованных критериев качества. Разработанный модуль автоматизирует проверку, сокращая время до 30 секунд и обеспечивая единообразие оценки.

Таким образом, предлагаемый гибридный подход сочетает преимущества всех рассмотренных альтернатив: детерминизм и интерпретируемость rule-base, семантический анализ на уровне, сопоставимом с ML-методами, гибкость LLM для вспомогательных задач, и при этом превосходит ручную проверку по скорости и объективности. Единственным ограничением является необходимость начальной настройки онтологии для новой предметной области, которая, однако, может быть автоматизирована с использованием LLM.

VIII. Заключение

В рамках работы разработана онтология сценариев использования, которая включает 6 классов, 10 свойств и 12 аксиом и служит формальной основой для заземления LLM и обеспечивает трассируемость от требований к правилам проверки. Модуль поддерживает три способа ввода и экспорт отчёта. Верификация на 28 тест-кейсах и валидация на реальном сценарии подтвердили 100% корректность и полное обнаружение нарушений. Внедрение модуля сократило время проверки сценария с 50 минут до 30 секунд, а число итераций согласования — с 5 до 2.

Перспективы развития включают создание плагинов для Jira и Confluence с целью бесшовной интеграции в существующие процессы разработки.

Библиография

[1] Boehm B., Basili V. Software Defect Reduction Top 10 List // IEEE Computer. 2001. Vol. 34, No. 1. P. 135–137.

[2] Pohl K. Requirements Engineering: Fundamentals, Principles, and Techniques. Springer, 2010. 813 p.

[3] Cockburn A. Writing Effective Use Cases. Addison-Wesley, 2001. 304 p.

[4] ISO/IEC/IEEE 29148:2018. Systems and software engineering — Life cycle processes — Requirements engineering.

[5] Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks // Proceedings of EMNLP-2019. 2019. P. 3982–3992.

[6] Sinha A., et al. Text2Test: Automated Inspection of Natural Language Use Cases // International Conference on Software Testing, Verification and Validation (ICST). 2009. P. 1–10.

[7] Evaluating LLMs for automated requirement and test case generation in railway signaling systems [Электронный ресурс]. URL: https://repository.utm.md/handle/5014/35278

[8] Обработка естественного языка (NLP — Natural Language Processing) [Электронный ресурс] // Витебский государственный технологический университет. URL: https://it.vstu.by/courses/information_systems/Development_and_optimization_of_intellectual_information_systems/practice/natural_language_processing/

[9] Хорошевский В.Ф. Проектирование систем программного обеспечения под управлением онтологий: модели, методы, реализации // Онтология проектирования. 2019. Т. 9, № 4 (34). С. 278–298.

[10] INCOSE. Guide to Writing Requirements. International Council on Systems Engineering, 2022. 115 p.

[11] Automatic use case identification in large requirements specifications [Электронный ресурс] // Friedrich-Alexander-Universität Erlangen-Nürnberg. 2025. URL: https://www.mfk.tf.fau.eu/2025/07/22/automatic-use-case-identification-in-large-requirements-specifications/

Hybrid validation of use cases: rule-based analysis, semantic NLP, and ontological grounding of LLM

Alina Karpova, Liliya Eranosyan

***Abstract* — Use case validation plays an important role in software development. One of the main tasks of system analysts is to prevent errors in use cases before they lead to a dramatic increase in the cost of fixes. However, manual validation often suffers from incompleteness, ambiguity, logical contradictions, subjectivity of assessments, inconsistency of the process, and high time costs. According to a survey of system analysts (N=6), the average time for manual validation of a single use case is 50 minutes, and the proportion of defects caused by poor requirements reaches 38%. This paper presents a hybrid approach to automated use case validation that combines rule-based analysis (25 formal rules based on INCOSE and ISO/IEC 29148), semantic analysis using the Sentence-BERT model (similarity thresholds: 0.85 for duplicate detection, 0.40 for goal relevance), and ontological grounding of LLMs. A colored use case ontology (6 classes, 10 properties, 12 axioms) serves as a formal grounding mechanism for LLMs, ensuring traceability and determinism. The developed software module supports three input methods (structured form, JSON, file upload) and exports structured reports. Verification on 28 test cases confirmed 100% correctness, while validation on a real-world scenario demonstrated 100% defect detection. The implementation reduces validation time per use case from 50 minutes to 30 seconds and decreases requirements-related defects from 38% to 10%.**

***Keywords* — Use case, LLM, NLP, ontology, software module, rule-based analysis, semantic analysis.**

REFERENCES

[1] Boehm B., Basili V. Software Defect Reduction Top 10 List // IEEE Computer. 2001. Vol. 34, No. 1. P. 135–137.

[2] Pohl K. Requirements Engineering: Fundamentals, Principles, and Techniques. Springer, 2010. 813 p.

[3] Cockburn A. Writing Effective Use Cases. Addison-Wesley, 2001. 304 p.

[4] ISO/IEC/IEEE 29148:2018. Systems and software engineering — Life cycle processes — Requirements engineering.

[5] Reimers N., Gurevych I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks // Proceedings of EMNLP-2019. 2019. P. 3982–3992.

[6] Sinha A., et al. Text2Test: Automated Inspection of Natural Language Use Cases // International Conference on Software Testing, Verification and Validation (ICST). 2009. P. 1–10.

[7] Evaluating LLMs for automated requirement and test case generation in railway signaling systems [Electronic resource]. URL: https://repository.utm.md/handle/5014/35278

[8] Natural Language Processing (NLP) [Electronic resource] // Vitebsk State Technological University. URL: https://it.vstu.by/courses/information_systems/Development_and_optimization_of_intellectual_information_systems/practice/natural_language_processing/

[9] Khoroshevsky V.F. Designing software systems under ontology management: models, methods, implementations // Design Ontology. 2019. Vol. 9, No. 4 (34). P. 278–298.

[10] INCOSE. Guide to Writing Requirements. International Council on Systems Engineering, 2022. 115 p.

[11] Automatic use case identification in large requirements specifications [Electronic resource] // Friedrich-Alexander-Universität Erlangen-Nürnberg. 2025. URL: https://www.mfk.tf.fau.eu/2025/07/22/automatic-use-case-identification-in-large-requirements-specifications/