УДК 004.415.2: 519.711.3

Мельников Егор Сергеевич, студент,
Московский университет имени С. Ю. Витте

РЕАЛИЗАЦИЯ МЕХАНИЗМОВ ТРАССИРУЕМОСТИ ЗАДАЧ К ВЕРСИЯМ ТРЕБОВАНИЙ В СИСТЕМЕ УПРАВЛЕНИЯ РАЗРАБОТКОЙ ПО

Аннотация. В статье описан механизм, который сохраняет связь задачи с определённой версией требования и регистрирует решение по каждой последующей публикации. Модель предусматривает раздельное хранение версии, по которой выполняется задача, и версии, влияние которой уже рассмотрено.

Ключевые слова: Управление требованиями, трассируемость, задача разработки, ссылочная целостность, управление изменениями.

Задачу разработки ставят по определённому содержанию требования. Пока задача выполняется, это содержание может измениться. Если в задаче сохранён только идентификатор требования, при переходе по ссылке исполнитель увидит текущую редакцию. Чтобы установить, какие условия были приняты в работу, ему придётся найти нужную запись в истории и сопоставить её с постановкой задачи.

Процессы разработки и сопровождения требований рассматриваются в ISO/IEC/IEEE 29148:2018 [1]. В обзоре Н. К. Горелица, Д. С. Кильдишева и А. В. Хорошилова трассируемость и управление изменениями входят в перечень задач инструментальной поддержки ответственных систем [2]. В работе Д. С. Кильдишева и А. В. Хорошилова метамодель Requality задаётся базовым деревом с типизированными атрибутами и декларативными моделями, сводимыми к базовой [3]. Официальная документация Requality описывает просмотр версий узлов, сравнение каталогов и работу со связями [5].

Далее предлагается реляционная модель для различения этих состояний и транзакционный алгоритм их изменения. Прототип разработан на Python 3.12.14 с модулем sqlite3 и SQLite 3.53.1. База данных размещалась в памяти. Все операции выполнялись последовательно через одно подключение. Набор данных создан для проверки ограничений и не содержит сведений о работе конкретной организации. Для каждого сценария задано ожидаемое состояние базы либо ожидаемый отказ.

Полученные значения сопоставляются с этими условиями программными проверками. При имитации сбоя дополнительно сравниваются полные SQL-дампы до и после операции. Для подключения включён контроль внешних ключей командой PRAGMA foreign_keys=ON. Изменяющие операции начинаются с BEGIN IMMEDIATE и завершаются COMMIT либо ROLLBACK. Согласно документации SQLite, BEGIN IMMEDIATE сразу начинает пишущую транзакцию; одновременно в базе может выполняться только одна такая транзакция. На данном стенде проверяется логика последовательных операций. Поведение при одновременных запросах требует самостоятельного испытания.

Постоянный идентификатор требования и номер его версии хранятся раздельно. Пара (req, ver) определяет одну опубликованную редакцию. Её содержание нельзя исправить на месте: для правки создаётся новая запись. Поле head в карточке требования указывает на последнюю опубликованную версию. Публикация в прототипе делает версию доступной для пересмотра связей; согласование черновиков находится за пределами реализации.

Схема включает пять таблиц (табл. 1).

Таблица 1

Основные сущности прототипа

Таблица	Поля	Назначение и ограничения
requirement	id, head	Постоянный идентификатор и последняя опубликованная версия; составная ссылка на revision
revision	req, ver, body	Содержание версии; первичный ключ (req, ver); изменение и удаление запрещены триггерами
task	id, state	Задача с состоянием open или done в сервисных операциях
link	task, req, ver, reviewed, token	Одна связь на пару (task, req); версия реализации, рассмотренная версия и счётчик изменений
audit	id, task, req, oldver, newver, head, actor, reason	Последовательность решений: прежняя и выбранная версии, рассмотренная публикация, исполнитель и причина

В таблице link поле ver хранит версию, по которой выполняется задача, а reviewed – версию, влияние которой уже рассмотрено. Например, сочетание ver = 1 и reviewed = 2 означает, что после рассмотрения второй версии для задачи сохранена первая. Причина такого решения записывается в audit. Если хранить только ver, это состояние невозможно отличить от случая, когда вторую версию ещё никто не рассматривал.

Условия согласованности распределены между базой данных и сервисными операциями. Составные внешние ключи (req, ver) и (req, reviewed) ссылаются на revision (req, ver). Они проверяют существование обеих версий именно для указанного требования. Первичный ключ link (task, req) запрещает повторную связь той же пары. Такой способ задания ссылочной целостности соответствует реляционным ограничениям, описанным в документации PostgreSQL [9].

Неизменяемость опубликованного содержания обеспечивают триггеры BEFORE UPDATE и BEFORE DELETE таблицы revision. При попытке изменения они вызывают RAISE (ABORT,...). Внешний ключ requirement (id, head) проверяется при фиксации транзакции: это позволяет совместно создать карточку требования и его первую версию. Изменение связи и запись в журнал выполняются в одной транзакции. Предполагается, что прикладной пользователь обращается к этим операциям через сервис и не редактирует таблицы напрямую. В прототипе исполнитель обозначен тестовой строкой analyst, а порядок записей задаётся целочисленным идентификатором. Псевдокод операции приведён в листинге 1.

Листинг 1

Транзакционный пересмотр связи

```
BEGIN IMMEDIATE
require reason is not empty
require task.state = open
read link and requirement
require expected_head = requirement.head
require expected_token = link.token
old_version = link.ver
```

```
if decision = migrate:
link.ver = requirement.head
link.reviewed = requirement.head
link.token = link.token + 1
append audit (old_version, link.ver, head, actor, reason)
COMMIT
on any error: ROLLBACK
```

Основной результат модели – различение двух причин, по которым задача остаётся связанной с прежней версией. При reviewed ≠ head изменение ещё не рассмотрено. При reviewed = head и ver ≠ head прежняя версия сохранена решением, причину которого можно найти в журнале, что позволяет строить очередь пересмотра без автоматической замены основания задачи. Такое решение уточняет одну из операций управления изменениями, рассматриваемых в работе. В существующих инструментах доступны более широкие средства работы с каталогами, версиями и базовыми линиями. Представленный прототип ограничен связями задач и одной последовательностью публикаций. Для сопровождения нескольких выпусков потребуется задать контекст каждой связи: набор версий, утверждённый для данного выпуска. Последняя публикация требования не обязательно входит в этот набор. Отдельной проверки потребует многопользовательский режим. В PostgreSQL для согласования операций можно использовать блокировки строк SELECT FOR UPDATE, сохраняемые до конца транзакции. Однако порядок блокировок при публикации, пересмотре и закрытии задачи должен быть общим: иначе возможны взаимоблокировки или изменение проверенных данных до записи решения. Перенос механизма на PostgreSQL в данной работе не выполнялся.

Таким образом, разработанный механизм сохраняет в задаче определённую версию требования и отдельно отмечает, до какой публикации рассмотрены изменения. Новая публикация не подменяет принятую в работу версию. Перенос ссылки или её сохранение выполняются явной операцией с записью причины в журнал. На прототипе проверены ограничения ссылочной целостности, неизменяемость опубликованного содержания и обработка устаревших запросов. Во всех 14 сценариях получены ожидаемые результаты. Для включения механизма в систему управления разработкой остаётся проверить конкурентное выполнение операций и связать решения с полномочиями пользователей и базовыми линиями выпусков.

Список литературы:

1. ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering. ISO, 2018. URL: https://www.iso.org/standard/72089.html (дата обращения: 09.10.2026).

2. Горелиц Н. К., Кильдишев Д. С., Хорошилов А. В. Управление требованиями к ответственным системам. Обзор решений // Труды ИСП РАН. 2019. Т. 31, вып. 1. С. 25–48. DOI: 10.15514/ISPRAS-2019-31 (1)-2.

3. Kildishev D. S., Khoroshilov A. V. Formalizing Metamodel of Requirements Management System // Proceedings of the Institute for System Programming of the RAS. 2018. Vol. 30, no. 5. P. 163–176. DOI: 10.15514/ISPRAS-2018-30 (5)-10.

5. ИСП РАН. Requality: инструмент управления требованиями. URL: https://www.ispras.ru/technologies/requality/ (дата обращения: 09.10.2026).