УДК 004.94:004.052:005.6

УПРАВЛЕНИЕ КАЧЕСТВОМ КОДА В ИМИТАЦИОННОМ МОДЕЛИРОВАНИИ: МЕТОД 80/20

Гаах Т.В.
старший преподаватель,
Калужский государственный университет им. К.Э. Циолковского,
Калуга, Россия
Каримов Д.Р.
магистрант,
Калужский государственный университет им. К.Э. Циолковского,
Калуга, Россия

Аннотация

Статья посвящена проблеме обеспечения качества кода в сложных программно-аппаратных комплексах имитационного моделирования на примере системы имитации фоново-целевой обстановки (СИФЦО). Обоснована высокая сложность таких систем, обусловленная многомерностью и стохастичностью моделируемой среды, распределённой архитектурой, жёсткими требованиями к производительности, а также накоплением технического долга и человеческим фактором. Показано, что традиционные подходы к тестированию «вслепую» неэффективны из-за ограниченности ресурсов и огромного объёма кодовой базы.

В качестве основного инструмента приоритизации усилий по улучшению качества предлагается диаграмма Парето (принцип 80/20). Рассмотрены преимущества метода: эмпирическая подтверждённость для программных проектов, наглядность, совместимость с существующими процессами (ABC-анализ, RCA) и универсальность применения (дефекты, время тестов, покрытие

кода). Приведена практическая методика построения диаграммы на основе данных баг-трекинга за отчётный квартал с категоризацией дефектов.
Результаты анализа показали, что более 77% всех зафиксированных ошибок сосредоточены в трёх категориях: ошибки физических моделей (38%), дефекты производительности (22%) и ошибки интеграции с аппаратурой потребителя (17%). На основе этого выделены зоны критического, высокого, среднего и низкого приоритета, а также сформулированы конкретные рекомендации по улучшению качества для каждой группы.
В заключении сделан вывод, что применение диаграммы Парето позволяет перейти от хаотичного тестирования к целенаправленному управлению качеством, сфокусировав 80% усилий на 20% наиболее проблемных компонентов, что обеспечивает максимальный эффект при ограниченных ресурсах.
Ключевые слова: имитационное моделирование, качество кода, диаграмма Парето, управление дефектами, тестирование.

CODE QUALITY MANAGEMENT IN SIMULATION MODELING: THE 80/20 METHOD

Gaakh T.V.
senior lecturer,
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia
Karimov D.R.
master's student,
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia

Abstract

This article examines the problem of code quality assurance in complex software and hardware simulation systems, using the background-target environment simulation system (BTES) as an example. The high complexity of such systems is explained, owing to the multidimensionality and stochasticity of the simulated environment, distributed architecture, stringent performance requirements, and the accumulation of technical debt and human error. It is shown that traditional blind testing approaches are ineffective due to limited resources and the enormous size of the codebase.

The Pareto diagram (80/20 principle) is proposed as the primary tool for prioritizing quality improvement efforts. The advantages of the method are discussed: empirical validation for software projects, clarity, compatibility with existing processes (ABC analysis, RCA), and versatility of application (defects, test time, code coverage). A practical method for constructing a diagram based on bug tracking data for the reporting quarter, categorizing defects, is presented.

The analysis revealed that over 77% of all recorded defects are concentrated in three categories: physical model errors (38%), performance defects (22%), and errors in integration with consumer hardware (17%). Based on this, critical, high, medium, and low priority zones are identified, and specific quality improvement recommendations are formulated for each group.

Finally, the authors conclude that using the Pareto diagram enables a transition from chaotic testing to targeted quality management, focusing 80% of efforts on the 20% most problematic components, ensuring maximum impact with limited resources.

Keywords: simulation modeling, code quality, Pareto chart, defect management, testing.

Система имитации фоново-целевой обстановки (СИФЦО) – это не просто набор скриптов, а сложный программно-аппаратный комплекс, предназначенный для создания реалистичной фоново-целевой обстановки в

лабораторных условиях. Её главная цель – заменить дорогостоящие и часто невозможные натурные испытания имитационными экспериментами.

Глубину этой сложности определяют несколько ключевых факторов.

1. Многомерность и стохастичность моделируемой среды.

Сложность СИФЦО в первую очередь обусловлена самой природой фоново-целевой обстановки. Это не статичная картинка, а динамическая, многопараметрическая и стохастическая среда. Модель должна учитывать:

- широкий спектр технических характеристик;
- внешние воздействия;
- случайный характер процессов.

Такое множество многокритериальных параметров и их стохастическая природа делают систему крайне сложной для прогнозирования и тестирования традиционными методами.

2. Архитектурная сложность и требование к производительности.

Современные СИФЦО часто строятся по распределенной многоагентной архитектуре. Это накладывает дополнительные требования:

- масштабируемость и открытость;
- асинхронность;
- высокая производительность.

Требования к производительности и сложная архитектура создают благодатную почву для трудноуловимых ошибок, связанных с гонками данных, взаимоблокировками и неэффективным использованием ресурсов.

3. Фактор "человеческого несовершенства" в коде.

И, наконец, нельзя сбрасывать со счетов человеческий фактор. В ходе разработки такой сложной системы:

- растет технический долг: код усложняется, появляются неоптимальные решения, принятые в спешке.

- нарушается целостность: изменения, вносимые в одну часть системы, могут непредсказуемо влиять на другие, особенно в условиях постоянной эволюции требований.

- накопление ошибок: чем сложнее система, тем выше вероятность появления новых дефектов при каждом изменении.

В условиях, когда ресурсы на тестирование ограничены, а система СИФЦО содержит тысячи модулей и миллионы строк кода, встаёт ключевой вопрос: на чём сосредоточить усилия в первую очередь? Перебирать все компоненты последовательно – значит растянуть улучшение качества на годы. Тратить время на равномерное форматирование всего кода – значит распылять бюджет впустую. Нужен инструмент, который позволил бы быстро, наглядно и обоснованно определить «болевые точки». Таким инструментом является диаграмма Парето (или принцип 80/20) по следующим причинам.

1. Эмпирическая подтверждённость для программных проектов.

Принцип Парето - не теоретическая абстракция. Многочисленные исследования в области инженерии программного обеспечения [1-3] подтверждают, что распределение дефектов по модулям почти всегда подчиняется закону «немногих важных»: около 80% всех ошибок и сбоев возникает в 20% файлов или классов; примерно 80% времени сопровождения уходит на исправление одних и тех же «горячих» участков [4].

Анализ и частота дефектов (багов), обнаруженных в течение этапа тестирования (квартал) стало прямым сигналом к применению Парето.

2. Простота и наглядность для всех участников.

Диаграмма Парето – это комбинация столбчатой гистограммы и кумулятивной кривой. Она не требует специальной математической подготовки – её поймёт и разработчик, и менеджер, и заказчик; визуально выделяет «красную зону» – первые столбцы, на которых стоит сосредоточиться; позволяет отслеживать динамику.

3. Фокус на приоритетах, а не на объёме.

В СИФЦО, как и в любой крупной имитационной системе, невозможно исправить все замечания статического анализатора или все «запахи» кода. Но можно выделить те немногие модули, которые: содержат наибольшее число критических уязвимостей; имеют высокую цикломатическую сложность (риск при изменениях); чаще всего становятся причиной сбоев в интеграционных тестах.

4. Совместимость с существующими процессами.

Данные из систем статистического анализа легко агрегируются в таблицу и преобразуются в Парето-график стандартными средствами Excel или Python.

Кроме того, Парето отлично сочетается с ABC-анализом и анализом первопричин (RCA). Сначала мы выделяем группу «А» (самые проблемные компоненты), а затем для каждого из них проводим глубокий RCA, чтобы понять, почему там так много ошибок (легаси-код, высокая связанность, недостаток тестов и т.п.).

5. Универсальность и масштабируемость.

Диаграмму Парето мы строим не только для дефектов, но и для: времени выполнения тестов (какие тесты самые долгие и частые); запросов в службу поддержки (какие вопросы пользователей повторяются чаще всего); покрытия кода (где тестов меньше всего, а рисков больше).

Таким образом, Парето не разовый инструмент, а постоянная метрика проекта, которая позволяет принимать решения на каждом этапе жизненного цикла СИФЦО.

Построение диаграммы начинается со сбора и категоризации данных. Данные собраны из системы управления тестированием и баг-трекинга за отчетный квартал.

Категории дефектов:

- ошибки физических моделей – неточности в математических моделях целей, фона, атмосферы, приводящие к нереалистичным выходным данным;

- дефекты производительности (RT) – нарушения требований к работе в реальном времени: превышение времени расчета кадра, нестабильная частота обновления;

- ошибки интеграции с АП – проблемы взаимодействия с аппаратурой потребителя (ГСН, блоки обработки): сбои протоколов, рассинхронизация;

- дефекты визуализации/рендеринга – артефакты графики, некорректное отображение тепловых/радиолокационных портретов, ошибки в сценах;

- ошибки конфигурации и данных – сбои при загрузке сценариев, некорректная интерпретация входных данных (траекторий, характеристик целей);

- дефекты надежности (сбои) – зависания, аварийные завершения работы системы, утечки памяти;

- вторичные ошибки (наследственные) – дефекты, возникшие как следствие исправления других ошибок.

Далее выполняем измерение, упорядочивание и подсчет данных для дальнейшего сведения их в таблицу 1.

Таблица 1 - Количество заведенных инцидентов (частота возникновения)

Тип дефекта	Количество багов	% от общего	Кумулятивный %
Ошибки физических моделей	38	38%	38%
Дефекты производительности	22	22%	60%
Ошибки интеграции с АП	17	17%	77%
Дефекты визуализации	10	10%	87%
Ошибки конфигурации и данных	7	7%	94%
Дефекты надежности (сбои)	4	4%	98%
Вторичные ошибки	2	2%	100%
Итого	100	100%	

По результатам, сведенным в таблицу, строим диаграмму Парето (рис.1).

Столбцы: количество багов по категориям (отсортировано по убыванию).

Линия: кумулятивный (накопленный) процент от общего числа дефектов.

Пунктирная линия: уровень 80%, ключевой для анализа по принципу Парето.

Анализ результатов построения позволяет выделить ключевые наблюдения:

- более 77% всех дефектов сконцентрировано в первых трех категориях (из семи). Это практически идеально соответствует принципу Парето (80/20), где меньшая доля причин (≈43% типов) порождает большую долю проблем (≈77%).

Кумулятивная линия пересекает уровень 80% между третьей и четвертой категорией, что четко отделяет критически важные области от второстепенных.

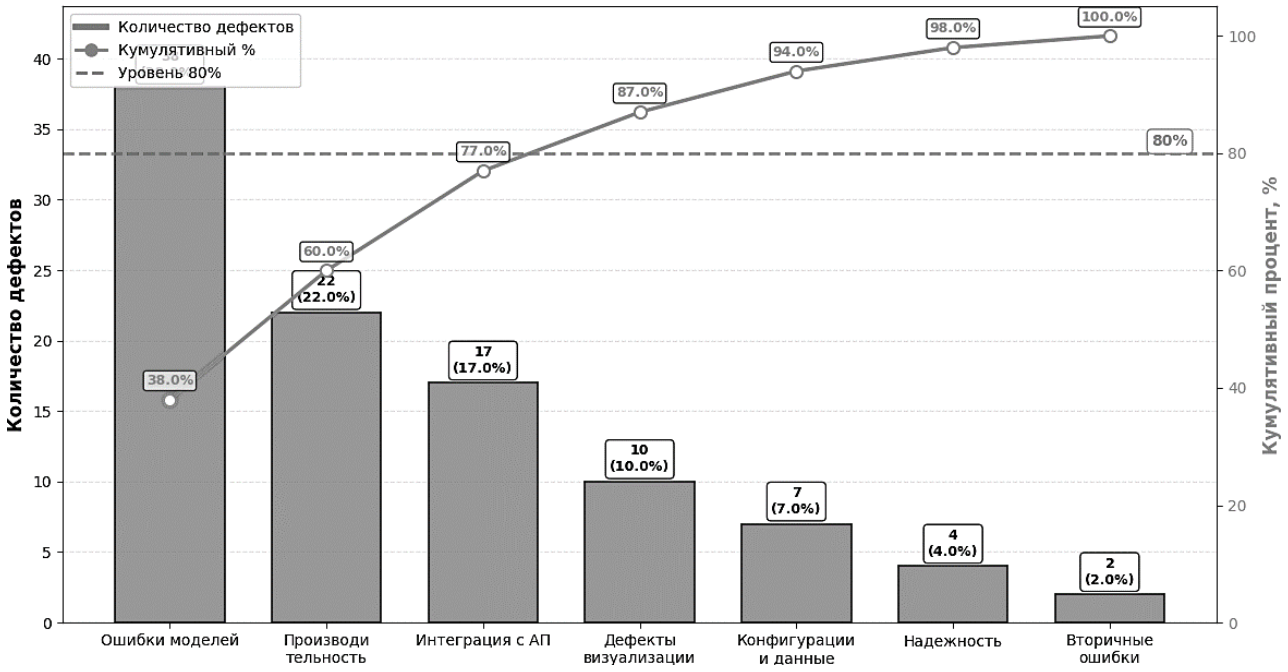

Рисунок 1 – Диаграмма Парето: анализ дефектов в системе имитации фоново-целевой обстановки (авторский рисунок)

Проанализировав диаграмму, производим детализацию по категориям:

Критическая зона (38%):

Ошибки физических моделей – абсолютный лидер (38%). Это указывает на фундаментальную проблему: неточность или неадекватность математического ядра системы.

Рекомендация: инициировать полномасштабный цикл верификации и валидации моделей на эталонных данных. Создать библиотеку тестовых сценариев с известным результатом. Усилить взаимодействие с физиками-модельерами на этапе проектирования.

Высокий приоритет (39%):

Дефекты производительности (22%) и Ошибки интеграции с АП (17%) – вместе составляют 39%. Это проблемы реализации и взаимодействия "железа" и ПО.

Рекомендация:

1. Провести профилирование кода для поиска "узких мест" и оптимизировать критические алгоритмы.
2. Разработать и внедрить детальные протоколы для тестирования интеграции с аппаратурой потребителя на ранних этапах.

Средний приоритет (17%):

Дефекты визуализации (10%) и Ошибки конфигурации (7%) – влияют на удобство использования и гибкость системы.

Рекомендация: внедрить автоматизированные регрессионные тесты для графического конвейера.

Низкий приоритет (6%):

Дефекты надежности (4%) и Вторичные ошибки (2%) – имеют наименьшую частоту.

Примечание: несмотря на низкий процент, дефекты надежности (сбои) являются крайне критичными для сложной стендовой системы, так как ведут к простою дорогостоящего оборудования. Требуют тщательного разбора каждого

случая. Вторичные ошибки указывают на недостаточное покрытие тестами после изменений кода.

Диаграмма Парето однозначно показывает, что ключевым драйвером низкого качества системы являются ошибки в фундаментальных физических моделях, за которыми следуют проблемы производительности и интеграции. Сфокусировав 80% усилий по улучшению качества на этих трех направлениях (верификация моделей, оптимизация кода, отладка аппаратных интерфейсов), команда достигнет максимального эффекта.

Заключение.

Сложность СИФЦО, определяемая многомерностью моделируемой среды, архитектурными вызовами и человеческим фактором, делает традиционное тестирование "вслепую" не просто неэффективным, а опасным. Оно создает иллюзию контроля, оставляя в тени самые критичные и трудноуловимые дефекты.

Для такой системы необходим иной подход: аналитический, основанный на данных. Вместо того чтобы пытаться проверить всё, нужно выявить наиболее критичные компоненты, где ошибки возникают чаще всего и имеют самые тяжелые последствия. Именно здесь на помощь приходит диаграмма Парето, позволяющая сфокусировать усилия на тех 20% кода, которые генерируют 80% проблем, делая процесс тестирования целенаправленным и эффективным.

Библиографический список:

1. Concas G. On the Distribution of Bugs in the Eclipse System / G. Concas, M. Marchesi, A. Murgia, R. Tonelli, I. Turnu // IEEE Transactions on Software Engineering. – 2011. – Vol. 37, no. 6. – P. 872–877.

2. Iqbal M. Applying 80/20 Rule in Waterfall Software Development Model / M. Iqbal, M. Rizwan// 2009 International Conference on Information and Communication Technologies. – Karachi, Pakistan, 2009. – P. 223–228.

3. Sorokin L. Can Search-Based Testing with Pareto Optimization Effectively Cover Failure-Revealing Test Inputs? /L. Sorokin, D. Safin, S. Nejati// Empirical Software Engineering. – 2025. – Vol. 30, no. 1. – Article 26.

4. Кох Р. Жизнь по принципу 80/20 / пер. с англ. – Минск: Попурри, 2004. – С. 1–6.