Посвящается всем тем, кто думал,
что у них ничего не получится.
– Лили Мара

Посвящается моей бабушке Джози и моей тете Алише,
которые привили мне любовь к чтению и технологиям.
– Джоэл Холмс

Краткое оглавление

1 ■ *Зачем перерабатывать исходный код на язык Rust* 19

2 ■ *Обзор языка Rust* 36

3 ■ *Введение в интерфейс с внешними функциями C (C FFI) и незащищенный язык Rust* 88

4 ■ *Продвинутый интерфейс с внешними функциями (FFI)* 126

5 ■ *Структурирование библиотек языка Rust* 178

6 ■ *Интеграция с динамическими языками* 206

7 ■ *Тестирование интеграций с Rust* 233

8 ■ *Асинхронный Python с языком Rust*.....261

9 ■ *WebAssembly для переработки исходного кода JavaScript*.....282

10 ■ *Интерфейс с WebAssembly для переработки исходного кода* ...304

Оглавление

Предисловие 10
Признательности 11
О книге 13
Кому стоит прочитать эту книгу 13
Как эта книга организована: дорожная карта 13
Об исходном коде 14
Дискуссионный форум liveBook 15
Об авторах 16
От издательства 17
Об иллюстрации на обложке 18
1 *Зачем перерабатывать исходный код на язык Rust* 19
1.1 Что такое переработка исходного кода? 20
1.2 Что из себя представляет язык Rust? 22
1.3 Почему именно язык Rust? 23
1.4 Следует ли перерабатывать исходный код на язык Rust? 23
1.4.1 Производительность 24
1.4.2 Защищенность памяти 26
1.4.3 Удобство технического сопровождения 27
1.5 Когда не следует перерабатывать исходный код на язык Rust 28
1.6 Как это работает? 29
1.7 Чему вы научитесь в этой книге? 30
1.7.1 Вызов функций Rust непосредственно из своей программы 30
1.7.2 Обмен со службой Rust по сети 33
1.8 Для кого предназначена эта книга? 34
1.9 Какие инструменты потребуются для начала работы? 34
Краткий итог 35
2 *Обзор языка Rust* 36
2.1 Владение и заимствование 37
2.2 Управление памятью в других языках 41
2.3 Времена жизни 45
2.3.1 Ссылки и заимствования 49
2.3.2 Контроль за мутируемостью 50
2.3.3 Ссылки и времена жизни 53
2.4 Строковые типы в языке Rust 56
2.4.1 Мутируемые строковые литералы 58
2.5 Перечисления и обработка ошибок 61
2.5.1 Перечисления 62
2.5.2 Обработка ошибок с помощью перечислений 66
2.5.3 Единичный тип 69

2.5.4 Типы ошибки 71
2.5.5 Преобразование ошибок 75
2.5.6 Поднятие паники при появлении ошибок 80
Краткий итог 86

3 *Введение в интерфейс с внешними функциями C и незащищенный язык Rust* 88

3.1 Незащищенный язык Rust 89
3.1.1 Обыкновенные указатели 90
3.2 Интерфейс с внешними функциями C 92
3.2.1 Включение пакета Rust 98
3.2.2 Создание динамической библиотеки с использованием языка Rust 100
3.2.3 Решение арифметических выражений на языке Rust 108
3.2.4 Общая черта Display 119
Краткий итог 125

4 *Продвинутый интерфейс с внешними функциями* 126

4.1 Скачивание исходного кода веб-сервера NGINX 127
4.2 Создание модуля для веб-сервера NGINX 128
4.3 Связывание языка C с языком Rust 132
4.3.1 Скрипты сборки 134
4.3.2 Инструмент bindgen 139
4.4 Чтение запроса веб-сервера NGINX 145
4.4.1 Аннотации времен жизни 152
4.4.2 Аннотации времен жизни в плагине для веб-сервера NGINX 158
4.5 Использование библиотеки-калькулятора 162
4.6 Написание HTTP-ответа 167
Краткий итог 176

5 *Структурирование библиотек языка Rust* 178

5.1 Модули 178
5.1.1 Какая разница? 182
5.1.2 Несколько файлов 183
5.2 Пути 186
5.2.1 Относительные и абсолютные пути 187
5.2.2 Псевдонимы путей 197
5.3. Восходящая видимость 202
Краткий итог 205

6 *Интеграция с динамическими языками* 206

6.1 Обработка данных в Python 206
6.2 Планирование перемещения на язык Rust 207
6.3 Разбор данных JSON 208
6.4. Написание модуля расширения Python на языке Rust 214
6.5 Измерение и тестирование производительности в Rust 219
6.6 Оптимизированные сборки 229
Краткий итог 231

7 *Тестирование интеграций с Rust* 233

7.1 Написание тестов на языке Rust 234
7.1.1 Документарные тесты 240
7.1.2 Добавление тестов в существующий исходный код 245

7.2 Тестирование исходного кода Rust с использованием языка Python ...250
7.2.1 Динамическое латание ...254
Краткий итог ... 259

8 Асинхронный Python с языком Rust ...261

8.1 Генерирование множества Мандельброта на языке Python ... 262
8.2 Масштабирование ... 265
8.3 Библиотека asyncio ... 268
8.4 Многопоточность исполнения ...271
8.5 Глобальная блокировка ...274
8.6 Библиотека PyO3 ... 276
Краткий итог ...280

9 WebAssembly для переработки исходного кода JavaScript ...282

9.1 Что такое WebAssembly? ...283
9.2 Перенос из JavaScript в Rust ...285
9.3 Язык Rust в браузере ... 286
9.3.1 Запрос данных ... 286
9.3.2 Компиляция в Wasm ... 290
9.3.3 Загрузка Wasm в браузер ... 291
9.4. Создание компонента библиотеки React ... 292
9.5 Веб-компоненты полностью на языке Rust ... 296
9.6 Еще раз о переработке JavaScript ...302
Краткий итог ...303

10 Интерфейс с WebAssembly для переработки исходного кода ...304

10.1 Универсальная среда исполнения системного интерфейса WASI ...308
10.2 Из браузера на машину ...312
10.3 Библиотека Wasm ...318
10.4 Потребление Wasm ...320
10.5 Еще о Wasm ...326
10.6 Память Wasm ...329
10.7 Это только начало ...335
Краткий итог ...335

Предметный указатель ...337

Предисловие

На протяжении всей нашей карьеры разработчиков программного обеспечения у нас была возможность участвовать в нескольких проектах по переработке кодовой базы. Их суть нередко состояла в одном и том же: нужно было масштабировать продукты и при этом уложиться в ограниченное время. Такая ситуация приводит к тому, что разработка занимает месяцы, заполненные обсуждениями шаблонов и языков.

Переработка кодовой базы с использованием языков Java и Go сопряжена со значительными трудностями, включая постоянное перемещение файлов, экспорт пакетов, создание системных оболочек и полное переписывание существующих систем. Пути к успеху редко были четко определены. Цель этой книги состоит в том, чтобы познакомить вас со многими из этих шаблонов, используя язык, предназначенный для разбивки и переписывания существующих систем. Книга «Переход на Rust. Рефакторинг исходного кода с других языков» демонстрирует легкость, с которой указанный язык может легко интегрироваться в вашу экосистему, обеспечивая преимущества масштабирования с первого дня благодаря особенностям языка.

Язык Rust привносит такие преимущества, как типобезопасность и защищенность памяти, а также связанный с этими свойствами прирост в производительности. Из этой книги вы узнаете, как язык Rust может улучшать практически любой проект. Язык Rust, который позиционируется как замена существующим языкам, таким как C и C++, отличается своей надежной цепочкой инструментов и функциональными компонентами защиты памяти. Мы также рассмотрим способы взаимодействия языка Rust с такими языками, как Python, и расскажем об улучшениях производительности при создании библиотек и модулей, работающих на обоих языках. Кроме того, мы откроем для вас неожиданные области применения языка Rust, например в веббраузерах и в качестве универсальной среды исполнения.

В целом цель этой книги – не только продемонстрировать мощь языка и экосистемы языка Rust, но и дать вам навыки уверенной переработки крупных систем.

Признательности

Благодарю своих партнеров за то, что они помогли довести эту книгу до конца, после того как я так долго откладывала ее в долгий ящик.

Спасибо моей маме, которая редактировала некоторые ранние наброски и чья любовь к чтению не давала мне отрываться от книг в течение многих лет.

Благодарю своих родителей, бабушек и дедушек за то, что они всегда поощряли меня в получении технического образования и делали это возможным.

Моему дяде Конраду спасибо за то, что помог развить любопытство к тому, как все устроено.

Спасибо всем сотрудникам издательства Manning. Эта книга не появилась бы на свет без самоотверженного труда редакторов, технических редакторов, графических редакторов, маркетологов, корректоров и многих других. Благодарю Энди Уолдрона за предоставленную возможность написать мою первую книгу.

Особая благодарность редакторам этой книги, Элеше Хайд и Сьюзан Этридж. Благодаря вашим советам я смогла воплотить свои фантазии в жизнь.

Компания OneSignal благодарит вас за то, что предоставили мне время и свободу для написания этой книги.

Благодарю учителей и профессоров, которые привили мне любовь к письменному слову, – Кристен Шумахер, Пола Хеберта и покойного Жана Лутца.

Спасибо тебе, Норм Крампе, за то, что удовлетворил мое техническое любопытство, и доктору Пэрис Франц за то, что вдохновила меня на такое важное дело, как написание книги.

Спасибо тебе, Ян Паскуаль, за то, что ты был подопытным кроликом в плане навыков технического общения, которые лежат в основе написания этой книги, и за твою поддержку на протяжении всего процесса.

Написание книги – это титанический труд, и я также благодарю всех, кто был одним из первых рецензентов или клиентов MEAP, оставивших отзывы о книге на онлайновых форумах.

– ЛИЛИ МАРА

Прежде всего хотел бы поблагодарить мою жену и партнера Челси, которая вдохновляет меня на осуществление моих мечтаний о писательстве и обучении.

Также благодарю двух моих сыновей, Эли и Абеля, которые являются неиссякаемым источником вдохновения. Посвящаю эту книгу двум

важным женщинам в моей жизни. Во-первых, это моя бабушка, которая привила мне любовь к книгам и чтению, и, во-вторых, моя тетя Алиша, которая в раннем возрасте повлияла на мою любовь к компьютерам.

Эта книга не была бы написана без огромной поддержки коллектива издательства Manning. Спасибо тебе, Энди Уолдрон, за предоставленную возможность написать книгу на столь интересную тему.

Как автор многих книг издательства Manning, я лично особенно ценю всех тех, кто оставил отзывы в рецензиях на книгу: Алена Куньо, Альфреда Томпсона, Амита Ламбу, Ариэля Отилибили, Крис. Карделла, Кристофера Вильянуэву, Клиффорда Тербера, Дэна Шейха, Даниэля Томаса Лареса, Диего Алонсо, Федерико Кирхейса, Фостера Хейнса, Габора Ласло Хайба, Жиля Якелини, Хаварда Уолл, Джареда Лав, Джеймса Блэчли, Джона Касевич, Джона Риддл, Джонатана Ривз, Жюльена Кастелейн, Кента Р. Спиллнер, Кшиштофа Камичек, Мацея Пшепира, Маркуса Гезелле, Мэттью Сармьенто, Макса Садрие, Михала Рутка, Мохсена Мостафа Йокар, Рамона Снир, Рани Шарим, Ричарда Рэндалл, Сальвадора Наваррете Гарсия, Сэма Ван Овермейр, Самбасиву Андалури, Сына джин Ким, Сейи Огуньеми, Тима Макнамара, Троя Эйслер, Уолта Стоунбернер, Уильяма Э. Уилер и Еркебулана Тулибергенова. И я признателен тем, кто приобрел эту книгу заранее через MEAP и предоставил отзывы и поддержку.

Очень благодарен Элеше Хайд за всю ту помощь, руководство и терпение, которые она проявила. На ее долю выпала трудная задача – с огромным мастерством руководить работой над этой книгой. Я очень ценю ее терпение и поддержку.

Также выражаю благодарность компании Regrow.ag, предоставившей мне свободу и вдохновившей на написание этой книги; моему другу Коди, который был рядом со мной с начальной школы; моим учителям английского языка в старших классах, которые поощряли мою писательскую деятельность и помогали мне обрести голос; и Отто, который всегда был рядом, чтобы выслушать и никогда не осуждать.

– ДЖОЭЛ ХОЛМС

О книге

В известной книге Мартина Фаулера (Martin Fowler) «Переработка исходного кода» (Refactoring) подчеркивается первичная цель переработки: улучшить конструктивное исполнение существующего исходного кода. Знакомые с указанной книгой читатели по достоинству оценили его метод представления различных сегментов исходного кода, за которым следуют улучшенные альтернативы, повышающие удобочитаемость, эффективность или простоту. Несмотря на то что во втором издании стратегии претерпели изменения, стержневая идея остается неизменной: функциональный исходный код всегда можно улучшить.

Книга «Переход на Rust. Рефакторинг исходного кода с других языков» описывает стратегии транзита с одного языка программирования на другой с сохранением внешнего поведения исходного кода. Как это достигается? Как вы увидите, Rust предназначен для постепенной замены других языков путем интеграции и декомпозиции существующего исходного кода – во многом напоминающих процесс ржавления железа – и замены его исходным кодом Rust. Изначально проект был нацелен на замену языка C++, но со временем расширился, включив в него языки JavaScript и Python.

Кому стоит прочитать эту книгу

Эта книга предназначена для разработчиков, которые специализируются на других языках, таких как C, C++, Python и JavaScript, но хотят изучить язык Rust. Хотя наша книга не дает подробного представления о данном языке, в ней приведены практические примеры и варианты использования для замены вашего исходного кода на язык Rust. Формального понимания языка Rust не требуется, хотя оно и полезно.

Как эта книга организована: дорожная карта

В соответствии с подходом Фаулера мы будем представлять задачи на одном языке и демонстрировать способы переработки этих сложностей на языке Rust. Цель состоит в том, чтобы поддерживать положенную в основу функциональность приложения, используя скорость и защищенность языка Rust с целью улучшения системы в целом.

Изложение начинается с ознакомления с языком Rust, обсуждения его механики и сравнения с такими языками, как C, C++ и Python. Эта информация представлена в контексте переработки исходного кода, делая упор на методах систематического улучшения системы, не позволяя им превращаться в неуправляемый исходный код. Кроме того, будут рассмотрены продвинутые функциональные компоненты языка Rust, такие как время жизни переменных и владение, которые имеют решающее значение для овладения языком.

Первостепенное внимание будет уделено языку C, который для многих является основополагающим. В главе 3 будет рассмотрена способность языка Rust создавать как защищенный, так и незащищенный исходный код, обследована тема обертывания опасного исходного кода на языке Rust и использование инструментов отладки. Эта основа подготовит вас к главе 4, где вы будете интегрировать Rust в существующую кодовую базу на языке C, управлять памятью и добавлять новую функциональность в веб-сервер NGINX.

После первоначальной интеграции в другую систему в главе 5 язык Rust будет рассмотрен как библиотечный инструмент. Создание пакетов, совместимых с другими проектами, является эффективным способом переработки приложений, при условии что эти библиотеки предлагают усиленную функциональность. Кроме того, будут обследованы метрики измерения и тестирования производительности, чтобы обосновать переход с более старых языков на язык Rust. В главах 6, 7 и 8 будут продемонстрированы способы применения этих пакетов для переработки исходного кода Python путем исполнения Python в экосистеме Rust либо путем встраивания экосистемы Rust в Python.

В последних двух главах вы познакомитесь с продвинутыми приложениями на языке Rust. Глава 9 посвящена компиляции Rust для работы в веб-браузерах с использованием нового двоичного формата под названием Wasm. В главе 10 эта технология будет использована для создания универсальной среды исполнения, предоставляющей гибкий (но сложный) метод переработки или взаимодействия с существующим исходным кодом.

Главы не обязательно читать по порядку, и если вы уже знакомы с языком Rust, то, вероятно, сможете пропустить первые две главы, при условии что не захотите освежить материал в памяти. Если вам не терпится перейти к определенному языку, то главы 3 и 4 посвящены интеграции с языками C и C++, главы 6–8 – интеграции с языком Python, а глава 9 – интеграции с языком JavaScript.

Главу 10 также можно прочитать отдельно, и в ней предлагается другой способ переработки, заключающийся не в изменении самого исходного кода, а в изменении среды, в которой исполняется приложение.

Переработка исходного кода – это скорее искусство, чем наука. И в нашей книге, и в книге Мартина Фаулера предлагаются шаблоны, которым предлагается следовать; ответственность за эффективное применение этих методов полностью лежит на вас.

Об исходном коде

Описанный в этой книге исходный код в основном посвящен языку Rust, но в контексте других языков. В начале рассматриваются основы языка Rust, а затем в оставшейся части книги описывается интеграция с языками C, Python и JavaScript. Эти языки не преподаются, но предполагается, что читатель их знает, если он занимается переработкой исходного кода на этом языке.

Какие-либо ограничения на используемое оборудование или программное обеспечение отсутствуют. Содержимое книги не относится к конкретной операционной системе и не нуждается в какой-либо спе-

циальной настройке, кроме установки языка Rust. В главах упоминаются дополнительные библиотеки и инструменты, но текст посвящен данной настройке, и читателю не требуется делать это заранее.

В дополнение к этому Rust – это развивающийся язык, и, следовательно, синтаксис и библиотеки могут со временем меняться. Для того чтобы максимально это учесть, в приводимых примерах мы позаботились о выборе стабильных библиотек.

Данная книга содержит много примеров исходного кода, как в пронумерованных листингах, так и в виде обычного текста. В обоих случаях исходный код отформатирован `вот таким шрифтом фиксированной ширины`, чтобы отделять его от обычного текста.

Во многих случаях изначальный исходный код был переформатирован; мы добавили разрывы строк и переработали отступы, чтобы уложиться в доступное пространство на странице книги. В некоторых случаях даже этого было недостаточно, и листинги содержали маркеры продолжения строки (➥). Кроме того, нередко комментарии из листингов удалялись, когда исходный код описывался в тексте. Многие листинги сопровождаются аннотациями к исходному коду, в которых подчеркиваются важные идеи.

Исполняемые фрагменты исходного кода можно получить из liveBook-версии (онлайновой версии) этой книги по адресу https://livebook.manning.com/book/refactoring-to-rust. Полный исходный код примеров их книги доступен для скачивания с веб-сайта Manning по адресу https://www.manning.com/books/refactoring-to-rust и из репозитория книги на GitHub по адресу https://github.com/lilymara/refactoring-to-rust.

Дискуссионный форум liveBook

Покупка данной книги включает в себя бесплатный доступ к онлайновой платформе чтения книг издательства Manning под названием liveBook. Используя эксклюзивные возможности обсуждения на liveBook, можно прикреплять комментарии к книге глобально либо к определенным разделам или абзацам. Там можно легко делать заметки для себя, задавать технические вопросы и отвечать на них, а также получать помощь от автора и других пользователей. В целях получения доступа к форуму перейдите по ссылке https://livebook.manning.com/book/duckdb-in-action/discussion. Подробнее о форумах издательства Manning и правилах поведения также можете узнать по адресу https://livebook.manning.com/discussion.

Издательство Manning видит свою обязанность перед читателями в том, чтобы предоставлять место, где может происходить содержательный диалог между отдельными читателями и между читателями и автором. Это обязательство не требует от автора какого-то конкретного объема участия, чей вклад в форум остается добровольным (и неоплачиваемым). Мы предлагаем вам попробовать задать автору несколько сложных вопросов, чтобы поддержать его интерес! Форум и архивы предыдущих обсуждений будут доступны на веб-сайте издателя на протяжении всего времени, пока книга находится в печати.

Об авторах

ЛИЛИ МАРА (LILY MARA) – разработчик программного обеспечения из Сан-Франциско, штат Калифорния. Много выступает с докладами о разработке программного обеспечения на языке Rust как внутри страны, так и за рубежом. Пишет на Rust с 2015 года и профессионально использует его для разработки высокопроизводительных масштабируемых систем. В настоящее время занимается написанием программного обеспечения в Discord.

ДЖОЭЛ ХОЛМС (JOEL HOLMES) – разработчик программного обеспечения, специализирующийся на разработке облачно-ориентированных приложений. Работал в нескольких стартапах, занимаясь конструированием и разработкой новых продуктов и служб, чтобы помогать компаниям-заказчикам развиваться и расти дальше. Попутно способствовал формированию инструментов и процессов, которые помогали в разработке и повышении качества. Живет в Питтсбурге со своей семьей и в настоящее время работает над созданием облачных приложений в Regrow.ag.

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем веб-сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об иллюстрации на обложке

На обложке книги «Переход на Rust. Рефакторинг исходного кода с других языков» изображена фигура под названием «Пьемонтуаз д'Асти», или «Женщина из города Асти в Пьемонте», взятая из сборника Жака Грассе де Сен-Совера, опубликованного в 1788 году. Эта иллюстрация тонко прорисована и раскрашена вручную.

В те времена по одежде можно было легко определять места проживания людей, их профессии или положение в обществе. В издательстве Manning превозносятся изобретательность и инициатива вычислительного бизнеса с помощью книжных обложек, в основу которых кладется богатое разнообразие региональной культуры столетней давности, ожившее благодаря фотографиям из коллекций, подобных этой.

1 Зачем перерабатывать исходный код на язык Rust

Эта глава охватывает следующие ниже темы:

- зачем может потребоваться переработка приложения;
- почему язык Rust хорошо подходит для переработки;
- когда стоит и не стоит начинать проект по переработке;
- общий обзор методов, которые используются для переработки исходного кода на язык Rust.

Если вы когда-либо слышали о языке программирования Rust, то, возможно, слышали и о компаниях-разработчиках программного обеспечения, которые переписывают свой исходный код с более медленного и интерпретируемого языка на язык Rust. Некоторые из этих компаний то и дело публикуют посты, в которых восхваляются преимущества системы Rust по сравнению с их предыдущими системами, и в них рассказывается очень интересная история: другие языки работают медленно, а Rust просто летает, и поэтому они рекомендуют переписывать исходный код на язык Rust, и системы станут быстрыми.

Хотя, возможно, покажется заманчивым думать, что все мы легко сможем переписать свой исходный код, когда появится что-то более совершенное, мы осознаем, что на самом деле программное обеспечение не существует в пузыре из бесконечных ресурсов. Повышение производительности и выплата технической задолженности должны балансироваться разработкой функциональных компонентов, запросами пользователей и миллионом других факторов, связанных с работой современного программного обеспечения. При повторной реализации функциональности на новом языке также необходимо обеспечивать, чтобы пользователи получали единообразную и надежную службу. Как же тогда разработчик сможет улучшить свою кодовую базу, сохраняя при этом ожидаемые высокие темпы разработки и надежность? Ответ заключается не в переписывании в стиле «большого взрыва», а в поступательной переработке исходного кода.

1.1 Что такое переработка исходного кода?

Рефакторинг (переработка/переделка исходного кода) – это процесс его реструктуризации таким образом, чтобы он исполнялся лучше, стал проще в плане технического сопровождения или соответствовал какому-либо другому определению «лучшего». Между переработкой и переписыванием существует различие, хотя и нечеткое. Разница между ними сводится к масштабу операции.

Переписывание – это повторная реализация с нуля целого приложения или значительной его части. Можно переписать систему, чтобы использовать преимущества нового языка программирования или модели хранения данных, или просто потому, что текущую систему сложно сопровождать в техническом плане и кажется, что проще будет ее выбросить и начать все сначала, чем улучшать.

Переработка – это переписывание в гораздо меньших масштабах. Вместо того чтобы стремиться к полной замене существующей системы, желательно найти те части системы, которые нуждаются в наибольшей помощи, и заменить как можно меньше исходного кода, чтобы улучшить систему. Преимущества переработки перед переписыванием многочисленны:

- Поскольку текущая система является «новой», она может продолжать работать и обслуживать клиентов, пока идет переработка. Можно разворачивать ряд очень небольших изменений в коде и смотреть, какое изменение вызвало проблему. Если переписывать и разворачивать всю новую систему одним махом, то как узнать, какая часть системы вызывает ошибки при их возникновении?
- Существующий исходный код, вероятно, уже имеет многолетний опыт разработки и мониторинга. Не следует недооценивать опыт других разработчиков в эксплуатации и отладке существующего кода. Если в новой системе возникает проблема, с которой у вас нет опыта работы, то как вы собираетесь ее устранять?