Посвящается Дженни. Я бы ничего не менял

Оглавление

Об авторе

Майкл Л. Перри (Michael L. Perry) опирался на работы таких математиков, как Бертран Мейер (Bertrand Meyer), Лесли Лампорт (Leslie Lamport) и Дональд Кнут (Donald Knuth), при создании математической системы для разработки программного обеспечения. Он воплотил эту систему в ряде проектов с открытым исходным кодом. Майкл часто выступает с докладами по математике и программному обеспечению на различных мероприятиях и в интернете. Вы можете узнать больше на сайте qedcode.com.

О техническом рецензенте

Карстен Томсен (Carsten Thomsen) – в основном бэкенд-разработчик, но работает и с небольшими фронтенд-разработками. Он является автором и рецензентом ряда книг и создал многочисленные учебные курсы для компании Microsoft, связанные с разработкой программного обеспечения. Он работает как фрилансер/подрядчик в разных странах Европы, используя такие инструменты, как Azure, Visual Studio, Azure DevOps и GitHub. Является исключительным специалистом по устранению неполадок путем постановки правильных вопросов, в том числе менее логичных, переходя от наиболее логичного к наименее логичному. Ему также нравится работать с архитектурой, исследованиями, анализом, разработкой, тестированием и исправлением ошибок в программах. Он обладает хорошими навыками наставничества и руководства командой, а также исследования и представления нового материала.

Благодарности

Как напоминает нам Толкиен, выходить за дверь – дело опасное. Первый шаг на пути, который привел к тому, что эта книга оказалась в ваших руках, был шатким. Я создавал свою первую распределенную систему и совершал все ошибки новичка. К счастью, у меня были хорошие друзья, с которыми я совершал эти ошибки.

Спасибо вам, Рассел Элледж (Russell Elledge) и Джерри Ферис (Jerry Feris), за то, что боролись вместе со мной. Вместе мы, трое друзей, изучили все неправильные способы использования TCP/IP и SOAP. Кто же знал, что трехстороннего рукопожатия недостаточно для создания нового?

Хотя первые попытки были нелегкими, мы начали разбираться. Рассел был моим постоянным соучастником, аудиторией и критиком на протяжении всего этого пути. Я должен также поблагодарить тебя за то, что ты познакомил меня с Крисом Гулдом (Chris Gould), который дал нам обоим свободу применить то, чему мы научились после той роковой первой попытки. Его поддержка позволила нам построить правильное решение на математически прочном фундаменте. Именно успех этого проекта дал мне окончательное подтверждение того, что этим понятиям *можно* научить.

Огромная благодарность Шону Уайтселлу (Sean Whitesell) за годы поддержки, ободрения и обсуждения. Ты всегда задаешь самые лучшие вопросы. И что не менее важно, умеешь мастерски объединять людей. Спасибо тебе за создание сообщества, которое помогло мне передавать идеи, вошедшие в эту книгу. И отдельное спасибо за установление окончательной связи для начала этого проекта.

Именно Шон познакомил меня с Флойдом Мэем (Floyd May). Флойд, ты такой глубокий мыслитель в области технологий, межличностных отношений и бизнеса. Ты заставил меня стать более общительным. Я не могу дождаться, когда увижу, куда тебя занесет.

Всем моим друзьям из «Улучшения»: Кори Дрю (Cori Drew), Гарольду Пулчеру (Harold Pulcher), Барри Форресту (Barry Forrest), Бену Кеннеди (Ben Kennedy), Дэвиду Вибберту (David Vibbert), Дэвиду Белчеру (David Belcher), Дэвиду О'Хара (David O'Hara)… всем Дэйвам. Мы выросли вместе. Я помню, как впервые встретил каждого из вас и все, что мы узнали с тех пор. Особое спасибо Тиму Рейберну (Tim Rayburn) за то, что помог мне вырасти как оратору, импровизатору и лидеру. И пожалуйста, не говорите Девлину Лайлзу (Devlin Liles), что я считаю его самым блестящим человеком в компании. Думаю, я бы не выдержал, если бы он узнал.

Особая благодарность Джоан Мюррей (Joan Murray) из Apress за веру в этот проект и Джилл Бальзано (Jill Balzano) за поддержку моего первого издательского опыта. И спасибо Карстену Томсену (Carsten Thomsen) за отзыв, полный улучшений и ободрения. Вы все сделали этот процесс самым увлекательным из всего, что я делал, выполняя самую сложную работу.

И наконец, моя самая искренняя благодарность моей семье. Папа, ты вдохновил меня на создание программного обеспечения. Ты обеспечил меня не только Apple II и IBM, на которых я проучился до конца школы, но и познакомил меня с первым человеком, зарабатывающим на жизнь тем, что я люблю. Ты продолжал присылать журналы Nibble и Byte, чтобы утолить мою жажду и, в конце концов, чтобы вдохновить меня писать о том, чему я научился. Я такой, какой я есть, благодаря тебе.

Дженни. Ты всегда верила в меня. Ты мой партнер и моя причина.

И Каэле. Ты заставляешь меня гордиться тобой. Я так счастлив, что мы закончили этот проект вместе.

А дорога ведет все дальше и дальше.

Введение

Шел 2001 год. Я присоединился к команде, использующей J2EE[1] версии 1.3 для создания распределенного процессора подарочных карт. Система точек продаж была написана на Microsoft Visual C++ 6.0. Мы только узнали об этой новой вещи под названием SOAP (Simple Object Access Protocol) – простой протокол доступа к объектам. Шутка заключалась в том, что он был слишком плохо определен, чтобы называться протоколом, он не был связан с доступом к объектам и был совсем не простым. Но он давал некоторые надежды на то, чтобы заставить клиента C++ обращаться к серверу Java.

Мы добавили в свои библиотеки три новые книги. Первая была посвящена реализации клиента SOAP на C++, вторая – JAXP, Java API для обработки XML, а третья подробно описывала работу и ограничения TCP/IP. Вооружившись этими инструментами, мы начали.

Сначала задача заключалась в том, чтобы заставить две платформы общаться друг с другом. Когда мы наконец остановились на подмножестве SOAP, с которым могли работать обе стороны, мы подумали, что преодолели этот бугор. Мы не знали, что по другую сторону были горы.

Возникли проблемы с надежностью сети. Мы создали лабораторию, которая постоянно проводила транзакции каждую ночь. Утром мы проверяли баланс карт и обнаруживали, что на некоторых машинах была неверная сумма. Это приводило к целому дню копания в журналах, настраивали следующий тест, а затем запускали его до утра.

Со временем мы разработали протокол обмена сообщениями (поверх SOAP), основанный на подтверждениях и квитанциях. Одна сторона отправляла сообщение. На следующее утро мы обнаруживали пропажу сообщений. Затем получатель подтверждал, что сообщение пришло. На следующее утро мы обнаруживали дубликаты. И тогда отправитель регистрировал подтверждение. Пропавших сообщений стало меньше, но все еще не было идеально.

Потребовалось много неудачных релизов и много лет работы, в том числе по праздникам, чтобы решить все проблемы. Мы узнали о «задаче двух генералов» (TGP) и поняли, почему наш протокол обмена сообщениями был несовершенен. Затем мы узнали о конечной согласованности и получили рабочее решение. Это решение требовало, чтобы существовала некоторая неопределенность в отношении того, сколько денег осталось на подарочной карте. Мы попытались провести разговор об этом с владельцем продукта. Банкиры получают конечную последовательность денег. Владелец продукта не был банкиром.

Уроки, которые мы извлекли при работе с подарочными картами, были усвоены тяжелым трудом. «Гарантированная доставка» означает не то, что вы

---

[1] Java 2 Enterprise Edition или J2EE – ранняя версия Jakarta EE – набора спецификаций и документации для языка Java, описывающей архитектуру серверной платформы. https://ru.wikipedia.org/wiki/Jakarta_EE.

думаете. Вам нужно сначала переместить данные, а затем обработать их. Удаленные вызовы процедур (RPC) – это не вызовы процедур. В системе «клиент–сервер» нет ни одной строки кода, до которой транзакция отменяется и после которой она фиксируется. Я бы не хотел получать эти уроки снова и снова.

И вот я начал собирать эти уроки воедино и определять систему, которую я назвал историческим моделированием. Она была основана на идее, что исторические факты не могут быть изменены или уничтожены. Она опиралась на отношения предшественников и преемников между фактами. И идентифицировала факты, основываясь только на их содержании, а не местоположении. Я заполнил ноутбук примерами исторических моделей. В конце концов, я интуитивно почувствовал, какие виды решений могут быть смоделированы исторически, а какие – нет. Тогда я понял, что должен поделиться этим. Надеюсь, я смогу избавить кого-то еще от необходимости усваивать эти уроки трудным путем.

С тех пор у меня было бесчисленное множество разговоров о неизменяемых архитектурах. Я разбил тему на легко усваиваемые фрагменты для выступлений на конференциях и в группах пользователей, а также создал два фреймворка с открытым исходным кодом – Correspondence и Jinaga. Однако по отдельности они не дали возможности другим начать применять неизменяемость на практике. Принятие только части идей оставляет пробелы, которые могут быть заполнены лишь с помощью остальной части системы.

Все это привело к книге, которую вы сейчас держите в руках. Это полное описание системы, шаблонов и техник. Она предвосхищает проблемы, которые создает историческое моделирование, и предлагает решения, позволяющие обеспечить его целостную реализацию. Самое главное, в ней представлена математическая основа, которая заставляет эту технику работать.

Если вы дочитали введение до конца, то наверняка сталкивались с некоторыми из этих проблем. Возможно, вы даже нашли похожие решения. Остается только еще несколько вопросов, которые, вероятно, у вас есть по поводу этой книги. Кто должен ее читать? Что я получу от нее? Как она организована? И как мне ее читать?

Рад, что вы спросили.

Кому следует ее прочитать

Эта книга предназначена в первую очередь для трех аудиторий: лиц, принимающих решения, конструкторов систем и создателей инструментов. Вы – лицо, принимающее решения, если вы определяете проблемы, для которых хотите создать решения. Ваша должность может быть технический директор, владелец продукта или аналитик бизнес-систем. Существуют проблемы, которые вы можете передать на аутсорсинг, есть проблемы, для которых вы можете купить решения, а есть проблемы, которые определяют основную ценность вашего бизнеса. Вам необходимо найти подходящих людей для решения проблем третьего типа. Чтобы найти их, вам нужно уметь с ними разговаривать. А после того, как вы пригласите их на работу, нужно понять, чем они занимаются. Если ваша основная бизнес-проблема похожа на ту, которую можно решить с помощью неизменяемой архитектуры, эта книга поможет вам создать такую команду и провести эти беседы.

Или, возможно, вы занимаетесь созданием систем. Вы являетесь членом команды, привлеченной для создания ценностей в основной области бизнеса. Ваша должность может быть разработчик, инженер по контролю качества или дизайнер пользовательского интерфейса. Вы знаете, как решать проблемы. Но было бы здорово иметь несколько готовых решений для наиболее распространенных проблем распределенных вычислений. Вы хотите знать, что все нестандартные ситуации учтены. Вы хотите иметь общий язык для обсуждения решений с людьми, которые помогают вам находить их. Если ваши задачи по разработке программного обеспечения требуют создания согласованных распределенных систем, то эта книга даст вам такие инструменты.

Наконец, вы, как и я, возможно, являетесь создателем инструментов. Вы – множитель силы. Ваши разработки дают возможность другим создавать решения быстрее, более предсказуемо и более эффективно. Вы можете быть архитектором решений или сопровождающим открытого исходного кода. Если у вас есть команда, вы хотите, чтобы ее члены были сосредоточены на создании бизнес-ценностей, а вы позаботитесь о «сантехнике». Если вы обслуживаете сообщество, вы хотите, чтобы потребители могли быстро изучить и применить ваш фреймворк для создания надежных систем. В любом случае, в этой книге изложены математика, алгоритмы и шаблоны, которые гарантируют корректность ваших решений.

Что вы получите от этого

У меня есть секрет. Это книга по математике. Не говорите об этом никому, кто не дочитал до конца введение.

Математика – величайшее изобретение человечества. Она удивительна в своей способности описывать мир природы. Она поразительно применима к широкому кругу проблем. И это единственный способ, с помощью которого мы можем быть уверены в чем-либо.

Обычно мы убеждаемся в том, что поняли что-то правильно, проверяя это. Мы применяем наше решение в одной ситуации и смотрим, получим ли мы ожидаемый результат. Затем пробуем другой сценарий и смотрим, что получится. Если мы действительно хороши, то сможем представить несколько непредвиденных условий и протестировать их. Но непредвиденное очень трудно предугадать.

Тестирование – это сбор опытных данных. Оно дает уверенность в том, что система ведет себя так, как ожидается, в определенных случаях, но не дает никакой уверенности в том, что вы ничего не упустили.

Знание требует математических выводов. Если что-то доказано математически, то вы можете быть уверены, что это будет истинно, независимо от того, какой тест вы попробуете провести. Теорема Пифагора верна для любого прямоугольного треугольника. Геометрия Евклида верна для всех фигур на плоскости. Если ваши рассуждения безупречны, вы можете быть уверены, что не пропустили ни одного крайнего случая.

Это не означает, что математические истины универсальны. Дело в том, что они имеют известные ограничения. Деление работает только для ненулевых делителей. Теорема Пифагора действует лишь на плоскости. Правила дедукции

говорят нам, как перенести эти ограничения на решение, чтобы точно знать, где это решение применимо, а где нет.

Эта книга применяет математическую строгость к проблеме распределенных вычислений. Она не является первой в данной области, но предлагает полное и практическое решение. Если вы будете следовать дедуктивным рассуждениям над проблемой и внесете ограничения распределенных систем в свои вычисления, то в итоге вы получите понимание границ решения. Эта книга – ваш путеводитель в этом процессе.

Как она организована

Книга условно разделена на три части, предназначенные для трех основных аудиторий. Лицам, принимающим решения, необходимо прочитать только первую часть, которая включает в себя первые три главы. В этой части вы сначала узнаете, почему неизменяемость так важна. Затем исследуете пространство альтернатив и познакомитесь с историческим моделированием. Наконец, вы узнаете, как читать историческую модель, чтобы более эффективно общаться со своей командой. Вы можете прекратить чтение, когда мы перейдем к глубокой математике.

Специалисты по созданию систем захотят прочесть вторую часть. Она включает в себя главы с четвертой по восьмую. Мы глубоко погружаемся в математические основы неизменяемости, причинности и бесконфликтных реплицируемых типов данных (conflict-free replicated data types – CRDT). Затем увидим, как применять эти математические рассуждения для анализа систем, построения машин состояний и соблюдения правил безопасности. Именно эти инструменты понадобятся вам для создания надежных распределенных систем. В завершение данного раздела приведен каталог основных шаблонов, которые помогут вам начать строить исторические модели.

Люди моей профессии, создатели инструментов захотят дочитать до конца. В третьей части мы разберем компоненты компьютерной системы и узнаем, как использовать их в неизменяемой архитектуре. Мы научимся обновлять пользовательский интерфейс с помощью инверторов запросов. Мы будем хранить неизменяемые записи в реляционной базе данных, будем надежно и безопасно обмениваться неизменяемыми сообщениями в сетях различных типов. В конце концов, мы собираем все вместе и описываем экосистему, состоящую из совместных приложений, генерирующих новое поведение на основе общих спецификаций. Это нечто поистине прекрасное и вдохновляющее, и я надеюсь, что вы последуете за мной до конца.

Как ее читать

Теперь, когда вы знаете, что это книга по математике, у вас могут возникнуть некоторые сомнения по поводу того, как вы будете ее читать. Возможно, вы с трудом проходили алгебру или забросили вычисления. Вы можете подумать, что математика не для вас.

Я считаю, что математика подходит всем. И моя цель в данной книге – доказать это. Математика – это не что иное, как применение логических рас-

суждений к символическим представлениям абстрактных понятий. С другой стороны, программирование – это применение логических операций к символическому языку, описывающему общие правила. Иными словами, это одно и то же. Если вы программист, то вы – прикладной математик.

Одна из проблем математики – это жаргон. Для того чтобы эффективно общаться друг с другом, математикам приходится придумывать слова для обозначения идей. К сожалению, естественный язык ограничен, и все хорошие слова уже заняты. И поэтому математики либо придумывают новые слова, либо используют термины, которые означают почти то, что нужно. В этой книге мы будем говорить, например, о свойствах полурешетки связности. Но я постараюсь не использовать эти слова, если смогу избежать этого. А если избежать не удастся, я буду давать им четкое определение.

Еще одна проблема с математикой – это то, как она написана. Математические работы имеют предсказуемую форму. Они начинаются с аннотации, затем полностью определяют проблему. Далее следуют раздел за разделом лемм и предложений, выстраивающих аргументацию. Каждое утверждение обосновывается предыдущими утверждениями. Наконец, как у М. Найт Шьямалана (M. Night Shyamalan), следует поворот сюжета, последнее утверждение рассматривает всю аргументацию в перспективе, и появляется результат.

Хотя мне очень нравятся хорошие математические статьи, я не читаю их так, как они написаны. Я бегло просматриваю первые несколько абзацев в поисках мотивации проблемы. Сканирую заголовки, чтобы понять суть аргументов. Я хочу знать, почему каждое утверждение доказано и каков будет его вклад в общее дело. Я хочу знать, как будет развиваться история, прежде чем потратить время на ее понимание.

Я написал эту книгу так, как я читаю статьи по математике. В каждом разделе вы поймете, чем обусловлен тот или иной результат. Затем вы увидите набросок основных рассуждений. Не будет загадкой, почему каждый шаг находится на отведенном ему месте. Потом каждый из этих шагов будет обоснован с той строгостью, которой он требует.

Я ожидаю, что это повлияет на то, как вы будете читать книгу. Если вам нужны только результаты, вы можете прочитать всего один-два абзаца после заголовка раздела. Если вы хотите узнать, почему или как, то продолжите читать дальше, чтобы понять аргументацию. А если вам нужно убедиться, то дочитайте весь раздел до конца. Важно, что вы можете прекратить чтение, когда оно становится слишком глубоким, и перейти к следующему разделу. Вы не пропустите ничего важного для себя.

Если вы прочитали этот раздел, ничего не пропустив, то я искренне рад, что вы у меня есть. Вы – один из моих людей. С вашей помощью мы сможем создать программное обеспечение, которое необходимо миру. Мы сделаем его надежным, эффективным и правильным. И это даст нашим пользователям автономию, необходимую для того, чтобы они могли выполнять свою работу творчески и уверенно, зная, что мы обеспечили математическую строгость.