Оглавление

Предисловие Марка Берджеса 19
Предисловие авторов 22
Условные обозначения 27
Использование примеров кода 27
Благодарности 29

Часть I. Введение

Глава 1. Вступление 35
Подход системного администратора к управлению сервисами 35
Подход Google к управлению сервисами: техника обеспечения надежности сайтов 37
Принципы SRE 40
Уделяем особое внимание инженерным задачам 40
Добиваемся максимальной скорости внедрения изменений без потери качества обслуживания 41
Мониторинг 42
Реагирование на критические ситуации 43
Управление изменениями 44
Прогнозирование нагрузки и планирование производительности 44
Материально-техническое обеспечение 45
Эффективность и производительность 45
Конец начала 46
Глава 2. Среда промышленной эксплуатации Google с точки зрения SRE 47
Оборудование 47
Системное ПО, которое «организует» оборудование 49
Управление машинами 49
Хранилище 51
Сеть 53

Другое системное ПО ... 54
Сервис блокировок ... 54
Мониторинг и оповещение ... 54
Наша инфраструктура ПО ... 55
Наша среда разработки ... 55
Shakespeare: пример сервиса ... 56
Жизненный цикл запроса ... 57
Организация задач и данных ... 59

Часть II. Принципы

Глава 3. Приручаем риски ... 64
Управление рисками ... 64
Измерение рисков, связанных с сервисом ... 65
Рискоустойчивость сервисов ... 67
Определение рискоустойчивости пользовательских сервисов ... 68
Определение рискоустойчивости инфраструктурных сервисов ... 71
Обоснование критерия суммарного уровня ошибок (бюджета ошибок) ... 73
Формируем бюджет ошибок ... 74
Преимущества ... 75

Глава 4. Целевой уровень качества обслуживания ... 77
Терминология для уровня качества обслуживания ... 78
Показатели ... 78
Целевые показатели ... 79
Соглашения ... 80
Показатели на практике ... 81
Что важно для вас и ваших пользователей ... 81
Сбор показателей ... 82
Агрегирование ... 83
Стандартизация показателей ... 84
Целевые показатели на практике ... 85
Определение целей ... 85
Выбор целевых показателей ... 86
SLO как инструмент управления ... 87
SLO формируют ожидания ... 88
Соглашения на практике ... 89

Глава 5. Избавляемся от рутины ... 90
Что такое рутина ... 90
Почему лучше иметь меньше рутинной работы ... 92
Что такое инженерная работа ... 93
Всегда ли рутина вредна ... 94
Итоги главы ... 95

Глава 6. Мониторинг распределенных систем 96
Определения 96
Зачем нужен мониторинг 97
Ставим для мониторинга реальные задачи 98
Симптомы и причины 100
Методы черного и белого ящика 100
Четыре золотых сигнала 101
Позаботимся о своем «хвосте» (или производительность измеряемая и наблюдаемая) 103
Выбор подходящего уровня детализации для измерений 104
Максимально просто, но не проще 104
Сводим все принципы воедино 105
Долгосрочное наблюдение 107
Bigtable SRE: когда оповещений слишком много 107
Gmail: предсказуемые ответы человека 108
Долгосрочная перспектива 109
Итоги главы 109

Глава 7. Эволюция автоматизации в Google 111
Польза автоматизации 111
Постоянство 112
Платформа 112
Быстрое восстановление 113
Быстродействие 113
Экономия времени 114
Польза автоматизации для Google SRE 114
Применение автоматизации 115
Применение автоматизации в Google SRE 116
Уровни автоматизации 117
Исключаем себя из процесса: автоматизируем все! 119
Облегчаем жизнь: автоматизируем процесс запуска кластера 121
Выявление несоответствий с помощью Prodtest 122
Идемпотентное разрешение несоответствий 124
Движение к специализации 126
Запуск кластера, ориентированный на сервисы 127
Borg: появление компьютера размером с дом 128
Основное качество — надежность 130
Рекомендации 132

Глава 8. Технологии выпуска ПО 133
Роль релиз-инженера 134
Основные положения 134
Модель самообслуживания 134
Высокая скорость 135

«Герметичные» сборки ... 135
Обязательные политики и процедуры ... 136
Непрерывная сборка и развертывание ... 136
Сборка ... 136
Ветвление ... 137
Тестирование ... 137
Пакеты ... 138
Rapid ... 138
Развертывание ... 140
Управление конфигурацией ... 140
Итоги главы ... 142
Это работает не только для Google ... 142
Управляйте релизами с самого начала ... 143

Глава 9. Простота ... 144
Стабильность или гибкость? ... 144
«Скучность» как добродетель ... 145
Не отдам свой код! ... 146
Показатель «Отрицательные строки кода» ... 146
Минимальные API ... 147
Модульность ... 147
Простота релизов ... 148
Итоги главы ... 148

Часть III. Практики

Мониторинг ... 150
Реагирование в критических ситуациях ... 150
Постмортем и анализ основных причин ... 152
Тестирование ... 152
Планирование пропускной способности ... 152
Разработка ... 153
Продукт ... 153
Другие источники от Google SRE ... 154

Глава 10. Оповещения на основании данных временных рядов ... 155
Укрепление позиций Borgmon ... 156
Инструментарий для приложений ... 157
Сбор экспортированных данных ... 158
Память временных рядов как хранилище данных ... 159
Метки и векторы ... 160
Вычисление правил ... 162
Оповещение ... 167
Разбиваем топологию системы мониторинга на части ... 168

Мониторинг методом черного ящика....170
Обслуживаем конфигурацию....170
Десять лет спустя....172

Глава 11. Быть на связи....173
Введение....173
Жизнь дежурного инженера....174
Сбалансированная организация дежурств....175
Баланс количества....175
Баланс качества....176
Компенсация....177
Чувство безопасности....177
Избегание неуместной нагрузки....179
Операционная перегрузка....179
Коварный враг: операционная недоработка....181
Итоги главы....181

Глава 12. Эффективная диагностика и решение проблем....182
Теория....183
Практика....185
Отчет об ошибках....185
Первичная обработка и сортировка....186
Обследование....187
Диагноз....189
Подтверждение диагноза....192
Волшебная сила отрицательных результатов....193
Лечение....195
Пример: анализ реальной ситуации....196
Как облегчить решение проблем....200
Итоги главы....200

Глава 13. Реагирование в критических ситуациях....201
Что делать, когда система сломалась....201
Авария, вызванная тестированием....202
Подробности....202
Развитие ситуации....202
Подведение итогов....203
Авария, вызванная изменениями конфигурации....203
Подробности....203
Развитие ситуации....204
Подведение итогов....204
Авария, вызванная процессом....206
Подробности....206
Развитие ситуации....206
Подведение итогов....207

У всех проблем есть решение ... 208
Учитесь на прошлом опыте. И не повторяйте его ... 208
Храните историю перебоев в работе ... 208
Задавайте себе сложные и даже неправдоподобные вопросы: «А что если?..» ... 209
Поощряйте упреждающее тестирование ... 209
Итоги главы ... 209

Глава 14. Управление в критических ситуациях ... 211
Неуправляемый инцидент ... 211
Анатомия неуправляемого инцидента ... 212
Излишняя сосредоточенность на технической стороне проблемы ... 212
Низкий уровень взаимодействия ... 213
«Вольный стрелок» ... 213
Элементы процесса управления в критических ситуациях ... 213
Рекурсивное разделение обязанностей ... 213
Выделенный центр управления ... 214
Обновляемый документ о состоянии инцидента ... 215
Четкая и оперативная передача полномочий ... 215
Управляемый инцидент ... 215
Когда следует сообщать об инциденте ... 217
Подведем итоги ... 217

Глава 15. Культура постмортема: учимся на ошибках ... 219
Философия постмортема от Google ... 220
Сотрудничайте и делитесь знаниями ... 222
Внедрение культуры постмортема ... 223
Итоги главы: непрерывные улучшения ... 225

Глава 16. Контроль неисправностей ... 227
Escalator ... 228
Outalator ... 228
Агрегирование ... 230
Маркировка ... 230
Анализ ... 231
Отчетность и общение ... 232
Неочевидная польза ... 232

Глава 17. Тестирование надежности систем ... 234
Виды тестирования ПО ... 236
Традиционное тестирование ... 236
Модульное тестирование ... 237
Интеграционное тестирование ... 237
Системное тестирование ... 238
Тестирование в промышленном окружении ... 239
Тестирование конфигураций ... 240
Нагрузочное тестирование ... 241

Окружения сборки и тестирования проекта....243
Масштабирование тестирования....246
Тестирование масштабируемых инструментов....247
Тестируем катастрофы....249
В погоне за скоростью....250
Передача в промышленную эксплуатацию....252
Ожидание сбоя тестов....253
Интеграция....255
Зондирование системы в промышленной эксплуатации....257
Итоги главы....260

Глава 18. Разработка ПО службой SRE....261
Почему так важна разработка ПО внутри службы SRE....262
Пример Auxon: история проекта и предметная область....263
Традиционное планирование производительности....263
Врожденная неустойчивость....264
Трудоемкость и неточность....265
Наше решение: планирование производительности, основанное на целях....266
Планирование производительности, основанное на целях....266
Предпосылки для целей....267
Знакомимся с Auxon....268
От требований до реализации: достижения и полученные уроки....271
Повышаем осведомленность и способствуем внедрению....273
Команда в развитии....276
Культивирование разработки ПО в службе SRE....277
Успешное создание культуры разработки ПО службой SRE: набор персонала и время разработки....278
Движемся к успеху....279
Итоги главы....281

Глава 19. Балансировка нагрузки на уровне фронтенда....282
Мощность — это не ответ....282
Балансировка нагрузки с использованием DNS....284
Балансировка нагрузки с использованием виртуального IP-адреса....287

Глава 20. Балансировка нагрузки в дата-центре....290
Идеальный случай....291
Выявляем плохие задачи: управление потоками и «хромые утки»....293
Простой подход к контролю работоспособности задач: управление потоками....293
Усовершенствованный подход к контролю работоспособности задач: состояние «хромой утки»....293
Ограничение пула соединений с помощью подмножеств....295
Выбираем правильное подмножество....295
Алгоритм выбора подмножества: случайное подмножество....296
Алгоритм определения подмножества: предопределенное подмножество....298

Политики балансировки нагрузки 301
Простой Round Robin 301
Least-Loaded Round Robin 304
Weighted Round Robin 306

Глава 21. Справляемся с перегрузками 308
Ловушки понятия «запросов в секунду» 309
Лимиты потребителей 310
Регулирование на стороне клиента 311
Критичность 313
Сигналы загруженности 314
Обработка ошибок, связанных с перегрузкой 315
Решаем выполнить повторную попытку 316
Нагрузка от соединений 319
Итоги главы 320

Глава 22. Справляемся с каскадными сбоями 322
Причины каскадных сбоев и способы их избежать 323
Перегруженность сервера 323
Истощение ресурсов 324
Процессор 325
Память 326
Потоки 326
Дескрипторы файлов 326
Зависимости между ресурсами 327
Недоступность сервисов 327
Предотвращаем перегруженность сервера 328
Управление очередями 330
Сегментация нагрузки и мягкая деградация 330
Повторные попытки 332
Задержка и дедлайны 335
Холодный запуск и холодное кэширование 338
Срабатывание условий каскадного сбоя 341
«Гибель» процесса 341
Обновления процессов 341
Новые релизы 341
Органический рост 342
Запланированные изменения, исчерпание ресурсов или отключения 342
Тестирование на предмет каскадных сбоев 343
Тестируйте до возникновения сбоев и после 343
Тестируйте популярные клиенты 345
Тестируйте некритические бэкенды 345
Мгновенное реагирование на каскадные сбои 345
Увеличьте количество ресурсов 346
Прекратите выполнять проверки на сбои/«гибель» 346

Перезапускайте серверы ... 346
Отбрасывайте трафик ... 346
Входите в деградированные режимы ... 347
Избавляйтесь от пакетной нагрузки ... 347
Избавляйтесь от плохого трафика ... 348
Итоги главы ... 348

Глава 23. Разрешение конфликтов: консенсус в распределенных системах и обеспечение надежности ... 350
Мотивация к использованию консенсуса: сбои координации распределенных систем ... 353
Пример 1: возникновение двух и более ведущих в схеме с ведущим и ведомыми ... 354
Пример 2: восстановление после сбоя требует вмешательства человека ... 354
Пример 3: некорректные алгоритмы членства в группе ... 355
Как работает распределенный консенсус ... 355
Шаблоны системной архитектуры для распределенного консенсуса ... 357
Надежные машины с реплицированным состоянием ... 358
Надежные реплицированные хранилища данных и конфигураций ... 359
Общедоступная обработка, применяющая алгоритм выбора лидера ... 359
Распределенная координация и блокировка сервисов ... 360
Надежный распределенный механизм помещения в очередь и отправки сообщений ... 361
Производительность систем с распределенным консенсусом ... 363
Multi-Paxos: детализированный поток сообщений ... 364
Масштабирование нагрузки, связанной с операциями чтения ... 366
Кворумная аренда ... 366
Производительность распределенного консенсуса и задержка сети ... 367
Размышляем о производительности: Fast Paxos ... 369
Стабильные лидеры ... 369
Пакетная обработка ... 370
Доступ к диску ... 370
Развертывание распределенных систем, основанных на консенсусе ... 372
Количество реплик ... 372
Расположение реплик ... 374
Производительность и балансировка нагрузки ... 375
Наблюдение за распределенными системами, основанными на консенсусе ... 381
Итоги главы ... 383

Глава 24. Cron: планирование и расписание в распределенных системах ... 384
Cron ... 385
Введение ... 385
Надежность ... 385
Задачи cron и идемпотентность ... 386

Масштабирование Cron 386
Расширенная инфраструктура 387
Расширенные требования 387
Создание cron в Google 388
Отслеживание состояния задач cron 389
Использование Paxos 389
Роли лидера и последователя 390
Сохраняем состояние 394
Запуск большого экземпляра Cron 395
Итоги главы 396

Глава 25. Конвейеры обработки данных 397
Происхождение шаблона проектирования «Конвейер» 397
Основной эффект, который большие данные оказывают на шаблон простого конвейера 398
Сложности, связанные с шаблоном циклического конвейера 398
Проблемы, вызванные неравномерным распределением работы 399
Недостатки циклических конвейеров в распределенных средах 400
Наблюдение за проблемами в циклических конвейерах 401
Проблемы «шумной толпы» 402
Интерференция нагрузки 402
Знакомство с Google Workflow 404
Workflow как шаблон «Модель — представление — контроллер» 404
Этапы выполнения в Workflow 406
Гарантии корректности Workflow 406
Гарантируем непрерывность бизнеса 408
Итоги главы и завершающие ремарки 410

Глава 26. Сохранность данных: как пишется, так и читается 411
Прямые требования к сохранности данных 412
Выбор стратегии для обеспечения отличной сохранности данных 413
Резервные копии или архивы? 415
Требования к облачному окружению на перспективу 416
Целевые значения показателей сохранности и доступности данных для SRE 417
Сохранность данных — это средство, доступность данных — цель 417
Создаем систему восстановления, а не систему резервного копирования 418
Типы сбоев, которые ведут к потере данных 418
Сложности поддержания высокой сохранности данных 420
Как служба SRE справляется справляется с трудностями обеспечения сохранности данных 422
Двадцать четыре комбинации типов сбоев с точки зрения сохранности данных 422
Первый уровень: мягкое удаление 423

Второй уровень: резервные копии и связанные с ними методы восстановления ... 426
Основной уровень: репликация ... 429
От терабайта к экзабайту — это не просто «больше» ... 429
Третий уровень: раннее обнаружение ... 430
Знаем, что восстановление данных сработает ... 434
Примеры ... 436
Gmail — февраль 2011 года: восстановление с помощью GTape ... 436
Google Music — март 2012 года: определение бесконтрольного удаления данных ... 438
Общие принципы SRE, применяемые для сохранности данных ... 444
Мышление новичка ... 444
Доверяй, но проверяй ... 444
Надежда — плохая стратегия ... 444
Глубокая защита ... 444
Итоги главы ... 445

Глава 27. Надежный масштабируемый выпуск продукта ... 447
Управление запусками ... 449
Роль инженера — координатора запуска ... 450
Настройка процесса запуска ... 450
Чек-лист для запуска ... 451
Сходимость и простота ... 452
Запускаем неожиданное ... 453
Разрабатываем чек-лист для запуска ... 453
Архитектура и зависимости ... 454
Интеграция ... 454
Планирование производительности ... 455
Типы сбоев ... 455
Поведение клиента ... 456
Процессы и автоматизация ... 456
Процесс разработки ... 457
Внешние зависимости ... 457
Планирование отправок ... 458
Избранные приемы выполнения надежных запусков ... 459
Постепенные отправки и отправки в несколько этапов ... 459
Фреймворки флагов функциональности ... 460
Справляемся со злоупотреблениями клиента ... 462
Поведение при перегрузке и нагрузочные тесты ... 463
Развитие службы LCE ... 464
Эволюция чек-листа LCE ... 465
Проблемы, которые инженеры LCE не решили ... 466
Итоги главы ... 467

Часть IV. Управление

Глава 28. Ускоренное обучение SR-инженеров для работы на дежурствах и не только 472
Вы наняли новых SR-инженеров, что теперь? 472
Первый тренинг: структурировать хаос 475
Кумулятивные и методичные пути обучения 476
Целевая, а не черновая работа над проектом 478
Подготовка выдающихся реверс-инженеров, умеющих импровизировать 479
Обратное проектирование: узнаем, как работают системы 480
Статистика и сравнение: последователи научного метода под давлением 480
Импровизация: когда случается неожиданное 481
Связываем все воедино: обратное проектирование сервиса, находящегося в промышленной эксплуатации 481
Пять приемов для вдохновления дежурных работников 482
Охотники за сбоями: чтение отчетов о происшествиях и обмен ими 483
Катастрофа: ролевая игра 484
Ломаем и чиним по-настоящему 485
Документация как обучение 486
Периодически проводите теневые дежурства 487
На дежурстве и не только: обряд перехода и практика постоянного обучения 488
Итоги главы 489

Глава 29. Справляемся с отвлекающими факторами и прерываниями 490
Управляем операционной нагрузкой 491
Факторы, определяющие обработку поступающих запросов 492
Неидеальные машины 492
Состояние когнитивного потока 493
Делаем хорошо что-то одно 494
Серьезно, скажите, что мне делать 496
Уменьшение количества прерываний 498

Глава 30. Добавляем в команду нового SR-инженера, чтобы предотвратить операционную перегрузку 500
Фаза 1: изучаем сервис и рабочее окружение 501
Определите главные источники стресса 502
Определите очаги возгорания 502
Фаза 2: делимся контекстом 503
Напишите для команды хороший постмортем 503
Сортируйте перегрузки в соответствии с их типами 504
Фаза 3: навязываем изменения 504
Начните с основ 505
Получите помощь в исправлении ошибок 505
Обосновывайте свои аргументы 505
Задавайте наводящие вопросы 506
Итоги главы 507

Глава 31. Общение и взаимодействие в службе SRE 508
Общение: производственные совещания 509
Повестка дня 510
Посещение 512
Взаимодействия внутри службы SRE 513
Структура команды 514
Приемы эффективной работы 514
Пример сотрудничества SR-инженеров: Viceroy 515
Пришествие Viceroy 515
Сложности 518
Рекомендации 519
Взаимодействие за пределами службы SRE 520
Пример: переход с DFP на F1 521
Итоги главы 524
Глава 32. Развитие модели вовлеченности SR-инженеров 525
Вовлеченность SR-инженеров: что, как и почему 525
Модель PRR 526
Модель вовлечения SR-инженеров 526
Альтернативная поддержка 527
Проверка готовности продукта: простая модель PRR 528
Вовлечение 529
Анализ 529
Улучшения и рефакторинг 530
Обучение 531
Освоение 531
Непрерывное улучшение 531
Развиваем простую модель PRR: раннее вовлечение 532
Кандидаты для раннего вовлечения 533
Преимущества модели раннего вовлечения 533
Развитие процесса разработки сервисов: фреймворки и платформа SRE 535
Усвоенные уроки 535
Внешние факторы, влияющие на SRE 536
На пути к структурному решению: фреймворки 536
Преимущества новой модели обслуживания и управления 538
Итоги главы 540

Часть V. Выводы

Глава 33. Полезные уроки из других отраслей 543
Познакомьтесь с нашими участниками 544
Готовность к катастрофам и их тестирование 546
Постоянная концентрация на безопасности 546
Внимание к деталям 547
Мгновенная производительность 547

Симуляции и прогоны....547
Тренировка и сертификация....548
Концентрация на сборе детальных требований и проектировании....548
Всесторонняя и глубокая защита....549
Культура написания постмортемов....549
Автоматизация и снижение служебной нагрузки....551
Структурированное и рациональное принятие решений....553
Итоги главы....554

Глава 34. Заключение....556

Приложения

Приложение А. Таблица доступности....560

Приложение Б. Практические рекомендации для сервисов в промышленной эксплуатации....561
Не бойтесь неудач....561
Постепенное выведение на рынок....562
Определяйте целевые значения с точки зрения пользователя....562
Бюджет ошибок....563
Мониторинг....563
Постмортемы....564
Планирование производительности....564
Перегрузки и отказы....565
Команды SRE....565

Приложение В. Пример документа о происшествиях....567

Приложение Г. Пример постмортема....569
Полученные уроки....571
Вспомогательная информация....573

Приложение Д. Список действий для координации запуска....574

Приложение Е. Пример протокола рабочего совещания....577

Об авторах....579

Библиография....580

Предисловие Марка Берджеса

История Google — это история роста. Это одна из величайших историй успеха в компьютерной индустрии, ознаменовавшая переход к IT-ориентированному бизнесу. Google в числе первых определила, что на практике означает союз бизнеса и IT-технологий, а также популяризовала концепцию DevOps — сращивание процессов разработки продуктов и информационно-технологического обслуживания. Данную книгу написали люди, воплотившие этот переход в реальность.

Компания Google развивалась в те времена, когда пересматривалась традиционная роль системного администратора. Google поставила под вопрос саму идею системного администрирования, говоря: «Мы не можем позволить себе подчиняться традициям, нам нужно придумать что-то новое, и у нас нет времени ждать, пока остальные нас догонят». Во введении к книге *Principles of Network and System Administration* [Burgess, 1999] я утверждал, что системное администрирование — человеко-машинная технология[1]. Некоторые критики резко отреагировали на эту идею, заявив, что «мы еще не достигли уровня развития, позволяющего называть это технологией». Уже тогда у меня было чувство, что вся эта отрасль зашла в тупик, заперта в заколдованном круге своей культуры, и я не видел пути вперед.

Компания Google сделала решительный шаг, разорвав этот круг и превратив грядущее в настоящее. Она изменила взгляд на саму роль системного администратора. Теперь название этой должности звучит так: *«инженер по обеспечению надежности информационных систем»* (Site Reliability Engineer, SRE). Некоторые мои друзья были в числе первых инженеров нового поколения; они формализовали эту роль с помощью специального ПО и средств автоматизации. Поначалу они

[1] В исходном тексте книги human-computer engineering. — *Примеч. пер.*

не распространялись о том, чем занимаются, а SRE-процессы, происходившие внутри Google и вовне, сильно отличались друг от друга: опыт Google был уникален. Но со временем методы работы Google становились известны остальным. Эта книга — шаг к тому, чтобы приподнять завесу над образом мышления SR-инженеров.

Из книги вы узнаете не только о том, как Google построила свою легендарную инфраструктуру, но и о том, как эта компания развивалась, училась и постепенно меняла свой взгляд на инструменты и технологии. Если мы будем открыты новому, то тоже сможем с легкостью справляться с самыми нетривиальными вызовами. Клановость IT-культуры зачастую тормозит всю отрасль. Если Google смогла преодолеть эту инерцию, то сможем и мы.

Эта книга представляет собой сборник статей, написанных специалистами из одной компании, разделяющими в общем одну точку зрения. Характерно, что все изложение строится вокруг единой цели компании. Некоторые темы и «персонажи» (программные системы) фигурируют в нескольких главах. При этом мы стараемся рассмотреть тему с разных сторон и в соответствии с различными интересами. Статьи (из них и состоит книга) похожи на личные блоги, они написаны в разных стилях и отражают взгляды разных специалистов, каждый из которых обладает своим набором навыков. Они написаны смело и честно, что необычно для большинства книг нашей отрасли. В некоторых статьях говорится «никогда не делайте так, всегда делайте так», другие же более абстрактны и философичны, что отражает разнообразие позиций их авторов в рамках IT-культуры, и это также играет свою роль в истории. Мы, в свою очередь, читаем их как скромные сторонние наблюдатели, которые не преодолевали этот путь и не располагают всей информацией о множестве противоречивых проблем и компромиссов. Именно вопросы таких наблюдателей — то, что прежде всего будет вынесено из этой книги. Почему они не сделали *X*? Что было бы, сделай они *Y*? Как мы посмотрим на эту книгу многие годы спустя? Сравнивая наши собственные идеи с информацией, предоставленной в книге, мы можем оценить собственные мысли и опыт.

Самое впечатляющее в этой книге — ее жизненность. Сегодня мы часто слышим безапелляционное: «Просто покажите мне код». Вокруг открытого исходного кода выросла культура «не задавай вопросов», где сообщество и принцип совместной разработки берут верх над профессионализмом[1]. Google — это компания, которая рискнула пересмотреть самые устои. Она нанимает самых талантливых людей, многие из которых имеют степень PhD. Инструментарий — лишь одно звено в цепи наряду с ПО, сотрудниками и данными. В книге нет универсальных решений задач, но в этом и вся суть. Подобные истории ценятся гораздо больше, чем конечный результат в виде кода или готовых программных продуктов. Реализация эфемерна,

[1] Это слишком категорично и однобоко. Инициатива Open Source, сообщества разработчиков, качество программного продукта — связь между ними, конечно, есть, но она не столь простая и однозначная. Закрытые «фирменные» разработки тоже, к сожалению, не всегда гарантируют качество. — *Примеч. пер.*

а задокументированное обоснование бесценно. Мы редко имеем доступ к подобным откровениям.

Эта книга — история успеха одной компании. Многие пересекающиеся истории показывают нам, что рост — это нечто гораздо большее, чем механическое увеличение классической вычислительной архитектуры. Это масштабирование бизнес-процессов, а не просто наращивание парка техники. Уже сам по себе этот урок стоит своего места на бумаге.

Мы не хотим втягиваться в самокритический обзор мира IT. Собственно, миру IT вообще свойственно повторять и заново изобретать решения. Многие годы проводилась лишь одна конференция, на которой обсуждалась инфраструктура IT, — USENIX LISA, и несколько конференций по операционным системам. И даже сейчас, когда ситуация значительно изменилась, эта книга остается бесценным даром: в ней детально задокументирован путь Google в переломную эпоху. Описанное не нужно копировать (хотя оно и может быть образцом для подражания) — книга должна вдохновить нас сделать свой следующий шаг. На ее страницах вы увидите множество правдивых примеров, демонстрирующих как превосходство, так и скромность. Вы увидите истории о надежде, страхе, успехах и провалах. Я приветствую мужество авторов и редакторов, позволивших рассказывать истории настолько откровенно, чтобы все, кто не является частью этого процесса, тоже смогли извлечь пользу из уроков, полученных в стенах Google.

Марк Берджес,
автор книги In Search of Certainty
Осло, март 2016

Предисловие авторов

Программное обеспечение (ПО) во многом подобно ребенку: для его *появления* на свет необходимо пройти через трудности и боль, но *после* его рождения усилий потребуется еще больше. Сегодня в обсуждении разработки ПО первому периоду уделяется намного больше внимания, несмотря на то что от 40 до 90 % затрат приходится на второй — *после* его выпуска[1]. Распространенное мнение о том, что развернутое и работающее ПО «стабильно» и, как результат, не требует пристального внимания разработчиков, неверно. Следовательно, если разработка программного обеспечения обычно сосредоточена на проектировании и построении программных систем, должна существовать и другая дисциплина, предметом которой будет *весь* жизненный цикл программных продуктов: их создание, развертывание, функционирование, обновление и в свое время — безболезненный вывод из эксплуатации. Эта дисциплина обращается — и должна обращаться — к широкому спектру навыков, но цель их применения иная, чем у других технических дисциплин. Такую дисциплину в Google называют *обеспечением надежности информационных систем* (Site Reliability Engineering, SRE).

Что же включает в себя понятие «обеспечение надежности информационных систем» (SRE)? Мы признаем, что название недостаточно хорошо отражает суть работы, — практически у каждого инженера, обеспечивающего надежность в Google, периодически спрашивают, что это такое и чем он занимается.

---

[1] Само по себе наличие такого большого разрыва может кое-что сказать нам о программной инженерии как о дисциплине. Для получения более подробной информации вы можете обратиться к [Glass, 2002].

Объясняя термин, стоит сказать, что специалисты SRE прежде всего *инженеры*. Мы используем принципы информатики и инженерии, чтобы проектировать и разрабатывать компьютерные системы, как правило, крупные и распределенные. Иногда нам приходится писать ПО для таких систем в тесном контакте с разработчиками программных продуктов. Иногда мы должны реализовать все служебные модули данной системы, вроде резервного копирования или балансировки нагрузки, при этом в идеале эти фрагменты должны быть пригодны для использования и в других системах. А иногда наша задача — выяснить, как применять существующие решения для решения новых задач.

Далее, мы заботимся об обеспечении *надежности* системы. Бен Трейнор Слосс, вице-президент Google и автор термина SRE, утверждает, что надежность — это наиболее важная характеристика любого продукта: система не будет успешной, если с ней невозможно работать! И поскольку надежность[1] настолько важна, специалисты SRE трудятся над тем, чтобы найти способы улучшить дизайн и функционирование систем в попытке сделать их более масштабируемыми, надежными и эффективными. Однако мы работаем в этом направлении только до определенного момента: когда система считается «достаточно надежной», мы переносим свое внимание на добавление новых функций или создание новых продуктов[2].

Наконец, SR-инженеры занимаются поддержанием работы *сервисов*, построенных на базе наших распределенных вычислительных систем, независимо от того, являются они хранилищами планетарных масштабов, сервисом электронной почты для сотен миллионов пользователей или поисковиком, с которого и началась история Google. Слово «сайт» в названии поначалу говорило, что мы поддерживали работу сайта google.com, однако сейчас у нас запущено гораздо больше сервисов, многие из которых не являются сайтами, — начиная с внутренней инфраструктуры вроде сервиса Bigtable и заканчивая продуктами для внешних разработчиков наподобие Google Cloud Platform.

Хотя мы преподносим SRE как универсальную дисциплину, ее появление именно в быстрорастущем мире веб-сервисов вполне закономерно. Возможно также, что ее зарождению способствовали и особенности нашей инфраструктуры. Неудивительно,

---

[1] В данном контексте надежность — это «вероятность того, что система будет выполнять требуемые функции без сбоев при заданных условиях в заданный период времени», согласно определению, приведенному в [O'Connor, 2012].

[2] Как правило, мы работаем над сайтами и другими подобными сервисами; мы не рассматриваем вопрос надежности ПО для атомных электростанций, авиационных систем, медицинского оборудования или иных систем, для которых критически важна безопасность. Однако в главе 33 мы сравниваем свой подход с подходами, используемыми в других областях.

что из всех характеристик ПО, которым мы можем уделить внимание после развертывания, надежность является одной из основных[1]. Веб-сервисы — как раз та сфера, где наиболее актуален наш подход, поскольку процесс обновления и совершенствования серверного ПО относительно самодостаточен, а процесс управления этими изменениями тесно связан со сбоями всех видов.

Несмотря на то что эта дисциплина появилась в Google и в веб-сообществе в целом, мы считаем, что наш опыт применим и в других компаниях и организациях. В этой книге мы пытаемся объяснить свой подход как для того, чтобы другие организации смогли использовать наши наработки, так и для того, чтобы лучше определить роль и значение SRE. Для этого мы структурировали книгу таким образом, чтобы общие принципы и более конкретизированные практики были по возможности отделены друг от друга. Кроме того, там, где это уместно, при рассмотрении тех или иных тем мы приводим примеры из практики Google. Мы верим, что читатель встретит это с пониманием и сможет сделать полезные для себя выводы.

Мы также предоставили вспомогательный материал — описание производственной среды Google и соответствия между нашим внутренним ПО и общедоступным, что должно помочь вам воспринимать весь изложенный материал в понятном вам контексте и применять свои знания на практике.

Наконец, нужно сказать, что создание ПО и разработка систем с учетом повышенной надежности — однозначное благо. Однако мы понимаем, что в небольших организациях могут задаваться непростым вопросом: как им лучше применить знания, полученные из этой книги? Это похоже на вопрос безопасности — чем раньше вы им займетесь, тем лучше. Даже если ваша небольшая организация имеет множество насущных проблем и ваше ПО отличается от выбранного в Google, вам все равно стоит заранее позаботиться о команде SRE, поскольку в этом случае последующее расширение структуры обойдется дешевле, чем построение ее с нуля. В части IV приведены практические рекомендации для обучения, коммуникации и проведения совещаний, опробованные у нас. Многие из них вы могли бы применить и для своей компании.

В компаниях среднего размера, скорее всего, уже имеется сотрудник, выполняющий функции SR-инженера (хотя, возможно, его должность называется как-то иначе). Поэтому путь к улучшению надежности ПО в такой организации — официально обозначить эту работу или найти людей, уже выполняющих ее, и побуждать их к ее дальнейшему выполнению, вознаграждая соответствующим образом. Эти люди стоят на границе разных взглядов на мир, подобно Ньютону, которого иногда называют не первым в мире физиком, а последним алхимиком.

---

[1] В этом мы отличаемся от направления DevOps: хотя мы рассматриваем инфраструктуру как код, нашей главной целью является надежность. Вдобавок мы стремимся к тому, чтобы избавляться от необходимости в отдельной службе операционной поддержки — для получения более подробной информации прочтите главу 7.