

Краткое содержание

Отзывы о книге «Совершенный софт»	19
Предисловие.....	24
Об авторе	34
Глава 1. Метод	36

ЧАСТЬ I ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Глава 2. Декомпозиция	46
Глава 3. Структура	91
Глава 4. Композиция	121
Глава 5. Пример проектирования системы	133

ЧАСТЬ II ПЛАНИРОВАНИЕ ПРОЕКТА

Глава 6. Мотивация.....	178
Глава 7. Обзор планирования проекта	184
Глава 8. Сеть и временные резервы	236
Глава 9. Время и затраты	250
Глава 10. Риск.....	278
Глава 11. Планирование проекта в действии	298

Глава 12. Расширенные методы планирования	352
Глава 13. Пример планирования проекта	382
Глава 14. Завершение.....	410

ПРИЛОЖЕНИЯ

Приложение А. Отслеживание проекта	436
Приложение Б. Проектирование контрактов сервисов	456
Приложение В. Стандарт проектирования	474

Оглавление

Отзывы о книге «Совершенный софт»	19
Предисловие	24
Структура книги	28
Часть I. Проектирование системы	28
Часть II. Планирование проекта.....	29
Приложения	31
Кто вы?	31
Что необходимо для работы с книгой	32
Дополнительные онлайн-ресурсы	32
Благодарности.....	33
Об авторе.....	34
От издательства	34
Глава 1. Метод	36
Что такое «Метод»?.....	37
Проверка результата проектирования	38
Аврал	39
Преодоление аналитического паралича	40
Обмен информацией	42
Чем Метод не является	43

ЧАСТЬ I ПРОЕКТИРОВАНИЕ СИСТЕМЫ

Глава 2. Декомпозиция	46
О вреде функциональной декомпозиции	47
Недостатки функциональной декомпозиции	47
Слишком много или слишком мало	48
О природе Вселенной (TANSTAAFL)	53
Антипроектирование	54
Недостатки декомпозиции предметной области	56
Построение дома с декомпозицией предметной области	57
Ошибочная мотивация	58
Тестируемость и проектирование	59
Пример: функциональная торговая система	62
Проблемы функциональной торговой системы	63
Декомпозиция на основе неустойчивости	65
Декомпозиция, сопровождение и разработка	67
Универсальный принцип	68
Декомпозиция на основе неустойчивости и тестирование	69
Проблемы неустойчивости	70
Проблема 2%	70
Эффект Даннинга–Крюгера	72
Выявление неустойчивости	72
Неустойчивость и изменчивость	73
Оси неустойчивости	73
Разбиение в ходе проектирования	74
Пример: декомпозиция дома на основе неустойчивости	75
Решения, замаскированные под требования	76
Список неустойчивостей	78
Пример: торговая система на основе неустойчивости	78
Ключевое наблюдение	81
Как противостоять зову сирен	84
Умозрительное проектирование	86
Проектирование для конкурентов	87
Неустойчивость и долговечность	88
О важности практики	89

Глава 3. Структура	91
Сценарии использования и требования	92
Требуемое поведение.....	92
Многоуровневый подход.....	95
Использование сервисов	95
Типичные уровни	96
Клиентский уровень	97
Уровень бизнес-логики.....	98
Уровень доступа к ресурсам	100
Использование атомарных бизнес-команд.....	100
Уровень ресурсов.....	101
Вспомогательные средства	101
Рекомендации по классификации	101
«Что в имени тебе моем»	102
Четыре вопроса.....	103
Соотношение Менеджеров и Ядер	104
Ключевые наблюдения.....	105
Убывание нестабильности сверху вниз	105
Возможность повторного использования возрастает сверху вниз	106
Почти расходные Менеджеры.....	106
Подсистемы и сервисы	107
Инкрементное построение.....	107
О микросервисах.....	110
Открытые и закрытые архитектуры	112
Открытая архитектура.....	112
Полузакрытая/полуоткрытая архитектура	113
Ослабление правил	114
Вызов вспомогательных средств.....	115
Запреты проектирования.....	117
Стремление к симметрии.....	119
Глава 4. Композиция	121
Требования и изменения	121
Соппротивление изменениям	122
Главная директива проектирования	122

Компоновочное проектирование.....	123
Базовые сценарии использования	124
Задача архитектора.....	125
Проверка архитектуры	125
Наименьший набор.....	128
Никаких функций нет	130
Обработка изменений.....	131
Ограничение изменений.....	131

Глава 5. Пример проектирования системы 133

Обзор системы	134
Унаследованная система	135
Новая система.....	137
Компания	137
Сценарии использования.....	138
Базовый сценарий использования	142
Упрощение сценариев использования	143
Антипроектирование	143
Монолит	144
Детализированные структурные элементы	144
Декомпозиция предметной области.....	147
Ориентация на потребности бизнеса	147
Концепция.....	148
Бизнес-цели	149
Формулировка миссии	150
Архитектура	151
Глоссарий TradeMe	151
Области нестабильности в TradeMe	152
Слабые нестабильности.....	155
Статическая архитектура.....	156
Оперативные концепции	159
Сообщения и приложения	160
Менеджеры потоков операций	162
Проверка архитектуры	164
Добавление мастера/подрядчика	164
Запрос мастера	166

Подбор мастера.....	167
Назначение мастера.....	170
Завершение найма	172
Оплата мастера.....	174
Создание проекта.....	174
Закрытие проекта.....	175
Что дальше?	176

ЧАСТЬ II ПЛАНИРОВАНИЕ ПРОЕКТА

Глава 6. Мотивация 178

Зачем нужно планирование проекта?	178
Планирование и адекватность проекта.....	180
Инструкции по сборке	180
Иерархия потребностей.....	180

Глава 7. Обзор планирования проекта 184

Определение успеха.....	184
Как сообщать об успехе	185
Исходный подбор персонала для проекта.....	186
Архитектор, а не архитекторы.....	186
Младшие архитекторы.....	188
Основная команда.....	188
Обоснованные решения.....	191
Планы, а не план	191
Критический анализ плана разработки программного продукта	192
Сервисы и разработчики.....	193
Проектирование и эффективность команды	195
Непрерывность задач.....	197
Оценка трудозатрат.....	197
Классические ошибки.....	198
Вероятность успеха	199
Методы оценки.....	200
Общая оценка проекта	202
Оценки отдельных активностей.....	204
Анализ критического пути	206

Сетевой график проекта.....	207
Критический путь.....	210
Назначение ресурсов	211
Численность персонала	212
Планирование активностей	217
Распределение кадров	217
Сглаживание кривой	224
Затраты на реализацию проекта.....	224
Эффективность как общая оценка.....	227
Планирование освоенной ценности	228
Классические ошибки.....	230
Пологая S-образная кривая.....	232
Роли и обязанности	234
Глава 8. Сеть и временные резервы	236
Сетевая диаграмма.....	236
Узловая диаграмма.....	237
Стрелочная диаграмма	237
Стрелочные и узловые диаграммы	238
Временные резервы.....	241
Общий временной резерв.....	241
Свободный временной резерв	242
Вычисление временного резерва.....	244
Наглядное представление временных резервов	244
Активное управление проектами	246
Составление графика на основе временных резервов	247
Временной резерв и риск	249
Глава 9. Время и затраты	250
Ускорение программных проектов.....	250
Уплотнение графика.....	254
Использование лучших ресурсов	254
Параллельная работа	254
Параллельная работа и затраты.....	257
Кривая зависимости затрат от времени	257
Точки на кривой «время-затраты»	258

Дискретное моделирование.....	260
Предотвращение классических ошибок	261
Реализуемость проекта	262
Поиск нормальных решений	264
Составляющие затрат проекта.....	266
Прямые затраты	266
Косвенные затраты	266
Бухгалтерия и ценность	267
Снова о мертвой зоне.....	267
Уплотнение и составляющие затрат	268
Нормальное решение и минимальные общие затраты	269
Персонал и составляющие затрат.....	272
Фиксированные затраты.....	274
Уплотнение сети.....	274
Поток уплотнения	275
Глава 10. Риск	278
Выбор вариантов.....	278
Кривая «время-риск»	279
Реальная кривая «время-риск»	280
Моделирование риска.....	282
Нормализация риска	282
Риски и временные резервы	283
Проектные риски.....	283
Риск и прямые затраты.....	284
Риск возникновения критичности	284
Настройка риска возникновения критичности	287
Риск Фибоначчи	287
Риск активности и риск возникновения критичности	291
Уплотнение и риск.....	292
Риск выполнения.....	293
Разуплотнение риска.....	293
Как выполняется разуплотнение	294
Цель разуплотнения.....	294
Метрики риска.....	296

Глава 11. Планирование проекта в действии.....	298
Формулировка миссии	298
Статическая архитектура.....	299
Цепочки вызовов.....	299
Список активностей.....	303
Диаграмма сети.....	304
Предпосылки планирования	306
Поиск нормального решения	308
Неограниченные ресурсы (итерация 1)	308
Сеть и проблемы с ресурсами.....	310
Инфраструктура в начале (итерация 2).....	310
Ограниченность ресурсов.....	312
Переход на субкритический уровень (итерация 7).....	316
Выбор нормального решения	319
Уплотнение сети.....	320
Уплотнение с использованием лучших ресурсов.....	320
Параллелизм в работе.....	322
Завершение итераций уплотнения	331
Анализ результативности	332
Анализ эффективности	333
Кривая «время-затраты»	334
Корреляционные модели «время-затраты».....	336
Мертвая зона	337
Планирование и риски.....	339
Разуплотнение риска.....	340
Переработка кривой «время-затраты»	344
Моделирование риска	345
Включение и исключение рисков	348
Анализ SDP.....	349
Представление вариантов	350
Глава 12. Расширенные методы планирования.....	352
Божественные активности	352
Решение проблемы божественных активностей.....	353
Точка пересечения риска	354
Вычисление точки пересечения.....	354

Поиск цели для разуплотнения.....	358
Геометрический риск.....	360
Геометрический риск возникновения критичности	361
Геометрический риск активности	363
Поведение геометрического риска	364
Сложность исполнения	366
Цикломатическая сложность	366
Тип проекта и сложность	367
Уплотнение и сложность	368
Очень большие проекты	370
Сложные системы и непрочность	371
Факторы сложности.....	372
Сеть сетей.....	374
Проектирование сети сетей.....	374
Малые проекты	378
Планирование по уровням.....	378
Достоинства и недостатки.....	379
Уровни и построение.....	381
Глава 13. Пример планирования проекта	382
Оценки	383
Оценки отдельных активностей.....	383
Общая оценка проекта	386
Зависимости и сеть проекта	386
Поведенческие зависимости.....	387
Зависимости, не связанные с поведением	388
Переопределение некоторых зависимостей.....	388
Проверки на здравый смысл.....	389
Нормальное решение	389
Диаграмма сети.....	390
Планируемый прогресс.....	392
Планируемое распределение комплектования.....	392
Затраты и эффективность	393
Сводка результатов.....	394
Уплотненное решение	394
Добавление вспомогательных активностей	394

Оценка продолжительности для Менеджеров.....	396
Назначение ресурсов	396
Планируемый прогресс.....	396
Планируемое распределение комплектования.....	396
Затраты и эффективность	398
Сводка результатов	398
Планирование по уровням	399
Планирование по уровням и риск.....	400
Распределение комплектования	400
Сводка результатов.....	401
Субкритическое решение.....	401
Продолжительность, планируемый прогресс и риск	401
Затраты и эффективность	402
Сводка результатов.....	402
Сравнение вариантов	403
Планирование и риск	404
Разуплотнение риска.....	404
Повторное вычисление затрат.....	407
Подготовка к анализу SDP	408
Глава 14. Завершение	410
Когда планировать проект.....	410
Настоящий ответ	411
Как добиться успеха в жизни	412
Общие рекомендации	414
Архитектура и оценки	415
Отношение к планированию.....	415
Альтернативность	416
Уплотнение	417
Планирование и риск	419
Реализация планирования проекта.....	420
Перспективы	422
Подсистемы и график.....	423
Из рук в руки.....	424
Джуниоры	425

Сеньоры	425
Сеньоры в роли архитекторов-джуниоров	426
На практике	427
Подведение итогов планирования проекта	428
О качестве.....	430
Активности контроля качества	431
Активности обеспечения качества	432
Качество и культура.....	433

ПРИЛОЖЕНИЯ

Приложение А. Отслеживание проекта 436

Жизненный цикл и статус активностей	437
Критерий выхода из фазы	439
Вес фазы	440
Статус активности	441
Статус проекта	442
Прогресс и осваиваемая ценность	442
Накапливаемый объем работы	443
Накапливаемые косвенные затраты	444
Отслеживание прогресса и объема работ	444
Прогнозирование.....	445
Прогнозирование и корректировка	447
Все хорошо	448
Заниженная оценка	449
Утечка ресурсов	451
Завышенная оценка	452
Подробнее о прогнозах	453
Неконтролируемое расширение масштаба проекта.....	454
Формирование доверия	454

Приложение Б. Проектирование контрактов сервисов 456

Оценка качества проектирования	456
Модульность и затраты.....	458
Затраты на сервис.....	459
Затраты на интеграцию.....	459
Область минимальных затрат	460

Сервисы и контракты.....	461
Контракты как грани	461
От проектирования сервиса к проектированию контрактов	462
Атрибуты хороших контрактов	462
Построение контрактов	464
Пример проектирования.....	465
Нисходящий рефакторинг	466
Горизонтальный рефакторинг	466
Восходящий рефакторинг	468
Метрики проектирования контрактов	469
Метрики контрактов.....	469
Метрики размера.....	469
Избегайте свойств	470
Ограничение количества контрактов.....	471
Использование метрик	472
Трудности проектирования контрактов.....	472
Приложение В. Стандарт проектирования	474
Главная директива	474
Директивы.....	475
Рекомендации по проектированию системы.....	475
Рекомендации по планированию проектов	477
Рекомендации по отслеживанию проектов	479
Рекомендации по проектированию контрактов сервисов	480

Отзывы о книге «Совершенный софт»

Я посещал как мастер-класс для архитекторов, так и мастер-класс по планированию проектов. До этого я почти потерял всякую надежду, что мне когда-нибудь удастся понять, почему работа моей команды никогда не приводит к успешному завершению, и лихорадочно пытался найти работоспособное решение, которое бы остановило нашу безумную гонку на выживание. Мастер-классы открыли передо мной мир, в котором разработка ПО поднимается на уровень всех остальных инженерных дисциплин и организуется профессионально, предсказуемо и надежно, что приводит к разработке высококачественных программных продуктов в рамках срока и бюджета. Полученные знания были бесценными! От создания основательной, проработанной архитектуры, которая может устоять перед вечно изменяющимися требованиями пользователей, до внутренних подробностей планирования и направления проекта к успешному завершению — все это было представлено с непревзойденным профессионализмом и компетентностью. А если учесть, что каждое слово истины, которым Джувел поделился с нами на мастер-классах, было получено, проверено и подтверждено реальной жизнью, этот усвоенный опыт превратился в мощный комплекс знаний, абсолютно необходимый для каждого, кто стремится стать архитектором в области программных систем.

*Россен Томев (Rossen Totev),
программный архитектор/руководитель проектов*

Мастер-класс по планированию проектов стал событием, изменившим мою карьеру. Я вырос в среде, в которой ограничения по срокам и бюджетам нарушались патологически, так что возможность учиться у Джувела стала даром небес. Шаг за шагом он представлял компоненты и необходимые инструменты для правильного планирования проектов. Это позволяет держать под контролем затраты и сроки в динамической и даже хаотичной среде современной разработки программных продуктов. Джувел говорит, что вы вступаете в асимме-

тричную войну против превышения затрат и сроков, а после его мастер-классов у вас появляется ощущение, что вступаєте в рукопашный бой с пистолетом. Никакого волшебства в этом нет — только применение базовых инженерных и производственных доктрин в области ПО, но, вернувшись в офис, вы будете чувствовать себя настоящим волшебником.

*Мэмм Роболд (Matt Robold),
руководитель разработки ПО, West Covina Service Group*

Фантастический опыт. Навсегда изменил мой образ мышления относительно того, как следует подходить к разработке программных продуктов. Я всегда знал, что какая-то часть моих ощущений относительно проектирования и программирования была правильной. Просто я не умел выразить ее словами, зато теперь эти слова у меня появились. Новые знания повлияли на мой подход не только к проектированию программных продуктов, но и к другим видам проектирования.

*Ли Мессик (Lee Messick),
ведущий архитектор*

Программный проект, над которым я работаю, годами страдал от невысказанно жестких сроков. Попытки разобраться в методологиях разработки и сформировать правильный процесс казались напрасной тратой сил, потому что помимо неразумных требований заказчиков приходилось бороться с нежеланием руководства идти на какие-либо компромиссы. Я сражался на двух фронтах, и поражение казалось неизбежным. Мастер-класс предоставил четкое видение ситуации, о котором я никогда не подозревал. Я получил именно те знания, которые искал. Освоил фундаментальные методы, которые изменили мое понимание того, как работают программные проекты. Теперь у меня есть инструменты для эффективного управления моими проектами в потоке бесконечно изменяющихся требований. В мире хаоса этот мастер-класс навел порядок. Я бесконечно благодарен IDesign. Моя жизнь никогда не будет прежней.

*Аарон Фридман (Aaron Friedman),
архитектор*

Жизнь изменилась. Я чувствую себя как хорошо настроенное пианино, на котором уже пару десятилетий копилась пыль.

*Джордан Ян (Jordan Jan),
технический директор/архитектор*

Курсы были просто фантастическими. Пожалуй, это была самая напряженная, но и самая плодотворная неделя моей профессиональной жизни.

Стоил Панков (Stoil Pankov), архитектор

Обучение у Джувела Лёве изменило мою жизнь. Я прошел путь от простого разработчика до настоящего архитектора, применяющего инженерные принципы из других дисциплин к проектированию не только программных продуктов, но и своей карьеры.

Кори Торгерсен (Kory Torgersen), архитектор

Мастер-класс для архитекторов — настоящий урок жизни из области проектирования и квалификации, который я прошел дважды. Знания преподносились на совершенно новом уровне, и я жалею, что не прошел это обучение десятки лет назад, в самом начале своей карьеры. «Перепрошивать мозг» и забывать то, что я знал ранее, неприятно, но я должен был пройти через это. Наконец, каждый последующий день я размышляю над тем, что сказал Джувел на мастер-классе, и пользуюсь новыми знаниями, помогая своей команде даже в мелочах — чтобы со временем все мы могли называть себя Профессиональными Инженерами. (P.S. При втором прохождении я написал более 100 страниц заметок!)

*Джейсу Джеячандрат,
руководитель разработки ПО, Nielsen*

Если вы расстроены, творческой энергии не осталось, если вы чувствуете отчаяние, видя множество неудач в нашей отрасли, этот мастер-класс вдохновит вас. Он поднимет вас на новый уровень профессиональной зрелости, а также придаст надежды и уверенности в том, что вы сможете правильно применить новые знания. Вы выйдете с мастер-класса по планированию проектов с новым образом мышления и достаточным количеством полезных инструментов, с которыми любая неудача программного проекта станет непростительной. Вы тренируетесь, решаете практические задачи, получаете представление о сути вещей и получаете опыт. Да, когда приходит время сообщать ключевым участникам данные о затратах, времени и риске проекта, МОЖНО быть точным. Не ждите, пока ваша компания отправит вас на этот мастер-класс. Если вы серьезно относитесь к своей карьере, поскорее запишитесь на этот или любой другой мастер-класс IDesign. Это лучшее вложение в свою карьеру, которое можно сделать. Спасибо всей команде IDesign за ее непрестанную работу по превращению отрасли разработки в серьезную инженерную дисциплину.

Люциан Мариан (Lucian Marian), архитектор, Mirabel

Как человек, который находится в начале своей карьеры, я могу честно заявить, что этот учебный курс изменил мою жизнь и мое отношение к работе. Я искренне считаю, что он стал поворотной точкой моей жизни.

Алекс Карпович (Alex Karpowich), архитектор

Хочу поблагодарить вас за неделю, изменившую мою (профессиональную) жизнь. Обычно я не могу высиживать на учебных курсах более 50% времени — они занудны и не учат меня ничему, чего бы я не знал или не мог узнать сам. На мастер-классе для архитекторов на протяжении 9 часов каждый день мне не было скучно: я узнал о своих обязанностях как архитектора (я-то думал, что архитектор — всего лишь проектировщик программных продуктов), об инженерном аспекте разработки, о важности реализации не только к заданному времени, но и в рамках бюджета и качества. Я научился не ждать, пока из меня «вырастет» архитектор, а активно управлять своей карьерой, научился оценивать и давать объективную оценку тому, что прежде считал интуитивными инсайтами. После этой недели многое встало на свои места. Не могу дождаться следующего мастер-класса.

Итай Золберг (Itai Zolberg), архитектор

Моему отцу
Томасу Чарльзу (Томми) Лёве

Предисловие

Вряд ли хоть кто-то из нас пришел в разработку программного обеспечения не по своей воле. Многие буквально влюбляются в программирование и решают, что хотят зарабатывать им на жизнь. Тем не менее между радужными представлениями о карьере и темной, угнетающей реальностью разработки лежит пропасть. Отрасль программирования в целом переживает глубокий кризис. Этот кризис оказался особенно острым из-за своей многомерности; нарушен буквально каждый аспект разработки:

- **Затраты.** Бюджет, выделенный для проекта, слабо связан с фактическими затратами на разработку системы. Многие организации даже не пытаются решить проблему с затратами — возможно, потому, что просто не знают, как к ней подойти, или потому, что это заставит их признать, что работа над системой окажется экономически неоправданной. Даже если затраты на первую версию новой системы будут оправданы, часто затраты на протяжении срока жизни системы намного выше положенных из-за некачественного проектирования и неспособности адаптироваться к изменениям. Со временем затраты на сопровождение становятся настолько неприемлемыми, что компании обычно решают начать с чистого листа — только для того, чтобы в ближайшем будущем получить не менее, а то и более дорогостоящую кучу хлама вместо новой системы. Ни в одной другой отрасли не встречается регулярный перезапуск систем просто из-за того, что поддержка старой системы экономически нецелесообразна. Авиакомпании эксплуатируют лайнеры десятилетиями, а жилой дом может быть построен сотню лет назад.
- **График.** Дедлайны часто оказываются взятыми из головы и являются ничем не обеспеченными конструкциями, так как не имеют никакого отношения ко времени, которое реально потребуется для разработки системы. Для большинства разработчиков дедлайны — всего лишь бесполезные ориентиры, которые проносятся мимо в процессе работы. Если команда разработки

выдерживает дедлайн, все удивляются, ибо всегда ждут, что сроки будут сорваны. И это тоже является прямым результатом плохого проектирования системы, что приводит к каскадному распространению в ней новых изменений и новой работы, с которой ранее завершенная работа становится недействительной. Более того, это является результатом чрезвычайно неэффективного процесса разработки, который не учитывает зависимости между активностями и не пытается отыскать самый быстрый и безопасный способ построения системы. Мало того что время реализации всей системы становится неприемлемо большим — и время реализации отдельных функций тоже может увеличиваться. Нарушение сроков в проекте — достаточно плохо; но еще хуже, когда нарушение скрывается от руководства и заказчиков, потому что никто не имеет ни малейшего понятия об истинном состоянии проекта.

- **Требования.** В конечном итоге часто оказывается, что разработчики решают не те задачи. Между заказчиками или их внутренними посредниками (например, отделом маркетинга) и группой разработки существует постоянное недопонимание. Также многие разработчики не признают свою неспособность сохранить суть требований. Даже идеально переданные требования с большой вероятностью со временем изменятся. Такие изменения делают решение недействительным и ставят под удар все, что пыталась построить команда.
- **Персонал.** Даже среднестатистические программные системы настолько сложны, что человеческий мозг просто не способен в них полностью разобраться. Внутренняя и внешняя сложность является прямым результатом неудачной архитектуры системы; в свою очередь, она ведет к появлению запутанных систем, создающих массу сложностей в сопровождении, расширении и повторном использовании.
- **Сопровождение.** Часто сопровождением программных систем занимаются совсем не те люди, которые их разрабатывали. Новые сотрудники не понимают, как работает система, и в результате попытки решения старых проблем постоянно приводят к созданию новых. Все это ведет к стремительному росту затрат на сопровождение и времени вывода продукта на рынок, а также к отмене проектов или их перезапуску с чистого листа.
- **Качество.** Возможно, ничто не нарушается в программных системах так серьезно, как качество. В программных продуктах есть баги, и можно сказать, что они стали неотъемлемой частью разработки. Говоришь «ПО» — имеешь в виду «баги». Разработчики не могут представить себе программную систему, избавленную от багов. Исправление дефектов (как и добавление новых функций или просто сопровождение) часто увеличивает счетчик ошибок. Плохое качество является прямым результатом малопонятной системной

архитектуры, которая создает проблемы с тестированием и сопровождением. Не менее важно то, что многие проекты не учитывают важнейшие активности по контролю качества и не выделяют достаточно времени для того, чтобы каждая активность была завершена идеально.

Несколько десятилетий назад в отрасли начали разрабатываться продукты для решения мировых проблем. В наши дни разработка сама по себе стала проблемой мирового уровня. Проблемы разработки часто проявляются в нетехнических аспектах: рабочих средах с высоким уровнем стресса, высокой текучести кадров, нервном истощении, отсутствии доверия, низкой самооценке и даже в соматических заболеваниях.

Ни одна из проблем разработки не нова¹. Некоторые люди за всю свою карьеру в разработке ни разу не видели, чтобы продукт был сделан правильно. Они начинают верить, что это в принципе невозможно, и отвергают любые попытки решения проблем, потому что «уж так сложилось». Они даже могут оказывать сопротивление людям, пытающимся улучшить процесс разработки. Они уже решили для себя, что цель недостижима, поэтому каждый, кто пытается добиться лучших результатов, желает невозможного, а это оскорбляет их самолюбие.

Мой опыт является контрпримером, который доказывает, что успешная разработка программных систем возможна. Каждый проект, за который я отвечал, выпускался в пределах срока, бюджета и без дефектов. История продолжилась и после основания компании IDesign, в которой мы помогаем своим клиентам снова и снова успешно выполнять свои обязательства.

Эта последовательная, систематичная история успеха не была случайна. Я получал образование в области проектирования систем — как физических, так и программных, поэтому было нетрудно узнать сходство этих двух миров. Применение практических принципов — идей, которые считаются проявлением здравого смысла в других инженерных областях, — было оправданно и в области программных систем. Мне никогда не приходило в голову рассматривать разработку программного продукта как техническую задачу или разрабатывать систему без плана. Я не видел смысла идти на компромиссы со своими убеждениями или поступаться принципами, потому что правильный подход работал, а устрашающие последствия отхода от него были очевидны. Мне повезло с учителями; я оказался в нужном месте в нужное время, видел, что работает, а что не работает; мне представилась возможность поучаствовать в больших важных проектах и быть частью культуры высших достижений.

В последние годы я заметил, что проблемы отрасли усугубляются. Все больше программных проектов завершаются неудачей. Эти неудачи обходятся все до-

¹ Edsger W. Dijkstra. The Humble Programmer: ACM Turing Lecture. Communications of the ACM 15, no. 10 (October 1972): 859–866. (*Дейкстра Эдсгер*. Смиренный программист.)

роже по времени и деньгам, и даже завершённые проекты обычно отклоняются от своих исходных обязательств. Кризис усугубляется не только тем, что системы становятся все больше, не только жесткими сроками или более высокой частотой изменений. Подозреваю, настоящая причина заключается в том, что знания о том, как правильно проектировать и планировать программные системы, медленно уходят из команд разработчиков. Когда-то в большинстве команд присутствовал ветеран, который обучал молодых и передавал коллективные знания. В наши дни эти учителя перешли на другую работу или вышли на пенсию. В их отсутствие рядовые сотрудники располагают бесконечным количеством информации при нуле знаний.

Я бы очень хотел, чтобы кризис в области программирования мог быть решен каким-то одним способом: применением процесса, методологии разработки, инструмента или технологии. К сожалению, для решения многомерной задачи требуются многомерные решения. В этой книге предлагается комплексное решение: исправление процесса разработки.

По сути, все, что я предлагаю, — проектирование и разработка программных систем с применением инженерных принципов. К счастью, нет необходимости заново изобретать велосипед. Другие инженерные дисциплины были достаточно успешны, поэтому отрасль разработки программного обеспечения может позаимствовать ключевые универсальные принципы проектирования и адаптировать их для программных систем и проектов. Чтобы добиться успеха, необходимо принять инженерную точку зрения. Вы хотите, чтобы программная система была простой в сопровождении и расширении, экономичной, пригодной для повторного использования и реализуемой по времени и риску? Все это инженерные, а не технические аспекты. Они напрямую обусловлены проектированием системы и планированием проекта. Появился специальный термин «архитектор программного продукта» (или просто «архитектор»), которым обозначается участник команды, который отвечает за все аспекты проекта, связанные с проектированием. Соответственно я буду называть читателя «архитектором».

Идеи, представленные в книге, не затрагивают всего, что необходимо знать для достижения успеха. Тем не менее они безусловно станут хорошей отправной точкой, так как направлены на исправление корневой причины всех проблем, упоминавшихся ранее. Корневой причиной является некачественное проектирование самой программной системы или планирование проекта, используемого для построения этой системы. Вы увидите, что программный продукт вполне может быть реализован в пределах графика и бюджета и что планировать системы, удовлетворяющие всем мыслимым требованиям, реально. Такие системы также не создают проблем в сопровождении, расширении и повторном использовании. Надеюсь, что использование этих идей выведет на правильный путь не только систему, над которой вы работаете, но и вашу карьеру, и что оно снова разожжет вашу страсть к программированию.

Структура книги

В книге представлен структурированный инженерный подход к проектированию систем и планированию проектов. Методология состоит из двух частей, нашедших отражение в структуре книги: проектирование системы (создание архитектуры) и планирование проекта. Обе части взаимно дополняют друг друга и являются обязательными. Приложения дополняют основной материал.

Обычно в технической литературе каждая глава посвящена отдельной теме. Такой подход упрощает написание книги, но обычно не соответствует особенностям нашего процесса познания. Напротив, в этой книге процесс обучения напоминает спираль. В обеих частях книги в каждой главе мы возвращаемся к идеям предыдущих глав, погружаясь в материал или развивая идеи на основании дополнительной информации, охватывающей разные аспекты. Такая подача материала повторяет естественный процесс обучения. Каждая глава строится на материале предшествующих глав, поэтому читать главы стоит по порядку. В обе части книги включен подробный практический пример, демонстрирующий основные идеи, а также дополнительные аспекты. В то же время, чтобы итерации были более компактными, обычно я стараюсь избегать самоповторений, так что даже ключевые моменты обсуждаются только один раз.

Ниже приведено саммари глав и приложений.

Глава 1. Метод

В главе 1 представлена ключевая идея: чтобы добиться успеха, вы должны спроектировать систему и спланировать проект для ее построения. Обе составляющие крайне важны для успеха. Проект невозможно планировать без архитектуры, а строить систему, которую вы не сможете построить, бессмысленно.

Часть I. Проектирование системы

Глава 2. Декомпозиция

Глава 2 посвящена разложению системы на компоненты, образующие ее архитектуру. Большинство архитекторов выполняет декомпозицию систем худшим из всех возможных способов, поэтому глава начинается с объяснения того, чего делать не следует. Разобравшись с ошибками, вы увидите, как правильно выполнить декомпозицию системы, и освоите простые средства анализа и наблюдения, которые помогут вам в этом процессе.

Глава 3. Структура

Глава 3 развивает идеи главы 2 и дополняет их структурой. Вы узнаете, как сохранять требования, как строить иерархию уровней в архитектуре, освоите

таксономию компонентов архитектуры, их взаимоотношения, узнаете конкретные рекомендации по классификации, а также некоторые сопутствующие аспекты, такие как проектирование подсистем.

Глава 4. Композиция

В главе 4 показано, как объединить компоненты системы в композицию, которая удовлетворяет заданным требованиям. В этой короткой главе приводятся некоторые из ключевых принципов книги; она использует материал двух предыдущих глав для создания мощного ментального инструмента, который вы будете применять в любой системе.

Глава 5. Пример проектирования системы

Глава 5 содержит обширный практический пример, который демонстрирует основные идеи проектирования систем, рассматривающиеся до этого момента. Последняя итерация спирали проектирования системы представляет реальную систему, выравнивает проектировочное решение с требованиями бизнеса и показывает, как создать архитектуру и проверить ее на жизнеспособность.

Часть II. Планирование проекта

Глава 6. Мотивация

Так как большинство людей никогда не слышало о планировании проектов (не говоря уже о том, чтобы применять его), в этой главе описана эта концепция и предоставлены побудительные причины для участия в планировании проекта. Это нулевая итерация спирали планирования проектов.

Глава 7. Обзор планирования проекта

В главе 7 приведен общий обзор планирования проектов. Глава начинается с определения успеха при разработке, после чего представляются ключевые концепции обоснованных решений, комплектования проекта, сети проекта, критического пути, сроков и затрат. В главе рассматриваются многие идеи и методы, используемые в последующих главах, а завершается она важным обсуждением ролей и обязанностей.

Глава 8. Сеть и временные резервы

В главе 8 подробно рассматривается сеть проекта и ее использование как инструмента проектирования. Вы увидите, как смоделировать проект в качестве диаграммы сети, узнаете ключевую концепцию временного резерва, научитесь