СОДЕРЖАНИЕ

ВВЕДЕНИЕ: РАБОТА И ПОТОК

Не тратьте время впустую: это материал, из которого соткана жизнь.

Бенджамин Франклин

Сразу после колледжа я работала билд-инженером, и в мои обязанности входила визуализация *билдов**. Я должна была отслеживать, какая версия какого файла на какой компьютер отправляется и в какой среде. Через три месяца я трудилась над билдом — брала код из репозитория исходных кодов, компилировала в исполнимые файлы, а затем устанавливала новые функциональные элементы туда, где остальные (аналитики, разработчики, тестеры и другие участники процесса) могли их увидеть. Билд никак не хотел компилироваться, и я сидела в офисе одна в два часа ночи, пытаясь исправить неполадки. От усталости допускала одну ошибку за другой и решила все-таки отправиться домой. Тогда я серьезно задумалась, правильно ли выбрала профессию. Видимо, технологические задания предполагали работу по ночам. Вздремнув пару часов, вернулась в офис, чтобы отследить кодовые зависимости между разными разработчиками и в итоге заставить билд функционировать.

Не могу точно сказать, сколько часов я потратила на отслеживание зависимостей за все годы слияний, билдинга и релиза продуктов, но уверена, что слишком много. Если бы мне платили по доллару за каждую минуту, когда я корректировала билды и неисправную среду, я сумела бы накопить приличную сумму. Отложенная работа (неважно, чем ее измерять, — часами, днями, неделями или

* Билд (build), релиз — версионированная сборка программного обеспечения. *Прим. ред.*

даже месяцами) не проходит бесследно. Тратить время на проблемы, которых можно избежать, — слишком дорогое удовольствие, к тому же это подрывает моральный дух команды. Жизнь коротка. Растраченное время никогда не вернется.

В научно-фантастическом фильме «Время» жизнь исчисляется секундами, которые буквально стоят денег: люди зарабатывают минуты, часы и дни, чтобы покупать еду, оплачивать проживание, транспорт и все остальное. Бандиты убивают людей, воруя их минуты. Растрачивать часы впустую — верный путь к смерти. В одном памятном эпизоде Уилл Салас, которого играет Джастин Тимберлейк, спасает жизнь богача — Генри Хамильтона, героя Мэтта Бомера. Когда оба добираются до безопасного укрытия, Генри рассказывает 28-летнему Уиллу, что ему 105 лет и он устал от жизни. Он спрашивает, что бы тот сделал, если бы жил сто лет. Уилл отвечает язвительно: «Я бы точно не стал тратить время понапрасну». Позже, когда парень засыпает, Генри передает ему свои сто лет и оставляет записку: «Не трать мое время понапрасну», а затем садится на край высокого моста и ждет, когда его часы отсчитают последние секунды [1].

Эта скомканная записка из научно-фантастической антиутопии прекрасно отражает нашу реальность. Время — это жизнь, пользуйтесь им мудро.

Мы мучаемся от бесконечных посягательств на наше время. Всем, от разработчиков до айтишников, тяжело соответствовать вечно растущим требованиям. В этом отношении не так уж много изменилось с моей первой работы после колледжа, когда я руководила конфигурацией софта в компании Boeing и занималась билдами и развертываниями на мейнфреймах IBM на базе ВВС Хикэм (Гавайи).

У моего рабочего стола выстраивались толпы коллег, которым не терпелось узнать статус билда. «Можно внести еще одну, последнюю коррективу?» Мне так хотелось сказать: «Встаньте в очередь. Я на пределе. Каждый раз, когда вы меня отвлекаете, дело затягивается еще на десять минут». То, что разработчики и тестеры приходили выяснить статус билда, было симптомом гораздо более серьезной проблемы, которую я в то время не распознавала.

Весь трудовой день был занят встречами и заседаниями. Редко удавалось поработать до вечера или хотя бы на выходных, ни на что не отвлекаясь. Однажды, через четыре месяца после того, как я получила эту должность, я всю ночь провела в офисе, стараясь максимально ускорить процесс и разгрести горы скопившихся дел. Руководитель программы, приехавший в 6:30, решил, что я только пришла. И не обрадовался, когда я сообщила, что отправляюсь домой поспать. Нехватка сна была еще одним тревожным признаком, о котором я тогда не задумывалась. Позже, спустя много лет работы в отрасли технологий, я поняла, что неразумный героизм — работа по ночам, совмещение двух должностей и постоянная гонка — оказывает разрушительное воздействие. Невозможно добиться высокого качества, когда спишь четыре часа в сутки.

Мы перегружаем себя и свою команду — это повседневная реальность в секторе информационных технологий. И поскольку нас постоянно прерывают, прекращаем работу над одной задачей и приступаем к другой, от проекта к проекту, никогда не уделяя ни одному столько времени, сколько следует. Это *переключение контекста* убивает возможность углубиться в работу и сосредоточиться. В итоге мы недовольны качеством, несмотря на все благие намерения.

Проблема в том, что мы имеем дело с неадекватными процессами — компании не смогли адаптироваться и найти более здоровые, эффективные способы выполнять рабочие требования. Напротив, преобладает устаревший подход: лишь бы все были постоянно заняты. Такие процессы уже не дают результата. Это как раз тот слон в комнате, которого никто в упор не замечает. Если бы все выполняли свои задачи качественно и в срок, не было бы проблем. Но это такая же редкость, как черный лебедь. Количество запросов (требований) практически всегда не соответствует времени, которое на них остается (производительность). Вот почему нам нужна *система вытягивания* — когда люди могут заниматься одной задачей достаточно долго, чтобы завершить ее, прежде чем взяться за новую, — такая, как Канбан. *Канбан* — наглядная система вытягивания, основанная на *ограничениях*, которые позволяют людям вытягивать работу, когда у них есть возможность, а не проталкивать без учета текущей загрузки.

Поскольку требования и производительность часто не сбалансированы и практически невозможно выполнить все задачи вовремя, такие *системы*, как Канбан, призваны помочь уравновесить эти моменты.

Чуть позже мы подробнее обсудим Канбан и его место в визуализации работы, а пока достаточно подчеркнуть, что эта система позволяет сделать работу и проблемы наглядными и повысить эффективность производственного процесса. Канбан помогает выполнить задание качественно без необходимости полуночничать.

«Цель Канбана — выявить все проблемы».
Тайити Оно

В 2000-х годах я работала в компании Билла Гейтса Corbis (Сиэтл), в крупнейшем международном фотобанке: возглавляла команду билда и конфигурации.

Мы пользовались вполне приличной репутацией в отделе инжиниринга до 2005 года, когда две наши предпродакшен-среды из семи серверов увеличились в четыре раза — то есть стало восемь предпродакшен-сред из двадцати пяти серверов. У нас было семнадцать баз данных. Каждая сконфигурирована вручную в рамках сильносвязанной архитектуры с огромным количеством зависимостей. Более того, компания поручила нам разработать новые крупные системы, причем так, чтобы каждую из них можно было развертывать по очереди. Зависимости между существующей системой и двумя новыми выросли до небес. И на мои плечи легло не двадцать пять серверов, а двести.

Чтобы справиться со всеми этими изменениями, мы создали дополнительные долгоживущие ветки в рамках *системы управления версиями* — где разработчики хранят свои коды. Ужасное решение, но это помогло командам не нарушать чужих корректировок. Долгоживущие ветки — это место, где код хранится обособленно, там невозможно проверить, как он будет воздействовать на код, уже переданный в продакшен. Это как взять из приюта старую кошку и надеяться, что она и ваш кот, который старше ее лет на сто, примут друг друга с распростертыми лапами. Когда предстоит конфигурировать и поддерживать больше двухсот серверов, требуется грамотное управление. Нужно было как минимум две недели, чтобы восстановить данные продакшена в среды предпродакшена.

Раз в шесть недель мы проводили день С (слияние), что отнимало немало времени у разработчиков.

Наша репутация ухудшилась. Разработчики жаловались, что на билды уходит слишком много времени. Это, естественно, оскорбляло меня. Я вознамерилась доказать, что они ошибаются, и записала сроки создания и развертывания билдов (рис. 1).

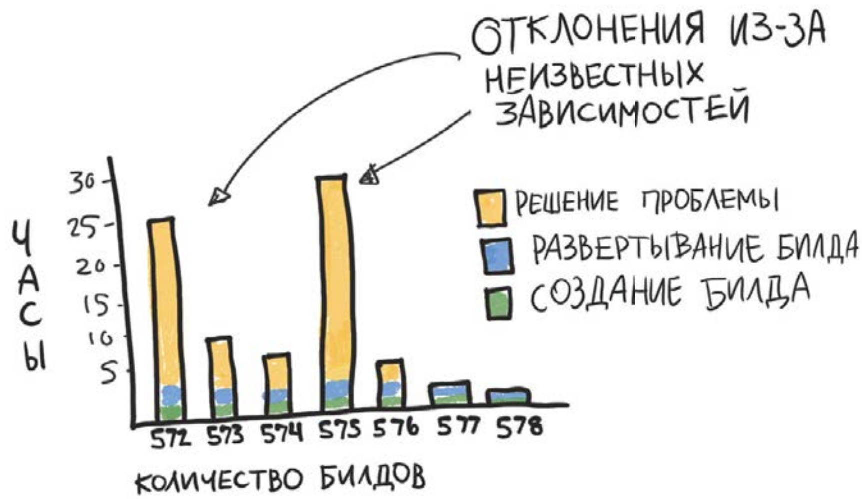

Рис. 1. Билды требуют не так уж много времени

Я отметила, что архитектурная катастрофа — система с нераспознаваемой структурой — делала развертывание и поддержку сред проблематичными. Ручные *smoke-тесты* (проверка функционала сайта) затягивают время, спустя которое разработчики и тестеры могут увидеть последние изменения, а отсутствие автоматического тестирования мешает быстро выявлять проблемы. Ручные smoke-тесты были нормой. Так что начальство посчитало обе проблемы надуманными. Но факт оставался: разработчиков и тестеров это не устраивало. Бизнес-аналитиков — тоже. И начальство было недовольно. Какая радость работать в команде, которая «не справляется». Преграды между группами были сильнее, чем связи. Типичная проблема неадекватной системы.

Мой опыт работы с неадекватной системой совпал с решением финансового директора заменить *систему планирования бизнес-ресурсов (ERP)* другим продуктом под названием SAP. *ERP* — информационная система менеджмента, которая охватывает планирование, закупку, материально-техническое обеспечение, продажи, маркетинг, финансы и HR. SAP — собственная система ERP, разработанная SAP AG, четвертой крупнейшей софтверной компанией в мире.

Босс спросил меня: «Не возьмешь ли ты на себя управление командой SAP Basis в рамках управления командой билдов и релизов?» И, как форменная идиотка, я согласилась. До сих пор не понимаю, как я могла обречь себя на новые неудачи. У меня не было никакого опыта с SAP, а новые обязанности только распыляли внимание — настолько, что все другие задачи я стала выполнять из рук вон плохо. Многозадачность — прекрасный способ подорвать прогресс, как многие из вас наверняка знают по собственному опыту.

В то время я еще не подозревала, что все это — тревожные сигналы неадекватной системы. Я видела лишь, что результаты моей работы никак нельзя назвать образцовыми, и расстроилась настолько, что задумалась об уходе.

И обновила резюме.

В 2006-м мы много времени уделили анализу и сравнению разных инструментов управления исходным кодом. Команда выбрала Team Foundation Server (TFS). В конце концов, мы принадлежали Microsoft, так что я установила, сконфигурировала и поддерживала TFS — и при этом изучала SAP, еженедельно интервьюировала кандидатов и помогала внедрять новый процесс сопровождения. Этот процесс позволил нам вносить корректировки каждые две недели, а не раз в полгода.

Разработчик пользовательского интерфейса (UI) по имени Дуэйн Джонсон увидел, насколько полезно поставлять небольшие, но частые корректировки, и стал отстаивать идею подобных регулярных улучшений. Дуэйн запустил постоянный процесс исправления багов UI, раз в два месяца. Появилась новая вещь, которую нужно было поддерживать, но она была стоящая. Эти инкрементальные и итеративные усовершенствования с систематической каденцией стали нашей *agile*-альтернативой традиционной разработке *каскадного типа*. Agile-методы влились в процесс и заставили задуматься о более эффективном подходе к работе.

В апреле 2006-го в Corbis появился шотландец из Microsoft. Дэвид Андерсон посещал нас ежемесячно и учил применять *теорию ограничений* (ТО) в обмен на разрешение написать историю agile-преобразования Corbis. ТО — способ найти самые важные лимитирующие факторы (ограничения), которые препятствуют достижению цели, а затем систематически совершенствовать их, пока они не перестанут быть лимитирующими. Мы воодушевленно читали его книгу Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results («Гибкое управление в разработке программного продукта: как применить теорию ограничений для бизнес-результатов») и планировали использовать *разработку на основе функционала (FDD)* — тип agile-разработки с межфункциональными, коллективными и ограниченными во времени методами создания функций. Как пишет Даррен Дэвис в блоге «Тайная история Канбан», методы Дэвида «…исключали из процесса однозначные расчеты и опирались на конкретные данные, чтобы прогнозировать, когда *вероятнее всего* будет завершена работа над продуктом» [2]. Дэвид провел с нами обзоры операций и объяснил, насколько важно оценивать прогресс (или его отсутствие). Когда я поняла, что именно нужно анализировать, мой мир изменился. Нытье не помогает, а вот оценка *cycle time* (времени, требующегося для выполнения работы) и представление этих данных руководству очень даже эффективны. Я смогла повлиять на начальство и получить одобрение на наем новых сотрудников для нашей команды.

Иногда очевидное теряется среди давления и нагрузки корпоративного мира. Мы интуитивно знали, что у нас в работе слишком много проектов. Но это было сложно увидеть, пока мы не принялись подсчитывать реальное время, уходящее на выполнение заданий. И тогда стало очевидно, что дела дольше всего находятся на *этапе ожидания*, чем на этапе выполнения. Мы ждали утверждений. Ждали, когда другие завершат свои части проекта, чтобы мы приступили к своим (или закончили их). Ждали возможности трудиться, ни на что не отвлекаясь, чтобы доделать наконец собственные задачи. Ждали подходящего дня/недели/месяца. И пока мы ждали, начинали новый проект, потому что, как вы понимаете, загрузка ресурсов — основная цель и нужно быть *постоянно* чем-то занятым.

Как пишет Кейт Мерфи в статье «Нет времени подумать», «одни из основных жалоб современного общества — чрезмерная занятость, повышенные обязательства и перегруженность работой. Спросите людей на любом социальном мероприятии, как у них дела, и ответ будет неизменным: "Я так занят", "С ума схожу от работы" или "Сил моих уже нет". Никто уже не говорит "хорошо"» [3]. Каждый день я вижу подтверждение этому. Когда выдается спокойная минутка для размышлений — допустим, пока ждешь очередной встречи, — звонит телефон. Для амбициозных личностей, которые стремятся постоянно быть на связи, занятость может перерасти в зависимость. Но занятость не приравнивается к росту, или совершенствованию, или ценности. Часто она означает, что вы взяли на себя столько задач, что не можете ни одну из них выполнить качественно. Иногда прогулка в парке или минутка размышления — оптимальный способ жить сегодняшним днем. Какой ужас: инженер сидит без дела целых пятнадцать минут и просто думает?!

В Corbis мы проанализировали причины, почему работаем над таким большим количеством задач одновременно, и нам многое открылось. Финансовый директор планировал внедрить новую систему расчетов. Старший вице-президент по глобальному маркетингу тоже чего-то хотел. Как и вице-президент по медиасервису. И глава отдела продаж. И все требовали результатов *прямо сейчас*. В итоге приоритеты сталкивались по всей иерархии, и это лишь бизнес-сторона дела. В сфере разработки не только нужно было внедрить все эти требования, но и внести внутренние корректировки и заняться сопровождением. Более того, когда возникали производственные проблемы, нам приходилось бросать все дела — нравится вам или нет, производство всегда на первом месте. Конфликтующие приоритеты стали очевидными, когда мы взглянули на множество долгоживущих веток кода, но в целом у нас не было четкого понимания последствий избыточного количества одновременных задач. Сложно управлять невидимой работой. Трудно заметить явные предупреждения, что наш интеллектуальный бюджет уже на грани. Нет времени просто подумать.

В сентябре 2008 года, после восьми лет работы в Corbis, меня, как и еще 41 сотрудника, сократили. Я решила попробовать что-то новое и устроилась в AT&T Mobile в команду управления разработкой. Но отказ от бережливой системы Канбан, которую я помогала создавать в Corbis, и возвращение к *каскадному подходу* (традиционный метод разработки продукта: нельзя приступить к работе, пока не будут выполнены все предыдущие этапы), когда прогнозы строятся на основе отчета по отработанным часам, стали для меня большим шагом назад. В июле 2010-го я уволилась.

В январе 2011-го Дэвид Андерсон предложил мне заняться разработкой и проведением нового курса для компании David J. Anderson & Associates под названием «Канбан для IT-операций». В то время Европа обгоняла США по популярности Канбан, так что в феврале мои исследования начались с Англии, Швеции и Германии. В марте мы провели первый пробный семинар в Бостоне, где я выступила на DevOpsDays Boston 2011 в Центре исследований и разработок Microsoft в Новой Англии.

Изначально я планировала написать рекомендации для участников семинара, чтобы помочь им спроектировать канбан-доски. Но эти рекомендации переросли в настоящее хранилище информации и здорово сэкономили мне время. Я стала записывать не только все, что узнавала о применении бережливого производства, Канбан и *потока* в собственной работе, но и выборочные формулы, теории и статистические данные экспертов. К примеру, что такое бережливое производство? Я предпочитаю определение Никласа Модига и Пэра Олстрема. В своей фантастической книге This Is Lean: Resolving the Efficiency Paradox («Это бережливое производство: как разрешить парадокс эффективности») они определили бережливое производство как «стратегию эффективности потока с ключевыми принципами “точно в срок” и визуального менеджмента» [4].

Итак, что же нам известно? Мы знаем, что к производству предъявляются высокие требования относительно бизнес-ценности, чтобы обеспечить конкурентоспособность. Мы в курсе, что многие организации применяют заторможенные и обременительные стратегии развертывания. Мы также понимаем, что способны достичь максимального результата, если четко видим, что получается, а что нет. Это очевидно, но регулярно игнорируется.

Технологический мир не планирует снижать обороты. Темпы, которыми мы должны выдавать новые функции и возможности, чтобы привлечь новых клиентов и удержать старых (то есть избежать *оттока*), практически приближены к сверхсветовой скорости. Многие компании сегодня функционируют в режиме выживания, хотя и не замечают этого. Значит, нет более подходящего времени, чтобы перейти на новый уровень. Как же это сделать?

Ответ прост и понятен. Он не требует ни крупных денежных затрат, ни гениев, ни специалистов. Нужно только перестать соглашаться на все подряд и обдуманно давать добро только на самое важное в этот момент. И делать это *визуально*.

Решение — спроектировать и использовать систему рабочего потока, которая выполняет пять основных задач:

1. Делает работу видимой.
2. Ограничивает количество незавершенных задач (WIP*).
3. Оценивает рабочий поток и управляет им.
4. Эффективно определяет приоритеты (это непросто, но потерпите — я покажу, как это сделать).
5. Вносит коррективы, опираясь на обратную связь и метрики.

Что мы обсудим в этой книге:

— как выявить пять расхитителей вашего времени;
— как обличить расхитителей времени, чтобы сделать работу видимой и оптимизировать *рабочий поток*;
— как узнать истинное положение дел с помощью метрик и обратной связи;
— какие практики способны довести до беды;
— как повлиять на решение начальства.

Все примеры, которые я привожу на этих страницах, основаны на опыте — моем и тех, кто был свидетелем расхищения времени. Некоторые избегают публичного обсуждения преступлений, совершенных в их компаниях, так что для их удобства я изменила имена, чтобы защитить и невиновных, и виноватых. Мы также рассмотрим системные организационные проблемы, которые необходимо решить, чтобы достичь успеха. Как сказал Эдвардс Деминг, «плохая система всегда побеждает хорошего человека» [5].

Эта книга — одновременно объяснение, руководство и бизнес-аргументация для методов бережливого производства, Канбан и потока, которые повышают темпы и эффективность работы.

Не все советы подойдут именно вам. В основном материал касается области IT, также приводится несколько примеров из других

* WIP, work in progress — работа в процессе, незавершенное производство. *Прим. ред.*

сфер — для большей убедительности. Заимствуйте то, что актуально для вас, а остальное примите в качестве информации, чтобы знать, с чем приходится сталкиваться сотрудникам других отделов вашей компании или конкурентам. Каждый параграф второй части включает упражнения из моих семинаров, где предложен план действий, который поможет сделать работу видимой, повысить эффективность процесса и выявить проблемы. Упражнения взаимосвязаны, так что лучше читать всё по порядку.

Объяснять концепции из этой книги кому-то нетрудно. Добиться одобрения и поддержки, чтобы внедрить предложенные подходы, уже намного сложнее. Люди не любят перемен. И поэтому, прежде чем заняться структурой рабочего потока, мы подробно исследуем, что именно мешает вам выполнять задачи быстро. Изучив преступления, совершенные против рабочей нагрузки, мы сможем собрать все необходимые данные и найти решение. Итак, приступим.