Предисловие

Джефф Сазерленд (Jeff Sutherland)

Впервые я начал применять методологию Scrum[1] в 1993 году в Easel Corporation в Барлингтоне (Burlington), Массачусетс[2], в сотрудничестве с нашим генеральным директором (CEO). В 1995 году я начал работать с Кеном Швайбером (Ken Schwaber), который формализовал процесс дистрибуции вновь созданного программного обеспечения по всему миру[3]. В четырех компаниях, где мне пришлось работать после Easel Corporation, я использовал природу самоорганизации Scrum для перехода от Type A Scrum к Type B Scrum и далее к Type C Scrum.

В 2000 году мне удалось применить методологию Scrum в компании PatientKeeper[4] и, как выяснилось, данная методология прекрасно сочетается с бережливым процессом разработки. С годами мы сократили время цикла до устраивающего заказчиков срока: неделя для срочных работ, месяц для мелких дополнений программного обеспечения и три месяца для значительных новых продуктов.

В PatientKeeper вся компания функционирует с применением методологии Scrum; каждый понедельник имеют место собрания MetaScrum[5], на которых принимаются решения, влияющие на управление, адаптацию, самоорганизацию и развитие компании. Подобным собранием всегда руководит Product Owner (или менеджер продукта) и на них присутствуют все учредители компании, а также генеральный директор. Бережливые концепции здесь тщательно рассматриваются и итерации Sprint начинаются, останавливаются или изменяются только на таких собраниях. В соответствии с решениями, принимаемыми на этих еженедельных собраниях MetaScrum, могут иметь место глобальные изменения, затрагивающие всю компанию (а также заказчиков, которых это касается).

Журнал предстоящей работы (product backlog) подготавливает менеджер продукта вместе с коллективом. Пятнадцатиминутные ежедневные совещания Scrum of Scrums проводит главный ScrumMaster. На подобных совещаниях рассматривается, что коллектив делал вчера, чем он будет заниматься сегодня и что мешает выполнению работы? Благодаря этим кратким собраниям, а также автоматизированной системе слежения за состоянием журналов предстоящей работы одновременных параллельных Sprint, нам удавалось осуществлять 45 релизов программного обеспечения ежегодно, предназначен-

---

[1] Scrum — одна из самых популярных методологий гибкой разработки программного обеспечения.

[2] Sutherland J. Agile Development: *Lessons Learned from the First Scrum*, Cutter Agile Project Management Advisory Service: Executive Update, 5(20), pp. 1-4. — *2004.*

[3] Schwaber K. *Scrum Development Process*, OOPSLA Business Object Design and Implementation Workshop, Springer: London, *1997.*

[4] Более подробно об этой компании речь пойдет в главе 5.

[5] Sutherland J. Future of Scrum: *Parëallel Pipelining of Sprints in Complex Projects*, AGILE 2005 Conference. *2005.*

ных для крупных, решающих важные задачи предприятий. Даты (очередных релизов) сообщались заказчикам до начала Sprint, и тысячи веб-пользователей и сотни врачей с PDA имели возможность оценить новое ПО к концу каждой Sprint. Для программы PatientKeeper интервал от концепции (или от журнала предстоящей работы) до получения прибыли (или продукта у потребителей) составил один месяц. Нам пришлось ликвидировать потери повсеместно — до, во время и после процесса реализации.

В 1993 году бережливый подход к разработке программного обеспечения не пользовался большой популярностью ни в одной отрасли. Методология Scrum явилась первой конкретной реализацией бережливого мышления для разработки ПО, позволившей различным компаниям организовать работу коллективов разработчиков на основе принципов бережливости и с использованием легкой в понимании стандартной модели. Основная трудность при этом заключалась в том, что трудно было объяснить, как реализовать эту модель, чтобы добиться стабильного повышения качества и производительности.

Сегодня печатные издания и семинары от Мэри и Тома Поппендик предоставляют проверенный практикой набор принципов, призванных помочь организациям внедрять предлагаемые средства, приемы и методы в своей уникальной среде и с учетом своих возможностей. Мы теперь можем объяснить, как использовать методологию Scrum для применения принципов бережливости в разработке программного обеспечения. Помимо моей компании, PatientKeeper, я использовал процедуры и процессы, описанные в этой книге для обучения (как оптимизировать Scrum) специалистов-практиков по всему миру.

Бережливый подход к созданию ПО рассматривает все гибкие (agile) методы в качестве эффективных, доказавших свою пригодность применений бережливого мышления. Однако данный подход предоставляет более широкую перспективу, позволяющую гибким методам быстро развиваться. Во-первых, бережливый подход позволяет держать под контролем всю цепочку создания ценности (от концепции до прибыли), а также выявлять все случаи потерь и задержек, имеющих место до и после разработки кода. Во-вторых, он создает менеджмент-среду, делающую возможным развитие гибких методов создания программного обеспечения. В-третьих, бережливый подход предоставляет набор доказавших свою эффективность принципов, которые каждая организация может использовать для внедрения предлагаемых средств, приемов и методов в своей уникальной среде и с учетом своих возможностей.

Каждая глава этой книги иллюстрирует набор принципов, применение которых позволит организовать более продуктивную работу коллективов разработчиков. Если вы хотите быть так же эффективны, как компания Toyota, где производительность вчетверо выше, чем у конкурентов (а качество в двенадцать раз лучше), вам не обойтись без этих принципов. Использование этих принципов сделает тщетной конкуренцию с вами и бесспорной вашу победу на рынке. Отдача от инвестиций при использовании практики, описанной в данной книге, может быть очень высокой.

Чтобы воспользоваться преимуществами бережливого подхода, мы полагаем, вам следует систематически применять соответствующие принципы на практике. Иными словами, вы должны глубоко понять японские концепции Muri, Mura и Muda. Авторы, Мэри и Том, представили их в своей книге в виде семи принципов бережливости и семи основных случаев потерь при разработке программного обеспечения, чтобы помочь читателю понять, как они работают и что нужно делать.

Концепция Muri связана с загрузкой системы, а Mura предписывает никогда не оказывать давление на индивидуума, систему или процесс. Тем не менее многие менеджеры

стремятся загрузить разработчиков на 110%. Они прилагают все усилия, чтобы создать атмосферу “срочности” и разработчики “работали напряженнее”. Они пытаются контролировать каждый шаг коллективов, что вредит их самоорганизации. Эти трудно постижимые идеи часто порождают задержки, необходимость переделывать уже сделанную работу и неудачные проекты.

Когда я спрашивал таких менеджеров, пытаются ли они загрузить свой ПК или ноутбук на 110 процентов, они дружно отвечали: “Конечно нет. Мой компьютер перестанет работать, если попытаться это сделать.”. Из-за перегрузки людей проекты часто задерживаются, программное обеспечение получается некачественным и его трудно поддерживать, и дела, в целом, идут все хуже. Необходимо понять, что в компании Toyota и в методологии Scrum используется система с “вытягиванием” (pull system), предписывающая избегать стрессов (Mura) и устранять “узкие места” (Muri). Разработчики берут из журнала предстоящей работы (Product Backlog) то, что им нужно, и когда нужно. Они выберут из журнала только ту работу, которая готова для Sprint, и они никогда не возьмут больше, чем они смогут выполнить в данной Sprint. Ускорить работу можно, выявляя и устраняя помехи и работая с менеджментом с целью ликвидации потерь (Muda). Если устранить потери, это уменьшит объем ожидающей выполнения работы. В результате производительность возрастает, и у разработчиков появляется время для повышения качества программного обеспечения и уделения большего внимания нуждам заказчиков.

Таким образом, оптимизируя загрузку системы (Muri) и избегая стрессов (Mura), вы сокращаете время цикла. Это делает очень заметными недостатки, вызывающие потери (Muda). Если устранить эти недостатки, коллектив разработчиков будет работать быстрее, будет делать больше при меньших затратах, повысит качество и создаст как раз такой продукт, который нужен заказчику.

Я рекомендую читателю постоянно держать книгу Мэри и Тома Поппендик на рабочем столе и пользоваться ею регулярно при реализации принципов бережливости. При некоторой настойчивости вы быстро сможете удвоить производительность, устраняя потери (и уменьшая при этом объем работы), а затем удвоите ее еще раз, устраняя разного рода помехи. Достигнув четырехкратного увеличения производительности (с помощью описанных в книге методов), вы быстро добьетесь и двенадцатикратного повышения качества, как компания Toyota.

— Джефф Сазерленд, Ph.D
Технический директор (Chief Technology Officer) компании PatientKeeper,
сертифицированный ScrumMaster,
один из создателей методологии разработки Scrum,
июль, 2006 год.

Кент Бек (Kent Beck)

Разработка программного обеспечения (ПО) — это цепь со множеством звеньев; повышение общей эффективности требует пристального взгляда на всю цепь. Данная книга предназначена для людей, имеющих отношение к созданию ПО, — не только для программистов, но также менеджеров, спонсоров, заказчиков, тестировщиков и проектировщиков. Изложенные здесь принципы в конечном счете касаются всех, связанных с разработкой программного обеспечения. Книга обращена к тем, кто готов взглянуть на процесс разработки как на единое целое.

Чтобы идеи представляли какую-то ценность, они должны основываться как на теории, так и на практике. Данная книга переносит хорошо зарекомендовавшие себя на производстве теорию и практику бережливости в разработку программного обеспечения. Многие фундаментальные проблемы производства являются также проблемами разработки ПО. Речь идет о неопределенностях и изменениях, постоянно совершенствующихся процессах и создании ценности.

Что эта книга может предложить? Во-первых, многообразие теорий, предоставляющих альтернативные подходы к совершенствованию процесса разработки ПО. Во-вторых, множество историй о применении этих теорий к реальным проектам. В-третьих, провоцирующие вопросы, призванные помочь читателю применять изложенные теории.

Мэри обладает уникальной квалификацией для написания этой книги. Она является опытным экспертом в области бережливого производства, знакомым с предметом не понаслышке (однажды, оказав помощь в преобразовании использовавшихся технологических процессов, ей удалось спасти от закрытия завод). Она также опытный разработчик программного обеспечения (как программист и менеджер). Мне однажды пришлось совместно с Мэри проводить семинар,и было удовольствием наблюдать, силу и уверенность, с которыми она подавала материал. Ее голос как будто слышится на страницах этой книги.

По сравнению с предыдущей книгой Мэри и Тома Поппендик, *Lean Software Development* данная книга предлагает новый взгляд на реализацию данного подхода. Здесь повторено изложение теории из предыдущей книги, однако с акцентом на применение идей к реальным ситуациям. В результате мы получили интересно написанное практическое руководство по применению идей, потенциально способных помочь вам трансформировать ваш процесс разработки.

Данная книга предназначена для практиков, стремящихся, чтобы их действия имели максимальный эффект. Если вы относите себя к этой категории, я рекомендую вам эту книгу в качестве источника идей и энергии для перемен к лучшему.

— Кент Бек,
Three Rivers Institute,
июль, 2006 год.