

---

---

Бережливое  
производство  
программного  
обеспечения  
*От идеи до прибыли*

---

---

# Implementing Lean Software Development

*From Concept to Cash*

Mary and Tom Poppendieck



**ADDISON-WESLEY**

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco  
New York • Toronto • Montreal • London • Munich • Paris • Madrid  
Capetown • Sydney • Tokio • Singapore • Mexico City

---

---

# Бережливое производство программного обеспечения

*От идеи до прибыли*

Мэри и Том Поппендик



Москва · Санкт-Петербург · Киев  
2010

ББК 32.973.26-018.2.75

П58

УДК 681.3.07

Издательский дом “Вильямс”

Зав. редакцией *С.Н. Тригуб*

Перевод с английского и редакция *О.А. Меженного*

По общим вопросам обращайтесь в Издательский дом “Вильямс” по адресу:  
info@williamspublishing.com, http://www.williamspublishing.com

**Поппендик, Мэри, Поппендик, Том.**

П58 Бережливое производство программного обеспечения: от идеи до прибыли. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2010. — 256 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1538-2 (рус.)

**ББК 32.973.26-018.2.75**

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства Addison-Wesley Publishing Company, Inc.

Authorized translation from the English language edition published by Prentice Hall, Inc., Copyright © 2007 Pearson Education, Inc.

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

Russian language edition is published by Williams Publishing House according to the Agreement with R&I Enterprises International, Copyright © 2010.

*Научно-популярное издание*

**Мэри Поппендик, Том Поппендик**

**Бережливое производство программного обеспечения:  
от идеи до прибыли**

Литературный редактор *Е.П. Перестюк*  
Верстка *О.В. Романенко*  
Художественный редактор *В.Г. Павлютин*  
Корректор *Л.А. Гордиенко*

Подписано в печать 26.03.2009. Формат 70x100/16.

Гарнитура Times. Печать офсетная.

Усл. печ. л. 20,64. Уч.-изд. л. 16,9.

Тираж 1000 экз. Заказ № 0000.

Отпечатано по технологии CtP  
в ОАО “Печатный двор” им. А. М. Горького  
197110, Санкт-Петербург, Чкаловский пр., 15.

ООО “И. Д. Вильямс”, 127055, г. Москва, ул. Лесная, д. 43, стр. 1

ISBN 978-5-8459-1538-2 (рус.)  
ISBN 0-321-43738-1 (англ.)

© Издательский дом “Вильямс”, 2010  
© Pearson Education, Inc., 2007

# Оглавление

<b>Предисловие</b>	<b>15</b>
<b>Введение</b>	<b>19</b>
<b>Глава 1. История</b>	<b>23</b>
<b>Глава 2. Принципы</b>	<b>41</b>
<b>Глава 3. Ценность</b>	<b>63</b>
<b>Глава 4. Потери</b>	<b>85</b>
<b>Глава 5. Скорость</b>	<b>113</b>
<b>Глава 6. Люди</b>	<b>131</b>
<b>Глава 7. Знание</b>	<b>159</b>
<b>Глава 8. Качество</b>	<b>181</b>
<b>Глава 9. Партнеры</b>	<b>209</b>
<b>Глава 10. Путешествие</b>	<b>223</b>
<b>Список литературы</b>	<b>245</b>
<b>Предметный указатель</b>	<b>252</b>

# Содержание

<b>Предисловие</b>	<b>15</b>
<b>Введение</b>	<b>19</b>
<b>Глава 1. История</b>	<b>23</b>
Взаимозаменяемые детали	23
Взаимозаменяемые люди	24
Семейство Тойода	24
Производственная система Toyota	26
Тайити Оно	26
Поставки “точно вовремя”	27
Автоматизация	27
Сигаэо Синго	27
Производство без запасов	28
Производство без проверок	28
“Точно вовремя”	28
Концепция бережливости	32
Бережливое производство и бережливые операции	33
Бережливая цепь поставок	33
Бережливая разработка продуктов	34
Бережливый подход к созданию программного обеспечения	38
Попробуйте это	38
<b>Глава 2. Принципы</b>	<b>41</b>
Принципы и практика	41
Разработка программного обеспечения	42
Разработка	42
Программное обеспечение	43
Семь принципов бережливого подхода при разработке ПО	44
Принцип 1: ликвидировать потери	44
Миф: созданная заранее спецификация сокращает потери	46
Принцип 2: встраивать качество	47
Миф: цель тестирования — выявление дефектов	49
Принцип 3: создавать знание	50
Миф: прогнозы обеспечивают предсказуемость	51
Принцип 4: откладывать необратимые решения	52
Миф: план — это обязательство	53

Принцип 5: доставлять быстро	54
Миф: спешка ведет к браку	54
Принцип 6: уважать людей	55
Миф: существует наилучший метод	57
Принцип 7: оптимизировать целое	57
Миф: оптимизировать путем декомпозиции	59
Попробуйте это	60
<b>Глава 3. Ценность</b>	<b>63</b>
Бережливые решения	63
От концепции до прибыли	66
Концепция	66
Осуществимость проекта	66
Пробная версия	67
Прибыль	68
Восхищенные потребители	68
Глубокое понимание потребителя	69
Фокусироваться на работе	70
Ориентированная на потребителя организация	71
Руководство	72
Главный инженер	72
Команда руководителей	74
Совместное руководство	74
Кто за все отвечает	75
Многофункциональные коллективы	76
Технологичность	77
Разработка на заказ	78
От проектов к продуктам	78
Сотрудничество между ОИТ и бизнесом	80
Ответственность	81
Попробуйте это	82
<b>Глава 4. Потери</b>	<b>85</b>
Меньше программного кода	85
Zaga	85
Сложность	87
Каждая функциональная возможность должна быть оправдана	87
Минимально-полезные наборы функций	89
Не следует автоматизировать запутанные процессы	89
Семь непроизводительных расходов	90
Частично выполненная работа	91
Избыточные функциональные возможности	92
Повторное приобретение знания	93
Передача работы	93
Переключение между задачами	94
Задержки	96

## 8 Содержание

Дефекты	97
Картирование потока создания ценности	98
Подготовка	99
Выбор процесса	99
Выбор начала и останова отсчета времени	99
Необходим ответственный	100
Не нужно усложнять	100
Примеры	101
Пример 1	101
Пример 2	103
Пример 3	105
Пример 4	107
Итоги	109
Перспективные карты потока создания ценности	110
Попробуйте это	110
<b>Глава 5. Скорость</b>	<b>113</b>
Поставлять быстро	113
PatientKeeper	113
Время — универсальная валюта	116
Теория очередей	117
Закон Литтла	118
Вариация и загрузка	118
Сокращение времени цикла	119
Добиться ритмичности поступления работы	120
Свести к минимуму количество одновременно выполняющейся работы	122
Минимизировать размеры объектов, одновременно находящихся в производстве	123
Создать постоянный ритм работы	124
Объем работы не должен превышать возможности организации	125
Использование планирования с “вытягиванием”	127
Резюме	129
Попробуйте это	130
<b>Глава 6. Люди</b>	<b>131</b>
Система менеджмента	131
Boeing 777	131
У. Эдвардс Деминг	134
Почему хорошие начинания оказываются неудачны	137
Коллективы	138
Что такое коллектив?	139
Компетентность	141
Руководство	143
Основанные на ответственности планирование и управление	144
Визуальная рабочая среда	146
Самоуправляющийся производственный процесс	147
Канбан	148

Андон	149
Наглядное информирование	150
Побудительные мотивы	151
Оценка эффективности	151
Система оценок	152
Вознаграждение	153
Совет 1: обеспечить, чтобы обоснованность повышений была неоспорима	153
Совет 2: не придавать особого значения ежегодным повышениям зарплаты	153
Совет 3: награждать в зависимости от вклада в общий результат, а не за личные достижения	154
Совет 4: найти лучшую мотивацию, чем деньги	155
Попробуйте это	156
<b>Глава 7. Знание</b>	<b>159</b>
Создание знания	159
В чем ваша проблема?	159
Научный метод мышления	160
Сохранение знания	161
Отчеты АЗ	163
Век Интернета	164
“Точно вовремя ”	165
Параллельная разработка альтернатив	165
Пример 1: интерфейс для медицинского устройства	167
Пример 2: компенсация эффекта красных глаз	168
Пример 3: сменные интерфейсы	168
Почему это не потери	169
Рефакторинг	169
Унаследованные системы	171
Решение проблем	172
Упорядоченный подход	173
1. Обозначить проблему	173
2. Проанализировать ситуацию	174
3. Создать гипотезу	175
4. Провести эксперименты	176
5. Оценить результаты	176
6. Стандартизировать метод и многократно его применять	177
Кайдзен-мероприятия	177
Решение проблем с участием больших коллективов	177
Попробуйте это	179
<b>Глава 8. Качество</b>	<b>181</b>
Обратная связь	181
Программа Polaris	181
Планирование релизов	183
Архитектура	185
Итерации	186

## 10 Содержание

Подготовка	187
Планирование	189
Реализация	190
Оценка	191
Вариация: интерфейс пользователя	192
Дисциплина	192
Пять “S”	194
Стандарты	196
Экспертиза программного кода	197
Парное программирование	198
Защита от ошибок	199
Автоматизация	200
Разработка через тестирование	201
Блочные тесты	201
Тесты “историй”	202
Тесты на простоту использования и диагностические тесты	203
Тестирование качества	203
Управление конфигурацией	203
Непрерывная интеграция	204
Вложенная синхронизация	205
Попробуйте это	206
<b>Глава 9. Партнеры</b>	<b>209</b>
Совместные усилия	209
Чрезвычайное происшествие!	209
Open Source	211
Глобальные сети	212
Аутсорсинг	215
Инфраструктура	215
Транзакции	216
Разработка	216
Контракты	218
Соглашение T5	218
Контракт PS 2000	219
Реляционные контракты	220
Попробуйте это	221
<b>Глава 10. Путешествие</b>	<b>223</b>
Куда бы вам хотелось отправиться ?	223
Компьютер на колесах	224
Долговременная перспектива	225
Люди важнее всего	227
Чему мы научились ?	228
Шесть сигм	228
Руководители производства и неофициальные лидеры	229
Средства и результаты	229

Теория ограничений	229
Критическая цепь	231
Привычка приспосабливаться	232
Гипотеза	233
Обучение	233
Стремление улучшать процессы	235
Критерии	236
Время цикла	237
Финансовая отдача	238
Удовлетворение заказчика	239
Дорожная карта	240
Попробуйте это	241
Оптимизировать целое	241
Уважать людей	241
Доставлять быстро	242
Не торопиться брать обязательства	242
Создавать знание	243
Встраивать качество	243
Ликвидировать потери	244
<b>Список литературы</b>	<b>245</b>
<b>Предметный указатель</b>	<b>252</b>



*Нашим родителям:  
Джону и Мардж Браст,  
а также  
Элмеру и Рут Поппендик*

# Отзывы о книге

“Эта замечательная книга объединяет практические советы, готовые к использованию методы и глубокое понимание, почему именно так следует создавать программное обеспечение. Мне приходилось наблюдать коллективы, полностью трансформировавшиеся благодаря идеям этой книги.”

—Майк Кон (Mike Cohn), автор книги *Agile Estimating and Planning*

“Я, сам практиковавший бережливый подход, обращался к первой книге этих авторов не раз. Когда вышла вторая книга, я был восхищен: она оказалась еще лучше. Если вас интересует, как принципы бережливости могут быть использованы в организации, занятой разработкой программного обеспечения, эта книга как раз для вас. Мэри и Том Поппендик предлагают читателям приятную смесь истории, теории и практики.”

—Алан Шеллоуэй (Alan Shalloway), соавтор книги *Design Patterns Explained*

“Я с большим удовольствием прочел данную книгу. Кажется, она еще лучше первой книги Тома и Мэри, хотя та книга была исключительно хороша! Мэри обладает глубокими познаниями в применении бережливых методов при разработке продукции и в производстве. Это редкий случай, когда рассматривается применение данных методов к разработке программного обеспечения. Этого не встретишь ни в одной другой книге (за исключением первой книги этих авторов).”

—Бас Водд (Bas Vodde)

“Эта новая книга Мэри и Тома Поппендик является хорошо написанным и понятным введением в принципы бережливости и доказавшую свою эффективность практику для менеджеров и инженеров, занятых созданием программного обеспечения. Упомянутые принципы и практика иллюстрируются очень удачными примерами успешного их применения. Я с удовольствием прочел данную книгу.”

—Роман Пичлер (Roman Pichler)

“В своей новой книге Мэри и Том Поппендик рассматривают более глубоко темы, с которыми они познакомили читателя в их первой книге *Lean Software Development*. Они начинают с убедительной и увлекательной истории о применении бережливых принципов, а затем переходят к обсуждению таких ключевых тем, как потребительская ценность, потери и люди. Каждая глава включает упражнения, призванные помочь читателю научиться применять изложенные методы. Если вы хотите лучше понять, как принципы бережливости используются при создании программного обеспечения, это книга для вас.”

—Билл Уэйк (Bill Wake), независимый консультант

# Предисловие

---

## Джефф Сазерленд (Jeff Sutherland)

Впервые я начал применять методологию Scrum<sup>1</sup> в 1993 году в Easel Corporation в Барлингтоне (Burlington), Массачусетс<sup>2</sup>, в сотрудничестве с нашим генеральным директором (СЕО). В 1995 году я начал работать с Кеном Швайбером (Ken Schwaber), который формализовал процесс дистрибуции вновь созданного программного обеспечения по всему миру<sup>3</sup>. 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<sup>4</sup> 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<sup>5</sup>, 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-

---

<sup>1</sup> Scrum — одна из самых популярных методологий гибкой разработки программного обеспечения.

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

<sup>3</sup> Schwaber K. *Scrum Development Process*, OOPSLA Business Object Design and Implementation Workshop, Springer: London, 1997.

<sup>4</sup> Более подробно об этой компании речь пойдет в главе 5.

<sup>5</sup> Sutherland J. *Future of Scrum: Parallel 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 беспорочной 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 (Muga) 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 (Muga), 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 удвойте 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 повторно 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.

# Введение

Идея *бережливости* пришла к нам из 90-х годов и нашла свое отражение в книге *Lean Software Development*, которую мы написали в 2003 году. Мы заметили, что для передовых идей, нашедших применение в производстве и логистике, часто требуется десятилетие или два, чтобы быть использованными в процессах разработки. Поэтому мы подумали, что еще не поздно воспользоваться хорошо себя зарекомендовавшими концепциями 80-х и 90-х годов, чтобы объяснить, почему гибкие (agile) методы являются очень эффективным подходом в разработке программного обеспечения.

Данная стратегия сработала. Книга *Lean Software Development* представляет собой описание набора методов, базирующихся на бережливом мышлении, которое руководители и менеджеры продолжают находить полезным для понимания гибких методов создания программного обеспечения. Книга была приобретена многими разработчиками, которые затем дали ее почитать своим менеджерам, и многие менеджеры затем распространили множество ее копий среди коллег, надеясь, что это поможет осуществить переход к бережливому и гибкому (lean/agile) методам разработки ПО.

Тем временем с идеей произошло нечто необычное. За последние пару лет бережливые инициативы пережили небывалый рост популярности. Слово *lean* (бережливость), которое первоначально приобрело популярность в 90-х годах, использовалось для обозначения японского подхода к производству автомобилей<sup>1</sup>. В последние годы деятельность компаний Honda и Toyota на рынке США была очень успешной, в то время как автопроизводители из Детройта были заняты реструктуризацией. Например, прибыли Toyota возросли с 8 млрд. долл. в фискальном году, который закончился 31 марта 2003 года, до более чем 10 млрд. долл. в 2004 году, 11 млрд. долл. в 2005 году и 12 млрд. долл. в 2006 году. Все это заставило многие компании еще раз присмотреться к концепции *бережливости*, чтобы попытаться понять, что стоит за таким устойчивым и непрерывным успехом.

Бережливые инициативы редко берут начало в подразделениях компании, занимающихся разработкой программного обеспечения или продуктов (изделий), но можно привести множество примеров того, как успешные бережливые инициативы заимствовались для разработки у производства или логистики. Однако практику бережливости в том виде, как она сложилась на производстве, не так то легко адаптировать к среде разработки. В то время как никто не оспаривает действенности принципов бережливости, часто оказывается очень непросто применить соответствующую практику и критерии в среде разработки. В случаях, когда бережливые инициативы “стопорились” в среде разработки, многие компании обнаруживали, что книга *Lean Software Development* предоставляет солидную основу для переосмысления подходов и адаптации идей бережливости к условиям организации, занятой разработкой.

Преимущества бережливого и гибкого подхода к разработке ПО в последние несколько лет приобрели широкую известность и признание, и многие организации изменили

---

<sup>1</sup> Womack J., Jones D., Roos D. *The Machine That Changed the World*, Rawson Associates, 1990.

свои методы создания программного обеспечения. Мы путешествовали по всему миру, посещая многие организации во время, когда делались попытки внедрить новые подходы, и мы многому научились в результате общения с людьми, упорно работающими над тем, чтобы изменить свои методы создания ПО. По мере того как рос объем наших знаний, то же происходило и со спросом на умение реализовать бережливый подход к разработке программного обеспечения. Мы осознали, что новая книга позволит нам поделиться тем, что мы узнали, с гораздо большим количеством людей, чем это возможно через личные контакты. Поэтому мы суммировали наш опыт в этой книге.

Это не поваренная книга, содержащая рецепты по реализации бережливого подхода. Так же как и наша предыдущая книга, это набор интеллектуальных методов, позволяющих подойти к адаптации принципов бережливости в вашей среде. Мы начали там, где закончилась первая книга, и окунулись глубже в вопросы и проблемы, с которыми приходится сталкиваться при попытке реализовать бережливые и гибкие методы в разработке ПО. Данная книга может рассматриваться как продолжение *Lean Software Development*. Вместо повторения того, что было в первой книге, здесь мы рассматриваем соответствующие вопросы под иным ракурсом. Мы исходим из того, что читатель убежден, что бережливый подход к разработке программного обеспечения — это хорошая идея, и сосредотачиваемся на основных вопросах его успешной реализации. Мы рассматриваем ключевые аспекты реализации и обсуждаем, что важно, что нет и почему. Нашей целью является помочь организациям перейти к более эффективной разработке программного обеспечения.

Первая глава книги посвящена истории бережливого подхода, а во второй рассматриваются семь принципов бережливого подхода к разработке ПО, представленных в *Lean Software Development*. Далее следуют главы, посвященные *ценности, потерям, скорости, людям, знанию, качеству, партнерам* и предстоящему *путешествию*. Каждая из этих восьми глав начинается с истории, иллюстрирующей, как некая организация сумела решить стоящую перед ней проблему. После этого следует обсуждение основных вопросов, которые мы (авторы) сочли важными; здесь же приводятся краткие истории, иллюстрирующие эти вопросы, а также содержащие ответы на типичные вопросы, которые мы часто слышим. Каждая глава завершается набором упражнений, призванных помочь читателю исследовать тему более глубоко.