ЧАСТЬ VI

Завершение и начало

В этой части

В этой части мы продолжим обсуждение нового подхода к объектно-ориентированному проектированию. В частности, будет рассмотрен вопрос о том, как этот новый подход применялся при проектировании и реализации шаблонов проектирования. В завершении приводится обширный список рекомендуемой литературы.

Глава	Предмет обсуждения
21	Обзор действующих сил и взаимосвязей в шаблонах проектирования в контексте нового подхода к объектно-ориентированному проектированию
22	Предлагается литература и указываются другие ресурсы, полезные для дальнейшего изучения

Глава 21

Шаблоны проектирования и новый взгляд на объектно-ориентированное проектирование

Введение

Завершая чтение книги, всегда полезно задать себе вопрос, что нового удалось из нее почерпнуть. В этой книге авторами была предпринята попытка дать читателю лучшее и, возможно, новое понимание принципов объектно-ориентированного проектирования, достигнутое за счет изучения шаблонов проектирования и разъяснения того, как эти шаблоны проектирования раскрывают объектно-ориентированную парадигму.

В этой главе мы выполним следующее.

- Познакомимся с новым взглядом на принципы объектно-ориентированного проектирования, базирующимся на понятии шаблонов проектирования.
- Выясним, как шаблоны проектирования могут помочь разработчику инкапсулировать реализацию.
- Обсудим метод анализа общности/изменчивости и выясним, как шаблоны проектирования помогают понять назначение абстрактных классов.
- Узнаем, как выполнить декомпозицию проблемной области на основе существующих в ней обязательств.
- Рассмотрим методы определения отношений между объектами.
- Еще раз вернемся к методу проектирования от контекста с использованием шаблонов проектирования.

В конце главы читателю будут предложены некоторые выводы и обобщения, основанные на практическом опыте авторов.

Перечень принципов объектно-ориентированного проектирования

В ходе обсуждения шаблонов проектирования упоминалось несколько важных принципов объектно-ориентированной парадигмы. Подводя итог, эти принципы можно сформулировать следующим образом.

- Объекты обладают четко определенными обязательствами.
- Объекты отвечают только за собственное поведение.
- Инкапсуляция подразумевает любой вид сокрытия, а именно:
  - сокрытие данных;
  - сокрытие класса (за абстрактным классом или интерфейсом);
  - сокрытие реализации.
- Выявление вариаций в поведении и данных с помощью анализа общности/изменчивости.
- Проектирование интерфейсов, а не реализации.
- Понимание наследования как метода концептуализации вариаций, а не механизма определения особых версий уже существующих объектов.
- Исключение взаимосвязанности различных вариаций в одном и том же классе.
- Стремление к поддержанию низкой связанности в системе.
- Стремление к поддержанию высокой связности в объектах.
- Повсеместное соблюдение правила "однажды и только однажды" в отношении реализации обязательств.

Как шаблоны проектирования инкапсулируют реализацию

Для некоторых из представленных в этой книге шаблонов проектирования характерно то, что они скрывают детали реализации от объекта-клиента. Например, шаблон Bridge скрывает от объекта-клиента подробности реализации классов, производных от класса ***`Absraction`***. Кроме того, интерфейс ***`Implementation`*** скрывает семейство классов реализации для класса ***`Absraction`*** и его производных классов. В шаблоне Strategy от объекта-клиента скрыта реализация каждого класса **`ConcreteStrategy`**. Это правило справедливо для большинства шаблонов, описанных "бандой четырех", — все они позволяют скрыть конкретную реализацию.

Ценность подобного сокрытия реализации состоит в том, что благодаря ему шаблоны позволяют легко добавлять новые реализации, так как объекты-клиенты ничего не знают о том, как работают уже существующие реализации.

Анализ общности/изменчивости и шаблоны проектирования

В главе 9, *Шаблон Bridge*, было показано, как с помощью анализа общности/изменчивости можно вывести шаблон Bridge. Аналогичным образом могут быть выведены и многие другие шаблоны, включая Strategy, Iterator, Proxy, State, Visitor, Template Method и Abstract Factory. Однако более важно то, сколько шаблонов может быть *применено* за счет проведения анализа общности/изменчивости, поскольку поиск общностей в проблемной области помогает обнаружить присутствие в ней шаблонов.

Например, в отношении шаблона Bridge, можно начать анализ с рассмотрения нескольких конкретных требований к системе:

- вычерчивание квадрата с помощью первой графической программы;
- вычерчивание окружности с помощью второй графической программы;
- вычерчивание прямоугольника с помощью первой графической программы.

Знание шаблона Bridge позволяет выделить в этих конкретных случаях две общности:

- графические программы;
- отображаемые геометрические фигуры.

Аналогично, знание шаблона Strategy подсказывает, что если существует несколько различных правил, то следует искать в них возможную общность, которую затем можно будет инкапсулировать.

Но изучение шаблонов на этом не заканчивается. Необходимо продолжать читать литературу. Шаблонам посвящены различные дискуссии, проводимые на основании практического опыта анализа и проектирования. Шаблоны предоставляют команде разработчиков единый инструментарий для обсуждения проблемы, а также позволяют включить в создаваемый код лучшие практические решения, найденные другими программистами.

Декомпозиция проблемной области в обязательства

Анализ общности/изменчивости позволяет найти концептуальное решение (общность) и подготовить решение на уровне реализации (каждая конкретная вариация). Если учитывать только общности и объекты, использующие их, то можно подойти к решению проблемы с другой стороны — с помощью метода декомпозиции в обязательства.

Например, в шаблоне Bridge проблемная область рассматривается как состоящая из двух различных типов сущностей (абстракций и реализаций). Следовательно, не стоит ограничивать себя только объектно-ориентированной декомпозицией (т.е. разложением проблемной области на объекты), будет полезно также попробовать разложить проблемную область в обязательства, что может оказаться даже проще. В этом случае возможно предварительно определить объекты, которые потребуются для реализации найденных обязательств, и лишь затем переходить к обычной объектной декомпозиции.

Выше сказанное — это всего лишь расширение правила, которое я уже упоминал выше: проектировщик не должен беспокоиться о том, как будут создаваться экземпляры объектов, пока не станет известно, в каких именно объектах он нуждается. Это правило можно понимать как требование декомпозиции поставленной задачи на две части:

- определить, какие объекты необходимы;
- принять решение, как будут создаваться экземпляры этих объектов.

Те или иные шаблоны часто подсказывают нам, как можно выполнить декомпозицию обязательств. Например, шаблон Decorator демонстрирует, как обеспечить гибкое комбинирование объектов, если после декомпозиции проблемная область включает набор основных обязательств, которые используются всегда (класс **`ConcreteComponent`**), и некоторое множество вариаций, используемых от случая к случаю (классы-декораторы). Шаблон Strategy демонстрирует декомпозицию проблемы на объект, который использует правила, и собственно используемые правила.

Отношения внутри шаблона

Должен заметить, что на проводимых мной занятиях я иногда позволяю себе некоторую вольность, приводя определенную цитату из книги Александера. После того как две трети дня были посвящены обсуждению того, насколько хороши шаблоны проектирования, я беру в руки его книгу *Timeless Way of Building*, открываю последнюю страницу и говорю так.

> В этой книге 549 страниц. На странице 545, которая, очевидно, одна из самых последних в книге, Александер говорит следующее: "На этой заключительной стадии шаблоны уже не важны…".[1]

Я делаю паузу и замечаю: "Было бы прекрасно, если бы он сказал об этом в начале книги — это могло бы сэкономить нам уйму времени". Затем я продолжаю цитировать книгу: "Шаблоны учат нас быть восприимчивыми к реальности".[2]

И заканчиваю я такими словами: "Если вы прочтете книгу Александера, то поймете, что реальность — это отношения и движущие силы, описываемые шаблонами".

Шаблоны предоставляют нам способ говорить об этой реальности. Однако для нас важны вовсе не шаблоны сами по себе. Это же справедливо и по отношению к шаблонам проектирования в области разработки программного обеспечения.

Шаблон описывает движущие силы, мотивы и отношения для определенной проблемы в определенном контексте и предлагает подход к ее решению. Например, шаблон Bridge представляет отношения между классами, производными от определенной абстракции, и их возможными реализациями. Шаблон Strategy описывает отношения между следующими элементами:

- классом, который использует один из наборов алгоритмов (**`Context`**);
- членами этого набора алгоритмов (классы стратегий);

---

[1] *Alexander C., Ishikawa S., Silverstein M. The Timeless Way of Building, NY: Oxford University Press, 1979, с. 545.*

[2] *Там же, с. 545.*

- классом-клиентом, который использует класс `Context` и определяет, какой из алгоритмов следует использовать.

Шаблоны и проектирование от контекста

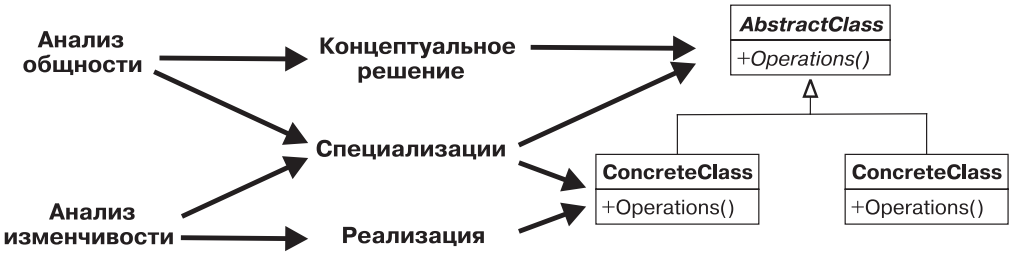
При обсуждении проблемы САПР в начале этой книги было показано, как шаблоны проектирования используются, если внимание сосредоточено на контексте, который они создают друг для друга. Шаблоны проектирования, функционирующие совместно, помогают найти оптимальную структуру приложения. Полезно будет указать, что многие шаблоны представляют собой типичные примеры проектирования от контекста.

Например:

- шаблон Bridge указывает на необходимость определять классы стороны реализации в контексте классов, производных от класса ***`Abstraction`***;
- шаблон Decorator требует проектировать классы-декораторы в пределах контекста исходного компонента;
- шаблон Abstract Factory требует определять создаваемые семейства объектов в пределах контекста проблемы в целом, что позволяет установить, экземпляры каких именно классов должны быть реализованы.

Фактически, разработку с использованием интерфейсов и полиморфизма в общем случае можно рассматривать как один из видов проектирования по контексту. Взгляните на рис. 21.1, который представляет собой повторение рис. 8.4. Обратите внимание на то, что интерфейс абстрактного класса определяет тот контекст, в пределах которого должны быть реализованы все производные от него классы.

Анализируя, что именно эти объекты должны выполнять (концептуальное решение), мы определяем, как к ним следует обращаться (спецификации).



При реализации этих классов следует убедиться, что выбранное API предоставляет достаточно информации для их правильной реализации и связывания.

РИС. 21.1. ***Взаимосвязи между анализом общности/изменчивости, уровнями детализации и абстрактными классами***

Дополнительные замечания

При изучении шаблонов проектирования будет весьма полезно рассмотреть следующие факторы и концепции.

- **Какие реализации этот шаблон скрывает?** Это позволит впоследствии изменять их.
- **Какие общности представлены в этом шаблоне?** Это поможет идентифицировать их.
- **Какими обязательствами обладают объекты в этом шаблоне?** Это может упростить выполнение декомпозиции обязательств.
- **Каковы отношения между заданными объектами?** Это предоставит информацию о движущих силах, представленных этими объектами.
- **Предоставляет ли сам шаблон некоторый пример проектирования по контексту?** Это позволит лучше понять, почему применение шаблонов является хорошим стилем проектирования.

Резюме

В этой главе подведен общий итог обсуждению нового подхода к объектно-ориентированному проектированию и показано, как шаблоны проектирования проявляются себя в этом. Рассматривая шаблоны проектирования, полезно ответить на вопросы, перечисленные ниже.

- Что инкапсулируют шаблоны проектирования?
- Как в них используется анализ общности/изменчивости?
- Как в таких шаблонах выполняется декомпозиция обязательств в проблемной области?
- Как они определяют отношения между объектами?
- Как они иллюстрируют проектирование по контексту?

Глава 22

Библиография

Эта книга — всего лишь введение. Введение в шаблоны проектирования, объектно-ориентированное проектирование и другие более мощные способы проектирования прикладных программных систем. Хотелось бы надеяться, что она обогатила вас определенными навыками, которые позволят вам реализовать на практике этот мощный и полезный стиль творческого мышления.

Каким может быть следующий этап в освоении идей, изложенных в этой книге? Чтобы помочь вам найти ответ на этот вопрос, в заключительной главе приведен аннотированный список источников, рекомендуемых к серьезному изучению.

В этой главе вы найдете следующее.

- Адреса Web-сайтов, дополняющих материал этой книги.
- Рекомендации для:
  - желающих углубить свои знания о шаблонах проектирования;
  - разработчиков на языке Java;
  - разработчиков на языке C++;
  - программистов на языке COBOL, которые хотят изучить объектно-ориентированную технологию;
  - всех желающих освоить новую мощную технологию разработки программного обеспечения, называемую XP (eXtreme Programming — экстремальное программирование).
- И, наконец, список книг, которые оказали влияние на меня лично и помогли понять, что жизнь — это нечто большее, чем программирование, и что гармоничное развитие личности поможет каждому стать действительно хорошим программистом.

Web-сайт поддержки данной книги

Web-сайт поддержки данной книги расположен по адресу:

```
http://www.netobjectives.com/dpexplained
```

На этом сайте можно найти такую дополнительную информацию о шаблонах проектирования.

- Примеры программного кода, ответы на часто задаваемые вопросы, материалы дискуссий, организованные по отдельным главам книги.
- Материалы дискуссий по проблемам рефакторинга.

- Сводные данные о шаблонах проектирования, представленные в удобном ссылочном формате.
- Описание курсов по изучению шаблонов проектирования и других близких тем.

Там же можно найти форму, чтобы отправить нам ваши комментарии и вопросы, касающиеся данной книги.

Кроме того, нами издается электронный журнал (e-zine), посвященный шаблонам проектирования и общим аспектам объектно-ориентированного проектирования. Для оформления подписки достаточно послать по электронной почте письмо с вашим именем и названием компании по адресу `info@netobjectives.com` со словом "`subscribe`" в поле `subject` (тема).

Рекомендуемая литература по шаблонам проектирования и объектно-ориентированной технологии

Я рекомендую к прочтению следующие книги и справочные руководства по объектно-ориентированному программированию и языку UML.

- Fowler M. *Refactoring: Improving the Design of Existing Code*, Reading, MA: Addison-Wesley, 2000.

  Это наиболее полная известная мне работа о рефакторинге.

- Fowler M., Scott K. *UML Distilled: A Brief Guide to the Standard Object Modeling Language, 2-nd Edition*, Reading, MA: Addison-Wesley, 2000.

  Этот источник я считаю лучшим для изучения языка UML. Данная книга будет полезной для начинающих, а также может использоваться как справочное руководство. Лично я пользуюсь ею постоянно.

- Meyer B. *Object-Oriented Software Construction*, Upper Saddle River, N.J.: Prentice Hall, 1997.

  Невероятно насыщенная книга, написанная одним из наиболее выдающихся умов в нашей области.

Предмет, называемый "шаблонами проектирования", продолжает развиваться и углубляться. Изучать его можно на различных уровнях и со многих точек зрения. Я рекомендую следующие книги и справочные пособия, которые помогут вам на этом пути.

- Alexander C., Ishikawa S., Silverstein M. *The Timeless Way of Building*, New York, NY: Oxford University Press, 1979.

  Это моя любимая книга как в профессиональном, так и в личном плане. Она одновременно интересна и глубока. Если вы прочитаете только одну книгу из данного списка, это должна быть именно она.

- Alexander C., Ishikawa S., Silverstein M. *A Pattern Language: Towns/Buildings/Construction*, New York, NY: Oxford University Press, 1977.
- Alexander C., Ishikawa S., Silverstein M. *Notes on Synthesis of Form*, New York, NY: Oxford University Press, 1970.

- Coplien J. *Multi-Paradigm Design for C++*, Reading, MA: Addison-Wesley, 1998.

  Главы 2–5 этой книги следует прочитать даже тем, кто разрабатывает программное обеспечение на языке, отличном от C++. Данная книга описывает метод анализа общности/изменчивости лучше, чем какая-либо другая. На Web-сайте нашей книги доступна интерактивная версия докторской диссертации Джима Коплина, которая представляет собой эквивалент этой книги.

- Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*, Reading, MA: Addison-Wesley, 1995.

  Это издание продолжает оставаться лучшей из всех изданных книг о шаблонах проектирования. Ознакомление с ней является обязательным для каждого разработчика, работающего на языке C++.

- Gardner K. *Cognitive Patterns: Problem-Solving Frameworks for Object Technology*, New York, NY: Cambridge University Press, 1998.

  Это взгляд на шаблоны проектирования с точки зрения науки о познании и искусственного интеллекта. На доктора Гарднера также оказала большое влияние книга Александера, приведенная первой в этом списке.

- Schmidt D., Stal M., Rohnert H., Busehmann F. *Pattern-Oriented Software Architecture, Vol. 2*, New York, NY: John Wiley, 2000.

  Книга об использовании шаблонов проектирования в многопоточных и распределенных средах.

- Vlissides J. *Pattern Hatching*, Reading, MA: Addison-Wesley, 1998.

  Это отличная книга о шаблонах проектирования для подготовленных читателей. Иллюстрирует несколько способов организации совместной работы шаблонов. Предварительно я рекомендую прочитать данную книгу и книгу "банды четырех".

Рекомендуемая литература для программистов на языке Java

Когда я только начинал изучение языка Java, моими любимыми книгами были следующие.

- Eckel B. *Thinking in Java, 2nd Edition*, Upper Saddle River, N.J.: Prentice Hall, 2000.

  Это одна из лучших книг о языке Java из числа изданных. Электронную версию этой книги можно найти по адресу `http://www.eckelobjects.com/DownloadSites`.

- Horstmann C. *Core Java 2, Volume 1, Fundamentals*, Palo Alto: Pearson Education, 1999. (Русский перевод этой книги Кея Хорстманна, *Java 2. Библиотека профессионала. Том 1. Основы*, готовится к выпуску издательским домом "Вильямс" в третьем квартале 2002 г.)

  Еще одна хорошая книга для изучения основ языка Java.

Каждый язык программирования имеет собственный набор компонентов для реализации шаблонов проектирования. Если речь идет о языке Java, то я могу порекомендовать следующие книги.

- Coad P. *Java Design*, Upper Saddle River, N.J.: Prentice Hall, 2000.

  Если вы считаете себя профессиональным разработчиком на языке Java, то я настоятельно рекомендую вам прочитать эту книгу. Здесь описывается большинство принципов и стратегий, которые будут весьма полезны при использовании шаблонов проектирования, несмотря на то, что собственно шаблоны в ней не упоминаются.

- Grand M. *Patterns in Java, Vol. 1*, New York, NY: John Wiley, 1998.

  Если вы программируете на языке Java, эта книга также принесет вам большую пользу. В ней приведены интересные примеры программного кода и используется язык UML. Однако, по нашему мнению, обсуждение движущих сил и мотиваций в книге "банды четырех" более полезно, чем то, которое представлено в этой книге. Тем не менее, изучение отличного набора предлагаемых примеров представляет большую ценность, особенно для тех, кто практически работает с языком Java.

- Информацию об API языка Java для классов `Observer` и `Observable` можно найти по адресу `http://java.sun.com/j2se/1.3/docs/api/index.html`.

Особого внимания в языке Java требует работа с потоками. Для углубленного изучения этой проблемы я рекомендую следующие издания.

- Hollub A. *Taming Java Threads*, Berkeley, CA: APress, 2000.
- Hyde P. *Java Thread Programming: The Authoritative Solution*, Indianapolis, IN: SAMS, 1999.
- Lea D. *Concurrent Programming in Java: Design Principles and Patterns, Second Edition*, Reading, MA: Addison-Wesley, 2000.

Рекомендуемая литература для программистов на языке C++

Для тех, кто программирует на языке C++ в среде UNIX, я нахожу необходимым ознакомиться со следующей книгой.

- Stevens W. *Advanced Programming in the UNIX Environment*, Reading, MA: Addison-Wesley, 1992.

  Это **обязательный** ресурс для каждого разработчика на языке C++ в среде UNIX.

Рекомендуемая литература для программистов на языке COBOL

Программистам, работающим с языком COBOL, которые хотят изучить объектно-ориентированное проектирование, можно порекомендовать следующий источник.

- Levey R. *Reengineering Cobol with Objects*, New York, NY: McGraw-Hill, 1995.

Полезная книга для программистов, работающих на языке COBOL и стремящихся освоить объектно-ориентированное проектирование.

Рекомендуемая литература для изучения технологии экстремального программирования

Для тех, кто хочет ознакомиться с технологией экстремального программирования (XP) или повысить свое мастерство в этой области, можно рекомендовать следующие два источника.

- `http://www.netobjectives.com/xp`

  Наш собственный Web-сайт, посвященный экстремальному программированию и включающий статьи и курсы лекций по технологии XP.

- Beck K. *Extreme Programming Explained: Embrace Change*, Reading, MA: Addison-Wesley, 2000.

  Это издание заслуживает внимания каждого, кто имеет отношение к разработке программного обеспечения, даже если он не планирует применять XP на практике. Я выбрал в этой книге 30 или около того страниц, содержащих, по моему мнению, важнейшие положения этой методологии, и поместил их список на нашем XP-сайте.

В настоящее время мы активно разрабатываем собственный метод проектирования программного обеспечения, который мы назвали Pattern-Accellerated Software Engineering. Он интегрирует несколько методов анализа и проектирования. Подробности можно найти по адресу `http://www.netobjectives.com/pase`.

Рекомендуемая литература по общим вопросам программирования

Данная книга отражает мою собственную философию, позволяет заглянуть в себя и увидеть, как каждому можно усовершенствоваться самому и улучшить свою работу.

- Hunt A., Thomas D. *The Pragmatic Programmer: From Journeyman to Master*, Reading, MA: Addison-Wesley, 2000.

  Это одна из тех прекрасных книг, которые я читаю по несколько страниц в день. Когда я встречаю упоминание о чем-то, что я уже использую, это укрепляет мою самооценку. Когда же я нахожу что-то новое, то получаю прекрасную возможность поучиться.

Наши любимые книги

Лично я полагаю, что лучший разработчик программного обеспечения — это не тот, кто живет и дышит одним лишь программированием. Пожалуй, именно способность думать и слушать, целостность и глубина личности, а также творческий подход — вот то, что, по моему мнению, присуще действительно хорошему разработчику. Такой человек

более коммуникабелен. Он способен находить полезные идеи в других дисциплинах (мы, например, успешно применили в своей практике достижения из области архитектуры и антропологии). Системы, созданные такими разработчиками, ориентированы на людей, для которых эти системы, собственно, и предназначаются.

Многие студенты спрашивают нас о том, что мы любим читать, что оказало влияние на формирование наших взглядов и развитие наших личностей. Вот то, что можно было бы ответить на этот вопрос.

Алан рекомендует следующие издания.

- Grieve B. *The Blue Day Book: A Lesson in Cheering You Up*, Kansas City, KA: Andrews McMeel Publishing, 2000.

  Это забавная и восхитительная книга. Обращайтесь к ней всякий раз, когда чувствуете себя не в своей тарелке.

- Hill N. *Think and Grow Rich*, New York, NY: Ballantine Books, 1960.

  "Богатство" означает здесь не только деньги — данное слово обозначает все то, чем вы желаете обладать в своей жизни. Эта книга оказала большое влияние на мой личный успех и успех моего бизнеса.

- Kundtz D. *Stopping: How to Be Still When You Have to Keep Going*, Berkeley, CA: Conari Press, 1998.

  Книга учит, как перестать быть трудоголиком. Она является прекрасным руководством в том, как отказаться от постоянного беспокойства и научиться наслаждаться жизнью, всегда добиваясь поставленных целей.

- Mandino O. *The Greatest Salesman in the World*, New York, NY: Bantam Press, 1968.

  Я прочитал и применил положения из этой книги на практике несколько лет назад. Это помогло мне найти в жизни тот путь, который я всегда хотел найти. Если вы обратитесь к данной книге, я настоятельно рекомендую делать все то, о чем говорится в свитках Хафида, а не ограничиваться лишь их прочтением (вы поймете, что я имею в виду, когда будете читать эту книгу).

- Pilzer P. *Unlimited Wealth: The Theory and Practice of Economic Alchemy*, Crown Publishers, 1990.

  Эти книга представляет как новую парадигму ресурсов и источников благосостояния, так и рекомендации, как ими можно воспользоваться. Она будет крайне полезна каждому, кто живет в наш век информации.

- Remen R. *My Grandfather's Blessings: Stories of Strength, Refuge, and Belonging*, New York, NY: Riverhead Books, 2000.

  Прекрасная книга, отражающая молитвы каждого достойного человека.

Джим рекомендует следующие издания.

- Buzan T., Buzan B. *The Mind Map Book: How to Use Radiant Thinking to Maximize Your Brain's Untapped Potential*, New York, NY: Dutton Books, 1994.

  Эта книга оказала революционное влияние на мой стиль преподавания, общения с людьми, мышления и ведения записей. Невероятно мощная техника. Я пользуюсь ей ежедневно.

- Cahill T. *How the Irish Saved Civilization*, New York, NY: Doubleday, 1995.

  Если в ваших жилах течет хоть немного ирландской крови, вы почувствуете гордость.

- Dawson C. *Religion and the Rise of Western Culture*, New York, NY: Doubleday, 1950.

  Здесь описано, как религия направляла развитие западной цивилизации и способствовала удержанию в узде "варварства, которое скрывается внутри каждого из нас". Излагаются важные взгляды на научное мышление.

- Jensen B. *Simplicity: The New Competitive Advantage in a World of More, Better, Faster*, Cambridge, MA: Perseus Books, 2000.

  Революция в мышлении и управлении знаниями. Системы проектирования будут проще в использовании для людей, если особенности человека были приняты во внимание при разработке процессов и технологий.

- Lingenfelter S. *Transforming Culture*, Grand Rapids: Baker Book House, 1998.

  Модель для понимания особенностей культур через теорию социальных игр.

- Spradely J. P. *The Ethnographic Interview*, New York, NY: Harcourt Brace Jovanovich College Publishers, 1979.

  Эту книгу следует прочесть каждому, кто хочет научиться брать интервью. Классический учебник, знакомый всем студентам, изучающим антропологию.

- Wiig K. *Knowledge Management Methods*, Dallas, AL: Schema Press, 1995.

  Виртуальная энциклопедия методов, помогающих организациям более эффективно использовать информационные ресурсы.