могут изменяться во время выполнения и потому более динамичны. Почти все паттерны в какой-то мере используют наследование. Поэтому к категории «паттерны классов» отнесены только те, что сфокусированы лишь на отношениях между классами. Обратите внимание: большинство паттернов действуют на уровне объектов.

Порождающие паттерны классов частично делегируют ответственность за создание объектов своим подклассам, тогда как порождающие паттерны объектов передают ответственность другому объекту. Структурные паттерны классов используют наследование для составления классов, в то время как структурные паттерны объектов описывают способы сборки объектов из частей. Поведенческие паттерны классов используют наследование для описания алгоритмов и потока управления, а поведенческие паттерны объектов описывают, как объекты, принадлежащие некоторой группе, совместно функционируют и выполняют задачу, которая ни одному отдельному объекту не под силу.

Существуют и другие способы классификации паттернов. Некоторые паттерны часто используются вместе. Например, компоновщик применяется с итератором или посетителем. Некоторыми паттернами предлагаются альтернативные решения. Так, прототип нередко можно использовать вместо абстрактной фабрики. Применение части паттернов приводит к схожему дизайну, хотя изначально их назначение различно. Например, структурные диаграммы компоновщика и декоратора похожи.

Классифицировать паттерны можно и по их ссылкам (см. разделы «Родственные паттерны»). На рис. 1.1 такие отношения изображены графически.

Ясно, что организовать паттерны проектирования допустимо многими способами. Оценивая паттерны с разных точек зрения, вы глубже поймете, как они функционируют, как их сравнивать и когда применять тот или другой.

1.6. Как решать задачи проектирования с помощью паттернов

Паттерны проектирования позволяют разными способами решать многие задачи, с которыми постоянно сталкиваются проектировщики объектно-ориентированных приложений. Поясним эту мысль примерами.

Поиск подходящих объектов

Объектно-ориентированные программы состоят из объектов. *Объект* сочетает данные и процедуры для их обработки. Такие процедуры обычно называют *методами* или *операциями*. Объект выполняет операцию, когда получает *запрос* (или *сообщение*) от *клиента*.

Посылка запроса – это *единственный* способ заставить объект выполнить операцию. А выполнение операции – *единственный* способ изменить внутреннее состояние объекта. Имея в виду два эти ограничения, говорят, что внутреннее состояние объекта *инкапсулировано*: к нему нельзя получить непосредственный доступ, то есть представление объекта закрыто от внешней программы.

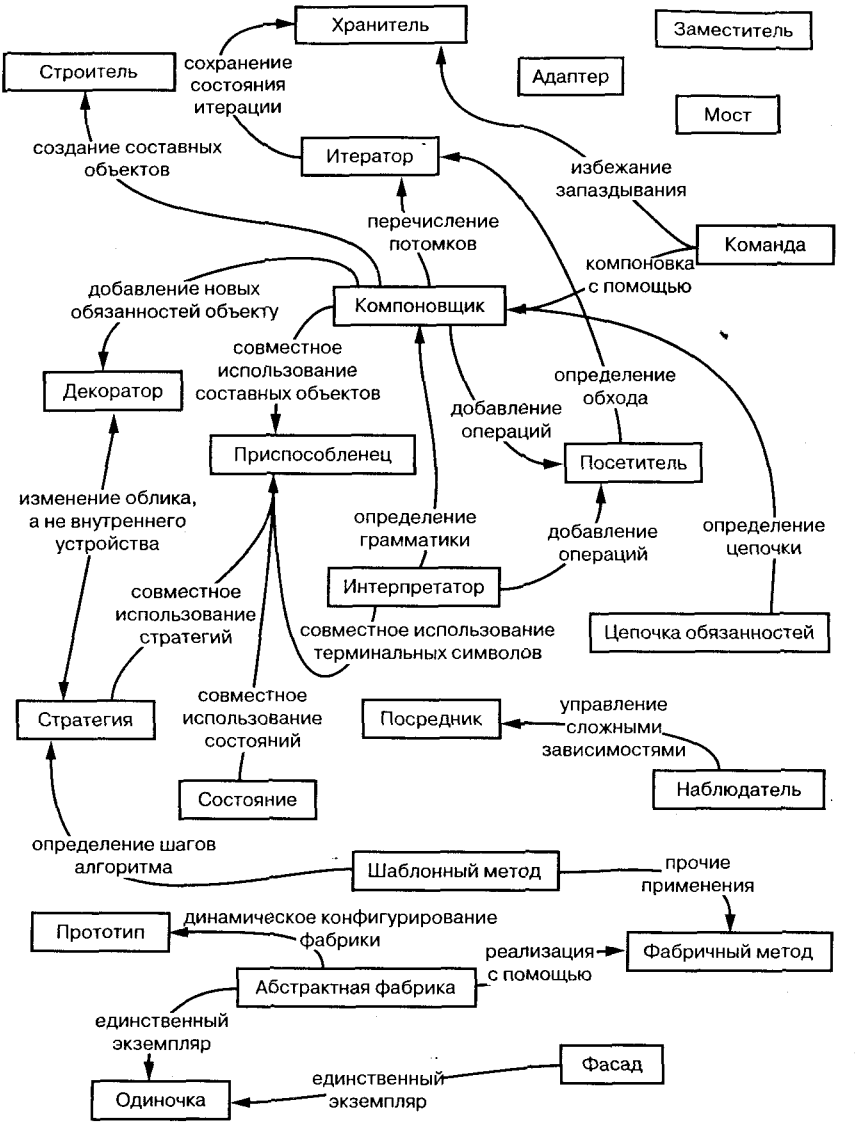

Рис. 1.1. Отношения между паттернами проектирования

Самая трудная задача в объектно-ориентированном проектировании – разложить систему на объекты. При решении приходится учитывать множество факторов: инкапсуляцию, глубину детализации, наличие зависимостей, гибкость, производительность, развитие, повторное использование и т.д. и т.п. Все это влияет на декомпозицию, причем часто противоречивым образом.

Методики объектно-ориентированного проектирования отражают разные подходы. Вы можете сформулировать задачу письменно, выделить из получившейся

фразы существительные и глаголы, после чего создать соответствующие классы и операции. Другой путь – сосредоточиться на отношениях и разделении обязанностей в системе. Можно построить модель реального мира или перенести выявленные при анализе объекты на свой дизайн. Согласие по поводу того, какой подход самый лучший, никогда не будет достигнуто.

Многие объекты возникают в проекте из построенной в ходе анализа модели. Но нередко появляются и классы, у которых нет прототипов в реальном мире. Это могут быть классы как низкого уровня, например массивы, так и высокого. Паттерн компоновщик вводит такую абстракцию для единообразной трактовки объектов, у которой нет физического аналога. Если придерживаться строгого моделирования и ориентироваться только на реальный мир, то получится система, отражающая сегодняшние потребности, но, возможно, не учитывающая будущего развития. Абстракции, возникающие в ходе проектирования, – ключ к гибкому дизайну.

Паттерны проектирования помогают выявить не вполне очевидные абстракции и объекты, которые могут их использовать. Например, объектов, представляющих процесс или алгоритм, в действительности нет, но они являются неотъемлемыми составляющими гибкого дизайна. Паттерн стратегия описывает способ реализации взаимозаменяемых семейств алгоритмов. Паттерн состояние позволяет представить состояние некоторой сущности в виде объекта. Эти объекты редко появляются во время анализа и даже на ранних стадиях проектирования. Работа с ними начинается позже, при попытках сделать дизайн более гибким и пригодным для повторного использования.

Определение степени детализации объекта

Размеры и число объектов могут сильно варьироваться. С их помощью может быть представлено все, начиная с уровня аппаратуры и до законченных приложений. Как же решить, что должен представлять собой объект?

Здесь и потребуются паттерны проектирования. Паттерн фасад показывает, как представить в виде объекта целые подсистемы, а паттерн приспособленец – как поддержать большое число объектов при высокой степени детализации. Другие паттерны указывают путь к разложению объекта на меньшие подобъекты. Абстрактная фабрика и строитель описывают объекты, единственной целью которых является создание других объектов, а посетитель и команда – объекты, отвечающие за реализацию запроса к другому объекту или группе.

Специфицирование интерфейсов объекта

При объявлении объектом любой операции должны быть заданы: имя операции, объекты, передаваемые в качестве параметров, и значение, возвращаемое операцией. Эту триаду называют *сигнатурой* операции. Множество сигнатур всех определенных для объекта операций называется *интерфейсом* этого объекта. Интерфейс описывает все множество запросов, которые можно отправить объекту. Любой запрос, сигнатура которого соответствует интерфейсу объекта, может быть ему послан.