- *паттерны проектирования менее специализированы, чем каркасы.* Каркас всегда создается для конкретной предметной области. В принципе каркас графического редактора можно использовать для моделирования работы фабрики, но его никогда не спутаешь с каркасом, предназначенным специально для моделирования. Напротив, включенные в наш каталог паттерны разрешается использовать в приложениях почти любого вида. Хотя, безусловно, существуют и более специализированные паттерны (скажем, паттерны для распределенных систем или параллельного программирования), но даже они не диктуют выбор архитектуры в той же мере, что и каркасы.

Значение каркасов возрастает. Именно с их помощью объектно-ориентированные системы можно использовать повторно в максимальной степени. Крупные объектно-ориентированные приложения составляются из слоев взаимодействующих друг с другом каркасов. Дизайн и код приложения в значительной мере определяются теми каркасами, которые применялись при его создании.

1.7. Как выбирать паттерн проектирования

Если в распоряжение проектировщика предоставлен каталог из более чем 20 паттернов, трудно решать, какой паттерн лучше всего подходит для решения конкретной задачи проектирования. Ниже представлены разные подходы к выбору подходящего паттерна:

- *подумайте, как паттерны решают проблемы проектирования.* В разделе 1.6 обсуждается то, как с помощью паттернов можно найти подходящие объекты, определить нужную степень их детализации, специфицировать их интерфейсы. Здесь же говорится и о некоторых иных подходах к решению задач с помощью паттернов;
- *пролистайте разделы каталога, описывающие назначение паттернов.* В разделе 1.4 перечислены назначения всех представленных паттернов. Ознакомьтесь с целью каждого паттерна, когда будете искать тот, что в наибольшей степени относится к вашей проблеме. Чтобы сузить поиск, воспользуйтесь схемой в таблице 1.1;
- *изучите взаимосвязи паттернов.* На рис. 1.1 графически изображены соотношения между различными паттернами проектирования. Данная информация поможет вам найти нужный паттерн или группы паттернов;
- *проанализируйте паттерны со сходными целями.* Каталог состоит из трех частей: порождающие паттерны, структурные паттерны и паттерны поведения. Каждая часть начинается со вступительных замечаний о паттернах соответствующего вида и заканчивается разделом, где они сравниваются друг с другом;
- *разберитесь в причинах, вызывающих перепроектирование.* Взгляните на перечень причин, приведенный выше. Быть может, в нем упомянута ваша проблема? Затем обратитесь к изучению паттернов, помогающих устранить эту причину;

- *посмотрите, что в вашем дизайне должно быть изменяющимся.* Такой подход противоположен исследованию причин, вызвавших необходимость перепроектирования. Вместо этого подумайте, что могло бы *заставить* изменить дизайн, а также о том, что бы вы хотели изменять без перепроектирования. Акцент здесь делается на *инкапсуляции сущностей, подверженных изменениям*, а это предмет многих паттернов. В таблице 1.2 перечислены те аспекты дизайна, которые разные паттерны позволяют варьировать независимо, устраняя тем самым необходимость в перепроектировании.

1.8. Как пользоваться паттерном проектирования

Как пользоваться паттерном проектирования, который вы выбрали для изучения и работы? Вот перечень шагов, которые помогут вам эффективно применить паттерн:

1. *Прочитайте описание паттерна, чтобы получить о нем общее представление.* Особое внимание обратите на разделы «Применимость» и «Результаты» – убедитесь, что выбранный вами паттерн действительно подходит для решения ваших задач.
2. *Вернитесь назад и изучите разделы «Структура», «Участники» и «Отношения».* Убедитесь, что понимаете упоминаемые в паттерне классы и объекты и то, как они взаимодействуют друг с другом.
3. *Посмотрите на раздел «Пример кода», где приведен конкретный пример использования паттерна в программе.* Изучение кода поможет понять, как нужно реализовывать паттерн.
4. *Выберите для участников паттерна подходящие имена.* Имена участников паттерна обычно слишком абстрактны, чтобы употреблять их непосредственно в коде. Тем не менее бывает полезно включить имя участника как имя в программе. Это помогает сделать паттерн более очевидным при реализации. Например, если вы пользуетесь паттерном стратегия в алгоритме размещения текста, то классы могли бы называться `SimpleLayoutStrategy` или `TeXLayoutStrategy`.
5. *Определите классы.* Объявите их интерфейсы, установите отношения наследования и определите переменные экземпляра, которыми будут представлены данные объекты и ссылки на другие объекты. Выявите имеющиеся в вашем приложении классы, на которые паттерн оказывает влияние, и соответствующим образом модифицируйте их.
6. *Определите имена операций, встречающихся в паттерне.* Здесь, как и в предыдущем случае, имена обычно зависят от приложения. Руководствуйтесь теми функциями и взаимодействиями, которые ассоциированы с каждой операцией. Кроме того, будьте последовательны при выборе имен. Например, для обозначения фабричного метода можно было бы всюду использовать префикс `Create-`.
7. *Реализуйте операции, которые выполняют обязанности и отвечают за отношения, определенные в паттерне.* Советы о том, как это лучше сделать, вы найдете в разделе «Реализация». Поможет и «Пример кода».