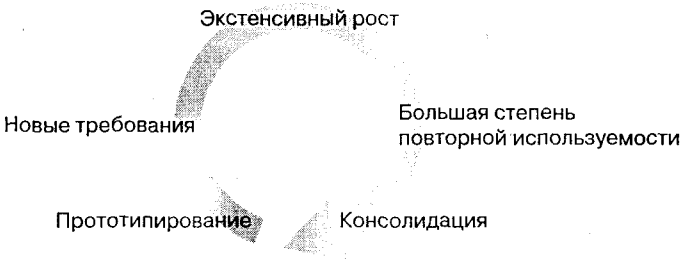
одной предметной области. Напротив, они отражают множество разных предметных областей, а классы определяют совершенно разнородные операции и переменные экземпляров.

Чтобы программа могла развиваться и дальше, ее необходимо пересмотреть и реорганизовать. Именно в этот период часто создаются каркасы. При реорганизации классы, описывающие специализированные и универсальные компоненты, отделяются друг от друга, операции перемещаются вверх и вниз по иерархии, а интерфейсы классов становятся более рациональными. В фазе консолидации появляется много новых объектов, часто в результате декомпозиции существующих и использования композиции вместо наследования. Поэтому прозрачный ящик заменяется черным. В связи с непрекращающимся потоком новых требований и стремлением ко все более высокой степени повторной используемости объектно-ориентированная программа должна вновь и вновь проходить через фазы экстенсивного роста и консолидации.



От этого цикла никуда не деться. Но хороший проектировщик предвидит изменения, которые могут потребовать реорганизации. Он знает, какие структуры классов и объектов позволят избежать реорганизации, его проект устойчив к изменениям требований. Требования проанализированы, выявлены те из них, которые, по всей вероятности, изменятся на протяжении жизненного цикла программы, и в проекте учтено все это.

В наших паттернах проектирования нашли свое применение многие структуры, появившиеся в результате реорганизации. Использование паттернов на ранних стадиях проекта предотвратит реорганизацию в будущем. Но даже если вы не увидели, как применить паттерн, пока не создали всю систему, он все равно «подскажет», как ее можно изменить.

6.2. Краткая история

Начало этому каталогу положила докторская диссертация Эриха [Gam91, Gam92]. Почти половина всех паттернов была представлена в этой работе. К моменту проведения конференции OOPSLA '91 диссертацию официально признали независимым каталогом, и для продолжения работы над ним к Эриху присоединился Ричард. Вскоре после этого к компании примкнул и Джон. Незадолго до конференции OOPSLA '92 в группу влился Ральф. Мы изо всех сил старались, чтобы каталог был готов к публикации в трудах ECOOP '93, но вскоре осознали,

что статью на 90 страницах не примут. Поэтому пришлось составить краткий реферат каталога, который и был опубликован. Вскоре после этого мы решили превратить каталог в книгу.

Названия, которые мы присваивали паттернам, по ходу дела менялись. «Обертка» (Wrapper) стала **декоратором**, «клей» (Glue) – **фасадом**, «холостяк» (Solitaire) – **одиночкой**, «бродяга» (Walker) – **посетителем**. Парочка паттернов осталась за бортом, поскольку мы не сочли их достаточно важными. Но в остальном набор паттернов в каталоге почти не менялся с конца 1992 г. Однако сами паттерны эволюционировали очень сильно.

На самом деле заметить, что нечто представляет собой паттерн, несложно. Мы все четверо активно трудимся над созданием объектно-ориентированных систем и обнаружили, что, когда поработаешь с достаточно большим числом таких систем, выявлять паттерны становится просто. Но *найти* паттерн куда проще, чем *описать* его, тем более так, чтобы проектировщики, незнакомые с ним, поняли назначение этого приема и уяснили, почему он важен.

Специалисты уже на самых ранних стадиях осознали ценность каталога. Но понять паттерны могли лишь те, кто их уже использовал.

Так как одной из главных целей книги было научить объектно-ориентированному проектированию, то мы пришли к выводу о необходимости улучшить каталог. Увеличили средний объем описания одного паттерна с двух до десяти с лишним страниц, включив подробный раздел «Мотивация» и пример кода. Мы также решили рассматривать различные компромиссы и подходы к реализации паттернов. В результате изучать паттерны стало легче.

Еще одно важное изменение, внесенное не так давно: задаче, которую призван решить тот или иной паттерн, стало уделяться более пристальное внимание. Проще взглянуть на паттерн как на решение, а не как на прием, который можно приспособить под собственные нужды и повторно использовать. Труднее увидеть, когда паттерн *подходит*, то есть охарактеризовать те задачи, которые он решает, и тот контекст, в котором он является наилучшим вариантом. Проще разглядеть, *что* делается, чем понять, *почему* так делается. Понимать назначение паттерна тоже важно, поскольку это помогает определить, какой паттерн стоит применить.

6.3. Проектировщики паттернов

Мы не единственные, кому интересно писать книги, которые содержат каталог паттернов, применяемых специалистами. Мы – часть обширного сообщества, заинтересованного в паттернах вообще и в паттернах, имеющих отношение к программному обеспечению, в частности. Кристофер Александр – это архитектор, который первым начал изучать паттерны в строительстве и разработал «язык паттернов» для их генерирования. Работа Александра постоянно вдохновляла нас. Поэтому уместно и поучительно сравнить его и наши старания. Затем мы обратимся к работам других авторов по паттернам, связанным с программным обеспечением.