ТЕМА 10
10.1 Классификация шаблонов программных проектов

При реализации проектов по разработке программных систем и моделированию бизнес-процессов встречаются ситуации, когда решение проблем в различных проектах имеют сходные структурные черты. Попытки выявить похожие схемы или структуры в рамках объектно-ориентированного анализа и проектирования привели к появлению понятия шаблона (паттерна), которое из абстрактной категории превратилось в непременный атрибут современных CASE-средств.

Паттерны различаются степенью детализации и уровнем абстракции. Используется следующая общая классификация паттернов по категориям их применения:

- Архитектурные паттерны;
- Паттерны проектирования;
- Паттерны анализа;
- Паттерны тестирования;
- Паттерны реализации.

Архитектурные паттерны (*Architectural patterns*) – множество предварительно определенных подсистем со спецификацией их ответственности, правил и базовых принципов установления отношений между ними.

Архитектурные паттерны предназначены для спецификации фундаментальных схем структуризации программных систем. Наиболее известными паттернами этой категории являются паттерны *GRASP (General Responsibility Assignment Software Pattern*). Эти паттерны относятся к уровню системы и подсистем, но не к уровню классов. Как правило, формулируются в обобщенной форме, используют обычную терминологию и не зависят от области приложения.

Паттерны проектирования (*Design patterns*) – специальные схемы для уточнения структуры подсистем или компонентов программной системы и отношений между ними. Паттерны проектирования описывают общую структуру взаимодействия элементов программной системы, которые реализуют исходную проблему проектирования в конкретном контексте. Наиболее известными паттернами этой категории являются паттерны *GoF (Gang of Four)*, названные в честь Э. Гаммы, Р. Хелма, Р. Джонсона и Дж. Влиссидеса, которые систематизировали их и представили общее описание. Паттерны *GoF* включают в себя 23 паттерна. Они не зависят от языка реализации, но их реализация зависит от области приложения.

Паттерны анализа (*Analysis patterns*) – специальные схемы для представления общей организации процесса моделирования. Паттерны анализа относятся к одной или нескольким предметным областям и описываются в терминах предметной области. Наиболее известными паттернами этой группы являются паттерны бизнес-моделирования *ARIS*

(Architecture of Integrated Information Systems), которые характеризуют абстрактный уровень представления бизнес-процессов. В дальнейшем паттерны анализа конкретизируются в типовых моделях с целью выполнения аналитических оценок или имитационного моделирования бизнес-процессов.

Паттерны тестирования (*Test patterns*) – специальные схемы для представления общей организации процесса тестирования программных систем. К этой категории паттернов относятся такие паттерны, как тестирование черного ящика, белого ящика, отдельных классов, системы. Паттерны этой категории систематизировал и описал М. Гранд. Некоторые из них реализованы в инструментальных средствах, наиболее известными из которых является IBM Test Studio. В связи с этим паттерны тестирования иногда называют стратегиями или схемами тестирования.

Паттерны реализации (*Implementation patterns*) – совокупность компонентов и других элементов реализации, используемых в структуре модели при написании программного кода. Эта категория паттернов делится на следующие подкатегории: паттерны организации программного кода, паттерны оптимизации программного кода, паттерны устойчивости кода, паттерны разработки графического интерфейса пользователя и др. Паттерны этой категории описаны в работах М. Гранда, К. Бека, Дж. Тидвелла и др. Некоторые из них реализованы в популярных интегрированных средах программирования в форме шаблонов создаваемых проектов. В этом случае выбор шаблона программного приложения позволяет получить некоторую заготовку программного кода.

10.2 Паттерны проектирования в нотации языка UML

В сфере разработки программных систем наибольшее применение получили паттерны проектирования GoF, некоторые из них реализованы в популярных средах программирования. При этом паттерны проектирования могут быть представлены в наглядной форме с помощью рассмотренных обозначений языка UML. Паттерн проектирования в контексте языка UML представляет собой параметризованную кооперацию вместе с описанием базовых принципов ее использования.

При изображении паттерна используется обозначение параметризованной кооперации языка UML (рис. 10.1), которая обозначается пунктирным эллипсом. В правый верхний угол эллипса встроен пунктирный прямоугольник, в котором перечислены параметры кооперации, которая представляет тот или иной паттерн.

Рисунок 10.1 – Изображение паттерна в форме параметризованной кооперации

В последующем параметры паттерна могут быть заменены различными классами, чтобы получить реализацию паттерна в рамках конкретной кооперации. Эти параметры специфицируют используемые классы в форме ролей классов в рассматриваемой подсистеме. При связывании или реализации паттерна любая линия помечается именем параметра паттерна, которое является именем роли соответствующей ассоциации. В дополнение к диаграммам кооперации особенности реализации отдельных паттернов представляются с помощью диаграмм последовательности.

Паттерны проектирования позволяют решать различные задачи, с которыми постоянно сталкиваются проектировщики объектно-ориентированных приложений. Ниже представлен полный список паттернов проектирования GoF и краткое описание назначения каждого из них

В качестве примеров рассматриваются два паттерна проектирования, которые нашли наибольшее применение при проектировании программных систем: паттерны *Фасад* и *Наблюдатель*.

10.3 Паттерн Фасад и его обозначение в нотации языка UML

Паттерн *Фасад* предназначен для замены нескольких разнотипных интерфейсов доступа к определенной подсистеме некоторым унифицированным интерфейсом, что существенно упрощает использование рассматриваемой подсистемы. Общее представление паттерна проектирования *Фасад* может быть изображено с помощью следующей диаграммы параметризованной кооперации (рис. 10.2).

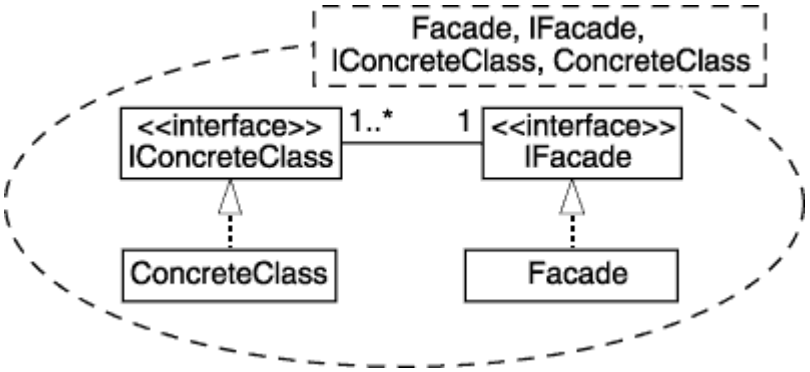

Рисунок 10.2 – Общее представление паттерна проектирования Фасад

Изображенная параметризованная кооперация содержит 4 параметра: класс Facade (Фасад), интерфейс IFacade, интерфейсы IConcreteClass и конкретные классы ConcreteClass, в которых реализованы интерфейсы IConcreteClass. Пунктирная линия со стрелкой в форме треугольника служит для обозначения отношения реализации (не путать с отношением обобщения классов).

При решении конкретных задач проектирования данный паттерн может быть конкретизирован. В этом случае вместо параметров изображенной

кооперации должны быть указаны классы, предназначенные для решения отдельных задач.

Ниже приведен пример, который иллюстрирует использование паттерна Фасад для выполнения операций по заданию и считыванию адресов из базы данных сотрудников. Фрагмент соответствующей диаграммы классов содержит 2 класса: Адрес и интерфейс к операциям этого класса IАдрес (рис. 10.3). При задании адреса нового сотрудника необходимо обратиться к этому интерфейсу и последовательно выполнить операции: задатьУлицу(), задатьДом(), задатьКорпус(), задатьКвартиру(), используя в качестве аргумента идентификационный номер нового сотрудника. Для получения информации об адресе сотрудника, необходимо также обратиться к этому интерфейсу и последовательно выполнить операции: прочитатьУлицу(), прочитатьДом(), прочитатьКорпус(), прочитатьКвартиру(), используя в качестве аргумента идентификационный номер интересующего сотрудника.

Очевидно, отслеживать при каждом обращении правильность выполнения этих последовательностей операций неудобно. С этой целью к данному фрагменту следует добавить еще один интерфейс, реализацию паттерна Фасад для рассматриваемой ситуации. Соответствующий фрагмент модифицированной диаграммы классов будет содержать 4 класса (рис. 10.4), изображенные таким образом, чтобы иллюстрировать реализацию параметризованной кооперации.

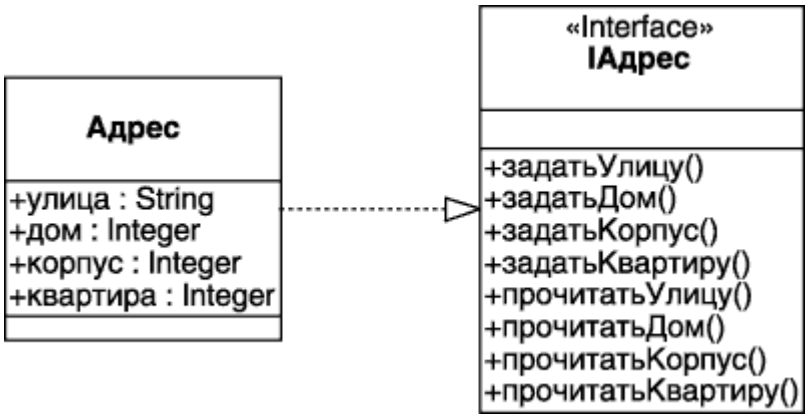

Рисунок 10.3 – Фрагмент диаграммы классов до применения паттерна Фасад

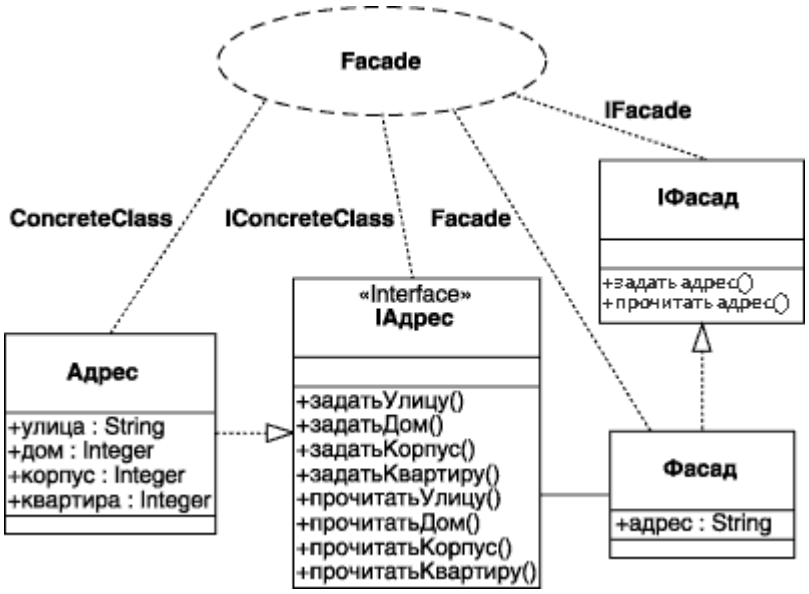

Рисунок 10.4 – Конкретная реализация паттерна проектирования Фасад

При задании адреса нового сотрудника в этом случае достаточно обратиться к интерфейсу IФасад и выполнить единственную операцию: задатьАдрес(), используя в качестве аргумента идентификационный номер нового сотрудника. Для получения информации об адресе сотрудника также достаточно обратиться к этому интерфейсу и выполнить единственную операцию: прочитатьАдрес(), используя в качестве аргумента идентификационный номер интересующего сотрудника. Реализацию данных операций следует предусмотреть в классе Фасад. Взаимодействие объектов этих классов может быть представлено с помощью диаграммы последовательности (рис. 10.5).

Аналогичная диаграмма последовательности может быть построена для выполнения операции по чтению адреса. Использование паттерна Фасад обеспечивает для клиента не только простоту доступа к информации об адресах, но и независимость представления объектов класса Адрес от запросов клиентов. Это обстоятельство особенно актуально при изменении формата представления информации или смене соответствующей базы данных. В этом случае потребуется внести изменения только в реализацию операций класса Фасад.

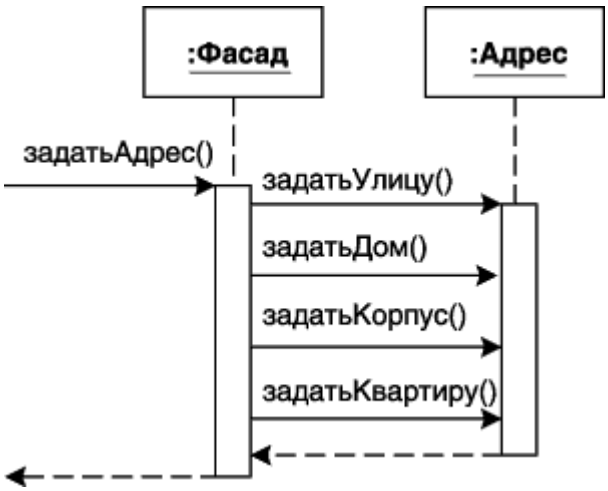

Рисунок 10.5 – Диаграмма последовательности для выполнения операции задания адреса

10.4 Паттерн Наблюдатель и его обозначение в нотации языка UML

Паттерн Наблюдатель предназначен для контроля изменений состояния объекта и передачи информации об изменении этого состояния множеству клиентов. В общем случае паттерн Наблюдатель также может быть изображен в виде параметризованной кооперации (рис. 10.6).

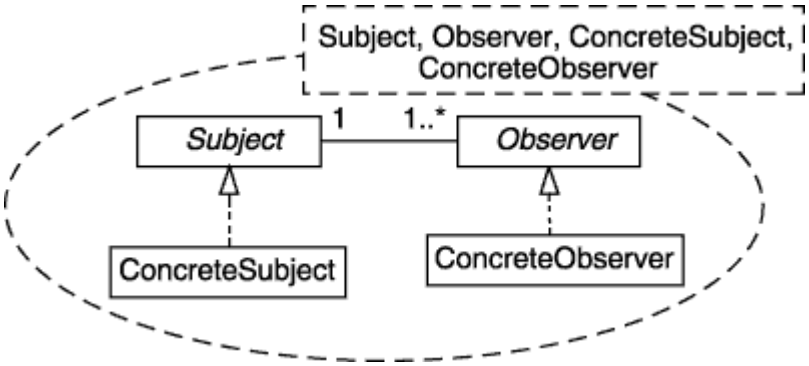

Рисунок 10.6 – Общее представление паттерна проектирования Наблюдатель

Изображенная параметризованная кооперация содержит 4 параметра: абстрактный класс Subject (Субъект), класс ConcreteSubject (Конкретный Субъект), абстрактный класс Observer (Наблюдатель) и класс ConcreteObserver (Конкретный Наблюдатель). Пунктирная линия со стрелкой в форме треугольника служит для обозначения отношения обобщения классов.

При решении конкретных задач проектирования данный паттерн также может быть конкретизирован. В этом случае вместо параметров изображенной кооперации должны быть указаны классы, предназначенные для решения отдельных задач.

Рассмотрим пример, который иллюстрирует использование паттерна Наблюдатель для отслеживания изменений в таблице БД и отражении этих изменений на диаграммах. Для определенности можно использовать таблицу

БД и две диаграммы – круговую и столбиковую. Фрагмент соответствующей диаграммы классов содержит 5 классов (рис. 10.7).

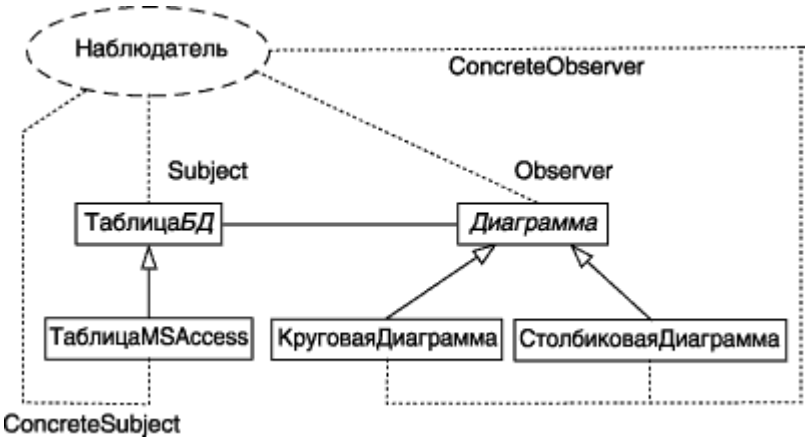

Рисунок 10.7 – Реализация паттерна проектирования Наблюдатель

В этом случае за субъектом Таблицей MS Access может "следить" произвольное число наблюдателей, причем их добавление или удаление не влияет на представление информации в БД. Класс Таблица MS Access реализует операции по отслеживанию изменений в соответствующей таблице, и при их наличии сразу информирует абстрактного наблюдателя. Тот в свою очередь вызывает операции по перерисовке соответствующих диаграмм у конкретных наблюдателей, в качестве которых выступают классы Круговая Диаграмма и Столбиковая Диаграмма.

Использование паттерна Наблюдатель не только упрощает взаимодействие между объектами соответствующих классов, но и позволяет вносить изменения в реализацию операций классов субъекта и наблюдателей независимо друг от друга. При этом процесс добавления или удаления наблюдателей никак не влияет на особенности реализации класса субъекта.

В заключении следует отметить, что язык UML представляет собой нотацию для визуального моделирования программных систем и бизнес-процессов. В то же время описание языка UML не содержит сведений относительно того, каким образом и в какой последовательности следует разрабатывать канонические диаграммы при выполнении конкретных проектов. Соответствующая информация относится к области методологии проектирования программных систем.