1. Основные понятия объектно-ориентированного подхода

Объектно-ориентированный подход основан на систематическом использовании моделей для языково-независимой разработки программной системы, на основе из ее прагматики.

Последний термин нуждается в пояснении. Прагматика определяется целью разработки программной системы: для обслуживания клиентов банка, для управления работой аэропорта, для обслуживания чемпионата мира по футболу и т.п. В формулировке цели участвуют предметы и понятия реального мира, имеющие отношение к разрабатываемой программной системе (см. рисунок 1.1). При объектно-ориентированном подходе эти предметы и понятия заменяются их моделями, т.е. определенными формальными конструкциями, представляющими их в программной системе.

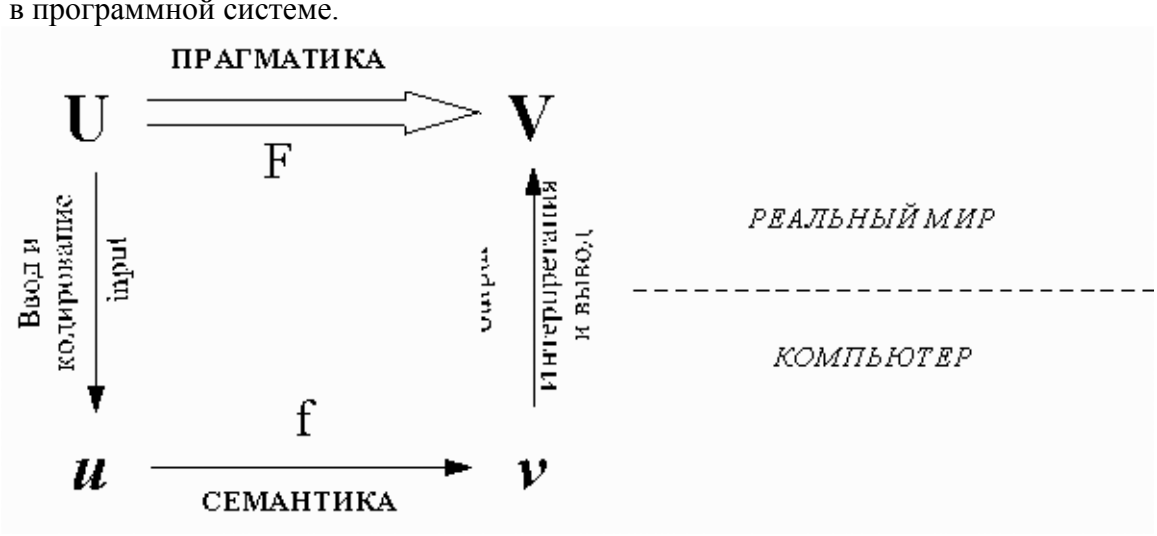

Рис. 1.1. Семантика (смысл программы с точки зрения выполняющего ее компьютера) и прагматика (смысл программы с точки зрения ее пользователей)

Модель содержит не все признаки и свойства представляемого ею предмета (понятия), а только те, которые существенны для разрабатываемой программной системы. Тем самым модель "беднее", а, следовательно, проще представляемого ею предмета (понятия). Но главное даже не в этом, а в том, что модель есть формальная конструкция: формальный характер моделей позволяет определить формальные зависимости между ними и формальные операции над ними. Это упрощает как разработку и изучение (анализ) моделей, так и их реализацию на компьютере. В частности, формальный характер моделей позволяет получить формальную модель разрабатываемой программной системы как композицию формальных моделей ее компонентов.

Таким образом, объектно-ориентированный подход помогает справиться с такими сложными проблемами, как уменьшение сложности программного обеспечения; повышение надежности программного обеспечения; обеспечение возможности модификации отдельных компонентов программного обеспечения без изменения остальных его компонентов; обеспечение возможности повторного использования отдельных компонентов программного обеспечения.

Систематическое применение объектно-ориентированного подхода позволяет разрабатывать хорошо структурированные, надежные в эксплуатации, достаточно просто модифицируемые программные системы. Этим объясняется интерес программистов к объектно-ориентированному подходу и объектно-ориентированным языкам программирования. Объектно-ориентированный подход является одним из наиболее интенсивно развивающихся направлений теоретического и прикладного программирования.

Цель данного курса лекций - введение в объектно-ориентированный подход к разработке и реализации прикладных программных систем. Я попытаюсь убедить вас в целесообразности и плодотворности систематического применения объектно-ориентированного подхода на всех этапах жизненного цикла прикладной программной системы (см. рисунок 1.2), начиная с анализа требований к программной системе и ее предварительного проектирования, и кончая ее
реализацией, тестированием и последующим сопровождением.

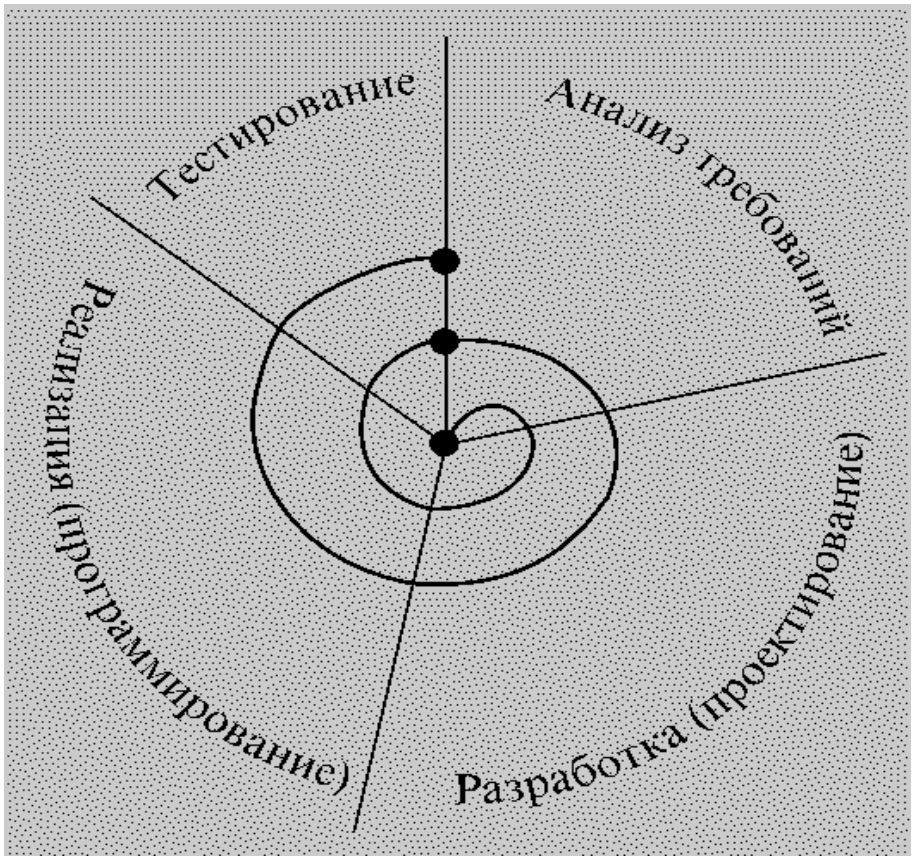

Рис. 1.2. Жизненный цикл программной системы

Объектно-ориентированный подход имеет два аспекта:

1. объектно-ориентированная разработка программного обеспечения;
2. объектно-ориентированная реализация программного обеспечения.

1.1. Объектно-ориентированная разработка программ

Объектно-ориентированная разработка программного обеспечения связана с применением объектно-ориентированных моделей при разработке программных систем и их компонентов. Говоря об объектно-ориентированной разработке, я имею

в виду:

- объектно-ориентированные методологии (технологии) разработки программных систем;
- инструментальные средства, поддерживающие эти технологии.

Объектно-ориентированная разработка может начаться на самом первом этапе жизненного цикла; она не связана с языком программирования, на котором предполагается реализовать разрабатываемую программную систему: этот язык может и не быть объектно-ориентированным. На этапе разработки объекты - это некоторые формальные конструкции (например, прямоугольники с закругленными углами, с помощью которых они изображаются на схемах), никак пока не связанные с их будущей реализацией на одном из языков программирования.

Объектно-ориентированная разработка программного обеспечения связана с применением объектно-ориентированных методологий (технологий). Обычно эти объектно-ориентированные методологии поддерживаются инструментальными программными средствами, но и без таких средств они полезны, так как позволяют хорошо понять различные аспекты и свойства разрабатываемой программной системы, что в последующем существенно облегчает ее реализацию, тестирование, сопровождение, разработку новых версий и более существенную модификацию.

Различные объектно-ориентированные методологии разработки программного обеспечения рассматриваются в разделах 2, 3 и 4. Наиболее подробно рассматривается объектно-ориентированная методология OMT (Object Modeling Technique) (разделы 2 и 3), которая поддерживает две первые стадии жизненного цикла программных систем. Эта методология выбрана для демонстрации объектно-ориентированного подхода, потому что является одной из наиболее продвинутых и популярных объектно-ориентированных методологий. Более того, ее
графический язык (система обозначений для диаграмм) получил достаточно широкое распространение и используется в некоторых других объектно-ориентированных методологиях, а также в большинстве публикаций по объектно-ориентированным методологиям. Методология OMT поддерживается системой Paradigm+, одной из наиболее известных инструментальных систем объектно-ориентированной разработки.

Будут рассмотрены и некоторые другие объектно-ориентированные методологии разработки программных систем в сравнении с методологией OMT (раздел 4):

1. SA/SD (Structured Analysis/Structured Design);
2. JSD (Jackson Structured Development);
3. OSA (Object-Oriented System Analysis).

1.2. Объектно-ориентированные языки программирования

Вопросы реализации программного обеспечения, разработка которого велась с применением одной из объектно-ориентированных методологий, рассматриваются в разделе 5. Реализация программного обеспечения связана с использованием одного из языков программирования.

Показано, что наиболее удобными для реализации программных систем, разработанных в рамках объектно-ориентированного подхода, являются объектно-ориентированные языки программирования, хотя возможна реализация и на обычных (не объектно-ориентированных) языках (например, на языке С и на языке Fortran).

Объектно-ориентированные языки программирования пользуются в последнее время большой популярностью среди программистов, так как они позволяют использовать преимущества объектно-ориентированного подхода не только на этапах проектирования и конструирования программных систем, но и на этапах их реализации, тестирования и сопровождения.

Первый объектно-ориентированный язык программирования Simula 67 был разработан в конце 60-х годов в Норвегии. Авторы этого языка очень точно угадали перспективы развития программирования: их язык намного опередил свое время. Однако современники (программисты 60-х годов) оказались не готовы воспринять ценности языка Simula 67, и он не выдержал конкуренции с другими языками программирования (прежде всего, с языком Fortran). Прохладному отношению к языку Simula 67 способствовало и то обстоятельство, что он был реализован как интерпретируемый (а не компилируемый) язык, что было совершенно неприемлемым в 60-е годы, так как интерпретация связана со снижением эффективности (скорости выполнения) программ.

Но достоинства языка Simula 67 были замечены некоторыми программистами, и в 70-е годы было разработано большое число экспериментальных объектно-ориентированных языков программирования: например, языки CLU, Alphard, Concurrent Pascal и др. Эти языки так и остались экспериментальными, но в результате их исследования были разработаны современные объектно-ориентированные языки программирования: C++, Smalltalk, Eiffel и др.

Наиболее распространенным объектно-ориентированным языком программирования безусловно является C++. Свободно распространяемые коммерческие системы программирования C++ существуют практически на любой платформе. Широко известна свободно распространяемая система программирования G++, которая дает возможность всем желающим разобрать достаточно хорошо и подробно прокомментированный исходный текст одного из образцовых компиляторов языка C++. Завершается работа по стандартизации языка
C++: последний Draft стандарта C++ выпущен в июне 1995 г. (он доступен по Internet).

Разработка новых объектно-ориентированных языков программирования продолжается. С 1995 года стал широко распространяться новый объектно-ориентированный язык программирования Java, ориентированный на сети компьютеров и, прежде всего, на Internet. Синтаксис этого языка напоминает синтаксис языка C++, однако эти языки имеют мало общего. Java интерпретируемый язык: для него определены внутреннее представление (bytecode) и интерпретатор этого представления, которые уже сейчас реализованы на большинстве платформ. Интерпретатор упрощает отладку программ, написанных на языке Java, обеспечивает их переносимость на новые платформы и адаптируемость к новым окружениям. Он позволяет исключить влияние программ, написанных на языке Java, на другие программы и файлы, имеющиеся на новой платформе, и тем самым обеспечить безопасность при выполнении этих программ. Эти свойства языка Java позволяют использовать его как основной язык программирования для программ, распространяемых по сетям (в частности, по сети Internet).

1.3. Сквозной пример

В качестве сквозного примера будет рассмотрена задача о разработке программного обеспечения банковской сети (рисунок 1.3). В состав этой сети входит центральный компьютер, принадлежащий объединению банков (консорциуму), компьютеры банков, к которым непосредственно (минуя центральный компьютер) присоединены кассовые терминалы, обслуживаемые кассирами, и сеть терминалов для клиентов банка (банкоматов).

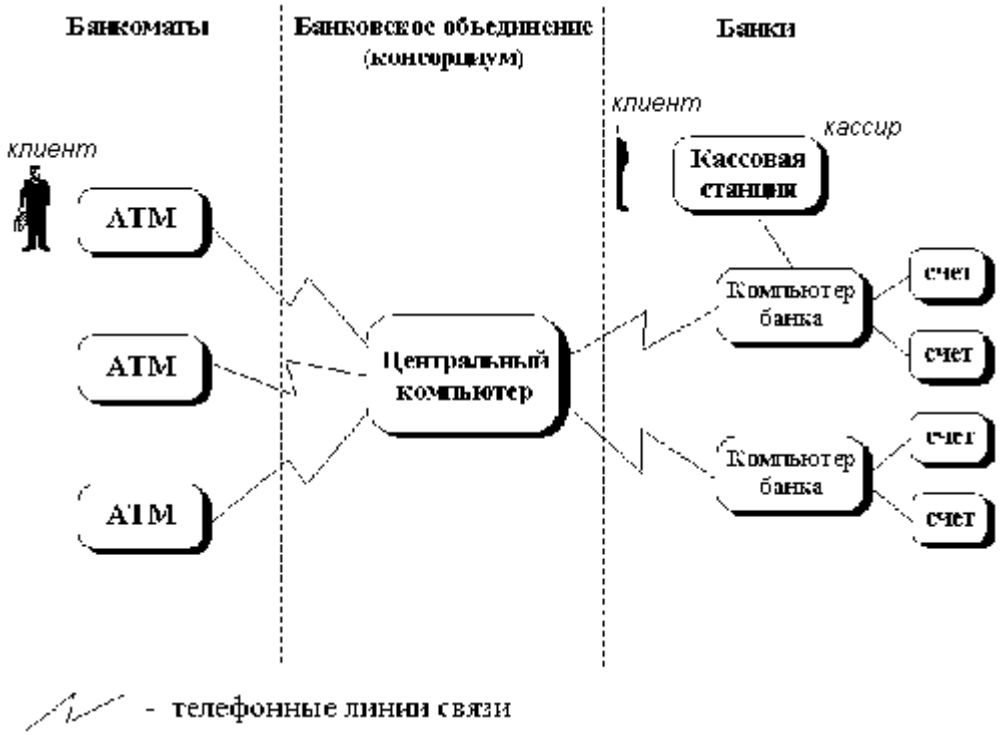

Рис. 1.3. Схема банковской сети

Клиенты банков имеют пластиковые банковские карточки (один клиент может иметь несколько карточек); карточка содержит код карточки, код банка, код клиента и другую информацию, обеспечивающую доступ к счету (счетам) клиента в этом банке. Клиент может вставить свою карточку в АТМ (банкомат) и, при условии, что код карточки и код банка верны, начать банковскую проводку. Данные с карточки поступают в центральный компьютер, который распределяет их по компьютерам банков в соответствии с кодами банков до начала проводки; после окончания проводки ее результаты поступают снова в центральный компьютер, который распределяет их по АТМ. Являясь распределителем данных между компьютерами банков и АТМ, центральный компьютер должен регулировать коллективный доступ со стороны клиентов и банков, организуя и поддерживая соответствующие очереди.

Проводка состоит в согласованном изменении данных на счетах клиента и отчетной документации банка, хранящихся в базе данных банка, в соответствии с данными проводки. Проводка включает в себя и проверку права клиента на доступ к его счетам на момент проводки (проверка безопасности), и проверку соответствия суммы, затребованной клиентом, текущему состоянию его счета. Если проверки прошли успешно, клиент получает из АТМ затребованную им сумму денег и квитанцию, в противном случае он получает только квитанцию. Во время осуществления проводки могут произойти сбои в работе аппаратуры, либо клиент может раздумать получать деньги и отменить уже начавшуюся проводку. В этом случае все счета и отчетные документы должны быть восстановлены в том

состоянии, в котором они были до начала проводки (откат). Для реализации отката используется служба ведения записей об изменениях, вносимых в базу данных банка при выполнении проводки. Все действия, связанные с выполнением проводки (в том числе протоколирование и обеспечение безопасности проводки), производятся программным обеспечением системы управления банковской сетью, процесс разработки которого и составляет содержание сквозного примера.

Компьютер банка поддерживает счета клиентов, т.е. хранит их в своей базе данных и выполняет проводки над этими счетами по запросам с АТМ (удаленная проводка) или с кассовых терминалов (проводка кассира, данные о которой вводятся кассиром).
Несмотря на внешнюю простоту, эта задача достаточно сложна, чтобы на ее примере можно было продемонстрировать основные особенности объектно-ориентированного подхода к разработке программных систем: в этой задаче есть и необходимость распределения по сети компьютеров (банкомат, который мы в дальнейшем будем для краткости называть АТМ, тоже можно рассматривать как специализированный компьютер - см. рисунок 1.4), и асинхронные процессы, и необходимость синхронизации таких процессов для организации параллельного обслуживания нескольких клиентов, и работа с базами данных (информация о клиентах хранится в базах данных банков), в частности, обслуживание транзакций (транзакциями являются банковские проводки).

Рис. 1.4. Схема банкомата (АТМ)

Наряду с описанным примером будет рассмотрено достаточно большое число других примеров, предназначенных для демонстрации особенностей, не охватываемых этим основным примером.