ООП программирование

Изменения в отношении к программированию вызваны с появлением сетевых технологий, WWW, WWW программирования и идеологии распределенных информационно-вычислительных систем (GRID).

ПРОЦЕССЫ на смену **ОБЪЕКТОВ**

Парадигмы программирования

- Следует обратить внимание и на потенциальные преимущества процессов перед объектами. Они, сами по себе, поддерживают параллелизм решаемой задачи, а также позволяют задавать несколько точек ожидания вместо одного пассивного состояния.
- С каждой из таких точек можно связать свой интерфейс, обеспечив, тем самым дополнительные возможности. Вообще, понятие процесса гораздо шире, чем класса, а это позволяет говорить о неисповедимых путях дальнейшее развития данного направления технологии программирования.

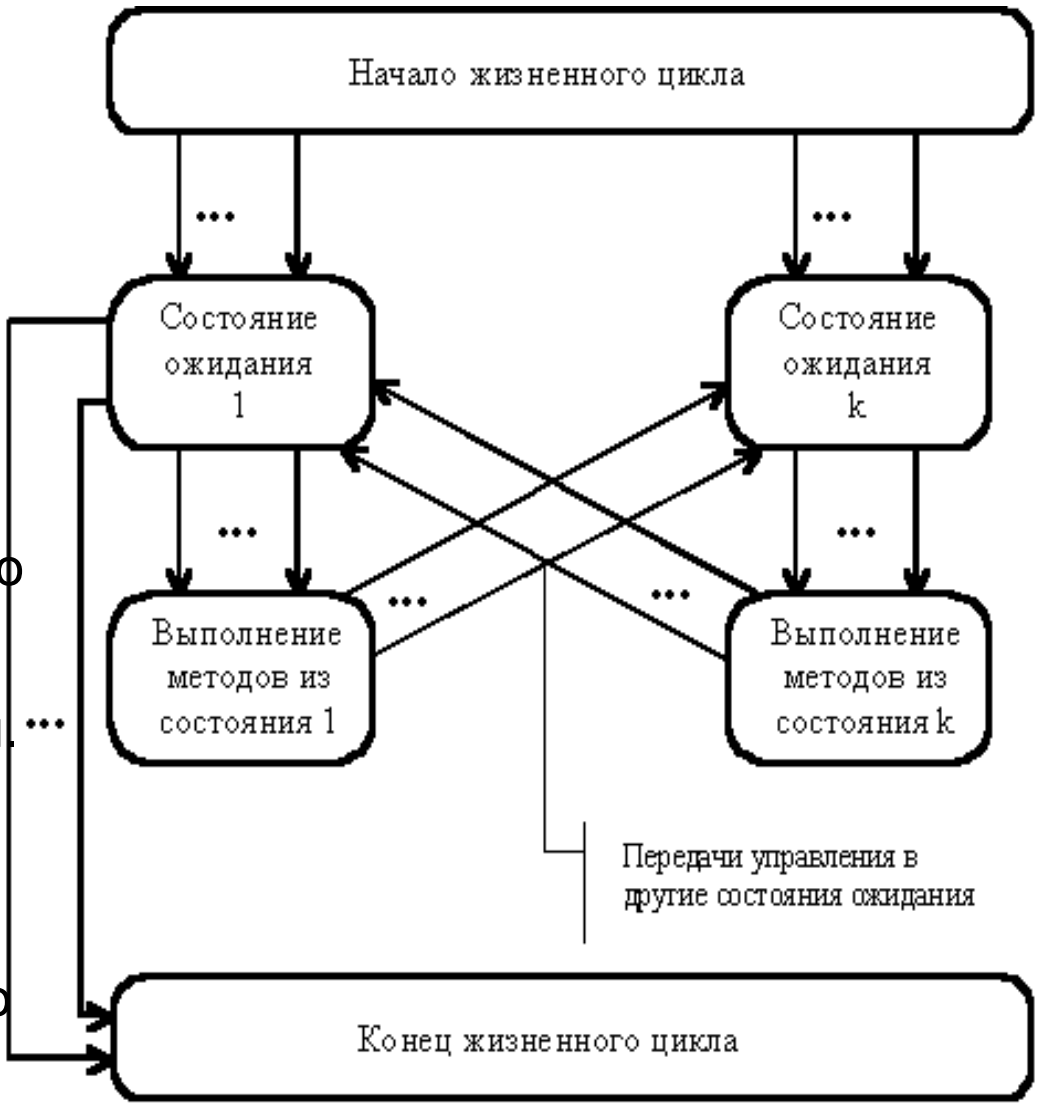


Рис. 3. Более детальная модель жизненного цикла процесса

Парадигмы программирования

- Описывая основные достоинства ООМ Гради Буч совершенно серьезно говорит, что: "Объектная модель позволяет в полной мере использовать выразительные возможности объектных и объектно-ориентированных языков программирования".

- Как будто построение объектной модели происходило не на основе этих самых языков! Ведь любая методология разрабатывается с использованием уже накопленных в предметной области эмпирических фактов и практических результатов, коими, в данном случае, и являлись существующие языки программирования. Ведь до появления ООМ прошел не один десяток лет, если плясать от Симулы-67.

- Только идиоты, политики и менеджеры могут создавать методологии, которые не используют выразительные возможности языков. Вряд ли можно отнести Буча к первым двум категориям. Это голова и гигант мысли в одном флаконе! Следовательно, он немного перестарался, чтобы раскрутить свои методы. Хотя, и политики в программировании более чем достаточно.

Парадигмы программирования

- **ОО подход породил множество постулатов и догм, определяющих правильный стиль программирования. Со временем они перекочевали в различные книжки.**

- **Ряд зафиксированных приемов оказались связанными со спецификой конкретных языков программирования. Другие были заявлены как универсальные и обосновались в "приемах объектно-ориентированного проектирования".**

Парадигмы программирования

- Не отрицая полезность накопленного эмпирического опыта, но, вместе с тем, создается впечатление, что мы пользуемся не той системой отсчета.
- Что-то напоминает геоцентрическую систему Птолемея, которая поставила землю в центр вселенной. Для простейших расчетов все идет нормально. Но как только переходим к более сложным и точным расчетам, каждая планета начинает двигаться по каким-то непонятным, дополнительным окружностям. И без уточняющих правил, исключений, измерительных инструментов, а также «плясок с бубном» не обойтись.

- Я не думаю, что в мгновение ока поставлю в центр нашей системы Солнце. Скорее всего, у меня на этом месте расположится Марс.

Парадигмы программирования

- Центром вселенной в мире ООП являются интерфейсы, точнее сигнатуры процедур и функций. Именно через сигнатуры должно идти все взаимодействие классов. В противном случае нарушается инкапсуляция, затрудняется модификация, а также возникает множество других проблем. Нельзя напрямую обратиться к переменной экземпляра класса. Только через методы интерфейса.

- Все это происходит оттого, что между клиентом, использующим класс и состоянием класса должно обязательно стоять промежуточное звено, служащее посредником при передаче данных. Роль этого звена обычно исполняет тело процедуры, которое и решает проблемы взаимодействия сигнатуры с данными класса.

Парадигмы программирования

- Даже в том случае, когда необходимо осуществлять только прием или передачу простейших данных, приходится пользоваться методом типа GetSetPut.
- Вместе с тем, метафора непосредственно доступной переменной часто бывает полезной и облегчает программирование. Поэтому, в ряде ОО языков использование методов скрывается под таким понятием как свойство.
- Однако и в этой ситуации реализация остается прежней.

Парадигмы программирования

- Однако косвенное взаимодействие необязательно должно обеспечиваться через явный вызов процедуры. Еще на заре СИСТЕМНОГО программирования был придуман обмен данными между процессами через почтовые ящики. Одни процессы загружали почту (информацию), а другие забирали ее.
- Внутренняя организация процессов оставалась при этом закрытой. Достаточно было только договориться о формате обмена данными и подключиться к «почтовым ящикам». При этом обмен мог вестись через простые переменные очереди, стеки и любые другие объекты одинаковым для процессов способом.
- Любые внутренние изменения данных в процессах никоим образом не влияли на их взаимодействие.

Интерфейсы

- Центром вселенной в мире ООП являются интерфейсы, точнее сигнатуры процедур и функций. Именно через сигнатуры должно идти все взаимодействие классов.
- В противном случае нарушается инкапсуляция, затрудняется модификация, а также возникает множество других проблем.
- Нельзя напрямую обратиться к переменной экземпляра класса.
- Только через методы интерфейса.

Интерфейсы

- Так что, интерфейс - это не только процедуры.
- Он может быть выстроен и на основе модели данных.
- При этом неважно, что посредники между почтовыми ящиками и процессами являются процедурами, так как нас совершенно не волнуют ни их сигнатуры, ни внутренние модификации процессов.

Интерфейсы

- Можно легко менять процедуры, обслуживающие почтовые ящики.
- Можно даже использовать несколько разных процедур процесса для обслуживания одного ящика, выбирая нужную процедуру в зависимости от текущего состояния ожидания.

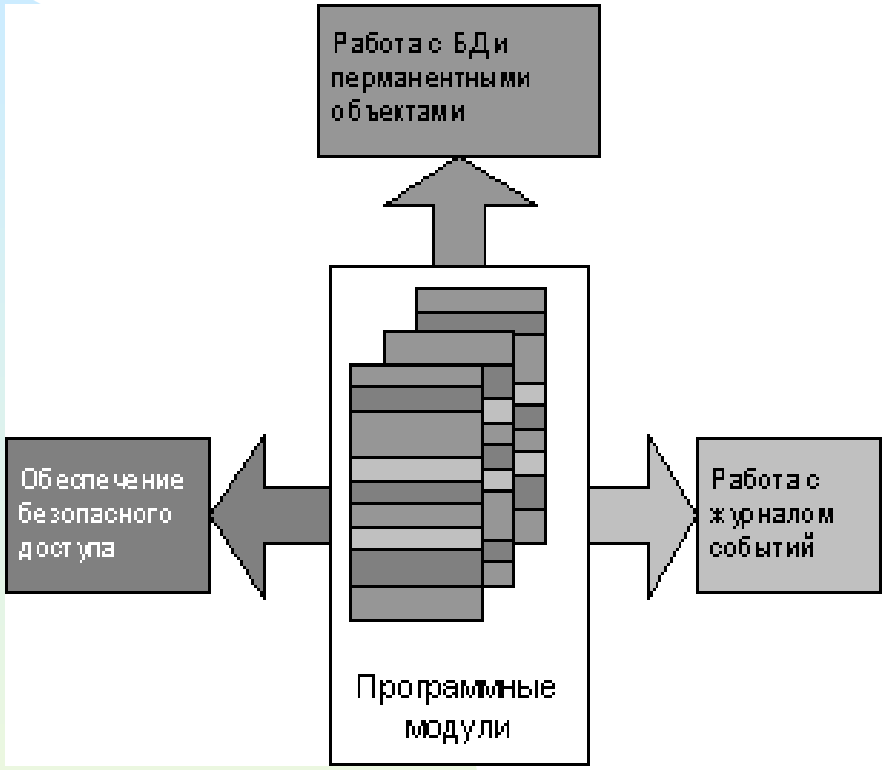
Аспектно-ориентированное программирование

- Аспектно-ориентированное программирование (АОП) представляет собой одну из концепций программирования, которая является симбиозом процедурного и объектно-ориентированного программирования.
- Эта методология призвана снизить время, стоимость и сложность разработки ПО. В современном ПО, как привило, можно выделить определенные части, или аспекты, отвечающие за ту или иную функциональность, реализация которой рассредоточена по коду программы, но состоит из схожих кусков кода.
- По оценкам, около 70% времени в проектах тратится на сопровождение и внесение изменений в готовый программный код. Поэтому в ближайшей перспективе роль АОП и подобных трансформационных подходов становится достаточно важной.
- *АСПЕКТ (от лат. aspectus — вид), точка зрения, с которой рассматривается какое-либо явление, понятие, перспектива*

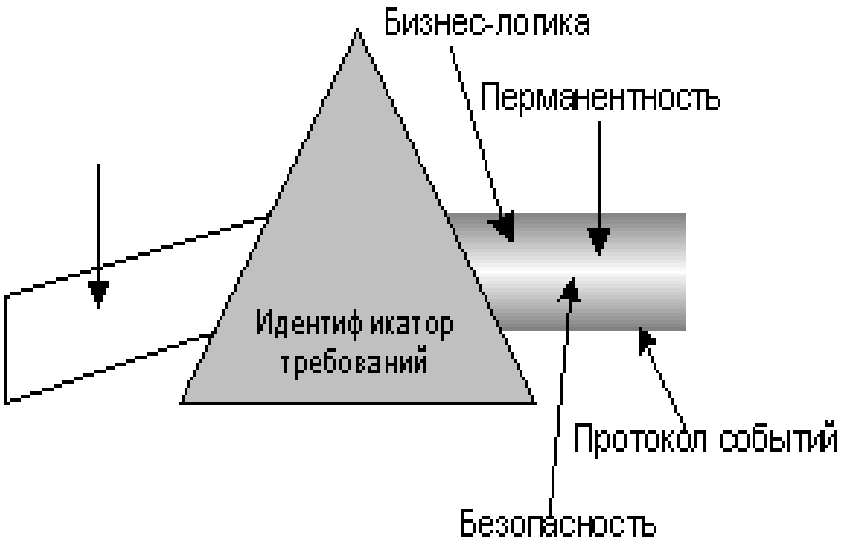
Аспектно-ориентированное программирование

- Как правило, любая программная система состоит из основной (предметно-ориентированной) и системной частей. Например, ядро системы обработки кредитных карт предназначено для работы с платежами, тогда как функциональность системного уровня предназначена для ведения журнала событий, целостности транзакций, авторизации, безопасности, производительности и т.д.
- Большинство подобных частей системы, известных как сквозная функциональность, затрагивает множество основных предметно-ориентированных модулей. Можно рассматривать сложную программную систему как комбинацию модулей, каждый из которых включает в себя, кроме бизнес-логики, часть сквозной функциональности из набора требований к системе.
- При разработке программной системы с использованием существующих методологий программирования сквозная функциональность будет включена во все модули, в результате система будет сложной в проектировании, понимании, реализации и поддержке.

Аспектно-ориентированное программирование



Система как набор функциональных модулей.



Декомпозиция требований: выделение функциональности.

Аспектно-ориентированное программирование

- Аспектно-ориентированное программирование дает решение для выделения в отдельные модули сквозной функциональности в сложных программных системах.
- АОП предлагает средства выделения сквозной функциональности в отдельные программные модули — аспекты.
- С точки зрения АОП в процессе разработки достаточно сложной системы программист решает две ортогональные задачи:
  - Разработка компонентов, то есть выявление классов и объектов, составляющих словарь предметной области (C++).
  - Разработка сервисов, поддерживающих взаимодействие компонентов, то есть построение структур, обеспечивающих взаимодействие объектов, при котором выполняются требования задачи.

Основные концепции АОП

- Аспектно-ориентированный подход рассматривает программную систему как набор модулей, каждый из которых отражает определенный аспект — цель, особенность функционирования системы.
- Набор модулей, образующих программу, зависит от требований к программе, особенностей ее предметной области. Наряду с функциональными требованиями, к программе предъявляются и общесистемные требования, например: целостность транзакций, авторизованный доступ к данным, ведение журнала событий и т. д.
- При проектировании программной системы разработчик выбирает модули так, чтобы каждый из них реализовывал определенное функциональное требование к системе.
- Однако реализация некоторых требований к программе зачастую не может быть локализована в отдельном модуле в рамках процедурного или объектно-ориентированного подхода. В результате код, отражающий такие аспекты функционирования системы, будет встречаться в нескольких различных модулях.

Основные концепции АОП

- Традиционные парадигмы программирования используют при проектировании программы функциональную декомпозицию и не позволяют локализовать сквозную функциональность в отдельных модулях.
- Необходимость реализации сквозной функциональности имеющимися средствами ведет к тому, что некоторый компонент содержит код, отражающий множество ортогональных требований к системе. Это делает такой модуль узкоспециализированным, ухудшает возможности его повторного использования и в некоторых случаях приводит к дублированию кода. В свою очередь, это вызывает повышение вероятности внесения ошибок, увеличение времени отладки, снижает качество программы и в большой степени затрудняет ее сопровождение.
- Аспектно-ориентированный подход в некоторых случаях позволяет избежать описанных проблем и улучшить общий дизайн системы, обеспечивая возможность локализации сквозной функциональности в специальных модулях — аспектах.

Основные концепции АОП

- АОП позволяет реализовывать отдельные концепции в слабосвязанном виде, и, комбинируя такие реализации, формирует конечную систему. АОП позволяет построить систему, используя слабосвязанные, разбитые на отдельные модули (аспекты) реализации общесистемных требований.
- Разработка в рамках АОП состоит из трех отдельных шагов:

1) Аспектная декомпозиция: разбиение требований для выделения общей и сквозной функциональности. На этом шаге необходимо выделить функциональность для модульного уровня из сквозной функциональности системного уровня. Например, в примере с кредитными картами можно выделить три вещи: ядро обработки кредитных карт, журнал событий, аутентификация.

Основные концепции АОП

- Разработка в рамках АОП состоит из трех отдельных шагов:

2)Реализация функциональности: Реализовать каждое требование отдельно. В примере с кредитными картами необходимо отдельно реализовать модули обработки кредитных карт, журнала и аутентификации.

3)Компоновка аспектов: На этом шаге аспектный интегратор определяет правила для создания своих модулей — аспектов, составляя конечную систему.

В примере с кредитными картами необходимо определить в терминах языка, реализующего АОП, при вызове каких операций необходимо вносить запись в журнал, и по завершению каких действий необходимо сообщать об успехе/неуспехе операции. Также можно определить правила, по которым будет вызываться модуль аутентификации перед доступом к бизнес-логике обработки кредитных карт.

Преимущества использования АОП

- Улучшение декомпозиции системы на отдельные модули: АОП позволяет инкапсулировать функциональность, которая не может быть представлена в виде отдельной процедуры или компонента.
  - АОП позволяет реализовать каждое требование отдельно с минимальным связыванием. В результате получается модуль, содержащий данное требование к системе без лишних внешних зависимостей, даже если это требование – сквозная функциональность.
  - При такой реализации модули содержат минимальное количество дублирующегося кода.

Преимущества использования АОП

- Упрощение сопровождения программной системы и внесения в нее изменений:
  - Так как могут существовать модули, на которые могут воздействовать аспекты, становится достаточно легко добавлять новую функциональность путем создания новых аспектов.
  - Более того, если в систему добавляется новый модуль, то существующие аспекты начинают воздействовать и на него без дополнительных усилий.
  - При использовании АОП системный архитектор может отложить решения, касающиеся потенциально возможных требований, поскольку впоследствии эти решения смогут быть реализованы как отдельные аспекты, что не затронет существующую функциональность.

Преимущества использования АОП

- Появление возможностей повторного использования кода, реализующего сквозную функциональность:
  - Следует из того, что при использовании АОП сквозная функциональность может быть реализована в виде аспектов и в виде слабосвязанных модулей

Технология Взаимодействия Объектов (ТВО)

- есть ни что иное, как программирование процессов ***универсальный протокол обмена данными между универсальными объектами***.

Протокол назван универсальным потому, что ***задается*** только один ***единственный способ*** обмена данными между объектами. Объект назван универсальным потому, что он ***задан*** только в одной ***единственной форме,*** специально приспособленной для работы по универсальному протоколу.

ТВО

Этап I	Этап II	Этап III	Этап IV
Низкоуровневое программирование (Ассемблер)	Процедурное программирование (Fortran, C, Pascal, ...)	Объектно-ориентированное программирование, ООП (C++, Oak, SmallTalk, ...)	Функциональное программированиеФП – (ТВО) (вне языковая технология)

Методики работы с БД, основанные на технологии файл-сервер, умерли и серьезно стоит рассматривать лишь методики, основанные на технологии клиент-сервер (клиенты-сервера).

Но при этом при ***правильном*** проектировании баз данных и приложений, работающих с ними, иметь дело приходится не с двумя уровнями, как это следует из названия клиент-сервер, а как минимум с пятью!