ВВЕДЕНИЕ В АСПЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

С. В. Жемжицкий

ВВЕДЕНИЕ

Аспекты являются эволюцией объектно-ориентированной парадигмы. Они предлагают решения для проблем, с которыми можно столкнуться при разработке программного обеспечения с использованием объектно-ориентированного подхода. Можно без труда заметить, что с применением объектно-ориентированного подхода некоторые участки кода повторяются во многих классах, поскольку функциональность, реализуемая данным кодом, присуща им всем. Хорошими примерами данной особенности являются аудит, управление транзакциями, ведение журнала событий. С появлением АОП функциональность может быть представлена в виде отдельного модуля, называемого аспектом.

АСПЕКТ

Функциональное требование – это некоторое требование, которое позволяет реализовать некоторую возможность, концепцию или вид функциональности.

Аспект – это код, реализующий заданное функциональное требование, который запускается другими требованиями в различных ситуациях. Если функциональное требование не было выделено в аспект, то функциональность этого требования будет вызываться явно в коде, реализующем другой вид требования, что приводит к «спутыванию» этих требований. Кроме того, необходимость вызовов кода функционального требования в различных местах, приводит к «рассеиванию» реализации требования по всем участкам системы. Наличие модели программирования, которая предполагает реализацию таких требований единожды и их использование только в одном месте программы позволяет значительно сократить время ее разработки.

АСПЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Сквозная функциональность – это функциональность, которая может встречаться в различных местах приложения.

Аспектно-ориентированное программирование (АОП) — это парадигма программирования, основанная на идее разделения сквозной функциональности, для улучшения разбиения программы на модули. АОП ставит своей целью разработать механизм реализации сквозной функциональности в объектно-ориентированных системах.

НОВОЕ ИЗМЕРЕНИЕ ПРИ ОРГАНИЗАЦИИ МОДУЛЬНОСТИ

При использовании АОП приложение состоит из классов и аспектов. Аспект отличается от класса тем, что реализует сквозную функциональность. Включение классов и аспектов в одно и тоже приложение означает, что модульность может быть достигнута двумя способами:

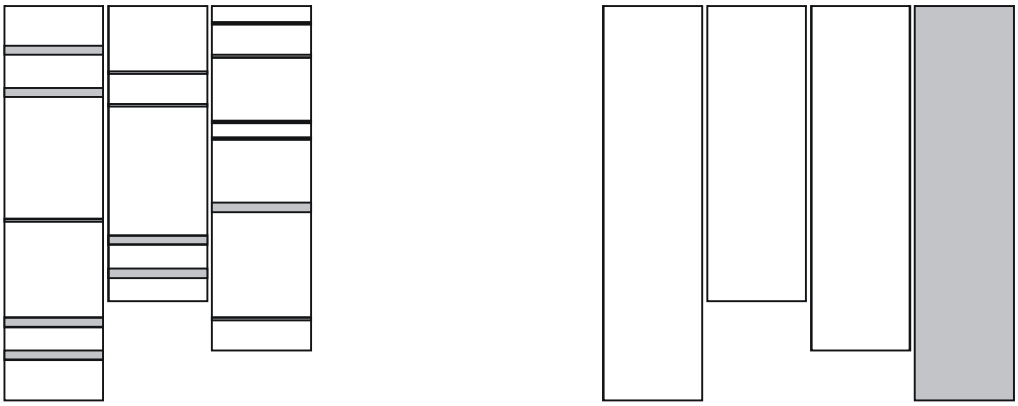
1. основная функциональность реализуется классами;
2. сквозная функциональность реализуется аспектами

Рисунок 1 наглядно иллюстрирует эффект от использования аспекта в коде приложения. Слева изображено приложение, состоящее из трех классов. Горизонтальные линии изображают код, который относится к сквозной функциональности. Эта функциональность пересекает все приложение из-за того, что все классы подвержены ее влиянию. Справа изображено то же приложение, но использующее аспект (закрашенный прямоугольник). Код, реализующий сквозную функциональность, полностью содержится в аспекте. Таким образом, приложение, спроектированное с использованием аспектов, является более простым для написания, поддержки и расширения.

ПРЕИМУЩЕСТВА АОП

АОП помогает избежать проблем, вызванных запутанным и рассредоточенным кодом. Ниже представлены дополнительные преимущества, предоставляемые 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.
- Упрощение сопровождения программной системы и простота



Без использования аспекта С использованием аспекта

Рисунок 1. Влияние аспекта на размещение сквозной функциональности

внесения в нее изменений. Поскольку 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.

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

Технология вынесения сквозной функциональности в отдельные аспектные модули стала важным эволюционным шагом в развитии таких концепций, как абстрагирование и повторное использование программного кода.

НЕДОСТАТКИ АСПЕКТНОГО ПОДХОДА

- Законченные и оттестированные компиляторы имеются только для нескольких языков программирования, что ограничивает применение аспектно-ориентированного программирования.
- Недостаточно качественная реализация расширений языков.
- Недостаточно проработан механизм привязки аспектов к компонентам. В распространенных в настоящее время реализациях АОП точки связывания описываются в терминах программных конструкций — классов, методов, полей класса. При этом получается тесная связь между аспектом и компонентом, к которому он применяется, что нарушает одну из основных идей АОП — независимость компонентов от применяемых к ним аспектов.
- Не полностью исследованы случаи, когда аспекты удобно и целесообразно применять.

ЗАКЛЮЧЕНИЕ

Применение новой методологии программирования позволяет значительно сократить время проектирования, разработки и отладки программного обеспечения, позволяя уже на этапе проектирования не задумываться о возможных требованиях, которые можно отнести к сквозной функциональности. Такие требования могут быть рассмотрены позже, когда в их реализации действительно возникнет необходимость. АОП не предлагает абсолютно новый подход к процессу проектирования, оно дает возможность учесть потенциальные требования без нарушения основной архитектуры проекта, и потратить меньше времени на сквозную функциональность на начальных стадиях.

Немаловажную роль играет и повторное использование кода, как программных модулей, так и аспектов в целом. Программный код избавляется от функциональности, которая не имеет непосредственно к нему никакого отношения, что способствует его повторному использованию, в то время как при традиционном подходе к разработке ПО повторно использовать код практически не представляется возможным из-за сквозной функциональности, присутствующей в нем.

Литература

1. Robert E. *Filman*, Tzilla *Elrad*, Siobhan *Clarke*, Mehmet *Aksit*. Aspect-Oriented Software Development // Addison-Wesley Professional. 2004.
2. Siobhán *Clarke*, Elisa *Baniassad*. Aspect-Oriented Analysis and Design: The Theme Approach // Addison-Wesley Professional. 2005.
3. Интернет адрес: http://en.wikipedia.org/wiki/Aspect-oriented_programming.
4. Интернет адрес: http://www.optim.su/CS/2003/4/AOP2/AOP.asp.

РАЗРАБОТКА СИСТЕМЫ ОБСЛЕДОВАНИЯ НА ОСНОВЕ 2D КИНЕМАТИЧЕСКОЙ МОДЕЛИ ОПОРНО-ДВИГАТЕЛЬНОГО АППАРАТА

А. Г. Кирковский

Эффективность работы организаций, в том числе и лечебных, зависит от автоматизации процессов, определяющих их функциональность. Ее роль состоит в упрощении и ускорении наиболее рутинных процессов.

Среди процессов, являющихся основой функционирования лечебных организаций центральную роль занимает лечебно-диагностический процесс. С развитием компьютерных технологий с одной стороны и накоплению богатой базы знаний с другой стороны стало возможным использование автоматизации и в этой области. Актуальность данной проблемы обусловлена как обилием рутинных процессов, требующих знаний, так и недостатком разработок в этом направлении. Сочетая компьютерные технологии и накопленные знания возможно значительно повысить эффективность диагностики и лечения. На сегодняшний день уже реализованы некоторые экспертные системы и системы поддержки принятия решений, направленные на решение этой проблемы.

Лечебно-диагностический процесс состоит из трех основных этапов: обследование, диагностика, лечение. Существующие системы практически полностью его охватывают. Каждый этап процесса очень важен. В частности в ходе обследования выявляются симптомы заболевания, от точности и качества определения которых непосредственно зависят остальные этапы. В этом процессе используются накопленные методики извлечения информации о состоянии здоровья человека[1].