поїздами мають показниковий розподіл. Встановлено, що кількість поїздів розподілено за нормальними законами, отримано диференційні функції розподілу поїздопотоків. Знання характеру розподілу поїздів дозволить виконати техніко-економічні розрахунки з визначення раціональних варіантів перерозподілу поїздопотоків у Дніпропетровському залізничному вузлі, а також визначити економічний ефект, що пов'язаний зі зменшенням експлуатаційних витрат на пропуск поїздів.

Список літератури: 1. *Шавзис, С. С.* Планирование поездообразования: новые подходы и решения [текст] / С. С. Шавзис // Залізничний транспорт – 2003. – №8 – С. 43–47. 2.*Ковалев, В. И.* Совершенствовать организацию и управление вагонопотоками [текст] / *В. И Ковалев, А. Т. Осьминин* // Железнодорожный транспорт – 2005. – №10 – С. 29–33. 3.*Липовец, Н. В.* Удосконалення організації пропускання вагонопотоків [текст] / Н. В. Липовец // Залізничний транспорт України – 2001. – №4 – С. 15–16. 4.*Ададуров, С. Є* Перевозочный процесс: направления инновационного развития [текст] / *С. Є Ададуров* // Железнодорожный транспорт – 2007. – №10 – С. 18–19. 5. *Музикіна, Г. І.* Проблема управління вантажними перевезеннями в умовах впливу економічних факторів [текст] / *Г. І. Музикіна, А. С. Савенко, П. В. Бех* // Вісник Академії митної служби України. – 2005. -№ 1 (25). - С. 51-57. 6.*Нагорный, Е. В.* Математическая модель функционирования каналов грузопотоков перевозки массовых грузов маршрутами [текст] / *Е. В. Нагорный, Н. Ю. Черныш* // Проблемы развития транспортных коммуникаций: Междун. сб. научн. тр. – Гомель: БелГУТ. –2000. –С. 75-83.7.*Авекитян, М. А.* Совершенствовать систему показателей эксплуатационной работы [текст] / *М. А. Авекитян* // Железнодорожный транспорт – 2005. – №10 – С. 10–18. 8.*Тулупов, Л. П.* Оптимизация управления перевозками на линейном уровне [текст] / *Л. П. Тулупов* // Железнодорожный транспорт – 2003. – №8 – С. 34–37. 9. *Долгополов, П. В.* Побудова моделі корпоративної мережі управління експлуатаційною роботою залізничного вузла [текст] / *П. В. Долгополов* // Збірник наукових праць УкрДАЗТ – Вип. 62 – Харків, 2004. – С. 31–37. 10. *Бутько, Т. В.* Удосконалення роботи залізничних вузлів при впровадженні варіантних технологій [текст] / *Т. В. Бутько, Д. В. Ломотько, О. А. Малахова* // Збірник праць КУЕТТ – 2003. Вип.4 – С. 56–60. 11.*Елисеев, С. Ю.* Концепция управления грузовыми перевозками в транспортных узлах с применением логистических центров [текст] / *С. Ю. Елисеев* // Вестник транспорта – 2006 – №2 – С. 12–14. 12. *Мишко, С. П.* Модель оперативного управління вантажними потоками залізничного вузла в термінах мереж Петрі [текст] / *С. П. Мишко* // Економіст, 2005. – № 8 – С. 77–79. 13. *Лаврухін, О. В.* Застосування апарату нечітких нейронних мереж для моделювання системи управління роботою залізничного вузла [текст] / *О. В. Лаврухін, В. В. Петрушов* // Збірник наукових праць ДонІЗТ, 2006 – №5 – С. 13–17. 14.*Федотов, Н. И.* Применение теории вероятности в транспортных расчетах [текст] / *Н. И. Федотов, А. В, Быкадоров* // Новосибирск – 1969. – 185 с.

Поступила в редколлегию 28.02.2011

УДК 004.05

С.Г.ЧЁРНЫЙ к.т.н., доц. Керченский государственный морской технологический университет

RE-ENGINEERING COMPONENT MODELS ДЛЯ ОПТИМИЗАЦИИ ТРАНСФОРМАЦИОННЫХ ПРОЦЕССОВ В ПРОГРАММНОМ КОДЕ

Розглянуто можливості редагування елементів програмного коду за допомогою процесу ре факторингу. Наведено приклади використання сегментації різноманітних структур для проектування компонентів програмних систем.

Ключові слова: модуль, рефакторинг, фрагмент, компонент.

Рассмотрены возможности редактирования элементов программного кода с помощью технологии рефакторинга. Приведены примеры использования сегментации различных структур для проектирования компонентов программных систем.
Ключевые слова: модуль, рефакторинг, компонент.

The possibilities of editing element code with refactoring technology. Examples of the use segmentation of different structures for the design components software systems.
Keywords: module, refactoring, components.

Постановка задачи в общем виде и ее актуальность

Практически невозможно представить сейчас существенное коммерческое приложение, не взаимодействующее с базой данных (БД), что подразумевает уровень архитектуры и обеспечение взаимодействие с БД. Технология рефакторинга для подобных структур достаточно является нелегким. В основе рефакторинга лежит последовательность небольших эквивалентных преобразований. Прежде всего рефакторинг обеспечивает постепенное развитие кода во времени, в результате чего реализуется эволюционный подход к программированию. Особенностью рефакторинга является то, что он сохраняет функциональную семантику базовости кода.

Одни из ведущих разработчиков этого направления, организующие свою работу на принципах адаптивного программирования, в частности, специалисты по экстремальному программированию (Extreme Programmer - XPer), полагают, что рефакторинг является ведущим подходом к разработке подобного рода систем и компонентов.

Наиболее часто употребимые методы рефакторинга: Change Method Signature; Encapsulate Field; Extract Class; Extract Interface; Extract Local Variable; Extract Method; Generalize Type; Inline; Introduce Factory; Introduce Parameter; Pull Up; Push Down; Replace Conditional with Polymorphism.

Рефакторинг желательно применять постоянно при разработке кода, что будет приводить к структурной детализации кода и стилизованной форме представления. Основными стимулами проведения являются задачи: необходимо добавить новую функцию, которая не достаточно укладывается в принятое архитектурное решение программного модуля; необходимо исправить ошибку, причины возникновения которой не выделены четко структурированной базовой внешней формой; проблематика в командной разработке, которая обусловлена сложностью логики программного продукта.

Основной материал

Операции рефакторинга БД концептуально являются более сложными, чем операции рефакторинга кода, поскольку при проведении операций рефакторинга кода необходимо заботиться лишь о сохранении функциональной семантики структуры фрагментации, а при осуществлении операций рефакторинга БД возникает процесс необходимости и сохранение информационной семантики [1-2].

При осуществлении процесса рефакторинга необходимо четко представлять какой из методов более лучший и оптимальный для данной структуры программного кода, ведь несвоеобразное использование методов приведет

разработчика к трудностям роботы программного модуля. При процессе рефакторинга разработчик старается сделать так, чтобы код стал удобнее для понимания и его поддержки, а при оптимизации кода приходится делать процедуры, которые приводят к обратному эффекту, что приводит к проблемам читаемости кода, но за счет этого возрастает скорость его выполнения. В общем аспекте проблемы, возникающие при проведении рефакторинга делятся на две категории: проблемы, связанные с БД; проблемы изменения интерфейсов.

Процесс переноса фрагментов кода можно разделить на обобщенные этапы: в оригинальном коде постепенно заменяется все, что использует специфические возможности исходного языка, на более простое, но эквивалентное по функционалу, что может привести медленной работе кода и к не корректному внешнему виду; редактируемый код приводится к виду, который сможет собрать новый компилятор; переносятся тесты, и код на новом языке доводится до совпадения по функционалу с оригинальным кодом. В настоящее время существует много ORM-прослоек, позволяющих общаться с БД в терминах сущностей. Инструменты рефакторинга могут быть реализованы с DMS для различных языков включают в себя [2]: поиск и удаление повторяющегося кода; форматирование кода для удобства просмотра; переименование идентификаторов; замена spaghetti-tangles gotos с структурированным кодом; использование CASE фрагментов вместо вложенных IF на той же переменной; включение функции / метода документации автоматизированного извлечения фактов; стиль проверки отладки; преобразование блоков кода в функции / подпрограмм / методов с соответствующими параметрами; удаление «мертвого кода»; реструктуризация APIs-интерфейсы для поддержки различных операционных систем.

Все коммуникации с БД приложение осуществляет через прослойку DataAccessLayer, которая предоставляет собой сервисы для доступа к данным. Семантические конструкции могут сделать это в качестве функции, или может предоставить программному модулю с помощью инструментов и обучение по использованию DMS для осуществления таких процессов рефакторинга (рис.1) [2].

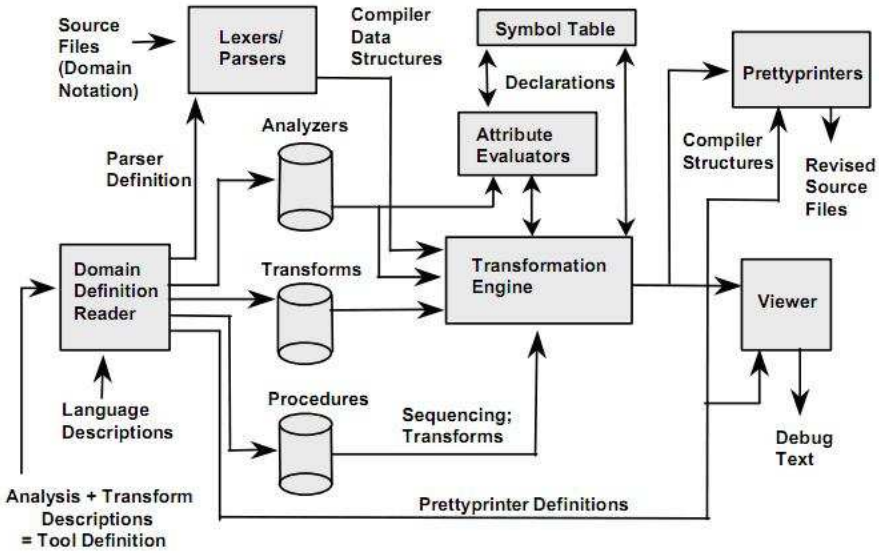

Рис.1. Архитектура DMS

На сегодняшний день выделяются три различные технологии, поддерживающие концепцию распределенных объектных систем. Это технологии RMI, CORBA и DCOM. Архитектура RMI (Remote Method Invocation, т.е. вызов удаленного метода), которая интегрирована с JDK1.1, является продуктом компании JavaSoft и реализует распределенную модель вычислений. RMI позволяет клиентским и серверным приложениям через сеть вызывать методы клиентов/серверов, выполняющихся в Java Virtual Machine. Хотя RMI считается легковесной и менее мощной, чем CORBA и DCOM тем не менее, она обладает рядом уникальных свойств, таких как распределенное, автоматическое управление объектами и возможность пересылать сами объекты от машине к машине.

Технология CORBA [3] (Common Object Request Broker Architecture), разрабатываемая OMG (Object Managment Group) с 1990-го года, позволяет вызывать методы у объектов, находящихся в сети где угодно, так, как если бы все они были локальными объектами.

Технология DCOM (Distributed Component Object Model) была разработана компанией Microsoft в качестве решения для распределенных систем. Сейчас DCOM является главным конкурентом CORBA, хотя контролируется он теперь уже не Microsoft, а группой TOG (The Open Group), аналогичной OMG. DCOM представляет собой расширение архитектуры COM до уровня сетевых приложений.

Прикладные компоненты реализуют определенную бизнес-логику, распределены по сети и могут быть использованы многократно и в последнее время завоевывают большую популярность в качестве строительных блоков для создания сложных распределенных приложений, что привлекает внимание к базовым объектным архитектурам для создания распределенных объектных программных систем. Основные компоненты: компонентная объектная модель Component Object Model (COM), разработанная корпорацией Microsoft, и общая архитектура брокеров объектных запросов Common Object Request Broker Architecture (CORBA), которую развивает Консорциум OMG.

Функции CORBA (рис.2 [4]) и COM - функции промежуточного программного обеспечения объектной среды.

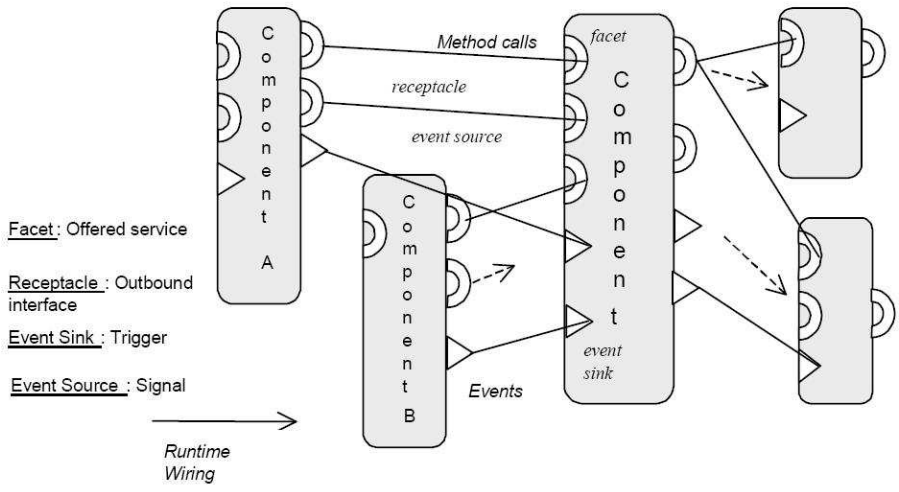

Рис.2. Структура компонента сущностей CORBA

Для обеспечения взаимодействие объектов и их интеграцию в цельную систему, архитектура промежуточного уровня должна реализовать несколько базовых принципов: независимость от физического размещения объекта (компоненты программного обеспечения не обязаны

находиться в одном исполняемом файле, выполняться в рамках одного процесса или размещаться на одной аппаратной системе);независимость от платформы (компоненты могут выполняться на различных аппаратных и операционных платформах, взаимодействуя друг с другом в рамках единой системы); независимость от языка программирования (различия в языках, которые используются при создании компонентов, не препятствуют их взаимодействию друг с другом).

Если посмотреть на структуру объектов, то можно заметить, что всем им присущи некоторые одинаковые поля – ID. Если воспользоваться терминами DDD (Domain Driven Design), то можно выделить общий интерфейс для всех объектов.

Выводы и предложения

Произведен анализ процесса рефакторинга в структуре оптимизации программного кода. Затронут аспект применения технологии, поддерживающий концепцию распределенных объектных систем. В рамках прикладных компонентов CORBA и DCOM рассмотрен процесс осуществления рефакторинга. Такой рефакторинг сервисов, предоставляющих работу с данными, открывает возможность для рефакторинга других частей приложения, позволяет выполнить преобразования некоторых частей приложения в соответствии с полезными методиками, однако тут есть и свои минусы – трудоемкость и порой невозможность проведения таких изменений, если приложение больших размеров и большая часть сервисов уже написана.

Список литературы: 1.Операции рефакторинга базы даннях [Електронний ресурс] ИД Вильямс // С.45-49. - Название с титул. экрана. - Режим доступа: http://www.williamspublishing.com/PDF/978-5-8459-1157-5/part.pdf . 2.Методы улучшения качества кода: рефакторинг. [Електронний ресурс] - Название с титул. экрана. - Режим доступа: http://social.msdn.microsoft.com/Forums/ru-ru/fordesktopru/thread/63d9ba19-491b-4e75-9796-6de5132e1a56 3. Сравнение COM и CORBA [Електронний ресурс] - Название с титул. экрана. - Режим доступа: http://kunegin.narod.ru/ref3/corba5/12.htm 4. Robert L., Ira D., Michael Mehlich. Re-engineering C++ Component Models Via Automatic Program Transformation. - Название с титул. экрана. - Режим доступа: http://www.semanticdesigns.com/Company/Publications/WCRE05.pdf

Поступила в редколлегию 19.03.2011

УДК 519.8

А. М. МЕЗЕРНИЙ, студент, НТУ «ХПІ»
Д. В. КУКЛЕНКО, канд. техн. наук, доц., НТУ «ХПІ»

ЗАСТОСУВАННЯ КОНЦЕПЦІЇ АВТОМАТИЗОВАНОЇ ПОБУДОВИ СПЕЦИФІКАЦІЇ ВИМОГ ДО ПРОЦЕСУ ОБЛІКУ ТА ФОРМАЛІЗАЦІЇ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В статті розглянуто основні проблеми та актуальність управління вимогами. Запропоновано підхід до виконання обліку та формалізації вимог до програмного забезпечення на основі концепції автоматизованого генератора специфікації вимог.