Схема компоновки червячного редуктора в сечении вдоль оси червяка с условно изображенной нагрузкой на схеме вала.

Новая технология процесса выполнения компоновки эффективно способствует интенсификации процесса курсового проектирования деталей машин не только при проведении индивидуальных и групповых консультаций со студентами дневной и заочной форм обучения, но и дает возможность студентам даже без консультаций преподавателя качественно и за сравнительно короткий период времени выполнить компоновку червячного редуктора.

Реализация на практике, в процессе выполнения компоновки, таких принципов обучения, как наглядность, доступность и научность, способствует повышению качества курсовых проектов. Компоновка редукторов увязана с действующей методикой подбора подшипников качения [3].

ЛИТЕРАТУРА

1. Кузин Н.А. Новый подход к решению вопросов, связанных с компоновкой редукторов // Информационные и сетевые технологии – образовательная среда XXI века: Материалы Республиканской научно-методической конференции. – Мн., 2003.– С.58 – 62. 2. Кузин Н.А. Комплекс специальных учебных пособий и новая методика проведения занятий по компоновке зубчатых и червячных редукторов // Машиностроение. – Мн., 2004. – Вып. 20. – С.322 – 328. 3. Кузин Н.А. Техническая механика. Выбор и расчет подшипников качения. – Мн.: УП «Технопринт», 2001. – 102 с.

УДК 658.512.011.56

Бочкарёва Л.В., Кирейцев М.В.

МЕТОДИКА ОБУЧЕНИЯ РАЗРАБОТКЕ ПРОГРАММНЫХ СРЕДСТВ В СРЕДЕ RATIONAL ROSE

Белорусский государственный университет информатики и радиоэлектроники,
Объединенный институт проблем информатики НАН Беларуси
Минск, Беларусь

В статье рассматриваются вопросы, связанные с изучением студентами курса «Системы автоматизированного проектирования программного обеспечения». Цель обучения заключается в приобретении студентами следующих знаний и навыков:

- Выбор и применение технологий для разработки программных средств (ПС);
- Применение языка UML (Unified Modeling Language) для моделирования и проектирования ПС;
- Применение CASE Rational Rose для создания ПС.

Сегодня становится очевидным, что построение модели будущей системы – это возможность значительно сократить время разработки, уложиться в бюджет и создать продукт высокого качества [1]. Модель помогает еще на стадии проектирования получить представление о поведении системы и избежать дорогостоящих ошибок в дальнейшем, когда в написание программного кода вложены значительные средства. Для моделирования предметной области на рынке программных продуктов представлен широкий спектр CASE-средств. Наиболее популярными среди них являются Rational Rose, BPwin, Silverrun, Process Analyst. Хотя на практике встречаются задачи, например, исследование линейных интервальных динамических систем методами корневых траекторий, которые не удается решить в рамках стандартных CASE-продуктов и приходится создавать специализированные инструментальные средства [2].

Важной характеристикой CASE-системы является комплексность подхода и использование единой унифицированной нотации не только на этапе моделирования, но и на последующих стадиях разработки программного продукта, как это представлено в Rational Rose. Для работы в среде необходим UML, который является графическим языком описания архитектуры системы. Программа на UML не кодируется, а описывается при помощи диаграмм, состоящих из объектов и связей между ними или из этапов процесса разработки [3]. В распоряжение проектировщика системы Rational Rose предоставляет следующие типы диаграмм, последовательное создание которых позволяет получить полное представление обо всей проектируемой системе и об отдельных ее компонентах [4]:

- Use case diagram (диаграмма сценариев);
- Deployment diagram (диаграмма топологии);
- Statechart diagram (диаграмма состояний);
- Activity diagram (диаграмма активности);
- Interaction diagram (диаграмма взаимодействия);

1. Sequence diagram (диаграмма последовательностей действий);
2. Collaboration diagram (диаграмма сотрудничества);

- Class diagram (диаграмма классов);
- Component diagram (диаграмма компонентов).

Модель создаваемой системы можно рассматривать с четырех точек зрения:

• *Use Case View* содержит в себе исходные данные проекта и того, кто их вводит, действия программы, выходные данные и того, кому они передаются.

• *Logical View* отражает логику программы: классы, их свойства и методы, отношения между классами. Рекомендуется приступать к работе с этим аспектом модели, когда уже определено, что и в какой последовательности будет выполнять программа.

• *Component View* показывает, на какие компоненты разбит проект и что помещено в каждый из них. На данном этапе проектируются модули и зависимости между ними, переходы от главной программы к подпрограммам;

• *Deployment View* включает в себя расположение физических устройств и связей между ними.

Следуя рекомендации и практическому опыту, Rational Rose используется проектировщиками и разработчиками. Графические возможности продукта позволяют решить задачи как на уровне общей модели процессов предприятия, так и на уровне конкретной модели класса в создаваемом программном обеспечении (ПО). Пользуясь Rational Rose, проектировщик может предложить заказчику не абстрактное словесное описание системы, а его конкретную модель, которая затем дополняется описанием классов на языке программирования. Rational Rose поддерживает проектирование, основанное на двух способах: прямом и обратном. В первом режиме разработчик строит диаграммы классов и их взаимодействия, а на выходе получает сгенерированный код на заданном языке программирования. Во втором режиме возможно построение модели на базе имеющегося исходного кода. Отсюда следует главная возможность для разработчика: повторное проектирование (round-trip). Он описывает классы в Rational Rose, генерирует код, вносит изменения в модель и снова пропускает ее через Rational Rose для получения обновленного результата.

В соединении с другими программными пакетами Rational Rose приобретает еще большие возможности. В сочетании со средствами документирования Rational SoDA он может давать полное представление о проекте. Полностью интегрируясь с Microsoft Visual Studio, этот пакет позволяет получать исходный код взаимодействующих классов и строить визуальные модели по существующему исходному коду. Возможность интеграции со средствами управления требованиями Requisite Pro, со средствами тестирования SQA Suite, Performance Studio, со средствами конфигурационного управления ClearCase, PVCS поднимает процесс ведения программного проекта на качественно новый уровень.

Подведя итог вышесказанному, можно выделить следующие преимущества от применения Rational Rose.

• сокращение цикла разработки приложения "заказчик-

программист-заказчик";

- увеличение продуктивности работы программиста;
- улучшение потребительских качеств создаваемых программ за счет ориентации на пользователей и бизнес;
- способность вести большие проекты и группы проектов;
- возможность повторного использования уже созданного программного обеспечения за счет упора на разбор их архитектуры и компонентов.

Для закрепления теоретических знаний студентам предлагается выполнить лабораторный практикум. Им необходимо создать модель программной системы в среде автоматизированного синтеза Rational Rose. Модель строится в виде набора диаграмм, последней из которых создается диаграмма классов и на ее основе синтезируется программный код. Исходные данные, требуемые для выполнения поставленной задачи, выдаются студентам в форме технического задания на проектирование. Кроме этого, студентам предоставляется информация по работе в среде Rational Rose: описание пунктов меню и команд, относящихся к ним, а также способы построения диаграмм каждого типа и наборы их инструментов. Процесс моделирования разбит на этапы. Каждый из них связан с построением диаграмм определенного типа и заключен в рамки лабораторной работы. Для выполнения всех лабораторных работ предлагается единый порядок, предусматривающий следующие шаги:

- Ознакомление с постановкой задачи и исходными данными;
- Разработка предлагаемой диаграммы;
- Реализация диаграммы в среде Rational Rose;
- Сохранение файла модели;
- Составление отчета по выполненной работе.

Порядок построения диаграмм (выполнения лабораторных работ) отвечает методологии разработки информационных систем. Сначала предлагается построить Use case diagram. Она позволяет создать список операций, которые выполняет система. На основе набора Use case создается список требований к системе и определяется множество выполняемых ею функций. Данный тип диаграмм используется при описании бизнес процессов автоматизируемой предметной области и определении требований к программной системе.

Вторая лабораторная работа посвящена диаграмме Deployment. Она предназначена для анализа аппаратной части системы. При помощи Deployment проектировщик может выполнить анализ необходимой аппаратной конфигурации, на которой будут работать отдельные процессы системы, и описать их взаимодействие между собой и с другими аппаратными устройствами. Этот тип диаграмм также позволяет анализировать взаимодействие процессов, работающих на разных компьютерах сети.

Третий этап – это построение диаграммы Statechart, предназначенной для определения состояний объектов и условий переходов между ними. Описание состояний позволяет точно установить модель поведения объекта при получении различных сообщений и взаимодействии с другими объектами. Диаграмма дает возможность четко представить все поведение будущего программного объекта в виде графических значков состояний. Второй тип диаграмм состояний – Activity diagram. Главное различие между Statechart и Activity diagram в том, что первая характеризует статичные состояния, а вторая – действия. При этом Activity больше подходит для моделирования последовательности действия, а Statechart – для моделирования дискретных состояний объекта.

Следующий этап моделирования связан с изучением взаимодействия имеющихся объектов, определением клиентов и серверов и порядка обмена сообщениями между ними. Диаграмма Sequence позволяет получить отражение процесса обмена сообщениями во времени. Второй тип диаграмм взаимодействия – Collaboration. Она отличается от предыдущей тем, что не акцентирует внимание на последовательности передачи сообщений, а отражает наличие взаимосвязей вообще, т.е. на ней отражается наличие сообщений от клиентов к серверам. Из-за отсутствия временной шкалы Collaboration получается более компактной и оптимально подходит для отражения взаимодействия всех объектов системы. Однако необходимо понимать, что взаимодействие объектов отличается от взаимодействия классов, являясь мгновенным снимком системы в некотором состоянии. Поскольку объекты создаются и уничтожаются на всем протяжении работы программы, и в каждый момент имеется конкретная группа объектов, с которыми осуществляется работа. В связи с этим появляются такие понятия, как время жизни и область видимости объектов.

Component diagram – диаграмма компонентов позволяет создать физическое отражение текущей модели. Диаграмма показывает организацию и взаимосвязи программных компонентов, представленных в исходном коде, двоичных или выполняемых файлах. Связи здесь представляют зависимости одного компонента от другого и имеют специальное отображение через значок “зависимости”. Кроме того, данный тип диаграмм иллюстрирует поведение компонентов по предоставляемому ими интерфейсу.

На последнем этапе моделирования создается диаграмма классов, которая является основной для получения кода приложения. При помощи Class diagram строится внутренняя структура системы, описывается наследование и взаимное положение классов относительно друг друга. Здесь создается логическое представление системы. Обычно Class diagram строится для всех классов системы в отличие от диаграмм взаимодействия, на которых отражены только отдельные объекты. Классы редко бывают изолированы, чаще они взаимодействуют друг с другом, что отражается при помощи различного

вида связей. Типы связей влияют на получаемый при генерации исходный код, поэтому студентам рекомендуется рассмотреть связи и их спецификации.

На основе диаграммы классов в среде Rational Rose создается код класса на выбранном языке. Для того чтобы воспользоваться данной возможностью, необходимо убедиться, что выбранный язык программирования установлен при помощи Add-Ins менеджера. Студентам предлагается генерировать код на Microsoft Visual C++. Для чего необходимо связать каждый класс с выбранным языком, определить компонент, в котором класс будет храниться. В случае работы с Microsoft Visual C++ программист получает доступ ко всей иерархии классов библиотеки MFC при помощи визуальных средств Model Assistant. На заключительном этапе разработки системы создается приложение на Microsoft Visual C++.

Требования, предъявляемые к современным программным продуктам, заставляют ставить процесс их разработки на индустриальную основу. Поэтому необходим инструмент, позволяющий уменьшить долю ручного труда при программировании, страхующий от ошибок и предоставляющий большие возможности для творчества. Таким инструментом является Rational Rose. Его можно использовать как отдельно, так и вместе с другими продуктами Rational Software, которые позволяют создавать сложные программные системы быстрее, качественнее и легче. Не случайно Rational Rose включается во все конфигурации Rational Suite – инструмента для аналитиков, программистов и тестировщиков.

ЛИТЕРАТУРА

1. Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1998. – 176 с. 2. Бочкарёва Л.В. Учебно-методическое пособие по курсу «Технология разработки и системы автоматизированного проектирования программного обеспечения» для студентов специальности ПОИТ. – Мн.: БГУИР, 2002. – 52 с. 3. Ларман К. Применение UML и шаблонов проектирования.: Пер. с англ.: Уч. Пос. – М.: Издательский дом “Вильямс”, 2001. – 496 с. 4. Трофимов С.А. CASE-технологии: практическая работа в Rational Rose. – М.:Бином-Пресс, 2002. – 288 с.