Федеральное агентство по образованию РФ

ГОУ ВПО Нижегородский государственный университет им. Н.И. Лобачевского

Факультет Вычислительной математики и кибернетики

Кафедра Математического обеспечения ЭВМ

УЧЕБНЫЙ КУРС

«Технологии программирования.   
Курс на базе Microsoft Solutions Framework (MSF)»

для подготовки по направлению «Информационные технологии»

Нижний Новгород  
2006

Содержание

HYPERLINK \l "\_Toc145664563"1.1.

HYPERLINK \l "\_Toc145664564"1.2.

HYPERLINK \l "\_Toc145664565"1.3.

HYPERLINK \l "\_Toc145664566"2.

HYPERLINK \l "\_Toc145664567"3.

HYPERLINK \l "\_Toc145664568"4.

HYPERLINK \l "\_Toc145664569"5.

HYPERLINK \l "\_Toc145664570"5.1.

HYPERLINK \l "\_Toc145664571"5.2.

HYPERLINK \l "\_Toc145664572"5.3.

HYPERLINK \l "\_Toc145664573"5.4.

HYPERLINK \l "\_Toc145664574"6.

Немного терминологии

Программирование

На протяжении всего времени обучения на факультете мы изучаем программирование. Программирование (Computer science) – молодая, активно развивающаяся область.

Долгое время человечество волнует вопрос о том, к какому роду деятельности относится программирование. В 60-х – 70-х годах XX века данный вопрос активно обсуждался на научных конференциях. Существовало 2 популярных точки зрения: «программирование это искусство» и «программирование это наука». К единому мнению придти так и не удалось. В настоящий момент мы можем добавить к этим популярным трактовкам еще одну: «программирование это бизнес». Чтобы понять, что программирование это бизнес, достаточно посмотреть, какими числами выражаются доходы современных IT-компаний. Так, например, по данным HYPERLINK["]( ") доход корпорации Microsoft за 2005 финансовый год составил 39,70 млрд. $. Впечатлены? Вам нравится этот бизнес? Тогда приступим к изучению курса.

IT-проекты

Будем понимать под IT-проектами проекты в области информационных технологий. Будем далее рассматривать лишь те IT-проекты, целью которых является разработка программного обеспечения.

Зададимся следующими вопросами:

 Что такое программное обеспечение (ПО)?

 Чем ПО отличается от обычной программы?

 Вчера мы с другом написали «Калькулятор». Определенно, это программа. Является ли она ПО?

Программы и программное обеспечение (программные продукты)

Программное обеспечение (Software) – набор компьютерных программ, процедур и связанной с ними документации и данных (ISO/IEC 12207).

Таким образом, программное обеспечение – это не просто программа. Это еще и документация и руководство пользователя.

Вместо словосочетания «программное обеспечение» часто используют другое –  «программный продукт». Будем далее считать, что это одно и то же. Одно из главных свойств программного продукта – продаваемость. Продаваемость – залог успеха бизнеса по разработке программного обеспечения. Если вы собираетесь что-то разработать, это должно быть востребовано на рынке. В противном случае вы потратите деньги на разработку (зарплата сотрудников, накладные расходы, налоги, аренда помещения...) и ничего не получите взамен. Вы можете написать замечательную программу. Реализовать там новый быстрый алгоритм. Она может великолепно работать, но если она никому не нужна, то вы (как компания) на пути банкротству. Допустим, в таких программах, как ваша, действительно есть потребность. Допустим, вы год упорно работали, и вот, казалось бы, настал ваш звездный час: все готово, все модули написаны, отлажены, собраны вместе и, как вам кажется, работают. Один «маленький» момент портит всю картину – если у вас нет хорошего (!) руководства пользователя (инструкции), желательно, в русскоязычном и англоязычном вариантах, то вашу программу никто не купит, особенно за границей. Если у вас все есть, но нет специалистов по рекламе, то про вашу программу никто не узнает. Если ...

Подытожим: программный продукт – это программа со всей сопутствующей документацией, программа, которую можно продать, либо извлечь из нею финансовую выгоду другим образом.

Вернитесь мысленно к пункту 1.2 и еще раз попробуйте ответить на поставленные вопросы. Получилось? Тогда перейдем к краткому обзору текущего состояния дел в отрасли разработки ПО в России и в мире.

Бизнес и IT-проекты. Рынок ПО в России и в мире. Немного статистики

Для того чтобы бизнес был успешным, необходимо (но не достаточно) выполнение многих условий:

 Продукт должен выходить на рынок

- надлежащего качества;

- вовремя;

- интересным потенциальным пользователям.

 Расходы должны соответствовать изначальному бюджету.

image

Рис. 1.1.  Статистика успешности IT-проектов.  
 По данным The Standish Group International, Extreme Chaos. [-1](#ftn0)

К сожалению, ситуация такова, что многие проекты не удовлетворяют этим, казалось бы естественным, условиям. Рассмотрим некоторую статистику (рис. 1.1).

Приведем расшифровку степени успешности проектов:

 Проваленные: закончились неудачей – цель вообще не была достигнута.

 Испытавшие большие проблемы: закончились созданием продукта, но превысили бюджет или (и) не уложились во время или (и) имеют лишь частичную функциональность.

 Успешные: закончились созданием продукта, уложились в бюджет и время. Вся планируемая функциональность реализована.

Как видите, доля успешных проектов неуклонно возрастает, оставаясь по-прежнему сенсационно малой. И это притом, что в 2004 году на разработку программных средств ушло около 3 700 000 000$.

Рассмотрим еще один любопытный график:

image

Рис. 1.2.  Доля успешных проектов в зависимости от их бюджета. [0](#ftn1)

Данная диаграмма говорит о том, что с ростом размера проекта (бюджет характеризует в данном случае размер и сложность задачи) шансы на его успех катастрофически падают.

Поговорим о текущем положении отрасли в России. В конце 90-х годов, обсуждая этот вопрос, мы приводили следующие качественные характеристики:

 Хорошие программисты.

 Грамотные аналитики.

 Недостаток хороших управленцев.

 Проблемы с документированием и локализацией.

 Проблемы с рекламой и продвижением.

Прошло всего 5 лет, и ситуация существенно изменилась к лучшему. Объем экспорта программного обеспечения из России в 2005г. превысил 1млрд.$ (для сравнения экспорт в автомобильной отрасли составил 380млн.$, в атомной энергетике – 850млн.$). Объем IT-рынка в 2004г. составил 9,2млрд.$, в 2005г. рост составил 22,1% (в то время как мире всего порядка 6%)! Конечно, доля нашего рынка в объемах мирового IT-рынка по-прежнему невелика (объем IT-рынка в мире в 2005г. составил 900млрд.$), но тенденция выглядит обнадеживающе. При этом объем рынка разработки программного обеспечения в России в 2005г. составил 1,4млрд.$ (OLE - object! от всего IT-рынка). В среднем этот показатель в России растет на 40-50% в год.[1](#ftn2)

Таким образом, основные тенденции на сегодняшний день представляются следующими:

 Быстрый рост объемов IT-рынка, рынка ПО.

 Укрепление позиций российских компаний.

 По-прежнему малая доля в мировых объемах.

Для того чтобы повысить объемы рынка, необходимо повысить показатель успешности проектов, для чего отрасль нуждается не только в новых технологиях, но и в грамотных специалистах, способных эти технологии применять. Надеемся, что вы пополните эту, без сомнения почетную и востребованную когорту.

О предмете

Задачи нашего предмета:

 Изучить причины неудач IT-проектов.

 Выявить способы устранения этих причин.

 Научиться применять эти способы на практике.

Причины неудачи IT-проектов

Почему IT-проекты терпят неудачи?

Почему, казалось бы, хорошо спланированный проект не укладывается во временные рамки?

Почему по прошествии некоторого времени выясняется, что имеющегося бюджета недостаточно?

Почему полученный в итоге продукт не пользуется спросом?

Проблема сложна и многогранна. Трудно перечислить все возможные причины неудачи. Остановимся кратко на некоторых из них, представляющихся нам наиболее существенными.

Причина 1. Нереалистичные временные рамки.

Правильно оценить время, необходимое для выполнения проекта, – сложная задача, решение которой часто не под силу даже опытным менеджерам. Существуют специальные критерии, которые помогают принимать правильные решения, такие как учет времени в человеко-часах и т.д. Тем не менее, задача остается сложной, колоссальное значение в ней имеет грамотный учет рисков (далее мы поговорим про это подробнее).

Причина 2. Недостаток количества исполнителей.

Иногда менеджер решает сэкономить, иногда переоценивает возможности своих сотрудников, иногда в ходе разработки выясняется, что задача сложнее, чем казалось на самом деле, – проблема недостатка рабочих рук, так или иначе, возникает достаточно часто.

Причина 3. Размытые границы проекта.

Одна из наиболее серьезных причин неудачи проекта – нечетко сформулированные цели, неоднократно меняющиеся в ходе разработки. Поверьте, многоэтажные дома и дачные домики строятся на основе применения разных технологий и материалов. Если вам доведется управлять проектом – сделайте все, чтобы четко сформулировать требования к системе в соответствии с пожеланиями пользователя. Мы поговорим про это подробнее в подразделе «Управление требованиями».

Причина 4. Недостаток средств.

Известны две крайности при планировании бюджета: чрезмерное раздувание (подход пессимиста) и чрезмерное уменьшение (подход оптимиста). Использование первого подхода чаще всего (если только ваш заказчик не совсем дилетант) приводит к тому, что ваша команда теряет проект. «Слишком дорого, сэр. Мы идем к Вашим конкурентам». Второй подход часто применяется не только в силу оптимизма менеджмента, но и в рекламных целях, чтобы любой ценой выиграть проект. «Мы сейчас напишем меньше всех, а там видно будет». Увы, в дальнейшем приходится расплачиваться за демпинговые меры. Качественно реализовать проект за выделенные деньги оказывается просто невозможным. Представляется разумным оценивать бюджет реально с некоторой перестраховкой на случай непредвиденных ситуаций (заболел ключевой сотрудник, вышло из строя дорогостоящее оборудование...). Не выиграем этот проект – выиграем другой. Хуже, если выиграем, но провалим. В нашу состоятельность больше могут и не поверить.

Причина 5. Нехватка квалифицированных кадров.

Нехватка квалифицированных специалистов – одна из существенных проблем отрасли. Технологии развиваются с такой скоростью, что профессионалы вынуждены все время обновлять свои знания. Относительная новизна самой области IT, с одной стороны, становящееся повсеместным внедрение информационных технологий во все сферы человеческой деятельности, с другой, а, значит, все возрастающий спрос на специалистов ведут к существенной нехватке квалифицированных кадров. Конечно, все хотят принять на работу лучших. Но опыт показывает, что их не так много, и на всех не хватает. Умение из потока кандидатов выбрать тех, кто вам нужен, очень важное качество специалистов по кадрам. Часто к подбору сотрудников рекомендуют привлекать всех членов команды. То, как новичок впишется в коллектив, совсем не последнее дело.

Технологии программирования – путь к успеху в разработке ПО

Технология – совокупность производственных процессов в определенной отрасли производства, а также научное описание способов производства [].

Уже в 60-х–70-х годах XX людям было ясно, что ввиду роста сложности решаемых при помощи компьютера задач неимоверно возрастает стоимость разработки программ. Причем, если стоимость аппаратуры растет умеренными темпами, а иногда и вовсе падает, то со стоимостью разработки программ ничего поделать не удается. Именно тогда вопрос о том, как оптимизировать процесс разработки, вышел на первый план.

Для того чтобы ответить на этот вопрос, потребовалось определить, куда уходят средства, за счет чего возрастает стоимость разработки программных систем?

Создание любой программной системы выполняется по некоторой схеме. Данная схема представляет собой последовательность стандартных этапов (очень приблизительно эта схема может выглядеть так: анализ, проектирование, разработка, тестирование, модификация). Именно на этих этапах и возникают существенные финансовые затраты. Для их оптимизации необходимо было понять, что программирование есть обычный технологический процесс, по характеру возникающих проблем мало чем отличающийся от, скажем, строительства дома или корабля.

Для сокращения затрат необходимо было конкретизировать схему, упорядочить действия, выполняемые на каждом этапе, разработать методы решения возникающих на разных этапах проблем. В довершении ко всему, схема подразумевает возвраты назад (циклы), в тех случаях, когда обнаруживается ошибка предыдущего этапа.

В результате кропотливой работы большого количества специалистов на каждом этапе и подэтапе возникли и продолжают появляться и совершенствоваться специальные технологии, позволяющие решать задачи в заданные сроки с заданным качеством.

Итак, технология программирования – совокупность методов, приемов и средств для сокращения стоимости и повышения качества разработки программных систем.

В любой серьезной компании, занимающейся разработкой программного обеспечения, на каждом этапе процесса разработки применяется большое количество разных технологий. Над созданием программного продукта работают представители таких специальностей как: аналитики, управленцы (менеджеры), тестеры, кодировщики, (программисты), технические писатели, системные администраторы, специалисты по повторному использованию, дизайнеры, специалисты по эргономике и др. Сейчас мы вспомним то, что вам должно быть известно из курса «Основы программирования», – ту часть технологий, которая наиболее устоялась и имеет отношение к общим вопросам анализа, проектирования и разработки: структурное, модульное, объектно-ориентированное и компонентное программирование.

Структурное программирование

Возникновение концепции структурного программирования [, , ] связывается с именем известного голландского ученого Э. Дейкстры – в 60-х годах прошлого века он сформулировал основные ее положения.

Принцип, на котором зиждется технология структурного программирования – фундаментальная научная и техническая идея о выделении множества базисных элементов, с помощью которых можно выразить (из которых можно собрать) любой объект из некоторого широкого набора.

Итак, основной принцип технологии структурного программирования гласит: для любой простой программы можно построить функционально эквивалентную ей структурную программу, т.е. программу, сформированную на основе фиксированного базисного множества, включающего структуру последовательного действия, структуру выбора одного из двух действий и структуру цикла, то есть многократного повторения некоторого действия с проверкой условия остановки повторения.

width:2px; height:1px; z-index:0;

width:2px; height:1px; z-index:0;

Рис. 1.3. Блок-схемы базисных алгоритмических конструкций

На рисунке 1.3 представлено изображение указанных алгоритмических конструкций в виде блок-схем. Здесь прямоугольник обозначает обобщенное действие, ромб – проверку условия, стрелки – переход от одного действия к другому.

Под простой программой в данном случае понимается программа, имеющая ровно один вход и один выход по управлению, такая, что через все ее функциональные блоки проходит путь от входа до выхода.

Изложенный принцип представляет собой теорему о структурировании. Ее точная формулировка и доказательство не входят в нашу задачу, желающие могут обратиться к монографии [6].

Базисные алгоритмические конструкции обладают важным свойством – они в точности удовлетворяют определению простой программы, то есть имеют один вход и один выход, что обеспечивает возможность осуществлять их суперпозицию. Любая из трех структур может быть подставлена в остальные или в саму себя.

Таким образом, центральный технологический принцип структурного программирования состоит в том, что формулировку алгоритма и его запись в виде программы рекомендуется выполнять на основе базиса из трех алгоритмических конструкций, применяя при необходимости их суперпозицию. Результатом последовательного применения этого принципа будет более ясная структура программы (в особенности, если использовать выделение структурных уровней с помощью отступов), что, несомненно, облегчит поиск в ней ошибок и упростит ее модификацию.

Модульное программирование

Технология модульного программирования – оформившаяся в начале 70-х годов XX века идея разработки больших программных систем []. Это фундаментальная концепция, являющаяся основой всех современных подходов к проектированию и реализации. В то же время суть ее проста и отражает широко известные научные и технические методы, заключающиеся в поиске и реализации некоторого базового набора элементов, комбинации которых дают решение всех задач из определенного круга.

Если концепция структурного программирования предлагает некоторый универсальный алгоритмический базис, то модульное программирование состоит в разработке под конкретную задачу или круг задач (предметную область) собственного базиса в виде набора модулей, позволяющего наиболее эффективно по целому ряду критериев построить программный комплекс. Модули, входящие в базис, это целые программы (в отличие от примитивов структурного программирования), решающие некоторые подзадачи основных задач.

С применением модульного программирования появляются возможности коллективной разработки программ как набора «независимых» частей, последовательного уменьшения сложности методом разбиения сложной задачи на более простые подзадачи, наконец, возможности повторного использования созданного ранее кода.

Объектно-ориентированное программирование

Развитие аппаратной базы привело к возможности решения при ее помощи все более или более сложных задач, а, значит, разработки все более и более сложных программ. Программы стали большими (даже очень большими), а разработка – коллективной. Объем работы увеличился, коллективы разрослись, код «разбух», и появились новые проблемы, которых не было раньше. Неожиданно выяснилось, что возможности структурного и модульного программирования ограничены и зачастую уже не позволяют добиваться желаемого результата (либо ничего не работает, либо проект не укладывается в сроки, либо в бюджет, либо через год после написания программы выясняется, что ее невозможно модифицировать и т.д.).

Объектно-ориентированная технология в некоторой степени решила большинство описанных проблем. В отличие от рассмотренных ранее технологий, объектно-ориентированная технология работает на стадиях анализа, проектирования и программирования. В основе технологии лежат объектная модель и объектная декомпозиция [].

К основные принципам объектной модели часто относят следующие:

 абстракция;

 инкапсуляция;

 иерархия (наследование, агрегация);

 полиморфизм;

 модульность.

Суть объектной декомпозиции состоит в выделении в предметной области классов и объектов, а также связей между ними, и лишь потом данных и алгоритмов, которыми характеризуется каждый класс. Таким образом, именно классы становятся основным «строительным блоком» в ООП, тогда как ранее таковыми блоками являлись алгоритмы.

Компонентное программирование

Компонентное программирование – представляет собой развитие объектно-ориентированной технологии. В отличие от ООП введен следующий уровень абстракции – классы объединяются в компоненты.

Компонент:

 программный код в виде самостоятельного модуля;

 может быть использован в неизменном виде;

 может допускать настройку;

 обладает поведением (функциональностью).

Основной принцип компонентного программирования: сборка приложения из готовых компонент, в общем случае написанных на разных языках.

Компонент изолирован от внешнего мира своим интерфейсом – набором методов (их сигнатурами).

Компонентная программа – набор независимых компонент, связанных друг с другом посредством интерфейсов.

Литература

1. С.И. Ожегов. Словарь русского языка. - М.: Советская энциклопедия, 1975.

2. И. Соммервиль. Инженерия программного обеспечения, 6 изд. – И.д. "Вильямс", 2002.

3. Г. Буч. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. Второе издание. – Бином, 1998.

4. N. Wirth. Program Development by Stepwise Refinement // Communications of the ACM vol.26(1).– 1971, 1983.

5. O. Dahl, E. Dijkstra, C.A.R. Hoare. Structured Programming.–London, England: Academic Press, 1972.

6. Р. Лингер, Х. Миллс, Б. Уитт. Теория и практика структурного программирования. – М.: Мир, 1982.

7. Э. Салливан. Время – деньги. – М.:Microsoft Press, Русская редакция, 2002.

"

[1](#ftnref2)  Источник: Светлана Шляхтина, Компьютер Пресс, 27 января 2006г.

Данные взяты с <http://www.aplana.ru/news/fullpublication.asp?id=139&page=1&year=2006>