Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Курсовая работа «Разработка прикладного программного обеспечения»

Руководство и методические указания для студентов направления 09.03.02 «Информационные системы и технологии».

Иркутск 2021г.

УДК 004.4'232: 004.4'234

Курсовая работа «Разработка прикладного программного обеспечения» Руководство и методические указания для студентов направления: 09.03.02 «Информационные системы и технологии». Составитель Бахвалова З.А. – Иркутск, Изд-во ИРНИТУ, 2018, 61 с.

Руководство и методические указания для студентов составлены в соответствии с Федеральным государственным образовательным стандартом высшего профессионального образования: по направлению **09.03.02 Информационные системы и технологии** программа бакалавриата **Информационные системы и технологии в административном управлении**

Руководство включает цели и задачи выполнения курсовой работы, порядок проведения и методические рекомендации по выполнению работы. Кроме того, в рамках данных указаний приведены требования по содержанию и оформлению отчета по курсовой работе.

Рецензент
доктор техн. наук, профессор кафедры информационных систем ИрГУПС
Л.В. Аршинский

© Иркутский
национальный исследовательский
технический университет, 2018

ОГЛАВЛЕНИЕ

Введение

Настоящие методические указания к выполнению курсовой работы по дисциплине «Технология разработки программных комплексов» написаны в соответствии с рабочей программой дисциплины.

Во время выполнения курсовой работы отрабатывается технологический процесс разработки программного продукта: разработка технического задания, внешнее и внутреннее проектирование, кодирование, тестирование, разработка и оформление документации.

При выполнении курсовой работы студент должен не только показать степень и глубину усвоения теоретического и практического материала, но проявить способность и умение самостоятельно и творчески решать конкретные инженерно–технические задачи по разработке современного программного обеспечения информационных систем.

Курсовая работа студента может стать составной частью (разделом, главой) выпускной квалификационной работы по данной специальности.

Цель работы – проектирование и программная реализация отказоустойчивого, дружественного к пользователю программного продукта в форме многопользовательского приложения.

Задачей курсовой работы является самостоятельное выполнение студентом этапов анализа, проектирования и разработки программного продукта. Студент при этом должен показать свой уровень подготовки, умение выбрать и обосновать решение стоящих перед ним проблем, навыки работы с технической и справочной литературой, умение применять вычислительную технику в своей деятельности.

Разработка программного продукта осуществляется на основе каскадной модели создания программного обеспечения.

Необходимые для выполнения курсовой работы знания

- Общие знания в области управления проектами и этапах разработки современного ПО.
- Знания в области моделирования программного обеспечения на языке UML;
- Владение языком Java, либо любым другим объектным языком программирования;
- Представление об автоматизированном тестировании приложении;

В процессе выполнения курсовой работы студенты должны:

– выполнить анализ задания;

– разработать развернутое техническое задание на программный продукт;

– выбрать технологию проектирования и разработать проект программного продукта (логическую и физическую модели с учетом динамических аспектов);

– выбрать структуры данных для реализации предметной области программного продукта;

– разработать интерфейс пользователя;

– выбрать стратегию тестирования и разработать тесты;

– выбрать язык и среду программирования;

– разработать алгоритмы и реализовать их в выбранной среде разработки;

– выполнить тестирование и отладку;

– разработать необходимую документацию, указанную в техническом задании.

Студент является единоличным автором курсовой работы и несет полную ответственность за принятые в курсовой работе решения, за правильность всех вычислений, за качество выполнения и оформления, а также за предоставление курсовой работы к установленному сроку для защиты.

Жизненный цикл программного обеспечения

Жизненный цикл программного обеспечения (ПО) — период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации.

Стандарт ГОСТ 34.601-90 предусматривает следующие стадии и этапы создания информационной системы (ИС):

1) Формирование требований к ИС
   - Обследование объекта и обоснование необходимости создания ИС
   - Формирование требований пользователя к ИС
   - Оформление отчета о выполнении работ и заявки на разработку ИС
2) Разработка технического задания
   - Изучение объекта
   - Проведение необходимых научно-исследовательских работ
   - Разработка вариантов концепции ИС и выбор варианта концепции ИС, удовлетворяющего требованиям пользователей
   - Оформление отчета о проделанной работе
   - Разработка и утверждение технического задания на создание ИС
3) Эскизный проект
   - Разработка предварительных проектных решений по системе и её частям
   - Разработка документации на ИС и её части
4) Техническое проектирование
   - Разработка проектных решений по системе и её частям
   - Разработка документации на ИС и её части
   - Разработка и оформление документации на поставку комплектующих изделий
   - Разработка заданий на проектирование в смежных частях проекта
5) Рабочее проектирование
   - Разработка рабочей документации на ИС и её части
   - Разработка и адаптация программ
6) Ввод в действие
   - Подготовка объекта автоматизации
   - Подготовка персонала
   - Комплектация ИС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями)
   - Строительно-монтажные работы
   - Пусконаладочные работы
   - Проведение предварительных испытаний
   - Проведение опытной эксплуатации

- Проведение приёмочных испытаний

7) Сопровождение ИС.
   - Выполнение работ в соответствии с гарантийными обязательствами
   - Послегарантийное обслуживание

В целях изучения взаимосвязанных приемов и методов канонического проектирования ЭИС перечисленные 7 стадий можно сгруппировать в часто используемые на практике четыре стадии процесса разработки ИС:

1. предпроектную,
2. проектирования,
3. внедрения,
4. эксплуатации и внедрения.

Предпроектная стадия.

Традиционно этапы исследования предметной области – предприятия, обоснование проекта ИС для него и разработки технического задания объединяют термином «Предпроектная стадия» («Предпроектное обследование»), поскольку результаты выполнения работ на данных этапах не являются законченным проектным решением. Основное назначение «Предпроектной стадии» заключается в обосновании экономической целесообразности создания ЭИС и формулировании требований к ней.

При изучении существующей системы разработчики должны уточнить границы изучения системы, определить круг пользователей будущей ЭИС различных уровней и выделить классы и типы объектов, подлежащих обследованию и последующей автоматизации.

Основной единицей обработки данных является задача. Поэтому функциональная структура проблемной области на стадии предпроектного обследования изучается в разрезе решаемых задач и комплексов задач. При этом задача в содержательном аспекте рассматривается как совокупность операций преобразования некоторого набора исходных данных для получения результатной информации, необходимой для выполнения функции управления или принятия управленческого решения. В большинстве случаев исходные данные и результаты их преобразований представляются в форме экономических документов. Поэтому к числу объектов обследования относятся компоненты потоков информации (документы, показатели, файлы, сообщения).

На первой, ***предпроектной стадии*** принято выделять три типа деятельности

- Сбор требований: общение с клиентами и пользователями, чтобы определить, каковы их требования.

- Анализ требований: определение, являются ли собранные требования неясными, неполными, неоднозначными, или противоречащими, и затем решение этих проблем.
- Документирование требований: Требования могут быть задокументированы в различных формах, таких как простое описание, ***сценарии использования, пользовательские истории, или спецификации процессов***.

Сценарии использования или же **пользовательский сценарий** (англ. Use Case) — в разработке ПО это описание поведения системы, которым она отвечает на внешние запросы. Другими словами, сценарий использования описывает, «кто» и «что» может сделать с рассматриваемой системой. Методика сценариев использования применяется для выявления требований к поведению системы, известных также как функциональные требования.

Пользовательские истории — быстрый способ документировать требования клиента, без необходимости разрабатывать обширные формализованные документы и впоследствии тратить ресурсы на их поддержание. Цель пользовательских историй состоит в том, чтобы быть в состоянии быстро и без накладных затрат реагировать на быстро изменяющиеся требования реального мира.

Спецификация требований программного обеспечения (англ. *Software Requirements Specification, SRS*) является полным описанием поведения системы, которая будет создана. Она включает ряд сценариев использования, которые описывают все виды взаимодействия пользователей с программным обеспечением. Сценарии использования также известны как функциональные требования.

В результате выполнения первого этапа проектировщики получают материалы обследования предметной области, которые должны содержать полную и достоверную информацию, описывающую изучаемую предметную область (предприятие), в том числе:

- цель функционирования системы. Под ***целью*** подразумевается получение определенных значений экономического эффекта в сфере управления какими-либо процессами системы или снижение стоимостных и трудовых затрат на обработку информации, улучшение качества и достоверности получаемой информации, повышение оперативности ее обработки и т.д., то есть получение косвенного и прямого эффекта от внедрения данной задачи;
- организационную структуру системы и объекта управления, т.е. его управленческие отделы, цехи, склады и хозяйственные службы;
- функции управления, выполняемые в этих подразделениях;

- протекающие в них технологические процессы обработки управленческой и экономической информации;
- материальные потоки и процессы их обработки, ресурсные ограничения.

После выполнения третьего этапа проектировщики получают количественные и качественные характеристики информационных потоков, описание их структуры и мест обработки, объемов выполняемых операций и трудоемкости их обработки. На основе этих материалов разрабатываются два документа: «Технико-экономическое обоснование проектных решений», содержащее расчеты и обоснование необходимости разработки ИС для предприятия и выбираемых технологических и проектных решений, и «Техническое задание», в состав которого входят требования к создаваемой системе и ее отдельным компонентам (программному, техническому и информационному обеспечению) и целевая установка на проектирование новой системы

Проектирование.

Эскизный, технический проекты и рабочая документация — это последовательное построение все более точных проектных решений. Допускается исключать стадию «Эскизный проект» и отдельные этапы работ на всех стадиях, объединять стадии «Технический проект» и «Рабочая документация» в «Технорабочий проект», параллельно выполнять различные этапы и работы, включать дополнительные.

На стадии технорабочего проектирования выполняются два этапа работ: **техническое и рабочее проектирование.**

На первом из них – «**Техническое проектирование**» осуществляется логическая проработка функциональной и системной архитектуры ЭИС, в процессе которой строится несколько вариантов всех компонентов системы; проводится оценка вариантов по показателям: стоимости, трудоемкости, достоверности получаемых результатов, и составляется «Технический проект» системы.

Все работы этапа технического проектирования можно разбить на две группы.

К первой группе относится разработка общесистемных проектных решений, в том числе:

- разработка общесистемных положений по ИС (уточняются цели создания ИС и выполняемые ею функции; устанавливается ее взаимосвязь с другими системами и формируется документ «Основные положения»);
- изменение организационной структуры (уточняется и изменяется организационная структура и получается описание организационной структуры;
- определение функциональной структуры (наиболее принципиальной в данном комплексе работ является разработка функциональной архитек-

туры ИС на базе принципов выделения функциональных подсистем (модулей, контуров): предметного, функционального, смешанного (предметно-функционального) и проблемного);
- разработка проектно-сметной документации и расчет экономической эффективности системы;
- разработка плана мероприятий по внедрению ИС.

Ко второй группе работ, выполняемых на этапе технического проектирования, относятся разработки локальных проектных решений, к числу которых относят следующие операции:
- разработка постановки задачи для задач, входящих в состав каждой функциональной подсистемы, включающей основные компоненты описания задачи и служащей основанием для разработки проектных решений по задаче;
- проектирование форм входных и выходных документов, системы ведения документов и макетов экранных форм документов;
- проектирование классификаторов экономической информации и системы ведения классификаторов;
- разработка структуры входных и выходных сообщений;
- проектирование состава и структур файлов информационной базы;
- проектирование внемашинной и внутримашинной технологии решения каждой задачи;
- уточнение состава технических средств.

На втором этапе – **рабочем проектировании** осуществляется техническая реализация выбранных наилучших вариантов и разрабатывается документация «Рабочий проект». Наиболее ответственной работой, выполняемой на этом этапе, являются ***кодирование и составление программной документации***. В ее состав входят следующие компоненты :
- описание программ;
- спецификация программ;
- тексты программ;
- контрольные примеры;
- инструкции для системного программиста, оператора и пользователя.

К числу работ, выполняемых на этапе рабочего поектирования, относится ***разработка технологической документации, правовых инструкций***, определяющих права и обязанности специалистов, работающих в условиях функционирования компонентов ИС на предприятии.

Заключительная операция рабочего проектирования – ***оформление документации рабочего проекта*** согласно требованиям стандартов

Этап «Рабочее проектирование» связан с физической реализацией выбранного варианта проекта и получением документации «Рабочего проекта».

Модель №4+1»

Архитектуру программного обеспечения можно проиллюстрировать с помощью нескольких **архитектурных представлений**. Каждое архитектурное представление связано с некоторым определенным набором вопросов, интересующих участников разработки: пользователей, проектировщиков, менеджеров, технических специалистов, обслуживающий персонал и так далее.

Архитектурные представления охватывают основные решения, принятые при выборе структуры программного обеспечения, и демонстрируют декомпозицию архитектуры на составляющие ее компоненты, **соединители** и **формы**. Решения, принимаемые при выборе структуры, обусловлены функциональными и дополнительными требованиями, а также другими ограничениями. В свою очередь, эти решения порождают новые **ограничения** в отношении требований и последующих решений на более низких уровнях.

В RUP отправной точкой при разработке архитектуры служит типичный набор архитектурных представлений, который называется "моделью 4+1".

Модель 4+1 была разработана Филипом Крухтеном для описания архитектуры преимущественно-программных систем, основанном на использовании нескольких взаимодополняющих представлений. Представления используются для описания системы с точки зрения разных заинтересованных лиц, таких как конечные пользователи, разработчики или руководители проекта. Четыре представления – это логическая структура, разработка, процессы и физическая структура системы. Дополнительно вводятся варианты использования, или сценарии, которые иллюстрируют описываемую архитектуру. Поэтому говорят, что модель состоит из 4+1 представления.

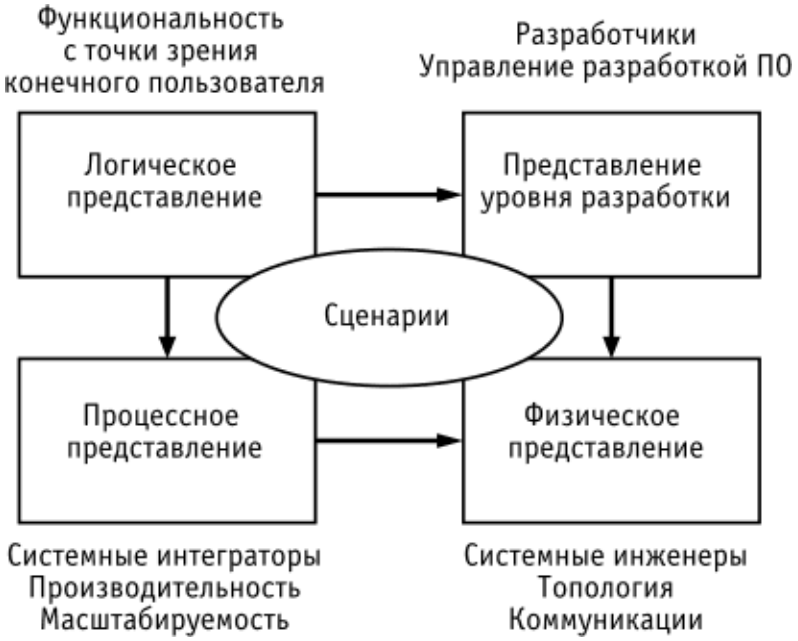


Рисунок 1 - Модель представления архитектуры "4+1"

Модель содержит следующие компоненты:

- Представление вариантов использования (сценарии), в состав которого входят сценарии и варианты использования, описывающие значимые для архитектуры технические риски, классы и поведение системы. Это подмножество модели вариантов использования. Описание архитектуры системы должно быть проиллюстрировано небольшим набором вариантов использования, или сценариев. Сценарии должны описывать взаимодействие между объектами и между процессами. Сценарии используются для идентификации архитектурных элементов и для иллюстрирования и проверки качества архитектуры системы. Они также служат отправной точкой для тестирования архитектурного прототипа системы. Сценарии описываются UML-диаграммами вариантов использования. *Представление пользователя позволяет увидеть, как система отражает предметную область и вписывается в повседневную работу непосредственных пользователей системы;*
- Логическое представление содержит важнейшие классы проекта, распределенные по пакетам и подсистемам, которые, в свою очередь, распределены по слоям. Кроме того, это представление содержит некоторые реализации вариантов использования. Данное представление представляет собой подмножество модели проекта. Логическое представление сфокусировано на функциональности, предоставляемой системой для конечных пользователей. В этом представлении используются UML-диаграммы классов, связей и последовательностей. *Представление разработки (или сопровождения) системы показывает, как система должна быть реализована (или уже была реализована);*
- Представление реализации содержит общие сведения о модели реализации и ее структуре с точки зрения модулей, пакетов и слоев. В это представление также входит информация о распределении пакетов и классов логического представления по пакетам и модулям представления реализации. Это подмножество модели реализации. Представление разработки показывает систему с точки зрения разработчика и касается управления программой. Это представление также известно как представление реализации. Здесь используются UML-диаграммы компонентов и пакетов для описания компонентов системы и их объединения в логические пакеты (например, слои).
- Представление развертывания содержит описания физических узлов наиболее распространенных конфигураций платформ и информацию о распределении задач (из представления процессов) между физическими узлами. Это представление применяется только с распределенными системами. Оно представляет собой подмножество модели развертывания. Представление физической структуры показывает систему с точки зрения системного инженера. Она показывает распределение программных компонентов по физическим уровням и физические каналы связи между уровнями. Это представление известно также как представление развёртывания системы. Представление **физической структуры** системы использует

UML-диаграмму развёртывания. *Представление развёртывания даёт информацию о конфигурации системы в её окружении в процессе эксплуатации.*

- Представление процессов содержит описание задач (процессов и нитей), их взаимодействия и конфигурации, а также взаимосвязи между классами и объектами проекта и задачами. Это представление применяется только в системах, обладающих значительным параллелизмом. В RUP это подмножество модели проекта. Представление процесса отражает динамические аспекты системы, показывает происходящие в системе процессы и связи между ними. Представление процесса фокусируется на поведении системы во время выполнения программы. Представление процесса отражает параллелизм выполнения, распределение, интеграцию, производительность, масштабирование и т.п. Представление процесса использует UML-диаграммы активности. *Представление бизнес-процессов позволяет увидеть, как система вписывается в процессы деятельности организации и автоматизирует их;*

Внедрение проекта

Третья стадия «***Внедрение проекта***» включает в себя три этапа: подготовку объекта к внедрению проекта; опытное внедрение проекта и сдачу его в промышленную эксплуатацию.

На этапе «***Подготовка объекта к внедрению проекта***» осуществляется комплекс работ по подготовке предприятия к внедрению разработанного проекта ЭИС. На этапе «***Опытное внедрение***» осуществляют проверку правильности работы некоторых частей проекта и получают исправленную проектную документацию и «Акт о проведении опытного внедрения». На этапе «***Сдача проекта в промышленную эксплуатацию***» осуществляют комплексную системную проверку всех частей проекта, в результате которой получают доработанный технорабочий проект и акт приемки проекта в промышленную эксплуатацию.

Эксплуатация и сопровождение проекта

Четвертая стадия – «***Эксплуатация и сопровождение проекта***» включает этапы: эксплуатацию проекта; ***сопровождение и модернизацию проекта.***

На этапе «***Эксплуатация проекта***» получают информацию о работе всей системы в целом и отдельных ее компонентов и собирают статистику о сбоях системы в виде рекламаций и замечаний, которые накапливаются для выполнения следующего этапа. На этапе «***Сопровождение проекта***» выполняются два вида работ: ликвидируются последствия сбоев в работе системы и исправляются ошибки, не выявленные при внедрении проекта, а также осуществляется ***модернизация проекта.*** В процессе модернизации проект либо дорабатывается, т.е. расширяется по составу подсистем и задач, либо производится перенос системы на другую программную или техническую плат-

форму с целью адаптации ее к изменяющимся внешним и внутренним условиям функционирования, в результате чего получают документы модернизированного техно-рабочего проекта .

Понятие требования

Л.Новиков в русской редакции нотации RUP приводит следующее определение: "Требование - это условие или возможность, которой должна соответствовать система".

В IEEE Standard Glossary of Software Engineering Terminology (1990) данное понятие трактуется шире. Требование - это:

1. условия или возможности, необходимые пользователю для решения проблем или достижения целей;
2. условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам;
3. документированное представление условий или возможностей для пунктов 1 и 2

Введем еще одно определение. **Требования** - это исходные данные, на основании которых проектируются и создаются автоматизированные информационные системы.

Первичные данные поступают из различных источников, характеризуются противоречивостью, неполнотой, нечеткостью, изменчивостью. Требования нужны в частности для того, чтобы Разработчик мог определить и согласовать с Заказчиком временные и финансовые перспективы проекта автоматизации. Поэтому значительная часть требований должна быть собрана и обработана на ранних этапах создания АИС.

Требование должно обладать следующими характеристиками:

1. Единичность — требование описывает одну и только одну вещь.
2. Завершенность — требование полностью определено в одном месте и вся необходимая информация присутствует.
3. Последовательность — требование не противоречит другим требованиям и полностью соответствует документации.
4. Атомарность — требование нельзя разделить на более мелкие.
5. Отслеживаемость — требование полностью или частично соответствует деловым нуждам как заявлено заинтересованными лицами и задокументировано.
6. Актуальность — требование не стало устаревшим с течением времени.
7. Выполнимость — требование может быть реализовано в рамках проекта.
8. Недвусмысленность — требование определено без обращения к техническому жаргону, акронимам и другим скрытым формулировкам. Оно выражает объекты и факты, а не субъективные мнения. Возможна одна и только одна его интерпретация. Определение не содержит нечетких фраз, использование отрицательных и составных утверждений запрещено.

9. Обязательность — требование представляет собой определенную заинтересованным лицом характеристику, отсутствие которой ведет к неполноценности решения, которая не может быть проигнорирована. Необязательное требование — противоречие самому понятия требования.
10. Проверяемость — реализованность требования может быть проверена.

Уровни требований

Требования разделяются по уровням. Уровни требований связаны, с одной стороны, с уровнем абстракции системы, с другой - с уровнем управления на предприятии.

Обычно выделяют три уровня требований.

1. На верхнем уровне представлены так называемые **бизнес-требования.** Примеры бизнес-требования: система должна сократить срок оборачиваемости обрабатываемых на предприятии заказов в три раза. Бизнес-требования обычно формулируются топ-менеджерами, либо акционерами предприятия.

2. Следующий уровень - **уровень требований пользователей**. Пример требования пользователя: система должна представлять диалоговые средства для ввода исчерпывающей информации о заказе, последующей фиксации информации в хранилище (базе данных) и маршрутизации информации о заказе к сотруднику, отвечающему за его планирование и исполнение. Требования пользователей часто бывают плохо структурированными, дублирующимися, противоречивыми. Поэтому для создания системы важен третий уровень, в котором осуществляется формализация требований.

3. Третий уровень - **функциональный**. Пример функциональных требований (или просто функций) по работе с электронным заказом: заказ может быть создан, отредактирован, удален и перемещен с участка на участок.

Функциональные требования регламентируют функционирование или поведение системы (behavioral requirements). Функциональные требования отвечают на вопрос "что должна делать система" в тех или иных ситуациях. Функциональные требования определяют основной "фронт работ" Разработчика, и устанавливают цели, задачи и сервисы, предоставляемые системой Заказчику.

Функциональные требования записываются, как правило, при посредстве предписывающих правил: "система должна позволять кладовщику формировать приходные и расходные накладные". Другим способом являются так называемые варианты использования (uses cases) - популярный и весьма продуктивный способ представления требований.

Нефункциональные требования, соответственно, регламентируют внутренние и внешние условия или атрибуты функционирования системы. К. Вигерс выделяет следующие основные группы нефункциональных требований:

- Внешние интерфейсы,

- Атрибуты качества,
- Ограничения.

Среди **внешних интерфейсов** в большинстве современных ИС наиболее важным является интерфейс пользователя. Кроме того, выделяются интерфейсы с внешними устройствами (аппаратные интерфейсы), программные интерфейсы и интерфейсы передачи информации (коммуникационные интерфейсы).

Основные **атрибуты качества**:

- Применимость,
- Надежность,
- Производительность,
- Эксплуатационная пригодность.

Ограничения - формулировки условий, модифицирующих требования или наборы требований, сужая выбор возможных решений по их реализации. выбор платформы реализации и/или развертывания (протоколы, серверы приложений, баз данных, ...), которые, в свою очередь, могут относиться, например, к внешним интерфейсам (конец цитаты).

Свойства требований

Ф. Брукс в своем, теперь уже ставшим классическим, эссе, следующим образом охарактеризовал роль требований в разработке программного обеспечения.

Строжайшее и единственное правило построения систем программного обеспечения (ПО) - решить точно, что же строить. Никакая другая часть концептуальной работы не является такой трудной, как выяснение деталей технических требований, в том числе и взаимодействие с людьми, с механизмами и с иными системами ПО. Никакая другая часть работы так не портит результат, если она выполнена плохо. Ошибки никакого другого этапа работы не исправляются так трудно.

Наука извлечения и формализации качественных (иногда говорят "хороших", "правильных") требований носит во многом эмпирический характер. Однако, в практике разработки программных систем накопились определенные представления о том, какими свойствами должны обладать требования к программной системе. Это:

Полнота - это скорее тенденция, цель, к которой нужно постараться максимально приблизиться на как можно более ранних стадиях проекта.

Ясность (недвусмысленность, определенность, однозначность спецификаций).

Требование обладает свойством ясности, если оно сходным образом воспринимается всеми совладельцами системы. К.Вигерс дает следующий совет по повышению ясности документов: "Пишите документацию просто, кратко и точно, применяя лексику, понятную пользователям".

Корректность и согласованность (непротиворечивость). К. Вигерс вводит понятие корректности требования через точность описания функциональности. Кроме того, можно рассуждать о взаимной корректности требований или согласованности (непротиворечивости): если два требования вступают в конфликт, значит - как минимум одно из них некорректно. Так, требования пользователей не должны противоречить бизнес-требованиям, а функциональные требования - требованиям пользователя.

Верифицируемость (пригодность к проверке). Если требование изложено на языке, понятном и одинаково воспринимаемым участниками процесса создания информационной системы, причем оно является полным, т.е. ни одна из важных для реализации деталей не упущена - значит, это требование можно проверить.

Необходимость и полезность при эксплуатации. Одни из самых субъективных и трудно проверяемых свойств требований. Необходимыми следует считать свойства, без выполнения которых невозможно, либо затруднено выполнение автоматизированных бизнес-функций пользователей; полезными при эксплуатации следует считать любые свойства, повышающие эргономические качества продукта.

Осуществимость (выполнимость). Выполнимость требования на практике определяется разумным балансом между ценностью (степенью необходимости и полезности) и потребными ресурсами.

Трассируемость. Определяется возможностью отследить связь между ним и другими артефактами информационной системы (документами, моделями, текстами программ и пр.). Процесс трассировки позволяет, с одной стороны, выявить уже на стадии проектирования системы проектные артефакты, к которым не ведет связь ни от одного из артефактов, описывающих требования, с другой - артефакты, описывающие требования, не связанные с проектными артефактами.

Упорядоченность по важности и стабильности. Приоритеты требований обычно назначает представитель Заказчика. Разработчик, отталкиваясь от приоритетности требований, управляет процессом реализации информационной системы.

Наличие количественной метрики. В первую очередь это относится к нефункциональным требованиям, которые, как правило, должны иметь под собой количественную основу (запрос должен отрабатываться не более, чем ___ секунд; средняя наработка на отказ должна составлять не менее, чем ___ часов). Функциональные требования также могут расширяться количественными мерами при помощи так называемых аспектов применимости

Пример требований к программному продукту

Требования к модулю управления голосованиями веб-сайта ИрНИТУ.

Требования к системе в целом

В системе должно поддерживаться два типа голосований:

- с одиночным выбором;
- с множественным выбором;

У голосований должен быть ограниченный срок, когда посетители сайта могут оставить свой голос. Срок голосования определяется администратором веб-сайта. Администратор должен иметь возможность продлить срок голосования как для голосования, которое идёт в настоящий момент, так и для уже закончившегося голосования. При этом если срок был продлён для голосования, которое уже закончилось, оно должно вновь стать активным и доступным для посетителей сайта;

Голосования создаются администратором ресурса с указанием:

- вопроса для голосования,
- продолжительности голосования,
- типа голосования,
- вариантов ответа,
- типа голосования: для всех посетителей, для зарегистрированных пользователей.
- раздел сайта, к которому относится голосование. (Необязательный параметр).

При создании ответа администратор указывает сам ответ и начальное количество голосов. Если количество голосов не указано, то оно считается равным 0.

Допускается редактирование текста вопроса голосования и вариантов ответа администратором сайта при условии, что по нему не было получено ни одного ответа посетителей.

При создании нового голосования должна осуществляться проверка существования голосования, даты проведения которого частично или полностью совпадают с новым. При возникновении конфликта подобного рода должно появляться сообщение об ошибке, позволяющее:

- удалить уже существующее голосование,
- изменить даты уже существующего голосования,
- изменить параметры создаваемого голосования,
- отказаться от создания нового голосования.

Допускается удаление голосования администратором сайта в любой момент времени.

Должен существовать архив голосований, доступный для просмотра посетителями ресурса, где можно посмотреть результаты прошедших ранее голосований.

Посетитель сайта может проголосовать при каждом посещении лишь один раз. После голосования ему отображаются текущие результаты. Для контроля того, что посетитель сайта отдал свой голос должно использоваться три механизма:

- Cookie;
- Сессия;
- Параметр о голосовании для зарегистрированных пользователей.

В случае если голосование является голосованием для зарегистрированных пользователей, то при входе на страницу голосования незарегистрированный пользователь должен быть перенаправлен на страницу авторизации.

Модуль голосования должен поддерживать возможность использования в качестве вопроса голосования и вариантов ответа HTML-фрагментов, содержащих изображения, стили, скрипты и текст со сложным форматированием.

Язык работы модуля – русский. Должна быть предусмотрена возможность перевода модуля на другие языки.

Одновременно в разных разделах сайта может проходить несколько голосований, при этом активное голосование для текущей страницы должно выбираться в следующем порядке:

- голосование для текущего раздела;
- ближайшее голосование вышестоящего раздела;
- голосование, для которого не указан раздел.

В случае, когда возникает конфликт при выборе голосования сообщение об этом должно быть предоставлено администратору. При этом ни одно из конфликтующих голосований отображаться не должно.

Требования к управлению модулем и настройке.

1. Первичная установка и настройка модуля производится разработчиком на платформе заказчика самостоятельно.
2. После установки и настройки модуля разработчик проводит обучения представителей заказчика работе с модулем, включая как способы создания, изменения и удаления голосования, так и способы ки конфигурационных параметров модуля.

Требования к эргономике

1. Интерфейс модуля голосования должен соответствовать общему оформлению официального веб-сайта Иркутского государственного технического университета.
2. Экранные формы модуля голосования должны быть согласованы с Заказчиком отдельным соглашением.
3. Административный интерфейс модуля голосования должен позволять квалифицированному администратору систему, прошедшему обучения работе с модулем, создавать новое голосование, содержащее вопрос с 1

изображением и 10 вариантов ответа, каждый из которых содержит по 1 изображению, за время не более 10 минут. При условии, что текст вопроса голосования и текст вариантов ответа были подготовлены заранее.

4. Модуль голосования должен позволять корректно отображать голосования с большим числом вариантов ответа. Для этого должен быть предусмотрен постраничный вывод вариантов ответа. Число вариантов ответа на одной странице должно настраиваться в параметрах модуля.
5. Административный интерфейс модуля должен быть удобен для использования.

Требования к отказоустойчивости

1. Модуль должен обладать высокой отказоустойчивостью.
2. Допустимое число ошибок, возникающих в административном интерфейсе при создании голосования – 1 на 50 создаваемых голосований.
3. Ошибок при ответе пользователя на вопрос голосования возникать не должно.
4. Должна быть предусмотрена возможность создания голосования в течение длительного времени без потери сессии.

Типовые случаи применения модуля

1. Проведение голосования в рамках конкурса «Мисс ИрГТУ»;
2. Проведение конкурса студенческих фоторабот.

Граничные условия эксплуатации

1. Максимальное число голосований в системе – 30000;
2. Максимальное число одновременно идущих голосований – 50;
3. Максимальный размер вопроса и ответа в голосовании – 8000 символов без учёта изображений;
4. Максимальное число ответов в одном голосовании – 500.
5. Максимальное число проголосовавших в одном голосовании – 300000.
6. Максимальная продолжительность голосования - 10 лет.

Тестирование

Очень часто при разработке программного обеспечения приходится сталкиваться с одной из двух проблем. Либо качество разработанного продукта много ниже самых минимальных разумных требований, либо затраты на тестирование превосходят все разумные пределы. К сожалению, бывает и так, что обе проблемы существуют одновременно. И денег на тестирование истрачено много, а качества достичь так и не удалось.

Одной из причин такой ситуации является объективная сложность процесса тестирования ПО. Ведь под словом Тестирование может скрываться множество самых различных действий, направленных на решение множества разнообразных задач. Тут и запуск и исполнение программы с целью проверки отсутствия ошибок, и оценка производительности, и контроль наличия и полноты документации и даже качества принятых проектных решений.

В соответствие с RUP Тестирование - одна из дисциплин RUP. Она ориентирована в первую очередь на оценку качества с помощью следующих методов:

- поиск и документирование дефектов качества;
- общие рекомендации относительно качества;
- проверка выполнения основных предположений и требований на конкретных примерах;
- проверка, что продукт функционирует так, как было запроектировано;
- проверка, что требования выполнены соответствующим образом.

В соответствие с IEEE Std 829-1983 Тестирование - это процесс анализа ПО, направленный на выявление отличий между его реально существующими и требуемыми свойствами (дефект) и на оценку свойств ПО.

Виды тестирования

Принято разделять тестирование по уровням задач и объектов на разных стадиях и этапах разработки ПО (см. таблицу1):

- тестирование частей ПО (модулей, компонентов) с целью проверки правильности реализации алгоритмов -- выполняется разработчиками;
- функциональное тестирование подсистем и ПО в целом с целью проверки степени выполнения функциональных требований к ПО -- рекомендуется проводить отдельной группой тестировщиков, не подчиненной руководителю разработки;
- нагрузочное тестирование (в том числе стрессовое) для выявления характеристик функционирования ПО при изменении нагрузки (интенсивности обращений к нему, наполнения базы данных и т. п.) -- для выполнения этой работы требуются высококвалифицированные тестировщики и дорогостоящие средства автоматизации экспериментов.

Этапы тестирования

Таблица 1 – Виды тестирования

Вид тестирования	Стадия, этап	Объект	Критерий
Структурное, надежности	Разработка	Компоненты	Покрытие ветвлений, функции
Сборочное	Разработка	Подсистемы	Функциональность, степень проверки компонентов
Функциональное	Разработка	Система в целом	Соответствие функциональным требованиям ТЗ
Регрессионное	Разработка, сопровождение	Система в целом	Проверка качества внесения изменений
Нагрузочное	Разработка, сопровождение	Система в целом	Оценка статистических характеристик системы, соответствие ТЗ, ТТХ, подбор конфигурации оборудования
Стрессовое	Разработка, сопровождение	Система в целом	Корректность работы системы при предельных нагрузках

Автоматизированное тестирование

Автоматическое тестирование является его составной частью. Оно использует программное обеспечение для проверки выполнения проводимых тестов, что помогает сократить время тестирования и упростить его процесс.

В наши дни создается большое количество инструментов с графическим интерфейсом, использование которых помогает облегчить работу программистов и повышает их производительность. Эти процессы увеличили требования к тестировщикам. Сейчас они должны обрабатывать большое количество информации за короткий срок. Потому использование автоматических тестов является необходимым условием для экономии средств и времени тестировщиков.

Современные средства разработки создают довольно сложные приложения, и их ручное тестирование является очень трудоемким процессом. Недостаток ручного тестирования также в том, что результаты выполнения тестов не сохраняются и их трудно повторить заново. Автоматические тесты позволяют упростить процесс ручного тестирования, сделать его наиболее удобным и точным.

Для автоматизации тестирования существует большое количество приложений. Наиболее популярные из них:

- JUnit и TestNG для Java
- DUnit для Delphi
- Selenium для веб-приложений
- Mercury LoadRunner
- Segue SilkPerformer
- Rational TestStudio

Использование этих инструментов помогает тестировщикам автоматизировать следующие задачи:

- установка продукта
- создание тестовых данных
- GUI взаимодействие
- определение проблемы

Однако автоматические тесты не могут полностью заменить ручное тестирование. Автоматизация всех испытаний очень дорогой процесс, и потому автоматическое тестирование является лишь частью ручного тестирования.

Почему полезно автоматизировать тестирование?

Test-Driven Development (TDD, разработка на основе автоматизированных тестов) – одна из наиболее популярных концепций, используемых в настоящее время при разработке приложений. Основа такой популярности TDD – широкое распространение Agile-методик разработки и значительный рост их популярности. Почему?

Во-первых, большинство Agile-техник разработки ПО требуют использования тестов для верификации качества программного кода.

Во-вторых, любое применение рефакторинга к программному коду требует наличия тестов для проверки его работоспособности. Чем более полным является набор тестов для приложения, тем более «агрессивный» рефакторинг допустимо применять к программному коду. Например, все современные среды разработки позволяют применять шаблон рефакторинга «переименование метода». Однако это не всегда безопасно. Если этот метод вызывается через java reflection, то среда разработки не сможет установить этот факт и изменить исходный код. Соответственно в данной ситуации может возникнуть ошибка, которая достаточно легко может быть найдена на уровне модульного тестирования.

В-третьих, написание тестов по своей сути является одной из моделей документирования исходного кода. В данном случае под документацией понимается набор исполняемых файлов, представляющих собой исполняемую спецификацию продукта. Тем самым получается, что внесение изменений в программный продукт начинается с изменения тестов, что приводит к переходу в красную стадию разработки, когда некоторые из тестов не проходят. После исправления исходного кода переходят к зелёной стадии, когда все те-

сты проходят успешно. А уже далее применяют рефакторинг для повышения качества и читаемости кода приложения.

В-четвёртых, модульное тестирование позволяет снизить общее время отладки приложения за счёт сокращения времени на локализацию и исправление ошибок. Особенно это важно при внесении изменений в функционирующую систему.

В-пятых, тесты служат отличной основой для формирования многих метрик по проекту. Так общее количество тестов характеризует объём реализуемого кода. Соотношение между числом public-методов и тестов характеризует степень покрытия тестами функционала системы. Совокупное время выполнения всех тестов может служить мерой эффективности написанного кода и.т.д.

Понятие модульного тестирования

Модульное тестирование - это тестирование программы на уровне отдельно взятых модулей, функций или классов. Цель модульного тестирования состоит в выявлении локализованных в модуле ошибок в реализации алгоритмов, а также в определении степени готовности системы к переходу на следующий уровень разработки и тестирования. Модульное тестирование проводится по принципу "белого ящика", то есть основывается на знании внутренней структуры программы, и часто включает те или иные методы анализа покрытия кода.

Модульное тестирование обычно подразумевает создание вокруг каждого модуля определенной среды, включающей заглушки для всех интерфейсов тестируемого модуля. Некоторые из них могут использоваться для подачи входных значений, другие для анализа результатов, присутствие третьих может быть продиктовано требованиями, накладываемыми компилятором и сборщиком.

На уровне модульного тестирования проще всего обнаружить дефекты, связанные с алгоритмическими ошибками и ошибками кодирования алгоритмов, типа работы с условиями и счетчиками циклов, а также с использованием локальных переменных и ресурсов. Ошибки, связанные с неверной трактовкой данных, некорректной реализацией интерфейсов, совместимостью, производительностью и т.п. обычно пропускаются на уровне модульного тестирования и выявляются на более поздних стадиях тестирования.

Именно эффективность обнаружения тех или иных типов дефектов должна определять стратегию модульного тестирования, то есть расстановку акцентов при определении набора входных значений. У организации, занимающейся разработкой программного обеспечения, как правило, имеется историческая база данных (Repository) разработок, хранящая конкретные сведения о разработке предыдущих проектов: о версиях и сборках кода (build) зафиксированных в процессе разработки продукта, о принятых решениях, допущенных просчетах, ошибках, успехах и т.п. Проведя анализ характеристик прежних проектов, подобных заказанному организации, можно предо-

хранить новую разработку от старых ошибок, например, определив типы дефектов, поиск которых наиболее эффективен на различных этапах тестирования.

В данном случае анализируется этап модульного тестирования. Если анализ не дал нужной информации, например, в случае проектов, в которых соответствующие данные не собирались, то основным правилом становится поиск локальных дефектов, у которых код, ресурсы и информация, вовлеченные в дефект, характерны именно для данного модуля. В этом случае на модульном уровне ошибки, связанные, например, с неверным порядком или форматом параметров модуля, могут быть пропущены, поскольку они вовлекают информацию, затрагивающую другие модули (а именно, спецификацию интерфейса), в то время как ошибки в алгоритме обработки параметров довольно легко обнаруживаются.

Являясь по способу исполнения структурным тестированием или тестированием "белого ящика", модульное тестирование характеризуется степенью, в которой тесты выполняют или покрывают логику программы (исходный текст).

Пример написания модульного теста на JUNit

Для демонстрации основных возможностей JUnit Framework, напишем примитивный класс на языке Java. Этот класс будет иметь два метода - нахождение факториала и суммы двух чисел:

```
 public class MathFunc {
  private int variable;
public MathFunc() {
  variable = 0;
}

public MathFunc(int var) {
  variable = var;
}
public int getVariable() {
  return variable;
}

public void setVariable(int variable) {
   this.variable = variable;
}
public long factorial() {
  long result = 1;
  if (variable > 1) {
    for (int i=1; i<=variable; i++)
      result = result*i;
  }
  return result;
}
public long plus(int var) {
```

```
    long result = variable + var;
    return result;
  }
}
```

Для написания тестового класса нам нужно создать наследника junit.framework.TestCase. Затем необходимо определить конструктор, принимающий в качестве параметра строку (String) и передающую ее родительскому классу. Наконец, напишем столько тестовых методов, сколько захотим:

```
  public class TestClass extends TestCase {
    public TestClass(String testName) {
      super(testName);
    }
    public void testFactorialNull() {
      MathFunc math = new MathFunc();
      assertTrue(math.factorial() == 1);
    }
    public void testFactorialPositive() {
      MathFunc math = new MathFunc(5);
      assertTrue(math.factorial() == 120);
    }
    public void testPlus() {
      MathFunc math = new MathFunc(45);
      assertTrue(math.plus(123) == 168);
    }
  }
```

Метод assertTrue проверяет, является ли результат выражения верным. Некоторые другие методы, которые могут пригодиться - assertEquals, assertFalse, assertNull, assertNotNull, assertSame.

Для того, чтобы объединить тесты, можно воспользоваться классом TestSuite с его методом addTest. Наконец, для запуска всех тестов нужно воспользоваться TestRunner. Здесь используется текстовый junit.textui.TestRunner (есть также графические версии - junit.swingui.TestRunner, junit.awtui.TestRunner). В итоге мы получаем метод main следующего вида:

```
  public static void main(String[] args) {
    TestRunner runner = new TestRunner();
    TestSuite suite = new TestSuite();
    suite.addTest(new TestClass("testFactorialNull"));
    suite.addTest(new TestClass("testFactorialPositive"));
    suite.addTest(new TestClass("testPlus"));
    runner.doRun(suite);
  }
```

После запуска получим такие данные:

Time: 0,02

OK (3 tests)

Для тестирования более комплексных классов, могут пригодится методы setUp и tearDown. Первый метод может проинициализировать один или несколько экземпляров тестируемого класса для использования в нескольких TestCases, второй метод отпускает захваченные при инициализации ресурсы.

Тестирование с помощью JUnit становится все более популярным. Думаю, JUnit Framework во многом поможет вам.

Введение в DUnit

Идея использования DUnit состоит в том, что когда вы пишите или изменяете код, вы сразу пишите и соответствующие этому коду тесты, не откладывая на более позднюю фазу тестирования. При соблюдении этого подхода и пересмотрах через регулярные промежутки времени приложения превращаются в само тестируемые.

DUnit предлагает классы, которые позволяют вам просто организовать и выполнить созданные вами тесты. Предлагаются две опции для запуска ваших тестов:

– с графическим интерфейсом, когда вы можете выбрать индивидуальные тесты и группы тестов

– в виде консольных приложений

DUnit была создана на основе библиотеки JUnit, разработанной Кеннетом Беком(Kent Beck) и Эрихом Гамма(Erich Gamma) для языка Java, которая превратилась в мощный инструмент, для программирования на Delphi. Оригинальный порт под Delphi был сделан Juanco Anez и в настоящее время поддерживается DUnit Group на SourceForge.

Для демонстрации тестирования с библиотекой DUnit была создана программа SalesMgr. Есть четыре формы Склад, Клиенты, Новая продажа и архив продаж. Использовались сервер FireBird(+FibPlus) (рис.1).

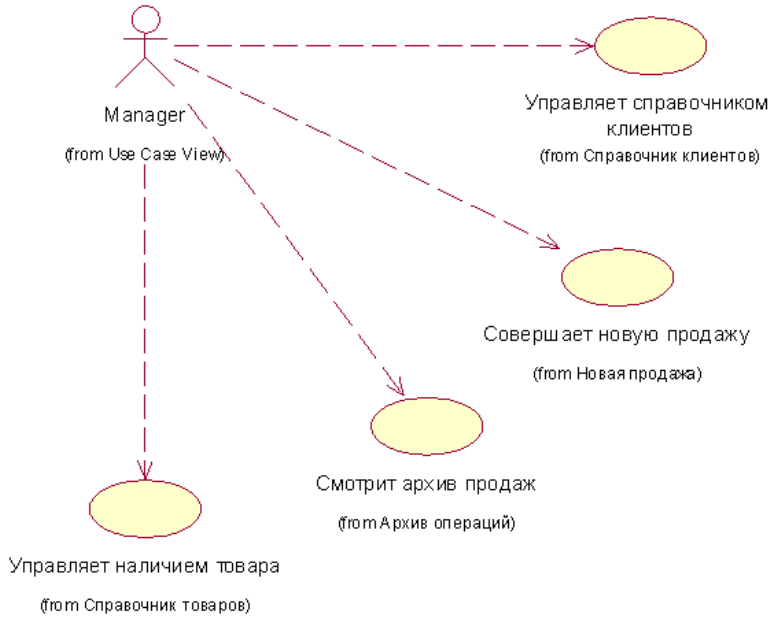


Рисунок 1 – Описание функций менеджера

Программа упрощенно имеет следующую структуру (рис.2)

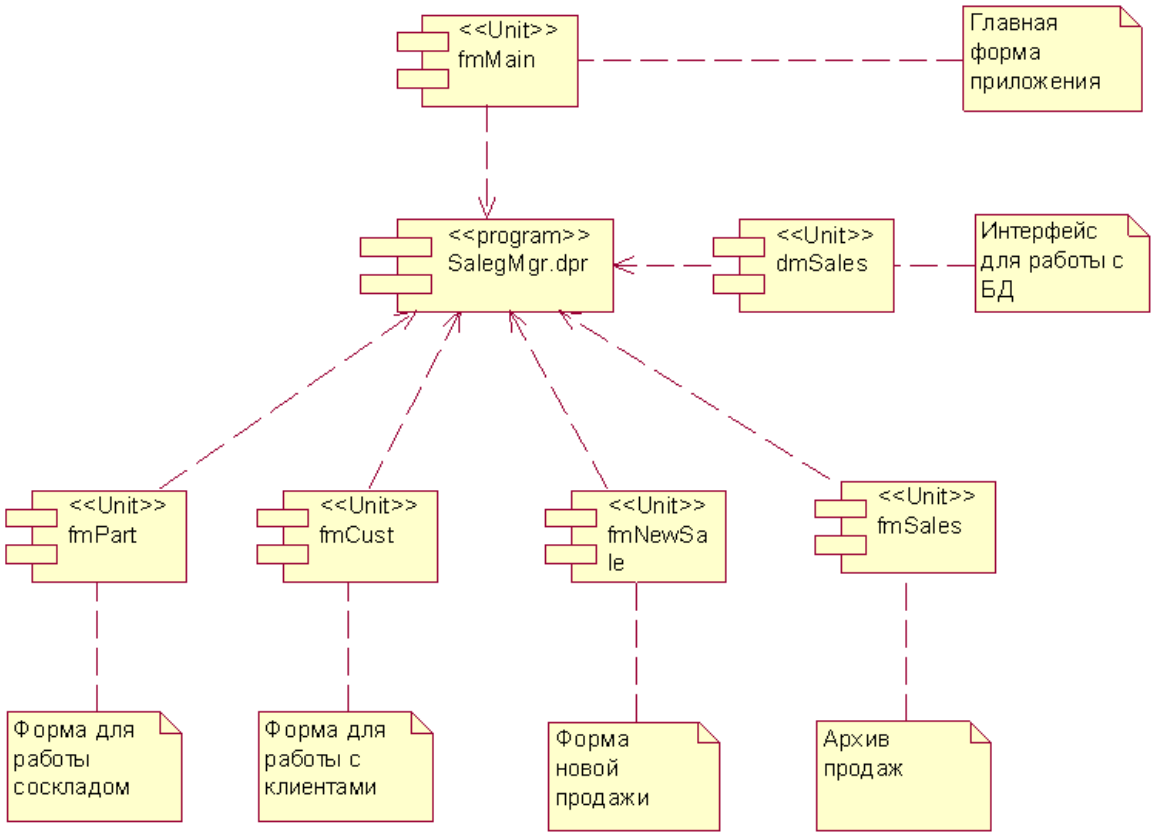


Рисунок 2 – Структура тестируемой программы

Посмотрим, чем нам может помочь DUnit. Все мы тестировать не будем, ограничимся основными операциями (рис.3).

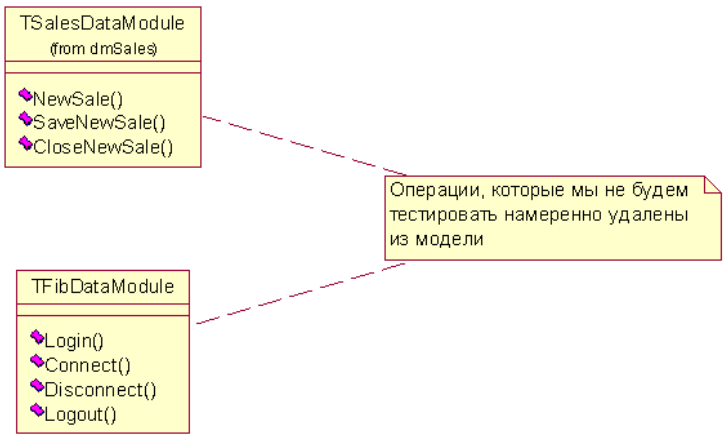


Рисунок 3- Тестируемые операции

При создании подобных программ, следует сделать шаблон проекта, в который входит модуль данных, занимающийся подключением к БД и хранением глобальных переменных.

Итак, займемся последовательным тестированием модулей:

- TFibDataModule
  - Login
  - Logout

- TSalesDataModule
  - NewSale
  - SaveSale
- TfrmNewSale
  - EnabledOp
  - sbAcceptClick
  - sbCancelClick

Чем нам может помочь DUnit? В каталоге ContribXPGen есть кодогенератор! Всего за пару минут можно сделать шаблон теста для модуля. Простая, но полезная утилитка, экономит массу времени.

Посмотрите Readme к DUnit, что бы не отвлекаться на основных моментах.

Итак, запускаем XPGen и подаем ему на вход dmFib.pas(TFibDataModule), немного правим и получаем код для тестирования функций dmFib:

```
unit Test_dmFib;
   interface
   uses
 TestFramework, SysUtils, Forms, dmFib;
type
   CRACK_TFibDataModule = class(TFibDataModule);
   Check_TFibDataModule = class(TTestCase)
   private
   DM: TFibDataModule;
   public
   //это для инициализации тестов
   procedure setUp; override;
   procedure tearDown; override;
   published
   //это непосредственно тесты
   procedure VerifyLogin;
   procedure VerifyLogout;
   end;
function Suite : ITestSuite;
implementation
function Suite : ITestSuite;
   begin
   result := TTestSuite.Create('dmFib Tests');
   // result.addTest(Check_TFibDataModule); почему-то не работает, хотя сге-
нерировано автомат.
   end;
   procedure Check_TFibDataModule.setUp;
   begin
   DM := TFibDataModule.Create(Application);
   end;
   procedure Check_TFibDataModule.tearDown;
```

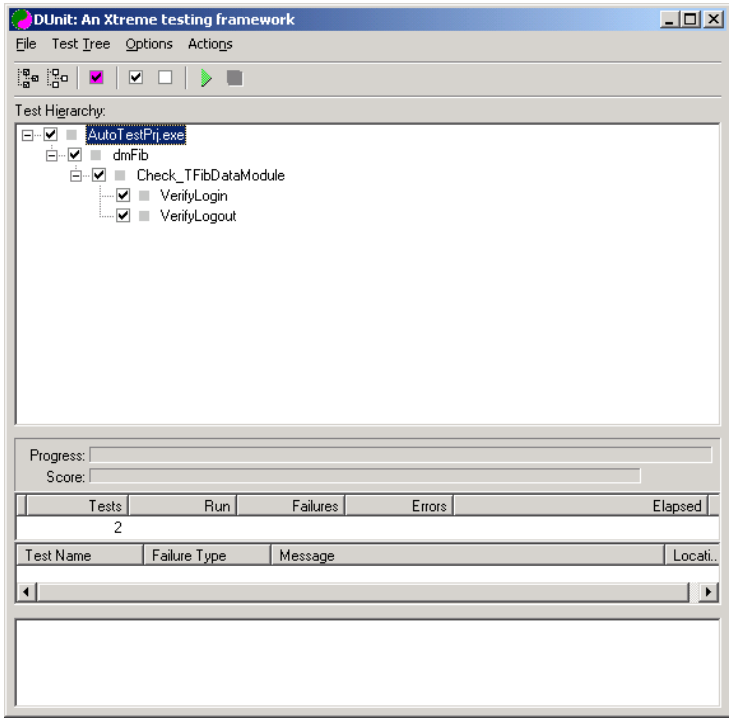
```
   begin
   DM.Free;
   end;
procedure Check_TFibDataModule.VerifyLogin;
   begin
   DM.Login;
   Check(DM.dbFib.Connected = True, 'Login failure!');
   end;
   procedure Check_TFibDataModule.VerifyLogout;
   begin
   DM.Login; DM.Logout;
   Check(not DM.dbFib.Connected, 'Logout failure!');
   end;
   initialization
   TestFramework.RegisterTest('dmFib', Check_TFibDataModule.Suite);
   end.
```

Код проекта:

```
program AutoTestPrj;

uses
   Forms,
   TestFrameWork,
   GUITestRunner,
   Test_dmFib in 'Test_dmFib.pas',
   dmFib in '..CommonFilesdmFib.pas' {FibDataModule: TDataModule},
   common in '..commonfilescommon.pas',
   Test_dmSales in 'Test_dmSales.pas';
   {$R *.res}
 begin
   Application.Initialize;
   GUITestRunner.RunRegisteredTests;
 end.
```

Запускаем и видим:



Нажимаем на всем знакомую зеленую треугольную кнопочку и получаем первый автоматизированный тест!

А для тестирования функций TsalesDataModule придется немного повозиться. Это связано с инициализацией входных данных.

```
unit Test_dmSales;
interface
uses
   TestFramework,
   SysUtils,
   Forms,
   dmSales;
type
CRACK_TSalesDataModule = class(TSalesDataModule);
Check_TSalesDataModule = class(TTestCase)
   public
   procedure setUp; override;
   procedure tearDown; override;
   published
   procedure VerifyNewSale;
   end;
function Suite : ITestSuite;
implementation
uses dmAttr, dmFib;
function Suite : ITestSuite;
   begin
   result := TTestSuite.Create('dmSales Tests');
   // result.addTest(testSuiteOf(Check_TSalesDataModule));
   end;
procedure Check_TSalesDataModule.setUp;
   var ctg: Integer; meas: string;
```

```
  begin
  FibDataModule := TFibDataModule.Create(Application);
  AttrDataModule := TAttrDataModule.Create(Application);
  SalesDataModule := TSalesDataModule.Create(Application);
  FibDataModule.Login;
 { TODO 1 -cDUnit_TEST : добавить тестовые ед.изм. и категории }
  AttrDataModule.OpenAttrs;
  if AttrDataModule.dtSection.RecordCount = 0 then
  AttrDataModule.__InsCatg('Test_Catg');
  if AttrDataModule.dtMeasure.RecordCount = 0 then
  AttrDataModule.__InsMeas('TESTMEA');
 ctg := AttrDataModule.GetCatgId;
  meas:= AttrDataModule.GetMeasure;
 { TODO 1 -cDUnit_TEST : добавить тестовое наличие товара на складе }
  SalesDataModule.OpenPart;
  SalesDataModule.__InsPart(1,ctg,'Part Position 1',50,5,6,meas);
  SalesDataModule.__InsPart(1,ctg,'Part Position 2',50,5,6,meas);
  SalesDataModule.__InsPart(1,ctg,'Part Position 3',50,5,6,meas);
  SalesDataModule.__InsPart(1,ctg,'Part Position 4',50,5,6,meas);
  SalesDataModule.__InsPart(1,ctg,'Part Position 5',50,5,6,meas);
 { TODO 1 -cDUnit_TEST : добавить тестовых покупателей }
  meas := DateToStr(Now);
  SalesDataModule.OpenCust;
  SalesDataModule.__InsCust(1,'Customer 1 '+meas,0,'Test Customer 1');
  SalesDataModule.__InsCust(1,'Customer 2 '+meas,0,'Test Customer 2');
 FibDataModule.CommitRetainingAll;
  end;
procedure Check_TSalesDataModule.tearDown;
  begin
  FibDataModule.Logout;
  SalesDataModule.Free;
  AttrDataModule.Free;
  FibDataModule.Free;
  end;
//////////////////////////
  procedure Check_TSalesDataModule.VerifyNewSale;
  begin
  with SalesDataModule do begin
  NewSale; //добавляем по единице каждого товара
  Check(dtPrice.RecordCount <> 0, 'нечего продавать');
  if dtPrice.RecordCount = 0 then Exit;
  dtPrice.First;
  while not dtPrice.Eof do begin
  dtPrice.Edit;
  dtPrice.FBN('QTY').AsInteger := 1;
  dtPrice.Next;
  end;
  SaveNewSale(Now, DateToStr(Now) + ' Test_dmSales');
  CloseNewSale;
  //ищем нашу новую продажу
```

```
   OpenSales; dtSales.Last;
   Check(dtSales.FBN('DESCR').AsString = DateToStr(Now) + '
Test_dmSales','NewSale failure!');
   end;
   end;
initialization
   TestFramework.RegisterTest('dmSales', Check_TSalesDataModule.Suite);
   end.
```

Тестирование интерфейса

Все как обычно – запускаем XPGen, даем ему на вход fmNewSale.pas, подправляем немного в соответствии нашими целями.

```
unit Test_fmNewSale;
   interface
   uses
   TestFramework,
   GUITesting,
   GUITestRunner,
   SysUtils,
   Graphics,
   Windows,
   Classes,
   Forms,
   fmNewSale;
type
   CRACK_TfrmNewSale = class(TfrmNewSale);
   Check_TfrmNewSale = class(TGUITestCase)
   private
   NewSaleFrm: TfrmNewSale;
   public
   procedure setUp; override;
   procedure tearDown; override;
   published
   procedure VerifyEnabledOp;
   procedure VerifysbAcceptClick;
   procedure VerifysbCancelClick;
   end;
function Suite : ITestSuite;
implementation
uses common, dmAttr, dmFib, dmSales, fmFilter, FrActionFrm;
function Suite : ITestSuite;
   begin
   result := TTestSuite.Create('fmNewSale Tests');
   end;
procedure Check_TfrmNewSale.setUp;
   begin
   inherited;
   FibDataModule := TFibDataModule.Create(nil);
   AttrDataModule := TAttrDataModule.Create(nil);
```

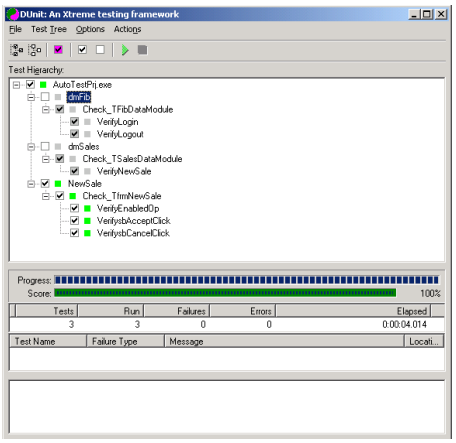
```
   SalesDataModule := TSalesDataModule.Create(nil);
   FibDataModule.Login;
   SalesDataModule.OpenCust;
 NewSaleFrm := TfrmNewSale.Create(nil);
   GUI := NewSaleFrm;
   end;
procedure Check_TfrmNewSale.tearDown;
   begin
   GUI := nil;
   NewSaleFrm.Free;
   inherited;
   end;
procedure Check_TfrmNewSale.VerifyEnabledOp;
   begin
   Show;
   Check(NewSaleFrm.sbAccept.Enabled = False, 'можно произвести пустую опера-
цию ?');
   Check(NewSaleFrm.sbCancel.Enabled = False, 'можно отменить пустую операцию
?');
   SalesDataModule.dtPrice.Edit;
   SalesDataModule.dtPrice.FBN('QTY').AsInteger := 0;
   SalesDataModule.dtPrice.Post;
   Check(NewSaleFrm.sbAccept.Enabled = False, 'можно произвести пустую опера-
цию ?');
   Check(NewSaleFrm.sbCancel.Enabled = False, 'можно отменить пустую операцию
?');
   end;
procedure Check_TfrmNewSale.VerifysbAcceptClick;
   begin
   Show;
   Check(NewSaleFrm.AddingOp = False, 'пустая операция ?');
   SalesDataModule.dtPrice.Edit;
   SalesDataModule.dtPrice.FBN('QTY').AsInteger :=1;
   SalesDataModule.dtPrice.Post;
   Check(NewSaleFrm.AddingOp, 'непустая операция ?');
   NewSaleFrm.sbAccept.Click;
   Check(NewSaleFrm.AddingOp = False, 'пустая операция ?');
   end;
procedure Check_TfrmNewSale.VerifysbCancelClick;
   begin
   Show;
   Check(NewSaleFrm.AddingOp = False, 'пустая операция ?');
   SalesDataModule.dtPrice.Edit;
   SalesDataModule.dtPrice.FBN('QTY').AsInteger :=1;
   SalesDataModule.dtPrice.Post;
   Check(NewSaleFrm.AddingOp, 'непустая операция ?');
   NewSaleFrm.sbCancel.Click;
   Check(NewSaleFrm.AddingOp = False, 'пустая операция ?');
   end;
initialization
   TestFramework.RegisterTest('NewSale', Check_TfrmNewSale.Suite);
```

```
   end.
```

Пришлось немного переделать свою форму-шаблон, так и не смог справиться со стандартными диалогами, которые создаются методом MessageDlg.

Окончательно проект принял такой вид:

```
program AutoTestPrj;
   uses
   Forms,
   TestFrameWork,
   GUITestRunner,
   Test_dmFib in 'Test_dmFib.pas',
   dmFib in '..CommonFilesdmFib.pas' {FibDataModule: TDataModule},
   common in '..commonfilescommon.pas',
   dmSales in '..SalesMgrdmSales.pas' {SalesDataModule: TDataModule},
   dmAttr in '..commonfilesdmAttr.pas' {AttrDataModule: TDataModule},
   fmFilter in '..commonfilesfmFilter.pas' {frmInvFilter},
   FrActionFrm in '..commonfilesFrActionFrm.pas' {FrActionForm},
   fmNewSale in '..SalesMgrfmNewSale.pas' {frmNewSale},
   OpTemplateFrm in '..commonfilesOpTemplateFrm.pas' {OpTemplateForm},
   ChildFrm in '..commonfilesChildFrm.pas' {ChildForm},
   ItemsFm in '..commonfilesItemsFm.pas' {ItemsFrame: TFrame},
   Test_dmSales in 'Test_dmSales.pas',
   Test_fmNewSale in 'Test_fmNewSale.pas';
{$R *.res}
begin
   Application.Initialize;
   GUITestRunner.RunRegisteredTests;
   end.
```



Понятие функционального тестирования

Кроме модульных тестов обязательной составляющей разрабатываемого решения должны являть функциональные тесты. Данный тип тестов не использует атомарные составляющие программного решения, такие как методы, а проверяет определённые заказчиком бизнес-функции в целом.

Основой для создания пакета функциональных тестов должен служить набор требований заказчика к разрабатываемой системе. При данном подходе бизнес-аналитикам, отвечающим за подготовку требований, необходимо

добиться высокой степени их детализации. В противном случае будет невозможно провести автоматизированное тестирование функционала. Например, вполне заслуживающее внимание, по мнению бизнес-аналитиков, требование «время простоя разрабатываемой системы по техническим причинам не должно превышать 10 часов в год» не может быть проверено на базе теста. С другой стороны требование «время отклика платёжного терминала клиента не должно превышать 5 секунд» является прекрасно контролируемым автоматически.

В мире сервис-ориентированных архитектур, где фактический интерфейс доступа к программной системе представляет собой Web-сервис, функциональные и модульные тесты становятся близки по своей сути. В тоже время, если программное решение имеет развитый GUI, то функционально тестирование становится значительно более сложным процессом.

Введение в Selenium

Selenium как проект был начат относительно недавно (в июне 2004 года, стал открытым в декабре 2004 года) и ранее велся под патронажем компании ThoughtWorks (владелец этой компании, кстати, Мартин Фаулер). Selenium выпускается под лицензией Apache License, Version 2.0.

Selenium доступен по этому адресу: http://www.openqa.org/selenium/

Selenium- это объектно-ориентированное JavaScript приложение, которое может анализировать файлы определенной структуры для того, чтобы находить в них команды для манипуляции браузером и команды для выполнения определенных проверок. В поставке Selenium входит документация, которая может помочь быстро разобраться в использовании этой системы для функционального тестирования.

Содержимое TestCase-а в Selenium

TestCase-ы Selenium-а – это обычные html-страницы с одной таблицей, содержащей команды. Каждая строка таблицы содержит 3 колонки. Первая из них является действием или проверкой (action и assertion/check), вторая – именем элемента (target), к которому применяется команда, и третья – значением (value). Вот пример файла TestCase-а Selenium-а, назовем его Test Login:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
   <html>
   <head>
   <meta content="text/html; charset=win-1251" http-equiv="content-type">
   <title>Test Login</title>
   </head>
   <body>
   <table cellpadding="1" cellspacing="1" border="1">
   <tbody>
   <tr>
   <td rowspan="1" colspan="3">Test Login<br>
   </td>
```

```
  </tr>
  <tr>
  <td>open</td>
  <td>/login</td>
  <td> </td>
  </tr>
  <tr>
  <td>type</td>
  <td>login</td>
  <td>vasa</td>
  </tr>
  <tr>
  <td>type</td>
  <td>password</td>
  <td>super_admin</td>
  </tr>
  <tr>
  <td>clickAndWait</td>
  <td>login_button</td>
  <td> </td>
  </tr>
  <tr>
  <td>verifyLocation</td>
  <td>/</td>
  <td> </td>
  </tr>
  <tr>
  <td>verifyTextPresent</td>
  <td>Wellcome, Vasa Pupkin</td>
  <td> </td>
  </tr>
  </tbody>
  </table>
  </body>
  </html>
```

Разберем по порядку, что же делает Selenium, выполняя данный TestCase. Стоит сразу указать, что Selenium пропускает первую строку таблицы. Здесь вы можете поместить краткое описание, что же будет осуществляться в данном тесте. Мы покажем ниже, как можно вставлять комментарии в тестовый код другими способами. Итак,

```
 <tr>
   <td>open</td>
   <td>/login</td>
   <td> </td>
</tr>
```

Выполняет действие open. Selenium дает команду браузеру перейти на указанную страницу - /login. Колонка для значения здесь не используется. Допустим в результате выполнения у нас открылась страница, содержащая форму авторизации:

```
<form id='login_form' name='login_form' action='' method='post'>
   Login : <input id='login' name='login' type='text' size='40'><br>
   Password : <input id='password' name='password' type='password' size='40'>
   <input id='login_button' type='submit' value='Login'>
</form>
```

Теперь Selenium при помощи действий type заполнит поля формы определенными значениями. Вы визуально увидите, что поля получили соответствующие значения.

```
<tr>
   <td>type</td>
   <td>login</td>
   <td>vasa</td>
   </tr>
   <tr>
   <td>type</td>
   <td>password</td>
   <td>super_admin</td>
</tr>
```

После этого действием clickAndWait Selenium нажмет кнопку для отправки формы и будет дожидаться ответа.

```
 <tr>
   <td>clickAndWait</td>
   <td>login_button</td>
   <td> </td>
 </tr>
```

После этого при помощи проверки verifyLocation Selenium проверит текущее положение браузера. После аутентификации мы должны оказаться на главной странице сайта

```
 <tr>
   <td>verifyLocation</td>
   <td>/</td>
   <td> </td></tr>
```

И, наконец, после успешной аутентификации на должны увидеть приглашение на главной странице. Selenium проверяет наличие соответствующего текста на странице при помощи проверки veritytextPresent

```
 <tr>
   <td>verifyTextPresent</td>
   <td>Wellcome, Vasa Pupkin</td>
   <td> </td>
</tr>
```

Чуть ниже мы разберем, какие команды и проверки есть в Selenium, как указывать определенные элементы на странице, а также как расширить набор доступных команд и проверок.

Организация тестов Selenium

Как уже было указано, TestCase-ы организованы в список, который называется TestSuite. TestSuite также является обыкновенной html страницей, приблизительно такого содержания:

```
<html>
  <head>
    <meta content="text/html; charset=win-1251"  http-equiv="content-type">
    <title>Test Suite</title>
  </head>
  <body>
    <table cellpadding="1" cellspacing="1" border="1">
     <tbody>
      <tr><td><b>Test Suite</b></td></tr>
      <tr><td><a href="./TestOpen.html">TestOpen</a></td></tr>
      <tr><td><a href="./TestLogin.html">TestLogin</a></td></tr>
     </tbody>
    </table>
  </body>
</html>
```

Теперь объясним чуть поподробнее то, как организуются тесты и как они хранятся на диске. В базовой поставке все файлы TestCase-ов и TestSuite-ов являются обычными текстовыми html файлами и лежат в папке /selenium/tests. При запуске Selenium пытается найти файл TestSuite.html в этой папке и загружает список тестов из TestSuite в левое верхнее окно. При щелчке на одну из ссылок файла TestSuite.html в окно текущего TestCase-а загружается содержимого соответствующего файла с командами. Вот в принципе и все. Для расширения набора тестов какого-либо приложения мы должны создать соответствующий TestCase html файл и поместить на него ссылку в TestSuite.html. Это, правда, самый простейший вариант использования Selenium-а, однако большинство разработчиков он устроит.

Отметим здесь же, что хотя Selenium воспринимает TestCase-ы только в виде html страниц, это вовсе не значит, что сами TestCase-ы должны быть html файлами. Они могут генериться при помощи PHP, JSP, CGI, в общем, при помощи любой технологии, которая вам может показаться удобной. Однако нельзя не согласиться, что Selenium – это одна из самых простых на сегодняшний момент систем, чтобы начать функциональное тестирование своих web-приложений.

Составляющие команд тестов

В Selenium существует несколько основных понятий:

- действия
- проверки
- локаторы

Действия

Действия (actions) используются для того, чтобы управлять браузером из-под Selenium. Набор действий достаточно широк. Ниже приведем список наиболее часто используемых действий. В скобках будет дана форма применения в формате wiki-table. Итак вот некоторые из действий Selenium:

- open – указывает браузеру открыть страницу по определенному адресу (|open|location||)
- click – указывает браузеру щелкнуть по ссылке или по элементу формы, например, для пометки checkbox-a (|click|target||)
- type – указывает браузеру ввести новое значение в элемент формы (|type|element|value|)
- select – указывает браузеру выбрать определенное значение <option> внутри <select> тега формы(|select|element|value)
- selectWindow – указывает браузеру переключить фокус на другое окно (|selectWindow|window_name||)
- goBack – указывает браузеру вернуться на предыдущую страницу
- close – указывает браузеру закрыть текущее окно

Полный список и примеры использования действий смотрите в документации, поставляемой вместе с Selenium.

Проверки

Проверки (checks) используются для проверки правильности поведения приложения. Любая проверка в Selenium представлена двумя типами – assert и verify. Если в тесте стоит проверка assertSomething и она не выполняется, тест прекращает свою работы, а если verifySomething – то выдается сообщение об ошибке, но тест продолжает работу. Вот список наиболее часто используемых проверок:

- verifyLocation / assertLocation – проверяет текущий URL окна браузера (|verifyLocation|url_needed||)
- verifyTitle / assertTitle – проверяет заголовок окна (|verifyTitle|title_needed||)
- verifyValue / assertValue – проверяет, что поле формы имеет указанное значение (|verifyValue|field|value_needed|)
- verifyTextPresent / assertTextPresent – проверяет, что текст страницы содержит указанный текст (|verifyTextPresent|text_needed||)
- verifyElementPresent / assertElementPresent – проверяет, что элемент присутствует на текущей странице (|verifyElementPresent|element_needed||)

Полный список и примеры использования проверок смотрите в документации, поставляемой вместе с Selenium.

Локаторы Selenium или как находятся элементы, к которым применяются команды

Локаторы используются для нахождения элементов, к которым относятся команды. Список локаторов Selenium достаточно большой:

- id – используется атрибут id (идентификатор) элемента
- name – используется атрибут name элемента
- identifier – используется атрибут id элемента, если по id-у элемент не найден, то поиск будет вестись по атрибуту name
- dom – используется для поиска элемента по DOM выражению, которое должно начинаться с document.
- xpath – используется для поиска элемента по XPath выражению, которые должно начинаться с //
- link – используется для нахождения ссылок с указанным текстом.
- Разберем, как используются локаторы в командах, на примере, чтобы все стало понятно. Возьмем знакомую форму аутентификации.

```
<form id='login_form' name='login_form' action='' method='post'>
   Login : <input id='login' name='login' type='text' size='40'><br>
   Password : <input id='password' name='password' type='password' size='40'>
   <input id='login_button' type='submit' value='Login'>
</form>
```

Для заполнения поля login формы login_form можно воспользоваться следующими командами:

```
<tr>
   <td>type</td>
   <td>id=login</td>
   <td>vasa</td>
</tr>
```

Использовали локатор id.

```
<tr>
<td>type</td>
<td>name=login</td>
<td>vasa</td>
</tr>
```

Использовали локатор name.

```
<tr>
<td>type</td>
<td>identifier=login</td>
<td>vasa</td>
</tr>
```

Использовали локатор identifier.

```
<tr>
<td>type</td>
<td>xpath=//input[@name='login']</td>
<td>vasa</td>
</tr>
```

Использовали локатор xpath.

```
<tr>
<td>type</td>
<td>dom=document.forms['login_form'].login</td>
```

```
<td>vasa</td>
</tr>
```

Использовали локатор dom.

Если в команде не указано явно название типа локатора, например не dom=document.forms[‘login_form’].login, а просто document.forms[‘login_form’].login , тогда Selenium последовательно пытается найти элемент при помощи следующих типов локаторов:

- identifier
- dom
- xpath

Обратите внимание, что для тегов <select> есть особые локаторы, впрочем, как и специальные действия и проверки. Мы не будем уделять этому внимание, так как все это достаточно хорошо описано в документации. Использование XPath локаторов

Использование XPath локаторов может значительно упростить жизнь тестировщику, особенно если у него нет прав доступа менять исходный код web-приложения, которое он тестирует. Стоит сразу отметить, что при работе Selenium с IE даже версии 6.0 при использовании XPath выражений, были замечены некоторые проблемы, однако в FireFox никаких нареканий нет.

Вот некоторые примеры использования XPath локаторов:

a[contains(text(), 'partial text')] – поиск ссылки по частичному совпадению *

input[@type='submit'] – поиск кнопки для отправления формы по типу

input[@value='Button Description'] - поиск кнопки по надписи на нет *

table[@id='yourTable']tr[td='uniqueCustomerCode']/tda/img[@alt='Edit'] – поиск ссылки по картинки, которая находится в определенной ячейке таблицы.

CASE-средства проектирования программного обеспечения

Проектирование предлагается произвести с помощью CASE-средства Enterprise Architect.

CASE-средство Enterprise Architect предназначено для построения визуальных моделей в процессе создания программного обеспечения (ПО). При этом оно охватывает все этапы жизненного цикла ПО: анализ, дизайн архитектуры, разработка программного кода, тестирование и размещение продукта на стороне заказчика.

Визуальным моделированием называется процесс графического представления модели с помощью некоторого стандартного набора графических элементов. Основной целью визуального моделирования является общение между всеми участниками проекта: пользователи визуализируют свое взаимодействие с системой; аналитики увидят взаимодействие между объектами

модели; разработчики поймут, какие объекты нужно создать и что эти объекты должны делать; менеджеры увидят как всю систему в целом, так и взаимодействие ее частей; наконец, руководители информационной службы, глядя на высокоуровневые модели, поймут, как взаимодействуют друг с другом системы в их организации.

Совокупность набора графических элементов, используемый в процессе визуального моделирования называется нотацией. В CASE-средстве Enterprise Architect реализована нотация UML - унифицированный язык моделирования (Unified Modelling Language, UML).

UML позволяет создавать несколько типов визуальных диаграмм. CASE-средство Rational Rose поддерживает разработку большинства этих моделей, в частности:

- Диаграммы вариантов использования или прецедентов (**Use Case**)
- Диаграммы классов (**Classes**)
- Диаграммы последовательности (**Sequence**)

и другие.

Порядок и этапы выполнения курсовой работы

При разработке программного обеспечения студенты должны придерживаться следующей последовательности выполнения работы:

а. Анализ

1. **Выработка системных требований (постановка задачи).** Сбор требований: общение с клиентами и пользователями, чтобы определить, каковы их требования.

Текстовое описание разрабатываемого программного продукта на русском языке. Формируется на основе задания к курсовой работе. В задание могут быть внесены дополнительные функции и ограничения (лучше не увлекаться). Также описываются условия эксплуатации программного продукта. Текстовое описание включает:

- текстовое описание предметной области (как есть);
- составление словарей: предметной области, по Абботу, объектно-ориентированного,
- постановка задачи (что хочет заказчик, кто будет пользоваться, последовательность и ограничения на выполняемые действия, описание входных/выходных параметров и ограничения накладываемые пользователем на эти параметры);
- формулировка цели работы,

2. **Анализ требований**: определение, являются ли собранные требования неясными, неполными, неоднозначными, или противоречащими, и затем решение этих проблем.

- описание функционального назначения системы, используя SADT диаграммы (детализация минимум до 3-го уровня),
- определение отношения между действующими лицами и вариантами использования для этого определить роли разрабатываемой системы и идентифицировать варианты использования системы,

3. **Документирование требований**: Формализованный набор требований, разбитый по ролям пользователей программного продукта. Обязательно должен включать ограничения, накладываемые на программный продукт. Определить, какие из вариантов использования (не менее трех) будут уточняться при последующем моделировании и будут реализованы в прототипе,

- полная диаграмма (или несколько диаграмм-по ролям) использования.

- реализация выбранных вариантов использования (один или два) в виде записи сценария на естественном языке и оставшиеся, из выбранных, вариантов использования реализовать диаграммами UML (всего не менее трех),
- определить нефункциональные и специальные требования, если они необходимы,
- описать поведение разрабатываемой системы;
- описать и сформировать интерфейсные классы;
- построить инфологическую модель разрабатываемой предметной области (нотация Чена).
-

3.1 Разработка технического задания (объединение всех требований в единый документ).

Рекомендуется придерживаться при формировании структуры документа видение из RUP:

- Введение,
- Позиционирование,
- Описания совладельцев и пользователей,
- Краткий обзор продукта,
- Возможности продукта,
- Ограничения,
- Показатели качества,
- Старшинство и приоритеты.
- Анализ и управление рисками,
- Другие требования к изделию,
- Требования к документации.

b. Проектирование программного продукта.

Начинается с выбора и обоснования набора программных инструментов, используемых при проектировании и реализации системы.

Проектированию обычно подлежат:

1. Архитектура ПО, Архитектура программного обеспечения — совокупность важнейших решений об организации программной системы. Архитектура включает: выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов и включает:

- проектирование классов разрабатываемой системы;
- проектирование хранилища данных, формирование логической и физических моделей);
- описание фактической организации модулей системы, разделение их на пакеты и объекты, описание подсистем (показывает, как система должна быть реализована);

- описание модели развертывания продукта (описание размещения программных компонент системы на аппаратных платформах);
- реализация ВИ с учетом спроектированных классов данных и интерфейсных классов (показать связи между этими компонентами системы);
- модели реализации процессов (рассматриваются одновременное выполнение и распределение процессов, интеграция системы, устойчивость к сбоям, а также то, как основные объекты абстракции, рассмотренные на уровне логического представления, соответствуют архитектуре процессов).

2. Пользовательские интерфейсы.

- разработка макетов интерфейсных классов, в соответствии с моделями реализации ВИ и процессов с описанием макетов.

c. Программная реализация:

- описание классов данных и реализованных методов
- кодирование (код должен быть описан);

d. Тестирование (автоматизированное).

- Модульное тестирование (тест, код с комментариями, результат тестирования)
- Функциональное тестирование (тест, код с комментариями, результат тестирования)
- Структурное тестирование (тест, код с комментариями, результат тестирования)
- Интерфейсное тестирование (тест, код с комментариями, результат тестирования)

c. Разработка документации.

В результате выполнения курсовой работы должен быть получен единый документ - описание проекта по разработке программного обеспечения, включающий все приведённые выше разделы. Данный документ сдаётся в печатном виде. Кроме того, данный документ должен сопровождать программным продуктом и тестами для него, а так же включать документацию пользователя.

- оформление документации пользователя;
- оформление документации по выполненной работе согласно требованиям стандартов ИРНИТУ.

В качестве средств программной реализации работы предлагается использовать:

- для описания преобразования – PHP;
- для хранения данных – СУБД MySQL;
- для реализации клиентской части настольного приложения – Delphi XE.

Варианты заданий

Вид приложения носит рекомендательный характер. Студентом разрабатываются не менее трех функций, автоматизируемых в этой предметной области, и плюс различные отчеты.

	Вид приложения	**Задание на проектирование и разработку программного продукта**
1.	Веб	**Call-центр технической поддержки сотового оператора,** обеспечивающее клиента функциями связи с менеджером, выдачи баланса, консалтинга по тарифам, подключения / отключения услуг и не менее 3 других функций.
2.	Веб	**Call-центр салона красоты**, обеспечивающее взаимодействие клиента, менеджера и базы учёта. Call-центр выдаёт стандартное приветствие и предоставляет клиенту голосовое меню, с помощью которого он может получить информацию об услугах салона, записаться на сеанс к стилисту, визажисту, массажисту, парикмахеру и т.д., связаться с менеджером, отменить, изменить время сеанса, записаться на повторный приём, получить справку. Менеджер может напрямую работать с базой учёта, внося туда сведения о клиентах и мастерах.
3.	Веб	**Формирование командировочного удостоверения** преподавателя вуза. Программное средство управляется бухгалтером. Преподаватель заполняет электронную анкету с типом командировки (по России или международная), местом назначения, временем пребывания, целью и способом проезда к месту командировки. Руководство вуза выдаёт (или не выдаёт) разрешение на командировку, в результате чего на основе анкетных данных преподавателя и реквизитов вуза, полученных из учётной базы данных, формируется проект приказа на командировку преподавателя, который получает преподаватель и предоставляет бухгалтеру. Бухгалтер оценивает по справочникам примерную стоимость проезда, проживания и величину

		суточных расходов и формирует величину аванса, который начисляется программным средством на зарплатную карточку преподавателя. По истечении командировки преподаватель предоставляет бухгалтеру чеки, подтверждающие использование аванса. Если потрачено меньше, чем запланировано, то осуществляется удержание с преподавателя разницы, если больше – то доначисление.
4.	Веб	**Библиотечный каталог вуза**, взаимодействующий со студентами, библиотекарями и преподавателями. Функции преподавателей – поиск учебников, заказ на закупку учебников; функции библиотекарей – внесение (исключение) записей в базу данных (или подтверждение по запросам студентов через интернет), распределение полномочий; функции студентов – алфавитный, тематический и не менее 2-х видов других поисков, запрос электронной версии, запрос на заказ учебника, запрос на справку, связь с библиотекарем через интернет или телефон.
5.	Веб	**Регистратура детской стоматологической клиники**, имеет агентов: секретарь, врачи и клиенты. Функции секретаря – внесение, исключение записей в базе данных (или подтверждение по запросам клиентов через интернет), запрос расписания врачей, определение времени приёма по справочнику, исключение или перенос записей по требованию врача с уведомлением клиентов, связь с врачом. Функции врача – анализ заявок на обслуживание и формирование предложений по оптимизации записей при наличии ошибок или иным обстоятельствам, заполнение карточки по результатам приёма; функции клиентов – поиск информации о расписании врачей, получение информации об опыте, образовании и достижениях врачей, запись на приём (по телефону, лично у секретаря или через интернет), запрос на справку у врача, связь с секретарём через интернет или телефон.
6.	Веб	**Регистрация клиентов в гостинице**, включающее агентов: администратор, персонал, клиент. Функции администратора: внесение паспортных и анкетных данных клиента, получение сведений о свободном номерном фонде, бронирование или немедленное предоставление ключа доступа в комнату клиента, выписка клиента, выдача справок клиенту, приём заявок на вызов персонала гостиницы (горничные, плотники, слесаря и т.д.); функции персонала – регистрация времени посещения клиента и сделанных работ; функции клиента – бронирование по телефону, лично у администратора или через интернет номера в гостинице; заполнение через интернет анкеты для заселения или передача информации администратору, получение

		справки у администратора, связь с администратором, получение карты доступа в комнату, использование карты доступа в номер.
7.	Веб	**Автоматизация работы приемной комиссии вуза**, включающее агентов: администратор, абитуриент. В вузе определен список специальностей и количество студентов для приема на каждую специальность, который может изменяться ежегодно. По каждой специальности вуза определен список предметов, сдаваемых абитуриентами, например: математика, русский язык и т. д. Функции администратора: внесение паспортных и анкетных данных абитуриента, ввод, коррекция и просмотр специальностей и сдаваемых предметов, обработка сведений о результатах сдачи экзаменов, вывод списка поступивших студентов по специальностям. Функции абитуриента-просмотр результатов экзамена и списка студентов, поступивших в вуз.
8.	Настоль-ное	**Информационная система ЗАРПЛАТА** для автоматизации начислений заработной платы, включающее агентов: бухгалтер и табельщик. На каждого работника хранятся следующие данные: табельный номер; Ф.И.О., отдел, должность; оклад; семейное положение и число детей; данные о невыходе на работу по болезни (даты заболевания и выздоровления, даты отпуска и т.д.) и т. д. . Функции бухгалтера – начисление зарплаты по сведениям из табеля учета рабочего времени, ввод, изменение анкетных данных работников, сведения о надбавках; ежемесячный перерасчет зарплаты с выдачей ведомости на экран и печать. Функции табельщика – ведение табеля учета рабочего времени (обработка сведений о посещаемости, сведений о болезнях, об отпусках, прогулах, опозданиях и т.д.) Зарплата начисляется работникам, имеющим установленные оклады. В период болезни работнику начисляется 50 % зарплаты. Работникам могут начисляться премии и другие надбавки. С общей суммы зарплаты отчисляется подоходный налог.
9.	Настоль-ное	**Информационная подсистема ДЕКАНАТ** - автоматизация работы деканата факультета (института) вуза, включающее агентов специалист деканата и декан. По каждой специальности имеется учебный план, который содержит список всех предметов, изучаемых студентами этой специальности, с указанием общего количества лекционных, практических, лабораторных часов, распределения предметов и курсовых работ по семестрам с указанием количества часов и видов отчетности (зачет, экзамен, КП, КР) за каждый семестр. На каждого студента заводится учебная карточка, в которую заносятся его

		анкетные данные, группа, специальность, а также список предметов, подлежащих сдаче согласно учебному плану специальности. По мере сдачи предметов и перехода с курса на курс учебная карточка заполняется соответствующими оценками. По окончании вуза копия учебной карточки выдается как приложение к диплому. Функция инспектора – вести учетную карточку студента Функция декана = обеспечивать ввод и обработку учебных планов специальностей, выдачу списков студентов по различным выборкам.
10.	Настоль-ное	**Информационная система КАДРЫ** для автоматизации работы отдела кадров предприятия. Система должна функционировать в двух режимах: первичной загрузки данных и текущей обработки данных. В режиме первичной загрузки данных система должна обеспечивать ввод данных из личных карточек работающих с контролем вводимой информации. В режиме текущей обработки данных система должна реализовывать действия: обработку данных по движению кадров (прием, увольнение, перемещение); получение статистической отчетной и справочной информации по уволенным и работающим (в т. ч. по различным категориям); ведение табельного учета по отсутствующим на рабочих местах.
11.	Веб	**Информационная система БИБЛИОТЕКА.** Библиотеке требуется вести списки читателей и списки книг и других изданий, при этом в фонде может быть несколько разных экземпляров каждого издания. Требуется вести учет книг (изданий) и их местонахождения, учет читателей, библиографических данных об изданиях (в каталогах нескольких видов), осуществлять печать каталожных карточек (библиографических описаний). В реальной ситуации также ведется учет читателей-должников и видов, осуществляется печать напоминаний им, учет сведений о потерях, заменах или оплатах книг и т. д. Читатель, зарегистрированный в библиотеке, может производить поиск книг по каталогу библиотеки, заказывать нужную книгу и просматривать свой абонемент.
12.	Настоль-ное	**Информационная система АРЕНДА ПОМЕЩЕНИЙ**, включающее агентов: жилищная организация, бухгалтерия. Жилищная организация сдает помещения в аренду различным нанимателям (предприятиям и организациям различных форм собственности, физическим лицам). Бухгалтерия - начисляет им помесячно арендную плату и платежи за коммунальные услуги и выставляет счета на оплату (с указанием юридических и платежных реквизитов арендаторов), ведет учет их оплаты на лицевых счетах арендаторов с подведением помесячного баланса (состояние лицевого счета на начало месяца,

		приход, расход и сальдо на конец месяца).
13.	Настоль-ное	**Информационная система ГОРОДСКОЙ СОВЕТ.** В базе данных муниципалитета хранятся имена, адреса, домашние и служебные телефоны всех членов городского совета. В совете много комиссий. Каждая комиссия имеет свой профиль – по вопросам образования, жилищная, торговая, энергетическая и т. д. В муниципальной базе данных записаны данные по каждой из комиссий, ее нынешний состав и председатель, прежние председатели и члены этой комиссии за последние 10 лет, даты включения и выхода из состава комиссии, избрания ее председателей. Многие члены городского совета заседают в нескольких комиссиях. Секретарь городского совета фиксирует дату, время и место проведения каждого заседания комиссий и оповещает членов комиссий, участвующих в текущем заседании Городского совета, о необходимости присутствовать на заседании. Требуется получать информацию по различным вопросам, в том числе и с выводом на печать.
14.	Веб	**Информационная система КОМПЬЮТЕРНАЯ ФИРМА.** Руководитель компьютерной фирмы, выполняющей сборку персональных компьютеров из готовых комплектующих, заказал разработку базы данных, основанной на двух представлениях данных о комплектующих. Одно представление (для клиентов) содержит данные, которые могут отображаться при согласовании с ними комплектности изделия – в ней указаны розничные цены на комплектующие. Цена комплектующих с течением времени может меняться. Второе представление предназначено для внутреннего пользования и анализа результатов деятельности фирмы – в нем содержатся оптовые цены на комплектующие и краткая информация о поставщиках (клиенты предприятия не имеют доступа к данным этого таблицы). Надо обеспечить ведение заказов от клиентов со сроком изготовления и пометкой «оплачено / не оплачено», расчетом суммарной стоимости различных комплектаций персонального компьютера (с указанием розничной стоимости его отдельных комплектующих и т. д.) и всего заказа клиента в целом. Обеспечить оперативный просмотр списка заказов по различным условиям (тип процессора, клиент, стоимость) на любую дату срока изготовления.
15.	Веб	**Информационная система УЧЕТ АКАДЕМИЧЕСКОЙ УСПЕВАЕМОСТИ В ВУЗЕ.** Сведения об академической успеваемости студентов содержатся в их зачетных книжках и в экзаменационных (зачетных) ведомостях: Ф.И.О. студента, номер зачетной книжки, год поступления, институт (факультет), специальности, учебная группа, семестр, название пред-

		мета (учебной дисциплины), вид отчетности (зачет, экзамен, КП, КР), сама оценка («зачтено», «не зачтено», «отлично», «хорошо», «удовлетворительно», «неудовлетворительно»), дата, фамилия преподавателя, количество часов и т. д. Требуется вести учет этих сведений с обеспечением печати заполненных экзаменационных (зачетных) ведомостей для любой группы по любому предмету, семестру и т. д., формирование и обеспечение печати итоговых ведомостей по результатам семестра с подсчетом средних баллов, качества знаний (процент успевающих на 4 и 5), процента успеваемости и неуспеваемости для групп, специальностей, курсов, институтов (факультетов) в среднем и по отдельным учебным дисциплинам в частности. Предоставить возможность студенту в онлайн режиме просматривать свою успеваемость.
16.	Веб	**Информационная система МУЗЫКАЛЬНЫЙ (ВИДЕО) МАГАЗИН**. Магазин музыкальных, компьютерных и видеозаписей ведет для покупателей каталог имеющихся в продаже записей с указанием их розничных цен, жанра, разновидностей жанра, вида носителя записи, имени (названия) исполнителя или автора, названия произведения или записи, года выпуска, производителя и т. д. Кроме того, магазин ведет внутренний учет текущих оптовых цен на записи, количества экземпляров, проданных за отдельные периоды в прошлом, числа еще не распроданных (имеющихся в наличии) экземпляров записей. В реальной ситуации также требуется вести учет заказов на отсутствующие записи, подсчет прибылей/убытков за прошедшие периоды и т. д. Покупатель может просматривать каталоги магазина, заказывать товар, имеющийся в магазине, и оставлять заявку на отсутствующий товар
17.	Веб	**Информационная система РЫБОЛОВНАЯ ФИРМА**. Рыболовной фирме принадлежит небольшая флотилия рыболовных судов. Каждое судно имеет «паспорт», куда занесены его название, тип, водоизмещение и дата постройки. Фирма регистрирует каждый выход на лов, записывая название судна, имена и адреса членов команды с указанием их должностей (капитан, боцман и т. д.), даты выхода и возвращения, а также вес пойманной рыбы отдельно по разным видам рыб. За время одного рейса судно может посетить несколько мест лова. Капитан фиксирует дату прихода на каждое место лова и дату отплытия, качество выловленной рыбы (отличное, хорошее, плохое). Фирма ведет учет лучших мест лова, поощряет команды выловившие наибольшее количество рыбы, формирует отчеты о работе флотилии..
18.	Настоль	**Информационная система КАФЕДРА**. Кафедре вуза тре-

	-ное	буется вести списки преподавателей и закрепленных за ними предметов и видов учебной нагрузки по этим предметам. Каждый преподаватель может выполнять разные виды учебной нагрузки по нескольким предметам, а по одному предмету разные виды учебной нагрузки могут проводиться несколькими преподавателями. Надо иметь возможность просматривать как список преподавателей по каждому предмету, так и перечень предметов (с видами учебных занятий) по каждому преподавателю. Требуется также хранить и распечатывать анкетные данные о преподавателях. Преподаватель может просматривать свою нагрузку по разным параметрам
19.	Веб	**Информационная система АГЕНТСТВО ПО ТРУДОУСТРОЙСТВУ.** Агентство по трудоустройству ведет списки лиц, ищущих работу, и списки вакансий. Вакансии поступают от организаций с указанием должности и оклада. В заявках претендентов, кроме анкетных данных, указываются желаемая должность и оклад. Каждая вакансия может заполнятся несколькими претендентами согласно их анкетным данным. Работодатель независимо от агентства отбирает одного из претендентов (или исключает всех), который и должен занять вакансию в базе данных агентства. После этого вакансия и претендент «аннулируются», т. е. они не должны в дальнейшем появляться в списках неудовлетворенных вакансий и претендентов.
20.	Веб	**Информационная система АУКЦИОНЫ.** Аукционная фирма занимается продажей с аукционов антикварных вещей и произведений искусства. Владельцы вещей, выставляемых на аукционах, юридически являются продавцами, а лица, приобретающие эти вещи, – покупателями. Получив от продавцов партию предметов, фирма решает, на котором из проводимых аукционов выставить конкретный предмет. Перед проведением очередного аукциона каждой из выставляемых на нем вещей присваивается отдельный номер лота. Две вещи, продаваемые на различных аукционах, могут иметь одинаковые номера лотов. В книгах фирмы делается запись о каждом аукционе: дата, время и место его проведения, о его специфике (например: картины до 1900 г., написанные маслом). Заносятся также сведения о каждом продаваемом предмете: аукцион, на который он заявлен, номер лота, продавец, начальная (стартовая) цена, краткое словесное описание. Продавцу разрешается выставлять любое количество вещей, и следить за их продажей (проданная цена), а покупатель имеет право приобретать онлайн или лично сколько ему угодно вещей. Одно и то же лицо или фирма может выступать и как продавец, и как покупатель.

		После аукциона служащие аукционной фирмы записывают фактическую цену, уплаченную за проданный предмет, и фиксируют данные покупателя.
21	Настольное	**Областной спорткомитет**. Команда клуба "Надежда" города принимает участие в соревнованиях женской баскетбольной суперлиги России. В этих соревнованиях участвуют порядка 12 команд из разных клубов и городов России. Участие в соревновании определяется документов, в котором указано: год проведения (2002 – 2003), клубы, участвующие в розыгрыше. Необходимо хранить информацию об клубах и участниках соревнований. Каждый клуб характеризуется следующей информацией: название, дата создания, город, спонсоры (ФИО, название организации, если это не частное лицо), главный тренер, который тренирует команду клуба в настоящее время (необходимо хранить историю о всех тренерах) – ФИО, возраст, звание. Также необходимо знать информацию о наличии залов клуба (название зала, адрес, вместимость, телефон, категория (низкая, средняя и т.п.), информацию о видах транспорта, предоставляемого клубом для перемещения участников соревнований (вид, вместимость). Необходимо также хранить информацию о всех участниках соревнований, которые в разное время играли за клуб – ФИО, дату рождения, звания, антропологические данные (дата, рост, вес), игровой номер, выполняемое амплуа. Эта информация может меняться с течением времени (игрок сменил амплуа, вырос), поэтому необходимо хранить историю.
22	Настольное	**Учреждение ГИБДД.** При проведении технического осмотра автомобиля необходимо фиксировать следующие данные: госномер автомобиля, проходящего технический осмотр, номер двигателя, цвет, марка, номер технического паспорта, номер водительского удостоверения, ФИО владельца, адрес прописки, год рождения, пол. Данные фиксируются на дату прохождения текущего осмотра, необходимо хранить историю осмотров – дата прохождения, результат. Необходимо также фиксировать ФИО, должность, звание сотрудника ГАИ, проводившего осмотр, заключение осмотра. Каждый день технический осмотр могут проходить много автомобилей, проводить осмотр могут разные сотрудники, но каждый сотрудник проводит за день не более 10 осмотров.
23	ВЕБ	**Провайдер, предоставляющий услуги доступа в Интернет в своем зале.** В БД заносится следующая информация о клиенте, пользующегося услугами Интернет: номер компьютера клиента, IP-адрес, дата, время начала соединения, окончания соединения, которые фиксируются автоматически при

		соединенияя. Цены за пользование услугами могут изменяться, эти данные хранятся в следующем виде: дата, стоимость одной минуты соединения, льготная стоимость с 20.00 до 2.00, льготная стоимость с 02.00 до 06.00. Для каждого абонента формируется квитанция об оплате, в которой содержится: название, адрес, телефон организации, выдавшей квитанцию, дата, время начала, окончания сеанса, количество минут, стоимость одной минуты, итоговая сумма, номер, ФИО оператора, выдавшего квитанцию, номер смены. В одной квитанции м.б. представлена информация о нескольких сеансах связи.
24	Веб	**«Комплектующие к станкам».** Предприятие, имеющее в своей структуре производственные участки и склады. Необходимо облегчить работу по учету комплектующих деталей, необходимых для ремонта и нормального функционирования станков предприятия. Каждый станок имеет номер, название (модель) и относится к определенному типу (токарные, фрезерные и т.п.). Необходимо фиксировать дату начала работы станка, эксплутационный срок и дату его списания. Каждому станку могут соответствовать разные комплектующие детали, каждая также имеет номер, название. Деталь получают со склада по накладной, в которой указано – с какого склада деталь получена, дата получения, цена детали на дату получения. На накладной расписывается ремонтник, производящий наладку и ремонт станка. Складов на предприятии м.б. несколько, каждый имеет номер, адрес (улица, номер дома), количество метров занимаемой площади.
25	Настольное	**«Кадры предприятия».** На предприятии существует ряд подразделений. Каждое подразделение имеет штатное расписание, в котором имеется перечень должностей. Каждая должность имеет название, краткое название, шифр, нижнюю и верхнюю границы разрядов единой тарифной сетки (от 1 до 18). Также известно, сколько единиц каждой должности выделено подразделению. О сотрудниках, работающих на предприятии, необходимо знать всю историю их перемещения – где, в каком подразделении работал сотрудник, на какой должности, какой имел разряд, дату начала и дату окончания работы. Также о сотруднике необходимо хранить личные данные: ФИО, возраст, пол, семейное положение.
26	Настольное	**«Банк данных товаров, производимых различными предприятиями»** (реклама). Рекламное предприятие. Необходимо хранить информацию о товаре, который производится предприятиями области – каждый товар имеет название, номер, относится к какой-либо группе товаров (канцелярские принадлежности, бумага, скобяные товары и т.п.). Цена товара

		меняется во времени и определяется позицией прайс-листа, выпускаемого периодически на предприятии, производящем товар. Предприятие характеризуется названием, имеет статистический код, адрес, телефон. Каждое предприятие может производит много товаров, и в тоже время один и тот же товар могут производить несколько предприятий. Также необходимо знать ФИО и должность руководителя предприятия, телефон отдела маркетинга предприятия, руководителя отдела маркетинга, ФИО контактного лица.
27	НАСТОЛЬНОЕ	**«Учет договоров страхования».** Предприятие – страховая организация. Страховая организация заключает договора с физическими лицами и юридическими организациями. Для организации оформляется коллективный договор, в котором перечислены страхуемые сотрудники: ФИО, возраст, категория риска (первая, вторая, высшая и т.п.). О предприятии хранится следующая информация: код, полное наименование, краткое наименование, адрес, банковские реквизиты (номер банка), специализация предприятия (медицинское учреждение, автотранспортное предприятие, учебное заведение и т.п.). В заключаемом коллективном договоре указывается дата заключения, срок договора (конец действия договора), сумма выплат по каждой категории сотрудников, выплаты по страховым случаям. Выплаты зависят от категории сотрудника. Необходимо также хранит информацию о страховом агенте, заключившем договор (ФИО, паспортные данные). Каждый агент может заключить много договоров, в каждом договоре м.б. оформлено несколько сотрудников. А каждый конкретный договор м.б. заключен только одним агентом.
28	ВЕБ	**«Учёт спроса и предложения».** Производственное предприятие, имеющее в структуре отдел маркетинга. Отдел маркетинга предприятия занимается спросом выпускаемого товара. Каждый товар характеризуется кодом, названием, категорией (промышленные, бытовые, торговое оборудование и т.п.). Продажа товара на предприятии осуществляется по накладным, в которых указано кому отправлен товар (юридическое или физическое лицо, название, имя, адрес, номер, серия документа, банковские реквизиты (номер и название банка). В накладной также указывается отпускная цена на текущую дату, количество отпущенного товара. Необходимо отслеживать название населенных пунктов, название региона России и страны ближнего или дальнего зарубежья куда отправлен товар. Каждая накладная соответствует одному пункту назначения и одному покупателю.
29	Веб	**«Банк данных насаждений парков».** Предприятие по бла-

		гоустройству парков. Предприятие оказывает такие виды услуг, как: формирование ландшафтов, насаждение парков, озеленение улиц и скверов. Фирма имеет название, юридический адрес. Каждый обслуживаемый парк делится на зоны. Каждому высаживаемому растению присваивается уникальный номер в пределах зоны. Необходимо хранить дату высадки растения и возраст растения. Растение м.б. высажено в парке в многолетнем возрасте. Каждое растение относится к какому-либо одному виду. Режим полив каждого растения зависит от возраста растения и его вида. Каждый полив характеризуется днем (каждый, один раз в неделю и т.п.) временем полива, нормой воды в литрах. Насаждения поливаются максимум один раз в день. Также необходимо иметь информацию о служителях парка, которых ухаживают за насаждениями (ФИО, телефон, адрес). Каждый служитель закрепляется за насаждением графиком (дата). на каждую дату закреплен за насаждением только один служитель. Также есть декораторы парка, о них необходимо хранить информацию о ФИО, телефоне, адресе, образовании, названием законченного учебного заведения, категорией (высшая, средняя и т.п.)
30	настольное	**«Музейные фонды».** Музейные предметы хранятся в музейных фондах. Существуют различные фонды: живопись, графика, икона, скульптура, декоративно-прикладное искусство (ДПИ), нумизматика, археология, рукописи и редкая книга и т.п. Для удобства работы в ряде фондов предусмотрены вспомогательные картотеки комплектов - сервизов и гарнитуров в ДПИ, альбомов в графике, иконостасов в древнерусском искусстве и т.п. Необходимо реализовать ведение карточек музейных предметов - инвентарный номер, название, дата создания, точно определена дата создания или приблизительно, авторах работы (только первый автор – ФИО, дата рождения, страна), выставки, в которых участвовал музейный предмет. Необходимо вести учет движения (прием на хранение, передача на выставку, возвращение с выставки, списание и т.п.) музейных предметов вне (знать информацию об организации, которой на время передается предмет – название, адрес, телефон, ФИО контактного лица, адрес где проводится выставка, название выставки, дата начала работы, дата окончания работы) и внутри музея (из фонда в фонд), осуществлять оформление актов движения. Акты подписывает руководитель музея и хранитель фонда, отвечающий за предметы в музейном фонде. Предметы могут передаваться как в составе целого комплектом, так и по отдельности.

Критерии оценивания работы

Таблица 3 -Требования и оценка курсовой работы

<table>
<tr><th></th><th>Требования</th><th colspan="3">оценка</th></tr>
<tr><td>1

2
3

4

5

6

7
8
9
10
11

12
13
14

15
16
17
18</td><td>Подробно и адекватно заданию выполнить описание предметной области,
Сформулировать цели разрабатываемой системы,
Создать словари предметной области, по Абботу и Объектно-ориентированный словарь,
Подробно описать функциональные и нефункциональные требования к разрабатываемой системе,
Подробно описать функциональную модель - цель, какой процесс описывает, точка зрения,
IDFE0 -модель должна содержать не менее трех уровней детализации,
Создать диаграмму, описывающую все сервисы системы,
Описать не менее трех сценариев использования (использовать и словесное описание и графическое),
Разработать техническое задание,
Разработать концептуальную модель хранилища данных,
Описать инструментарий, используемый при разработке программного обеспечения,
Разработать диаграмму интерфейсных классов.
Выполнить детальное описание трех ранее выбранных сценариев использования с описанием цели, входных и выходных параметров, с учетом спроектированных классов данных и интерфейсных классов (показать связи между этими компонентами системы);
Разработать макеты интерфейсных объектов,
Реализовать демо - версию разрабатываемого ПО.
Защита курсовой работы (ответить на все вопросы преподавателя, оформление документации по выполненной работе согласно требованиям стандартов ИРНИТУ).</td><td>3</td><td rowspan="10">4</td><td rowspan="11">5</td></tr>
<tr><td>19

20

21

22
23
24

25
26</td><td>Подробное словесное описание каждого уровня детализации IDEF0 модели.
Выполнить проектирование хранилища данных (логическая, физическая модели и их описание),
Описать поведения системы во времени,
Описать классы данных;
Описать не менее двух алгоритмов, по которым работают операции классов,
Описать архитектуры разрабатываемого ПО (фактическая организация модулей системы; модель развертывания и модель реализации процессов)
Реализовать не менее 3 функций разрабатываемого программного продукта.</td></tr>
<tr><td>27
28

29</td><td>Реализовать полную версию разрабатываемого программного продукта,
Создать не менее трех различных автотестов (модульный, функциональный, структурный, интерфейсный),
Разработать документацию пользователя.</td></tr>
</table>

Содержание отчета по курсовой работе

Отчет по курсовой работе сдается на проверку преподавателю, после которой он может быть возвращен на доработку или принят, в результате чего студент допускается (или не допускается) к защите курсовой работе. По результатам защиты и реализации курсовой работе выставляется оценка.

Отчет сдается в электронном виде и в виде документа вместе с носителем информации (диск, на котором записан

- отчет с подписью и оценкой преподавателя в форматах *.doc и *.pdf,
- исходный код разработанного приложения,
- исполняемый файл, разработанного приложения).

Отчет выполняется в соответствии со стандартами оформления документов в ИРНИТУ.

Список литературы:

1 Иванова Г.С. Технология программирования: Учебник. — М: Кнорус, 2010.
2 Лэнгсам Й., Огенстайн М., Тененбаум А. Структуры данных для персональных ЭВМ. — М: Мир,1989.-568с.
3 Диго С.М., Клешко Г.Н., Мишенин А.И., Петров Е.А. Сборник задач по курсу "Информационные системы и структуры данных". — М.:Статистика,1981. — 144с.
4 Вирт Н. Алгоритмы и структуры данных. -М: Мир,1989.-360с.
5 Буч Г. Объектно-ориентированное проектирование с примерами применения: Пер. С англ. — М.: Конкорд, 1992.- 519 с., ил.
6 Иванова Г.С., Ничушкина Т.Н., Пугачев Е.К. Объектно-ориентированное программирование: Учебник для вузов. — М.: МГТУ, 2007 – 368 с.
7 Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. – Питер, 2007
8 Орлов С.А. Технологии разработки программного обеспечения. Разработка сложных программных систем. Учебное пособие. 2-е изд. – СПб.: Питер, 2003.
9 Техника разработки программ : в 2 кн.: учеб. для вузов по направлениям "Информатика и вычислит. техника"... / Е. В. Крылов, В. А. Острейковский, Н. Г. Типикин Кн. 2 Технология, надежность и качество программного обеспечения, 2008. - 468 с. : а-ил
10 Брауде Э. Технология разработки программного обеспечения -СПб.: Питер, 2004. — 655 с.: ил.
11 Разработка программного обеспечения UML: https://dic.academic.ru/dic.nsf/ruwiki/30213 (Дата доступа 15.02.21)
12 Г.С. Иванова, Т.Н. Ничушкина Проектирование программного обеспечения Учебное пособие. М.: МВТУ-2002. С.74
13 Т. В. Черушева Проектирование программного обеспечения. Учебное пособие. Пенза. Издательство ПГУ .2014
14 Концепция: Архитектура программного обеспечения http://dit.isuct.ru/Publish_RUP/core.base_rup/guidances/concepts/software_architecture_4269A354.html (Дата доступа 15.02.21)

Учебное издание

Курсовая работа «Разработка прикладного программного обеспечения»
Руководство и методические указания для студентов направлений:
09.03.02 «Информационные системы и технологии»

Составитель
Бахвалова Зинаида Андреевна

Редактор З.А.Бахвалова