МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования   
**«Нижегородский государственный университет им. Н.И. Лобачевского»**

**Национальный исследовательский университет**

**Институт информационных технологий, математики и механики**

**Кафедра: Информатики и автоматизации научных исследований**

Направление подготовки: «Прикладная информатика»

Профиль подготовки: «Прикладная информатика в области принятия решений»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА**

**Тема:**

**«Проектирование и реализация системы обучения «Автоматизированное рабочее место проведения оценки качества программных продуктов (АРМ Эксперта)»**

Допущена к защите Выполнил: [ПИ - 381607м](https://vk.com/im?sel=c43)

Заведующий кафедрой: студент группы

Любимцев Дмитрий Олегович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель:

к.т.н. Банкрутенко Владимир Викторович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород  
2018

# 

# **Введение**

Тенденции развития современных информационных технологий приводят к постоянному возрастанию сложности информационных систем (ИС). Современные крупные проекты ИС, как правило, имеют сложное описание (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними), требующее тщательного моделирования и анализа данных и процессов.

Для успешной реализации проекта объект проектирования (ИС) должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые функциональные и информационные модели ИС. Накопленный к настоящему времени опыт проектирования ИС показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Однако до недавнего времени проектирование ИС выполнялось в основном на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС. Кроме того, в процессе создания и функционирования ИС информационные потребности пользователей могут изменяться или уточняться, что еще более усложняет разработку и сопровождение таких систем.

В 70-х и 80-х годах при разработке ИС достаточно широко применялась структурная методология, предоставляющая в распоряжение разработчиков строгие формализованные методы описания ИС и принимаемых технических решений. Она основана на наглядной графической технике: для описания различного рода моделей ИС используются схемы и диаграммы. Наглядность и строгость средств структурного анализа позволяла разработчикам и будущим пользователям системы с самого начала неформально участвовать в ее создании, обсуждать и закреплять понимание основных технических решений. Однако, широкое применение этой методологии и следование ее рекомендациям при разработке конкретных ИС, встречалось достаточно редко, поскольку при неавтоматизированной (ручной) разработке это практически невозможно. Действительно, вручную очень трудно разработать и графически представить строгие формальные спецификации системы, проверить их на полноту и непротиворечивость, и тем более изменить. Если все же удается создать строгую систему проектных документов, то ее переработка при появлении серьезных изменений практически неосуществима. Ручная разработка обычно порождала следующие проблемы:

* неадекватная спецификация требований;
* неспособность обнаруживать ошибки в проектных решениях;
* низкое качество документации, снижающее эксплуатационные качества;
* затяжной цикл и неудовлетворительные результаты тестирования.

С другой стороны, разработчики ИС исторически всегда стояли последними в ряду тех, кто использовал компьютерные технологии для повышения качества, надежности и производительности в своей собственной работе (феномен «сапожника без сапог») [6].

Перечисленные факторы способствовали появлению программно-технологических средств специального класса - CASE-средств, реализующих CASE-технологию создания и сопровождения ИС. Термин CASE (ComputerAidedSoftwareEngineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ИС в целом. Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы. CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС.

# **Глава 1. Обзор современных Case-средств.**

CASE-средства предназначены для реализации больших и средних проектов коллективами разработчиков и обычно базируются на охватывающих все этапы жизненного цикла этих систем, конкретных методологиях коллективной разработки и сопровождения информационных (программных) систем.

Основу современной CASE-технологии анализа и проектирования информационных систем составляют:

* поддержка всех этапов жизненного цикла ИС, начиная с самых общих описаний предметной области до получения и сопровождения программного продукта;
* методология структурного нисходящего анализа и проектирования, при которой разработка ИС представляется в виде последовательности четко определенных этапов;
* ориентация на реализацию приложений в архитектуре "клиент-сервер" с использованием всех особенностей современных серверов баз данных (включая декларативные ограничения целостности, хранимые процедуры, триггеры баз данных) и поддержкой в клиентской части всех современных стандартов и требований к графическому интерфейсу конечного пользователя;
* наличие централизованной базы данных - репозитория, обеспечивающего хранение моделей предметной области и спецификаций проекта прикладной системы на всех этапах ее разработки;
* возможность одновременной работы с репозиторием многих пользователей;
* централизованное хранение проекта системы и управление одновременным доступом к нему всех участников разработки;
* поддержка согласованности действий разработчиков, не допускающая ситуации, когда каждый аналитик или программист работает со своей версией проекта и модифицирует ее независимо от других;
* автоматизация последовательного перехода от одного этапа разработки к другому, использование специальных утилит, с помощью которых можно по спецификациям концептуального уровня автоматически получать первоначальные варианты спецификации уровня проектирования (описание структуры базы данных и состава программных модулей), а по последним, после всех необходимых уточнений и дополнений, автоматически генерировать готовые к выполнению программы;
* автоматизация стандартных действий по проектированию и реализации ИС, например, генерация многочисленных отчетов по содержимому репозитория, обеспечивающих полное документирование текущей версии системы на всех этапах ее разработки.

Реализация перечисленных технологических возможностей зависит от того, какая конкретная CASE-система используется коллективом аналитиков и разработчиков проекта. Ниже рассматриваются различные коммерческие CASE-системы.

## **1.1SILVERRUN**

CASE-средство Silverrun американской фирмы СomputerSystemsAdvisersInc. (CSA) используется для анализа и проектирования ИС бизнес-класса. Оно применимо для поддержки любой методологии, основанной на раздельном построении функциональной и информационной моделей (диаграмм потоков данных и диаграмм "сущность-связь"). Настройка на конкретную методологию обеспечивается выбором требуемой графической нотации моделей и набора правил проверки проектных спецификаций. В системе имеются готовые настройки для наиболее распространенных методологий: DATARUN (основная методология, поддерживаемая Silverrun), Gane/Sarson, Yourdon/DeMarco, Merise, Ward/Mellor, InformationEngineering. Для каждого понятия, введенного в проекте, имеется возможность добавления собственных описателей. Архитектура Silverrun позволяет наращивать среду разработки по мере необходимости.

Silverrun имеет модульную структуру и состоит из четырех модулей, каждый из которых является самостоятельным продуктом и может приобретаться, и использоваться без связи с остальными модулями:

* Модуль построения моделей бизнес-процессов в форме диаграмм потоков данных (BPM - BusinessProcessModeler).

*Позволяет моделировать функционирование обследуемой организации или создаваемой ИС. В модуле BPM обеспечена возможность работы с моделями большой сложности: автоматическая перенумерация, работа с деревом процессов (включая визуальное перетаскивание ветвей), отсоединение и присоединение частей модели для коллективной разработки. Диаграммы могут изображаться в нескольких предопределенных нотациях, включая Yourdon/DeMarco и Gane/Sarson. Имеется также возможность создавать собственные нотации, в том числе добавлять в число изображаемых на схеме дескрипторов определенные пользователем поля.*

* Модуль концептуального моделирования данных (ERX - Entity-RelationshipeXpert).

*Обеспечивает построение моделей данных "сущность-связь", не привязанных к конкретной реализации. Этот модуль имеет встроенную экспертную систему, позволяющую создать корректную нормализованную модель данных посредством ответов на содержательные вопросы о взаимосвязи данных. Возможно автоматическое построение модели данных из описаний структур данных. Анализ функциональных зависимостей атрибутов дает возможность проверить соответствие модели требованиям третьей нормальной формы и обеспечить их выполнение. Проверенная модель передается в модуль RDM.*

* Модуль реляционного моделирования (RDM - RelationalDataModeler).

*Позволяет создавать детализированные модели "сущность-связь", предназначенные для реализации в реляционной базе данных. В этом модуле документируются все конструкции, связанные с построением базы данных. Гибкая изменяемая нотация и расширяемость репозитория позволяют работать по любой методологии. Возможность создавать подсхемы соответствует подходу ANSI SPARC к представлению схемы базы данных. На языке подсхем моделируются как узлы распределенной обработки, так и пользовательские представления. Этот модуль обеспечивает проектирование и полное документирование реляционных баз данных.*

* Менеджер репозитория рабочей группы (WRM - WorkgroupRepositoryManager).

*Применяется как словарь данных для хранения общей для всех моделей информации, а также обеспечивает интеграцию модулей Silverrun в единую среду проектирования.*

Платой за высокую гибкость и разнообразие изобразительных средств построения моделей является такой недостаток Silverrun, как отсутствие жесткого взаимного контроля между компонентами различных моделей.

Для автоматической генерации схем баз данных у Silverrun существуют мосты к наиболее распространенным СУБД: Oracle, Informix, DB2, Ingres, Progress, SQL Server, SQLBase, Sybase. Для передачи данных в средства разработки приложений имеются мосты к языкам 4GL: JAM, PowerBuilder, SQL Windows, Uniface, NewEra, Delphi. Все мосты позволяют загрузить в Silverrun RDM информацию из каталогов, соответствующих СУБД или языков 4GL. Это позволяет документировать, перепроектировать или переносить на новые платформы уже находящиеся в эксплуатации базы данных и прикладные системы. При использовании моста Silverrun расширяет свой внутренний репозиторий специфичными для целевой системы атрибутами. При создании приложения на языке 4GL данные, перенесенные из репозиторияSilverrun, используются либо для автоматической генерации интерфейсных объектов, либо для быстрого их создания вручную.

Для обмена данными с другими средствами автоматизации проектирования, создания специализированных процедур анализа и проверки проектных спецификаций, составления специализированных отчетов в соответствии с различными стандартами в системе Silverrun имеется три способа выдачи проектной информации во внешние файлы:

* Система отчетов.

*Можно, определив содержимое отчета по репозиторию, выдать отчет в текстовый файл. Этот файл можно затем загрузить в текстовый редактор или включить в другой отчет.*

* Система экспорта/импорта.

*Для более полного контроля над структурой файлов в системе экспорта/импорта имеется возможность определять не только содержимое экспортного файла, но и разделители записей, полей в записях, маркеры начала и конца текстовых полей. Файлы с указанной структурой можно не только формировать, но и загружать в репозиторий. Это дает возможность обмениваться данными с различными системами: другими CASE-средствами,СУБД, текстовыми редакторами и электронными таблицами.*

* Хранение репозитория во внешних файлах через ODBC-драйверы.

*Для доступа к данным репозитория из наиболее распространенных систем управления базами данных обеспечена возможность хранить всю проектную информацию непосредственно в формате этих СУБД.*

В системе Silverrun поддерживается групповая работа. Используются два способа:

* В стандартной однопользовательской версии имеется механизм контролируемого разделения и слияния моделей. Разделив модель на части, можно раздать их нескольким разработчикам. После детальной доработки модели объединяются в единые спецификации.
* Сетевая версия Silverrun позволяет осуществлять одновременную групповую работу с моделями, хранящимися в сетевом репозитории на базе СУБД Oracle, Sybase или Informix. При этом несколько разработчиков могут работать с одной и той же моделью, так как блокировка объектов происходит на уровне отдельных элементов модели.

Имеются реализации Silverrun трех платформ - MS Windows, Macintosh и OS/2 PresentationManager - с возможностью обмена проектными данными между ними.

## **1.2 ProKit\*Workbench**

Система Prokit\*Workbench одна из первых интегрированных CASE-систем, появишаяся на российском рынке. Данная CASE-система, включает следующие этапы жизненного цикла программного продукта: планирование, проектирование и создание самого программного продукта. Данная система обеспечивала пошаговый процесс проектирования на стадиях жизненного цикла системы и включала подсистемы:

* Analyzer.

*Определяет прикладные функции, поддерживает многоуровневую детализацию, включает схемы потоков данных;*

* DataModeler.

*Определяет данные, поддерживает моделирование данных, включает модели типа "сущность-связь”;*

* Prototyper.

*Позволяет выявить, исследовать и специфицировать требования в прикладной системе, обеспечивает получение работающего прототипа, обеспечивает использование результатовпрототипирования на последующих стадиях разработки;*

* Designer.

*Обеспечивает переход от анализа к проекту, включает схемы потоков данных и структурные схемы, поддерживает разработку программных спецификаций с учетом особенностей реализации на языках программирования;*

* Repository.

*В максимальной степени поддерживает динамические связи графическими моделями, содержит всю проектную информацию, обеспечивает возможность экспорта и импорта данных;*

* Reports.

*Обеспечивает выдачу информации, содержащейся в репозитории и на схемах, дает возможность пользователю получать стандартные и нестандартные отчеты с использованием макросов, поддерживает весь жизненный цикл системы;*

* Import/Export.

*Дает возможность передавать данные как в рамках одного проекта, так и между проектами, обеспечивает возможность разработки проекта на многих рабочих станциях, создает дополнительные возможности для управления данными проекта.*

Analyzer является основной подсистемой Prokit\*Workbench, которая помогает разработчику программного обеспечения собрать и запомнить системные требования, используя схемы потоков данных и поддерживающий проект репозиторий.

Схемы потоков данных формируются с помощью четырех символов:

* **Внешний объект** (определяет границы системы, служит для представления источника и/или получателя информации)
* **Процесс** (осуществляет преобразование входных данных в выходные, служит для описания функции, может быть детализирован (раскрыт) в виде другой схемы потоков данных)
* **Хранилище данных** служит для представления места, где должны содержаться данные (между процессами (в интервалах между обработкой), для отражения свойств внешнего объекта)
* **Поток данных** (служит для представления перемещения данных при помощи определенной конфигурации символов (диаграммы), изображается с одним источником и одним получателем).

## **1.3 RationalRose**

RationalRose - CASE-средство фирмы RationalSoftwareCorporation (США) - предназначено для автоматизации этапов анализа и проектирования ПО, а также для генерации кодов на различных языках и выпуска проектной документации. RationalRose использует синтез-методологию объектно-ориентированного анализа и проектирования, основанную на подходах трех ведущих специалистов в данной области: Буча, Рамбо и Джекобсона. Разработанная ими универсальная нотация для моделирования объектов (UML - UnifiedModelingLanguage) претендует на роль стандарта в области объектно-ориентированного анализа и проектирования. Конкретный вариант RationalRose определяется языком, на котором генерируются коды программ (C++, Smalltalk, PowerBuilder, Ada, SQLWindows и ObjectPro). Основной вариант - RationalRose/C++ - позволяет разрабатывать проектную документацию в виде диаграмм и спецификаций, а также генерировать программные коды на С++. Кроме того, RationalRose содержит средства реинжиниринга программ, обеспечивающие повторное использование программных компонент в новых проектах.

В основе работы RationalRose лежит построение различного рода диаграмм и спецификаций, определяющих логическую и физическую структуры модели, ее статические и динамические аспекты. В их число входят диаграммы классов, состояний, сценариев, модулей, процессов.

В составе RationalRose можно выделить 6 основных структурных компонент: репозиторий, графический интерфейс пользователя, средства просмотра проекта (browser), средства контроля проекта, средства сбора статистики и генератор документов. К ним добавляются генератор кодов (индивидуальный для каждого языка) и анализатор для С++, обеспечивающий реинжиниринг - восстановление модели проекта по исходным текстам программ.

Репозиторий представляет собой объектно-ориентированную базу данных. Средства просмотра обеспечивают "навигацию" по проекту, в том числе, перемещение по иерархиям классов и подсистем, переключение от одного вида диаграмм к другому и т. д. Средства контроля и сбора статистики дают возможность находить и устранять ошибки по мере развития проекта, а не после завершения его описания. Генератор отчетов формирует тексты выходных документов на основе содержащейся в репозитории информации.

Средства автоматической генерации кодов программ на языке С++, используя информацию, содержащуюся в логической и физической моделях проекта, формируют файлы заголовков и файлы описаний классов и объектов. Создаваемый таким образом скелет программы может быть уточнен путем прямого программирования на языке С++. Анализатор кодов С++ реализован в виде отдельного программного модуля. Его назначение состоит в том, чтобы создавать модули проектов в форме RationalRose на основе информации, содержащейся в определяемых пользователем исходных текстах на С++. В процессе работы анализатор осуществляет контроль правильности исходных текстов и диагностику ошибок. Модель, полученная в результате его работы, может целиком или фрагментарно использоваться в различных проектах. Анализатор обладает широкими возможностями настройки по входу и выходу. Например, можно определить типы исходных файлов, базовый компилятор, задать, какая информация должна быть включена в формируемую модель и какие элементы выходной модели следует выводить на экран. Таким образом, RationalRose/С++ обеспечивает возможность повторного использования программных компонент.

В результате разработки проекта с помощью CASE-средства RationalRose формируются следующие документы:

* диаграммы классов;
* диаграммы состояний;
* диаграммы сценариев;
* диаграммы модулей;
* диаграммы процессов;
* спецификации классов, объектов, атрибутов и операций
* заготовки текстов программ;
* модель разрабатываемой программной системы.

Последний из перечисленных документов является текстовым файлом, содержащим всю необходимую информацию о проекте (в том числе необходимую для получения всех диаграмм и спецификаций).

Тексты программ являются заготовками для последующей работы программистов. Они формируются в рабочем каталоге в виде файлов типов .h (заголовки, содержащие описания классов) и .cpp (заготовки программ для методов). Система включает в программные файлы собственные комментарии. Состав информации, включаемой в программные файлы, определяется либо по умолчанию, либо по усмотрению пользователя. В дальнейшем эти исходные тексты развиваются программистами в полноценные программы.

RationalRose интегрируется со средством PVCS для организации групповой работы и управления проектом и со средством SoDA - для документирования проектов. Интеграция RationalRose и SoDA обеспечивается средствами SoDA.

Для организации групповой работы в RationalRose возможно разбиение модели на управляемые подмодели. Каждая из них независимо сохраняется на диске или загружается в модель. В качестве подмодели может выступать категория классов или подсистема.

Для управляемой подмодели предусмотрены операции:

* загрузка подмодели в память;
* выгрузка подмодели из памяти;
* сохранение подмодели на диске в виде отдельного файла;
* установка защиты от модификации;
* замена подмодели в памяти на новую.

Наиболее эффективно групповая работа организуется при интеграции RationalRose со специальными средствами управления конфигурацией и контроля версий (PVCS). В этом случае защита от модификации устанавливается на все управляемые подмодели, кроме тех, которые выделены конкретному разработчику. В этом случае признак защиты от записи устанавливается для файлов, которые содержат подмодели, поэтому при считывании "чужих" подмоделей защита их от модификации сохраняется и случайные воздействия окажутся невозможными.

RationalRose функционирует на различных платформах: IBM PC (в среде Windows), Sun SPARC stations (UNIX, Solaris, SunOS), Hewlett-Packard (HP UX), IBM RS/6000 (AIX).

## **1.4 Bpwin**

Мировой лидер в области CASE-технологии предлагает мощное средство системного анализа деловой и производственной активности, позволяющее отслеживать соответствие структуры бизнеса, документооборота, финансовых потоков жестким и динамичным требованиям современной экономики. Система BPwin поможет повысить конкурентоспособность, оптимизировать процессы управления. Результатом использования BPwin является исключение излишних и бесполезных действий, снижение затрат, повышение гибкости и эффективности. BPwin - это незаменимый инструмент менеджеров и бизнес-аналитиков, а начиная с версии 1.8, в которую включена поддержка диаграмм потоков данных и методики IDEF3 (BPwinProfessional), становится в руках системных аналитиков и разработчиков и мощным средством моделирования процессов при создании корпоративных информационных систем.

BPwin обладает интуитивно-понятным графическим интерфейсом, быстро и легко осваивается, что позволяет сосредоточиться на анализе самой предметной области, не отвлекаясь на изучение инструментальных средств. BPwin помогает быстро создавать и анализировать модели с целью оптимизации деловых и производственных процессов. Применение универсального графического языка бизнес-моделирования IDEF0 обеспечивает логическую целостность и полноту описания, необходимую для достижения точных и непротиворечивых результатов.

Посредством набора графических инструментов для отображения действий и объектов BPwin позволяет легко построить схему процесса, на которой показаны исходные данные, результаты операций, ресурсы, необходимые для их выполнения, управляющие воздействия, взаимные связи между отдельными работами.

Интерактивное выделение объектов обеспечивает постоянную визуальную обратную связь при построении модели. BРwin поддерживает ссылочную целостность, не допуская определения некорректных связей и гарантируя непротиворечивость отношений между объектами при моделировании.

Встроенный механизм вычисления стоимости позволяет оценивать и анализировать затраты на осуществление различных видов деловой активности Механизм вычисления расходов на основе выполняемых действий (Activity-BasedCosting, ABC) - это технология, применяемая для оценки затрат и используемых ресурсов. Она помогает распознать и выделить наиболее дорогостоящие операции для дальнейшего анализа. BPwin может генерировать отчеты непосредственно в формате MS Excel и Word для последующей обработки и использования в других приложениях. Связь с ERwin (моделирование данных в стандарте IDEF1X) позволяет сократить время проектирования и разработки сложных информационных систем. Для системных аналитиков тесная интеграция BРwin с инструментом проектирования баз данных открывает уникальные возможности по созданию действительно комплексных систем, в которых ERwin служит для описания информационных объектов системы, в то время как BPwin отражает функциональные особенности предметной области. Связывая сущности и атрибуты модели данных с информацией о выполняемых действиях, Вы можете продолжить анализ процессов на новом уровне с одновременной перекрестной проверкой моделей процессов и данных.

Основные характеристики BPwin:

* Развитая методология функционального моделирования на основе IDEF0
* Мощные редакторы для описания операций, связей и вычисления затрат на выполнение работ
* Иерархическая структура диаграмм, облегчающая последовательное уточнение элементов модели
* Контекстные диаграммы для описания границ системы, области действия, назначения объектов
* Декомпозиционные диаграммы для описания особенностей взаимодействия различных процессов
* Расширенные возможности по поддержанию ссылочной целостности
* Поддержка методологии IDEF3
* Экспорт моделей в средства имитационного моделирования
* Интеграция и связь со средством проектирования баз данных ERwin (методология IDEF1X)
* Поддержка свойств, определяемых пользователем. Описание моделей может быть расширено за счет свойств, определяемых пользователем, включая мультимедийные документы.
* Интеграцияс Logic Works ModelMart. Сервер приложений для программных продуктов LogicWorks, ModelMart поддерживает мощный набор инструментальных программных средств, обеспечивающих совместное (групповое) проектирование и разработку программных систем, включая механизмы объединения моделей и анализа изменений, контроль версий, возможность создания "компонент" модели и т.д. Для организации хранилища моделей в ModelMart используются СУБД на платформах Oracle, Sybase, Informix или SQL Server. Кроме того, поддерживаются прямые связи ModelMart с ERwin и BPwin.
* Удобный интерфейс пользователя. В распоряжении пользователей имеется проводник, ставший привычным в среде Windows 95/NT, позволяющий легко переходить с одной диаграммы на другую простым перемещением по "дереву" проводника.
* Расширенная архитектура. BPwin поддерживает 16- и 32-х разрядные системы, позволяя организовать совместную работу для всех участников проекта.
* Автоматическая поддержка изменения размеров. BPwin поддерживает автоматическую настройку размеров диаграмм и возможность изменения масштабов изображения моделей, таких как BPwin, ERwin и ModelMartSinchronizer.

## **1.5 PowerDesigner**

PowerDesigner (бывший S-Designor, принадлежавший компании PowerSoft) представляет собой инструмент, в состав которого входят средство создания концептуальных (то есть логических) моделей, средство создания физических моделей и средство объектно-ориентированного моделирования, используемое при генерации клиентских приложений. Средство создания физических моделей представляет собой отдельный продукт — PowerDesignerPhysicalArchitect. В состав продукта PowerDesignerDataArchitect входят средства создания концептуальных и физических моделей, в состав PowerDesignerDeveloper — средства объектно-ориентированного моделирования и создания физических моделей, а в состав PowerDesignerObjectArchitect — все три средства.

Физические и концептуальные модели в PowerDesignerDataArchitect хранятся в разных файлах, однако возможна генерация как физической модели на основе модели концептуальной, так и наоборот.

Помимо серверных СУБД производства Sybase (AdaptiveServerEnterprise 12.0, Sybase SQL Anywhere) PowerDesignerDataArchitect способен работать с любыми ODBC-источниками. Как и ERwin, он поддерживает генерацию триггеров серверных СУБД, осуществляющих стандартную обработку событий, связанных с нарушениями ссылочной целостности.

PowerDesignerDeveloper и PowerDesignerObjectArchitect могут генерировать код клиентских приложений для PowerBuilder, а также классы Java и компоненты JavaBeans. Возможно и обратное проектирование диаграмм классов из исходных текстов Java, байт-кодов и архивов Java. Поддерживается также генерация кода Web-приложений и объектов для SybaseEnterpriseApplicationServer на основе физической модели.

PowerDesignerDataArchitect может импортировать логические и физические модели ERwin.

PowerDesignerDataArchitect может хранить свои модели данных в коллективно разделяемом репозитарии, управляемом с помощью средства PowerDesignerMetaWorks и доступном как дополнительный модуль в составе любого из перечисленных выше продуктов.

## 1.6 ER/Studio

ER/Studio менееизвестенвнашейстране, чемERwinиPowerDesignerDataArchitect. Однако возможности этого продукта также заслуживают внимания.

По своему назначению этот продукт сходен с ERwin — он представляет собой специализированное средство проектирования данных и не содержит в своем составе инструментов для объектно-ориентированного моделирования или моделирования бизнес-процессов. Список поддерживаемых СУБД у этого продукта достаточно широк и включает все наиболее популярные серверные и настольные СУБД. В отличие от ERwin последняя версия этого продукта корректно поддерживает новые типы данных SQL Server 7.

ER/Studio поддерживает написание макросов на SAX Basic (клон VisualBasicforApplications). Этот язык позволяет создавать макросы для выполнения однотипных операций, например, добавления стандартных полей к вновь создаваемым сущностям. С помощью этого же языка можно генерировать стандартные триггеры и хранимые процедуры для вставки, удаления, изменения записей. Код на этом языке можно даже отлаживать и обращаться к свойствам сущностей для конструирования серверного кода. Однако, в отличие от ERwin, ER/Studio не позволяет добавить к каждой таблице свои шаблоны триггеров или просмотреть код конкретного триггера в процессе разработки модели — чтобы получить код одного триггера, нужно сгенерировать скрипт для всей модели.

Модели ER/Studio можно сохранить не только в виде DDL-скрипта, но и в формате XML. Можно также создать репозитарий для их хранения в любой серверной СУБД. ER/Studio может импортировать модели ERwin, но при импорте теряются связи шаблонов серверного кода с конкретными таблицами, и не все макросы ERwin корректно преобразуются в макросы SAX Basic.

ER/Studio позволяет сгенерировать Java-классы для клиентских приложений.

ER/Studio является COM-сервером, что позволяет использовать его в других приложениях, предоставляя им возможность просмотра и редактирования моделей данных, а также создавать другие решения на его основе.

## **1.7 SystemArchitect**

SystemArchitect 2001 представляет собой универсальное CASE-средство, позволяющее осуществить не только проектирование данных, но и структурное моделирование. Средство проектирования данных и создания ER-диаграмм является одной из составных частей этого продукта.

Этот продукт поддерживает СУБД практически всех ведущих производителей, включая Oracle (Oracle 8), Sybase, DB2, SQL Server, IBM (AS400, DB2), Informix, Sybase, Access, dBASE, Paradox и др.

В процессе логического моделирования можно проверить модель на соответствие правилам проектирования данных (например, на соответствие ее первой, второй или третьей нормальным формам). При генерации DDL-скрипта можно сгенерировать триггеры (в том числе и нестандартные).

Все компоненты SystemArchitect позволяют документировать процесс работы над проектом, включая техническое задание, план тестирования и др.

Модели SystemArchitect 2001, как и в случае других CASE-средств, можно сохранять в репозитарии. Однако в отличие от традиционных репозитариев, обладающих более или менее стандартной структурой хранимых данных, репозитарийSystemArchitect является настраиваемым — к сохраняемым объектам можно добавлять дополнительные свойства, определенные пользователем.

SystemArchitect обладает встроенным VisualBasicforApplication, что позволяет создавать разнообразные решения на базе этого продукта, включая автоматическую генерацию моделей и проектной документации.

SystemArchitect 2001 позволяет генерировать код клиентских приложений для VisualBasic, Delphi и PowerBuilder (на сегодняшний день это практически единственное CASE-средство, поддерживающее генерацию кода Delphi), классы C++, а также код и текстовые экранные формы COBOL.

## **1.8 VisioEnterprise**

Продукт под названием Visio, приобретенный в январе 2000 года корпорацией Microsoft вместе с его разработчиком — компанией VisioCorporation, позиционировался на рынке как одно из самых популярных средств создания схем и диаграмм. То, что один из членов семейства MicrosoftVisio 2000 — Visio 2000 Enterprise — содержит в своем составе полноценное CASE-средство, было в определенной степени сюрпризом для пользователей CASE-инструментов. Однако, если вдуматься, появление своих средств проектирования данных, моделирования бизнес-процессов и объектно-ориентированного моделирования у Microsoft — шаг вполне закономерный, поскольку такие средства появляются рано или поздно у большинства производителей популярных серверных СУБД и средств разработки, каковым Microsoft является уже довольно давно.

Как и подавляющее большинство средств проектирования данных, VisioEnterprise позволяет производить прямое и обратное проектирование данных, преобразовывать логическую модель в физическую. Этим средством поддерживаются все ODBC- и OLE DB-источники данных. С его помощью можно создавать триггеры для стандартной обработки нарушений ссылочной целостности в случае, если DDL-скрипт создается для Microsoft SQL Server, и серверные ограничения, если скрипт создается для другой СУБД. Можно отметить, что Visio при генерации скриптов позволяет указывать параметры организации физической памяти Oracle, Informix, Microsoft SQL Server, DB2 и некоторых других СУБД.

Помимо средств проектирования данных Visio включает средства объектно-ориентированного моделирования и генерации кода приложений VisualBasic 6, а также классов C++ и Java. Модели Visio можно сохранять в MicrosoftRepository.

Visio, в отличие от специализированных средств проектирования данных, не обладает скриптовым языком, позволяющим создавать серверный код, не связанный с конкретной СУБД. При использовании этого продукта такой код нужно создавать на этапе физического проектирования в уже созданном скрипте. К тому же этот продукт является сервером автоматизации, обладает весьма обширной объектной моделью и встроенным средством разработки — VisualBasicforApplications, что позволяет, создавать на его базе разнообразные решения, в том числе и автоматизировать разработку моделей данных.

## **1.9 VisibleAnalystWorkbench**

Система VisibleAnalystWorkbench (VAW) относится к сетевым многопользовательским CASE-системам, предназначенным для поддержки процесса создания ИС от этапов анализа текущей деятельности системы управления предприятия до создания законченных моделей ее реорганизованной деятельности, а также разработки конечных приложений в технологии "клиент-сервер". Продукт реализует широкий набор методов структурного системного анализа.

Система обеспечивает хранение всех моделей и спецификаций, относящихся к проекту прикладной системы и возникающих на различных этапах ее жизненного цикла, в централизованной базе данных - репозитории. Работа над проектом во многом сводится к работе с этой базой данных - вводу и коррекции различных описаний, поверке их согласованности и полноты, преобразованию одних моделей в другие и т.д.

Средства управления репозиторием с помощью удобного интерфейса реализуют административные функции управления, включая создание и удаление приложений, управление доступом к данным со стороны различных пользователей, предоставление прав одному приложению использовать часть спецификаций другого, экспорт и импорт отдельного приложения или всего репозитория.

Реализована возможность работы над одним проектом коллектива разработчиков. Используемый VAW механизм блокировки, гарантирует, что разработчик не может изменить какой-либо элемент проекта, пока с ним работает другой разработчик. Каждый пользователь имеет возможность самостоятельно определить пароль доступа к своей части проекта.

VAW снабжен мощными встроенными средствами импорта-экспорта. Поддерживается связь с такими популярными средами проектирования конечных приложений, как PowerBuilder и SQL Windows. Кроме того, VAW располагает стандартным форматом экспорта-импорта данных, который позволяет получить доступ к данным, хранящихся в репозитории, из других систем.

Также VAW с помощью встроенных средств может взаимодействовать с такими СУБД, как SQLBase, Oracle, Sybase, а также возможно проведение реинжиниринг баз данных, поддерживаемых этими СУБД.

VAW поддерживает следующие нотации для построения моделей: Yordon, Gane&Sarson, SDM, IE и имеет мощные средства проверки согласованности и корректности создаваемых моделей. Система может проверять диаграммы на соответствие синтаксису, осуществляет проверку правильности ключевых атрибутов сущностей, поиск не используемых элементов и т.д. VAW также имеет средства проверки моделей на сбалансированность и синхронность используемых данных в DF- и ER-диаграммах одного проекта.

В соответствии с общей технологией и архитектурой CASE-систем при работе с VAW выделяются следующие основные этапы процесса разработки информационной системы:

* моделирование и анализ функционирования существующей системы управления.

*Целью этапа является построение моделей существующих процессов управления, выявление их недостатков и возможных источников усовершенствования. Содержанием работы проектировщиков является построение моделей двух типов - функциональной, представленной в виде диаграмм функциональной декомпозиции, и поведенческой, представленной в виде диаграмм потоков данных, описывающих особенности процессов обработки данных при реализации соответствующих функций управления.*

* построение и анализ информационной модели предметной области.

*На данном этапе осуществляется детальное информационное моделирование существующей системы управления, описывающее информационные потребности предприятия. Результатом является информационная модель системы управления, отображающая ее единое информационное пространство.*

* проектирование концептуальной модели базы данных новой системы.

*Содержанием работ, проводимых на этапе концептуального проектирования базы данных, является построение формализованного информационного описания будущей предметной области. Этот этап является одним из ключевых в создании ИС. Конечным результатом этапа является СУБД-независимое описание базы данных. Достижение этого результата предполагает построение концептуальной модели базы.*

* проектирование функциональной структуры новой системы.

*Анализ полученных при выполнении предыдущих этапов результатов позволяет перейти к проектированию функциональной структуры новой информационной системы. Результаты этих работ представляются в виде диаграмм функциональной декомпозиции, описывающих функциональные возможности создаваемой системы.*

* проектирование процессов обработки данных.

*Результатом этапа является описание особенностей организации процессов обработки данных в новой информационной системе. Фиксация результатов проектирования на этом этапе осуществляется в виде диаграмм потоков данных. Основой для их построения служат, как правило, диаграммы потоков данных, полученные на этапе анализа функционирования существующей системы управления.*

* проектирование и реализация приложений.

*Заключительными этапами проектирования являются проектирование и реализация пользовательских приложений. В процессе реализации пользовательских приложений создаются программы, отвечающие всем требованиям проектных спецификаций.*

# **Глава 2. Проектирование и реализация системы обучения «Автоматизированное рабочее место проведения оценки качества программных продуктов (АРМ Эксперта)»**

### **2.1 Назначение информационной системы «АРМ Эксперта»**

Информационная система «АРМ Эксперта» предназначена для проведения оценки качества программных продуктов в соответствии с различными нормативными документами по оценке качества программных продуктов (ГОСТ 28195-89 и ГОСТ ИСО/МЭК 9126-93). Основное назначение данной информационной системы – использование студентами, изучающими предмет «Разработка и стандартизация программных средств и информационных технологий» на кафедре ИАНИ института ИТММ.

**2.2 Постановка задачи**

На данный момент в терминал классах кафедры ИАНИ установлена информационная система АРМ Эксперта 1995 года. С течением времени система морально устарела. Кроме того, АРМ Эксперта изначально создавался не для обучения, а для работы центров по сертификации программных продуктов.

В связи с этим изначально была поставлена задача спроектировать и реализовать аналог АРМ Эксперта с использованием современных средств разработки. Задача была выполнена, после чего было принято решение о создании фактически новой системы, которая будет являться уже не просто средством оценки качества ПП, но системой обучения проведения оценки качества.

Таким образом на текущий момент поставлена следующая задача:

**Спроектировать и реализовать систему обучения на основе АРМ Эксперта.**

**2.3 Проектирование**

**2.3.1 Техническое задание**

Техническое задание на систему обучения АРМ Эксперта представлено в Приложении 1.

**2.3.1 Диаграмма потоков данных (DFD)**

Диаграмма потоков данных является основным средством моделирования функциональных требований проектируемой системы. С их помощью эти требования разбиваются на функциональные компоненты и представляются в виде сети, связанной потоками данных. Главная цель таких средств - продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношение между этими процессами.

Основной сценарий работы выглядит так:

В начале нового семестра, преподаватель, получив список группы, регистрирует группу в системе. Для этого он входит в систему под своим логином и паролем. Для более эффективного проведения практических занятий, студенты разбиваются на группы по 2-3 человека и регистрируются в системе. Для каждой группы преподаватель имеет возможность назначить отдельное задание. Студенты после входа в систему, получают одну из базовых моделей, представляющих собой один из ГОСТов. На практических занятиях, задачей каждой рабочей группы является выбрать себе определённый программный продукт и оценить с помощью АРМ Эксперта. При этом в качестве задания может быть указано использование для оценки только часть показателей ГОСТа. В режиме редактирования студенты могут удалить или же наоборот добавить пункты в модель оценки. Таким образом, каждая группа создаёт для себя уникальную модель оценки. Для каждого пункта созданной модели группа выставляет коэффициент, соответствующий важности этого пункта для оценки ПП. После того, как модель оценки будет сформирована и все необходимые коэффициенты установлены, учебная группа переходит в режим Эксперта. В этом режиме проводится непосредственно оценка заданного ПП. Результатом процедуры оценки является итоговый отчёт, который генерирует АРМ Эксперта. Анализ отчета позволяет сформулировать вывод о качестве ПП с точки зрения выбранный модели оценки. Группа, получившая отчёт и готовая к сдаче работы информирует преподавателя нажатием кнопки «ГОТОВО». В результате чего преподаватель получает оповещение, что группа готова. Преподаватель удаленно открывает модель группы и осуществляет ее проверку. При возникновении вопросов к качеству модели или отчета, преподаватель отменяет готовность группы и отправляет ей свои комментарии. За успешное выполнение задания группе выставляется оценка, о чем сигнализирует соответствующее оповещение.

Проектирование диаграммы потоков данных является итерационным процессом, что и было подтверждено при создании приведенных ниже диаграмм. Это особенно заметно проявляется при проектировании диаграмм для больших задач.

Диаграмма потоков данных спроектированная на основе этой схемы представлена ниже

### **2.3.2 Диаграмма «сущность-связь»**

Диаграммы “Сущность-связь” (ERD) предназначены для разработки моделей данных и обеспечивают стандартный способ определения данных и отношений между ними. Фактически с помощью ERD осуществляется детализация хранилищ данных проектируемой системы, а также документируются сущности системы и способы их взаимодействия, включая идентификацию объектов, важных для предметной области (сущностей), свойств этих объектов (атрибутов) и их отношений с другими объектами (связей).

На основе схемы приведенной в предыдущем разделе спроектируем диаграмму «Сущность-связь», для этого выделим сущности:

* Заказчик
* Руководитель испытательного центра
* Программный продукт
* Очередь заявок
* Программа испытаний
* Протокол испытаний
* Проверка области аккредитации
* Сертификат

А также ассоциативная сущность:

* ГОСТы из области аккредитации

Диаграмма «Сущность-связь» спроектированная на основе этих сущностей и связей между ними представлена ниже:

## **2.4 Реализация**

Для реализации ядра программы был выбран язык C#, интерфейс реализован на языке XAML. Для взаимодействия с сервером используется язык SQL и TSQL. Разработка велась в Visual Studio 2015 с использованием библиотек .NetFramework, а так же библиотеки ITextSharp для формирования отчёта в формате PDF.

**2.4.1 Краткий обзор используемых средств разработки**

**2.4.1.1 C#**

C# — простой, современный объектно-ориентированный и типобезопасный язык программирования. C# относится к широко известному семейству языков C.

C# является объектно-ориентированным языком, но поддерживает также и компонентно-ориентированное программирование. Разработка современных приложений все больше тяготеет к созданию программных компонентов в форме автономных и самоописательных пакетов, реализующих отдельные функциональные возможности. Важная особенность таких компонентов — это модель программирования на основе свойств, методов и событий. Каждый компонент имеет атрибуты, предоставляющие декларативные сведения о компоненте, а также встроенные элементы документации. C# предоставляет языковые конструкции, непосредственно поддерживающие такую концепцию работы. Благодаря этому C# отлично подходит для создания и применения программных компонентов.

C# обладает рядом функций, обеспечивающих надежность и устойчивость приложений: сборка мусора автоматически освобождает память, занятую уничтоженными и неиспользуемыми объектами; обработка исключений дает структурированный и расширяемый способ выявлять и обрабатывать ошибки; строгая типизация языка не позволяет обращаться к неинициализированным переменным, выходить за пределы массива или выполнять неконтролируемое приведение типов.

В C# существует единая система типов. Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Таким образом, все типы используют общий набор операций, и значения любого типа можно хранить, передавать и обрабатывать схожим образом. Кроме того, C# поддерживает пользовательские ссылочные типы и типы значений, позволяя как динамически выделять память для объектов, так и хранить упрощенные структуры в стеке.

Чтобы обеспечить совместимость программ и библиотек C# при дальнейшем развитии, при разработке C# много внимания было уделено управлению версиями. Многие языки программирования обходят вниманием этот вопрос, и в результате программы на этих языках ломаются чаще, чем хотелось бы, при выходе новых версий зависимых библиотек. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

**2.4.1.2 XAML**

XAML — расширяемый, основанный на [XML](https://ru.wikipedia.org/wiki/XML), [язык разметки](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D1%80%D0%B0%D0%B7%D0%BC%D0%B5%D1%82%D0%BA%D0%B8) для декларативного программирования приложений, разработанный [Microsoft](https://ru.wikipedia.org/wiki/Microsoft).

С помощью XAML описывается, прежде всего, пользовательский интерфейс. Логика приложения по-прежнему управляется процедурным кодом (С#, VB, JavaScript и т. д.). XAML может использоваться как для браузер-базированных приложений, так и для настольных приложений.

**2.4.1.3 SQL и TSQL**

SQL — [декларативный](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), применяемый для создания, модификации и управления данными в [реляционной базе данных](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), управляемой соответствующей [системой управления базами данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85).

Язык SQL ориентирован на выполнение действий с таблицами БД и данными в этих таблицах, а также некоторых вспомогательных действий. В отличие от про­цедурных языков программирования, в нем нет операторов управления вычис­лительным процессом (циклов, переходов, ветвления) и средств ввода-вывода. Составленную на языке SQL программу также называют SQL-запросом.

Язык SQL обычно интегрируется в другие средства (оболочку), используясь в интерактивном режиме. Так как SQL не обладает возможностями полноценного языка программирова­ния, а ориентирован на доступ к данным, то его часто включают в средства раз­работки программ.

Transact-SQL (T-SQL) — процедурное расширение языка [SQL](https://ru.wikipedia.org/wiki/SQL), созданное компанией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) (для [Microsoft SQL Server](https://ru.wikipedia.org/wiki/Microsoft_SQL_Server)) и [Sybase](https://ru.wikipedia.org/wiki/Sybase) (для [Sybase ASE](https://ru.wikipedia.org/wiki/Sybase_ASE)).

SQL был расширен такими дополнительными возможностями как:

* управляющие операторы,
* локальные и глобальные переменные,
* различные дополнительные функции для обработки строк, дат, математики и т. п.,
* поддержка аутентификации Microsoft Windows.

Язык Transact-SQL является ключом к использованию MS SQL Server. Все приложения, взаимодействующие с экземпляром MS SQL Server, независимо от их реализации и пользовательского интерфейса, отправляют серверу инструкции Transact-SQL.

**2.4.1.4 Visual Studio**

Microsoft Visual Studio — линейка продуктов компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), включающих [интегрированную среду разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как [консольные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) [приложения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), так и приложения с [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), в том числе с поддержкой технологии [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), а также [веб-сайты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82), [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), [веб-службы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0) как в [родном](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), так и в [управляемом](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) кодах для всех платформ, поддерживаемых [Windows](https://ru.wikipedia.org/wiki/Windows), [Windows Mobile](https://ru.wikipedia.org/wiki/Windows_Mobile), [Windows CE](https://ru.wikipedia.org/wiki/Windows_CE), [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework), [Xbox](https://ru.wikipedia.org/wiki/Xbox), [Windows Phone](https://ru.wikipedia.org/wiki/Windows_Phone) [.NET Compact Framework](https://ru.wikipedia.org/wiki/.NET_Compact_Framework) и [Silverlight](https://ru.wikipedia.org/wiki/Silverlight).

Visual Studio включает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) с поддержкой технологии [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense) и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и дизайнер [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Visual Studio позволяет создавать и подключать сторонние дополнения ([плагины](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD)) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем [контроля версий исходного кода](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8) (как, например, [Subversion](https://ru.wikipedia.org/wiki/Subversion) и [Visual SourceSafe](https://ru.wikipedia.org/wiki/Microsoft_Visual_SourceSafe)), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на [предметно-ориентированных языках программирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)) или инструментов для прочих аспектов [процесса разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F)(например, клиент Team Explorer для работы с [Team Foundation Server](https://ru.wikipedia.org/wiki/Team_Foundation_Server)).

**2.4.1.5 .NET Framework**

.NET Framework — [программная платформа](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0#%D0%BA%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), выпущенная компанией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Основой платформы является общеязыковая среда исполнения [Common Language Runtime (CLR)](https://ru.wikipedia.org/wiki/Common_Language_Runtime), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Хотя .NET является патентованной технологией корпорации Microsoft и официально рассчитана на работу под операционными системами семейства [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows), существуют независимые проекты (прежде всего это [Mono](https://ru.wikipedia.org/wiki/Mono) и [Portable.NET](https://ru.wikipedia.org/wiki/Portable.NET)), позволяющие запускать программы .NET на некоторых других операционных системах. В настоящее время .NET Framework получает развитие в виде [.NET Core](https://ru.wikipedia.org/wiki/.NET_Core), изначально предполагающей кросcплатформенную разработку и эксплуатацию.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) [Common Intermediate Language (CIL)](https://ru.wikipedia.org/wiki/Common_Intermediate_Language). В терминах .NET получается сборка. Затем код либо исполняется виртуальной машиной [Common Language Runtime (CLR)](https://ru.wikipedia.org/wiki/Common_Language_Runtime), либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора. Использование виртуальной машины предпочтительно, так как избавляет разработчиков от необходимости заботиться об особенностях аппаратной части. В случае использования виртуальной машины CLR встроенный в неё [JIT-компилятор](https://ru.wikipedia.org/wiki/JIT) «на лету» (just in time) преобразует промежуточный [байт-код](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) в машинные коды нужного процессора. Современная технология динамической компиляции позволяет достигнуть высокого уровня быстродействия. Виртуальная машина CLR также сама заботится о базовой безопасности, управлении памятью и [системе исключений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), избавляя разработчика от части работы.

Архитектура .NET Framework описана и опубликована в спецификации [Common Language Infrastructure (CLI)](https://ru.wikipedia.org/wiki/Common_Language_Infrastructure), разработанной Microsoft и утверждённой [ISO](https://ru.wikipedia.org/wiki/ISO) и [ECMA](https://ru.wikipedia.org/wiki/ECMA). В CLI описаны типы данных .NET, формат метаданных о структуре программы, система исполнения [байт-кода](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D1%82-%D0%BA%D0%BE%D0%B4) и многое другое.

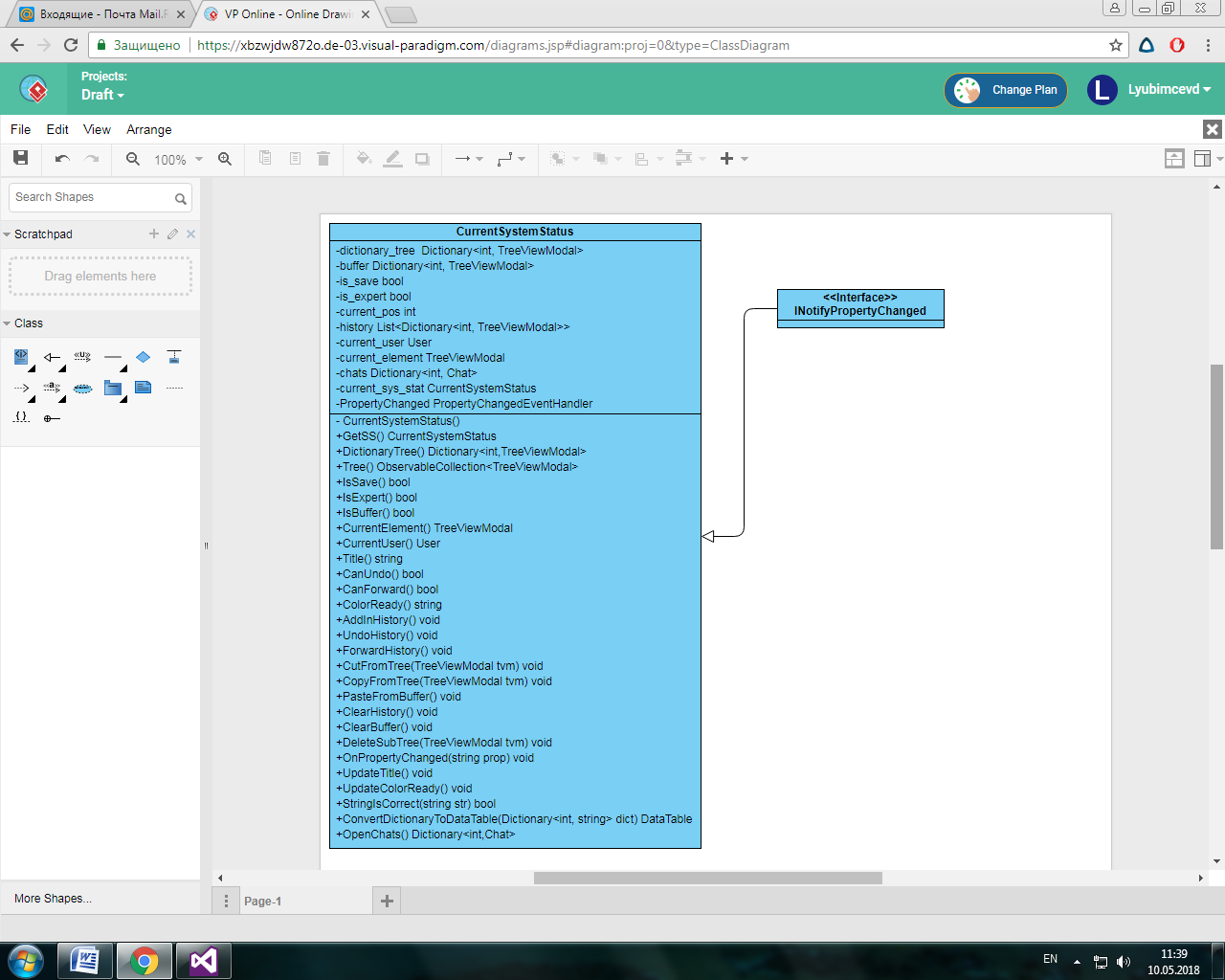
Объектные классы .NET, доступные для всех поддерживаемых языков программирования, содержатся в библиотеке [Framework Class Library (FCL)](https://ru.wikipedia.org/wiki/Framework_Class_Library). В FCL входят классы [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), [ADO.NET](https://ru.wikipedia.org/wiki/ADO.NET), [ASP.NET](https://ru.wikipedia.org/wiki/ASP.NET), [Language Integrated Query](https://ru.wikipedia.org/wiki/Language_Integrated_Query), [Windows Presentation Foundation](https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation), [Windows Communication Foundation](https://ru.wikipedia.org/wiki/Windows_Communication_Foundation) и другие. Ядро FCL называется [Base Class Library (BCL)](https://ru.wikipedia.org/wiki/Base_Class_Library).

**2.4.2 Архитектура приложения**

Будем рассматривать архитектуру системы с помощью UML диаграмм. Так как классы в системе имеют большое количество полей и методов, представить их на одной диаграмме наглядно не представляется возможным. Поэтому рассмотрим классы системы по отдельности.

**2.4.2.1 Классы**

**2.4.2.1.1 CurrentSystemStatus**

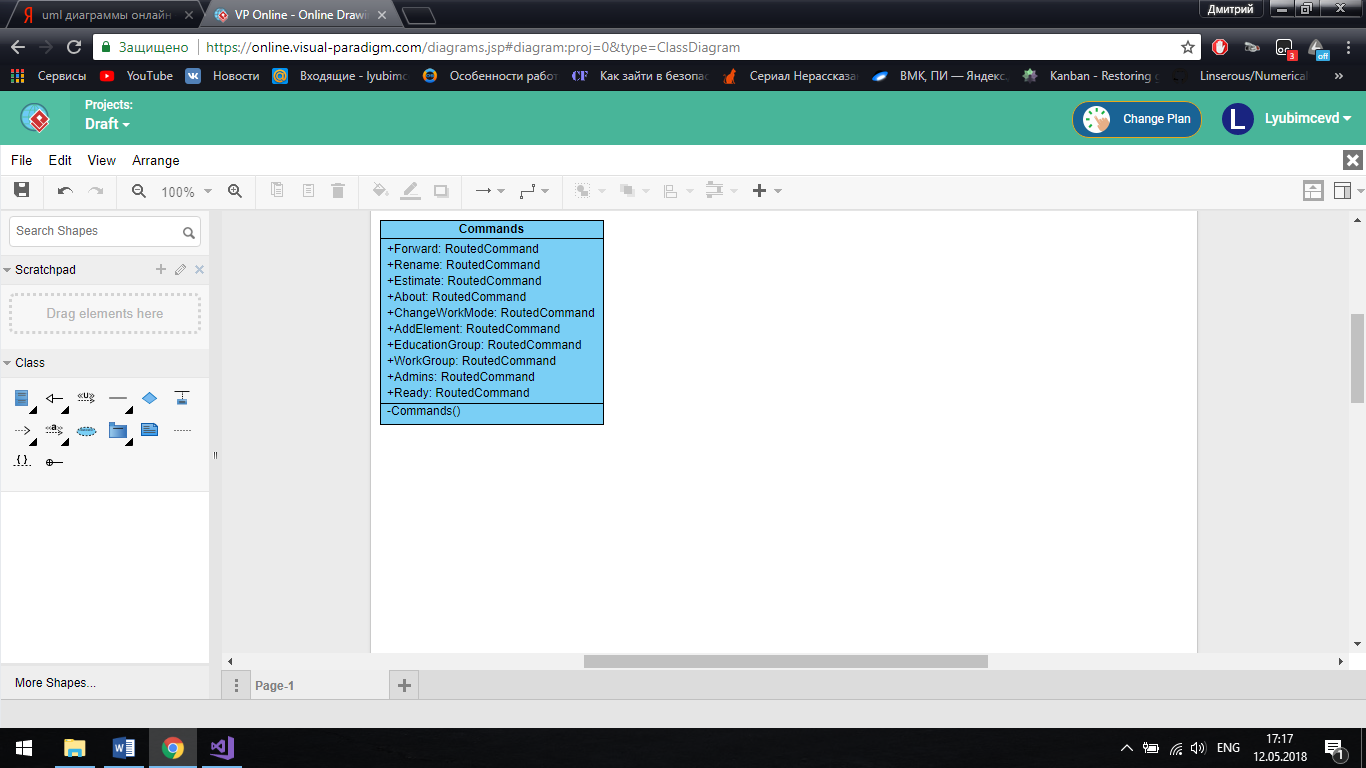
****

CurrentSystemStatus предоставляет сведения о текущем состоянии системы, а также некоторые общие функции, используемые во многих блоках системы. Класс создан с использованием шаблона Singleton(Одиночка), что видно по закрытому конструктору и методу GetSS, возвращающему экземпляр класса. Так же класс наследуется от интерфейса INotifyPropertyChanged, что необходимо для динамического обновления контента окон системы.

Методы класса:

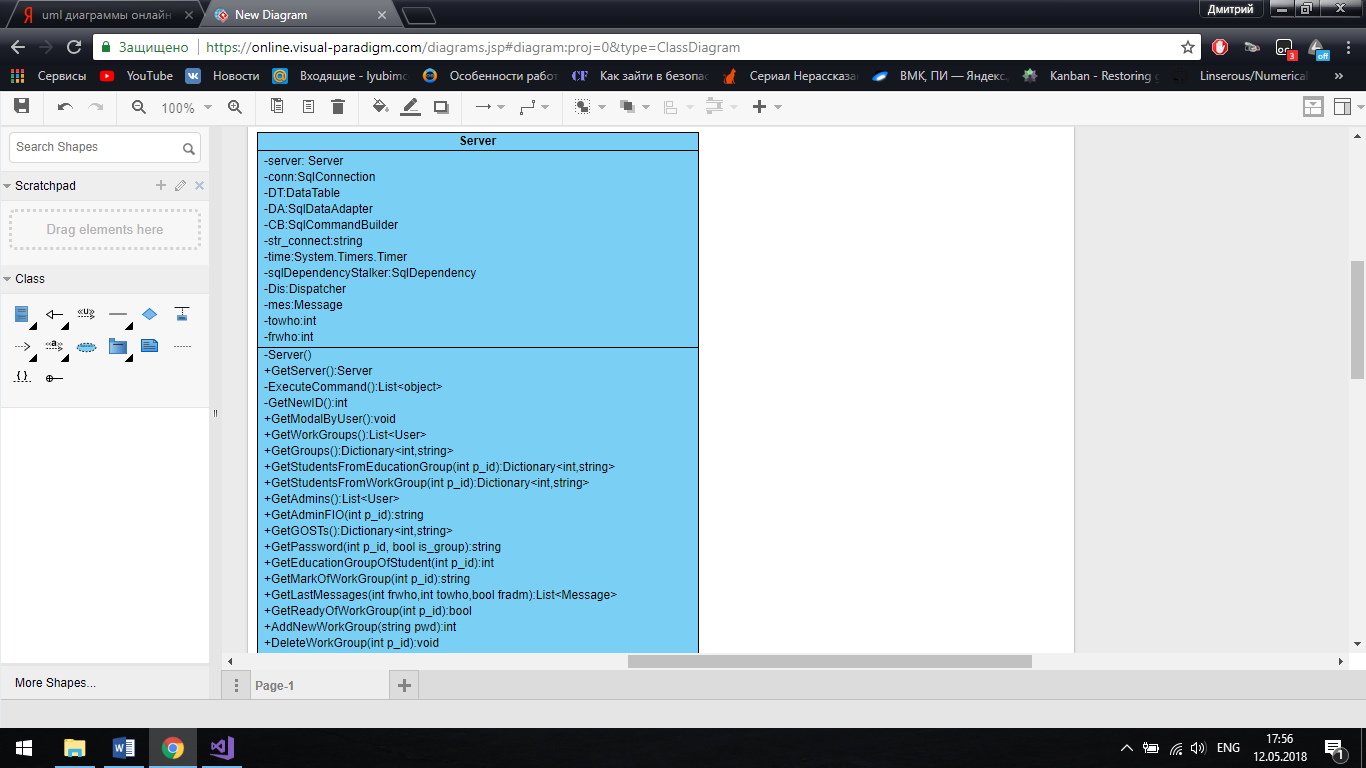
* DictionaryTree. Возвращает открытую в данный момент модель
* Tree. Возвращает объект для отображения модели на экране в виде дерева
* IsSave. Определяет, сохранено текущее состояние модели или нет
* IsExpert. Определяет текущий рабочий режим (Эксперт или Администратор)
* IsBuffer. Определяет пуст ли буфер в текущий момент
* CurrentElement. Возвращает выбранный элемент модели
* CurrentUser. Возвращает текущего пользователя
* Title. Возвращает строку заголовка главного окна системы
* CanUndo. Определяет, можно ли откатить состояние системы на предыдущее
* CanForward. Определяет, можно ли вернуть систему в ранее откаченное состояние
* ColorReady. Определяет текущий цвет кнопки «Готово»
* AddInHistory. Добавляет в историю текущее состояние системы
* UndoHistory. Возвращает систему в предыдущее состояние
* ForwardHistory. Возвращает систему в ранее отменённое состояние
* CutFromTree. Вырезает поддерево из текущей модели
* CopyFromTree. Копирует поддерево из текущей модели
* PasteFromBuffer. Вставляет поддерево из буфера в выбранный элемент модели
* ClearHistory. Отчистка истории изменений системы
* ClearBuffer. Отчистка буфера системы
* DeleteSubTree. Удаляет поддерево из текущей модели
* OnPropertyChanged. Позволяет динамически изменять контент окон системы
* UpdateTitle. Обновляет заголовок главного окна системы
* UpdateColorReady. Обновляет цвет кнопки «Готово»
* StringCorrect. Определяет, корректна ли переданная строка
* ConvertDictionaryToDataTable. Возвращает DataTable, соответствующую текущему Dictionary.
* OpenChats. Возвращает список открытых чатов

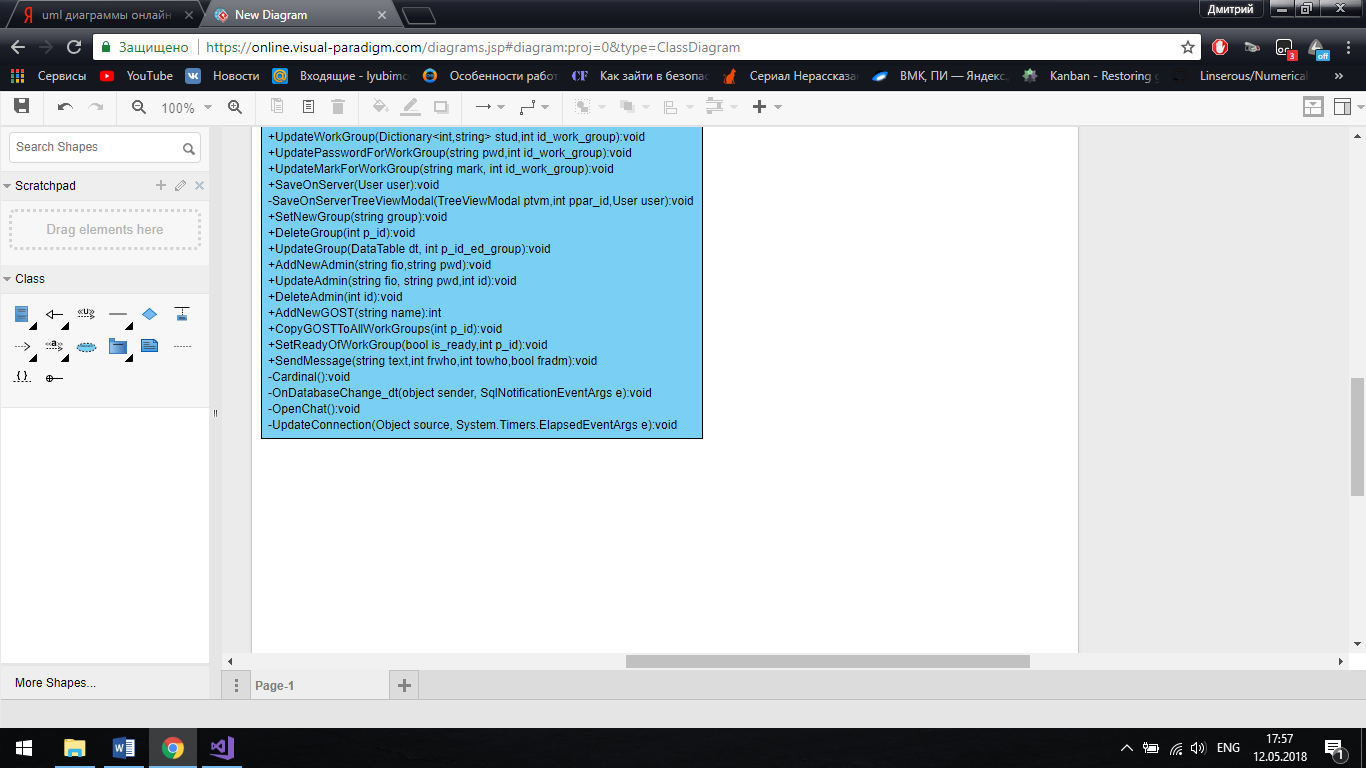
**2.4.2.1.2 Commands**



Класс Commands представляет собой набор команд, которые не являются стандартными (например, Copy, Help итд.). Было принято решение использовать именно команды, чтобы соблюсти единообразие кода в XAML.

**2.4.2.1.3 Server и ServerFiles**





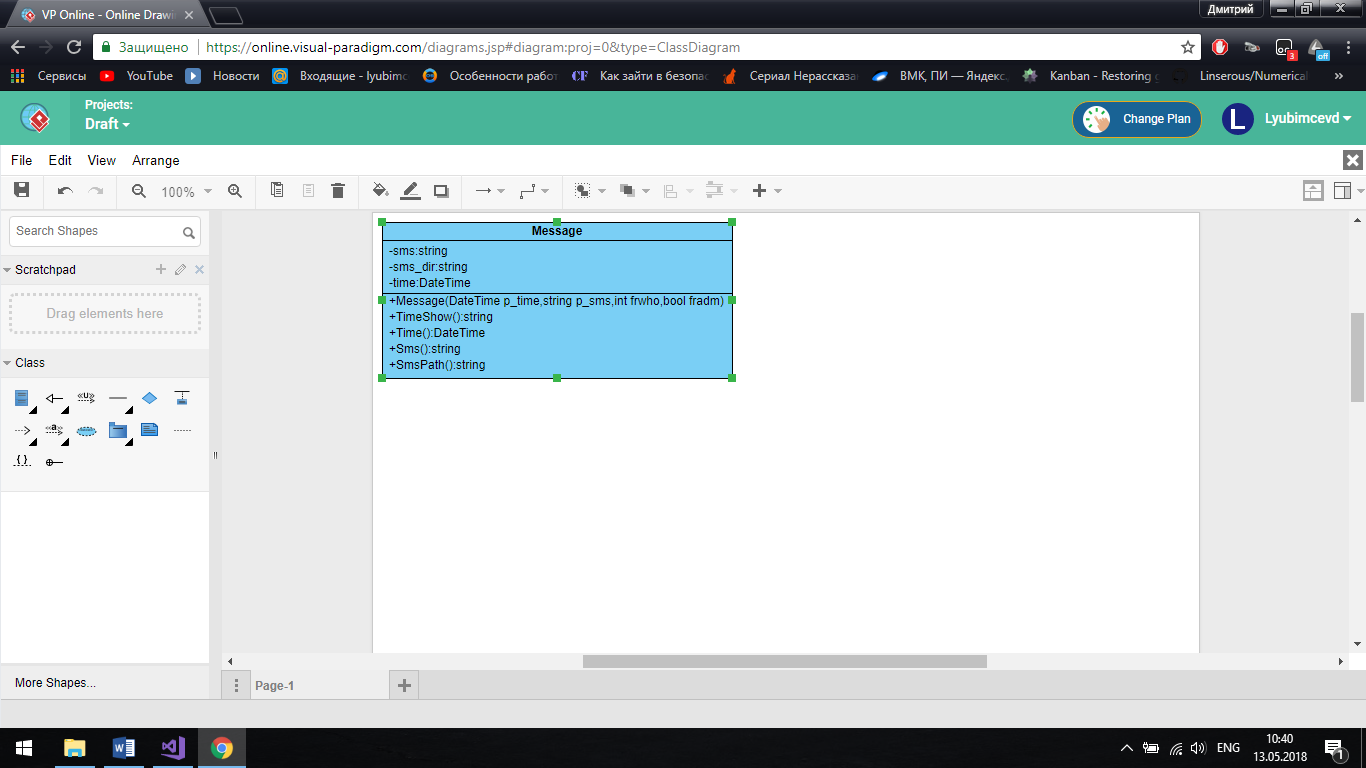
Классы Server и ServerFiles представляют собой модули для взаимодействия системы и сервера (Server) или системы и базы данных в формате Microsoft Access (ServerFiles). Класс ServerFiles является почти точной копией класса Server, поэтому отдельно демонстрировать его структуру не будем. Класс создан с использованием шаблона Singleton(Одиночка).

Методы класса:

* ExecuteCommand. Отправляет SQL команду на сервер и возвращает результат
* GetNewID. Получает новый id для выбранной таблицы
* GetModalByUser. Выгружает модель выбранного пользователя
* GetWorkGroups. Выгружает список рабочих групп
* GetGroups. Выгружает список учебных групп
* GetStudentsFromEducationGroup. Выгружает список студентов из выбранной учебной группы
* GetStudentsFromWorkGroup. Выгружает список студентов из выбранной рабочей группы
* GetAdmins. Выгружает список администраторов системы
* GetAdminFIO. Выгружает ФИО выбранного админа
* GetGOSTs. Выгружает список ГОСТов, введённых в систему
* GetPassword. Выгружает пароль выбранного админа или рабочей группы
* GetEducationGroupOfStudent. Выгружает учебную группу выбранного студента
* GetMarkOfWorkGroup. Выгружает оценку, поставленную рабочей группе
* GetLastMessages. Выгружает последние сообщения из диалога выбранной рабочей группы и выбранного админа
* GetReadyOfWorkGroup. Выгружает отметку о готовности выбранной рабочей группы
* AddNewWorkGroup. Добавляет новую рабочую группу в систему и возвращает id этой группы
* DeleteWorkGroup. Удаляет рабочую группу по её id
* UpdateWorkGroup. Обновляет список студентов в выбранной рабочей группе
* UpdatePasswordForWorkGroup. Обновляет пароль выбранной рабочей группы
* UpdateMarkForWorkGroup. Обновляет оценку выбранной рабочей группы
* SaveOnServer. Инициирует сохранение открытой модели на сервер
* SaveOnServerTreeViewModal. Сохраняет открытую модель на сервер
* SetNewGroup. Добавляет новую учебную группу в систему
* DeleteGroup. Удаляет выбранную учебную группу
* UpdateGroup. Обновляет список студентов выбранной учебной группы
* AddNewAdmin. Добавляет нового администратора в систему
* UpdateAdmin. Обновляет данные о выбранном администраторе
* DeleteAdmin. Удаляет выбранного администратора
* AddNewGOST. Добавляет новый ГОСТ в систему
* CopyGOSTToAllWorkGroups. Раздаёт ГОСТ сем рабочим группам
* SetReadyOfWorkGroup. Устанавливает готовность выбранной рабочей группы
* SendMessage. Отправляет сообщение в чат между рабочей группой и преподавателем

Функции OnDatabaseChange\_dt, OpenChat и UpdateConnection предназначены для системы, отслеживающей в реальном времени сообщения для текущего пользователя.

**2.4.2.1.4 Message**

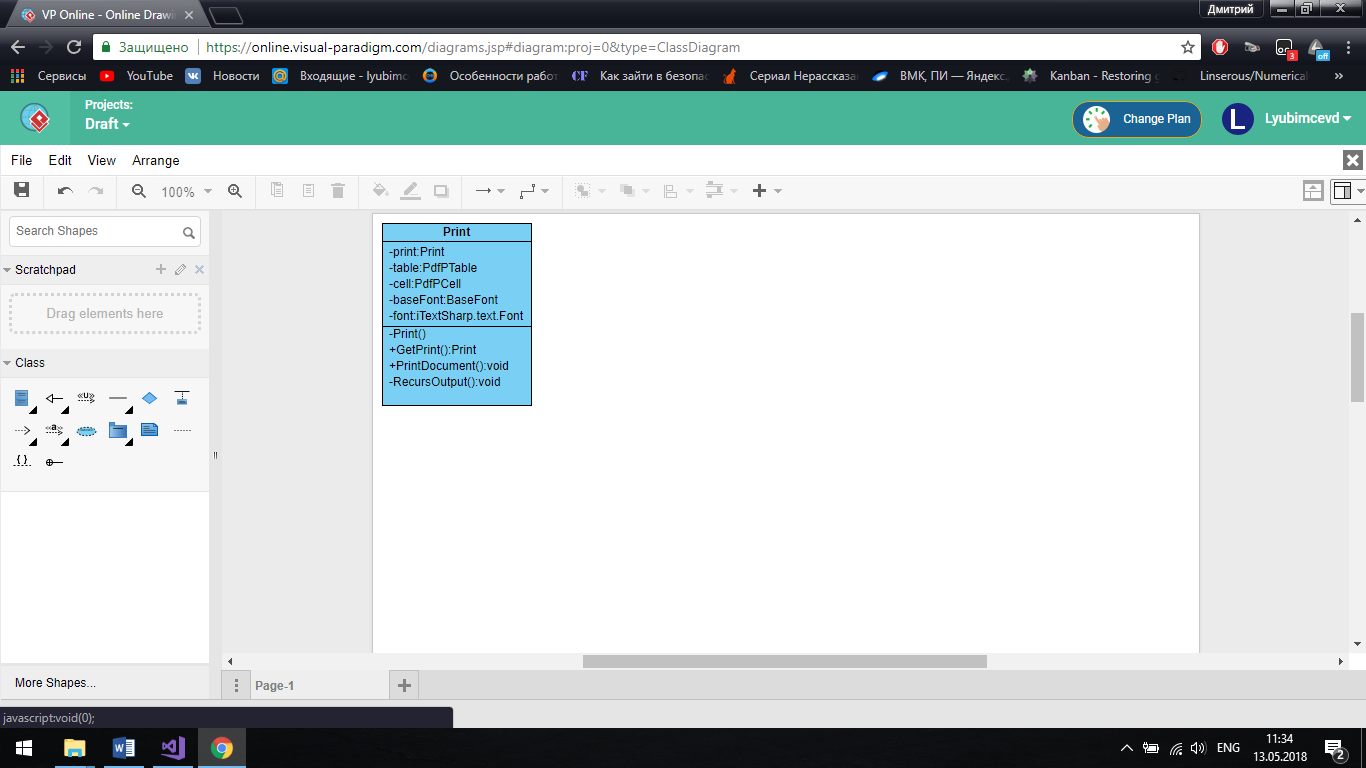


Класс Message представляет собой класс для хранения сообщения.

Методы класса:

* TimeShow. Возвращает время для отображения в чате
* Time. Возвращает время написания сообщения
* Sms. Возвращает содержание сообщения
* SmsPath. Возвращает от кого, кому было послано сообщение

**2.4.2.1.5 Print**

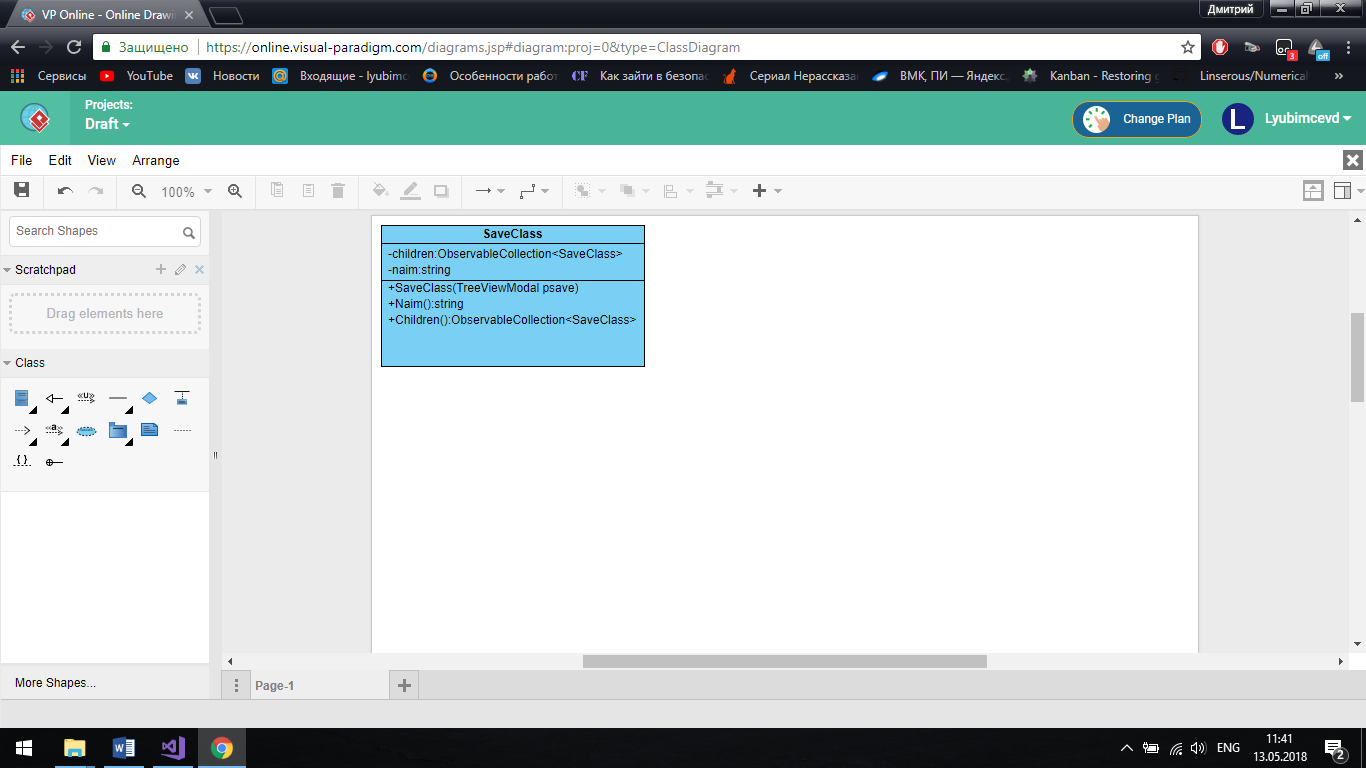


Класс Print предназначен для формирования и печати отчёта по текущей модели. Класс создан с использованием шаблона Singleton(Одиночка).

Методы класса:

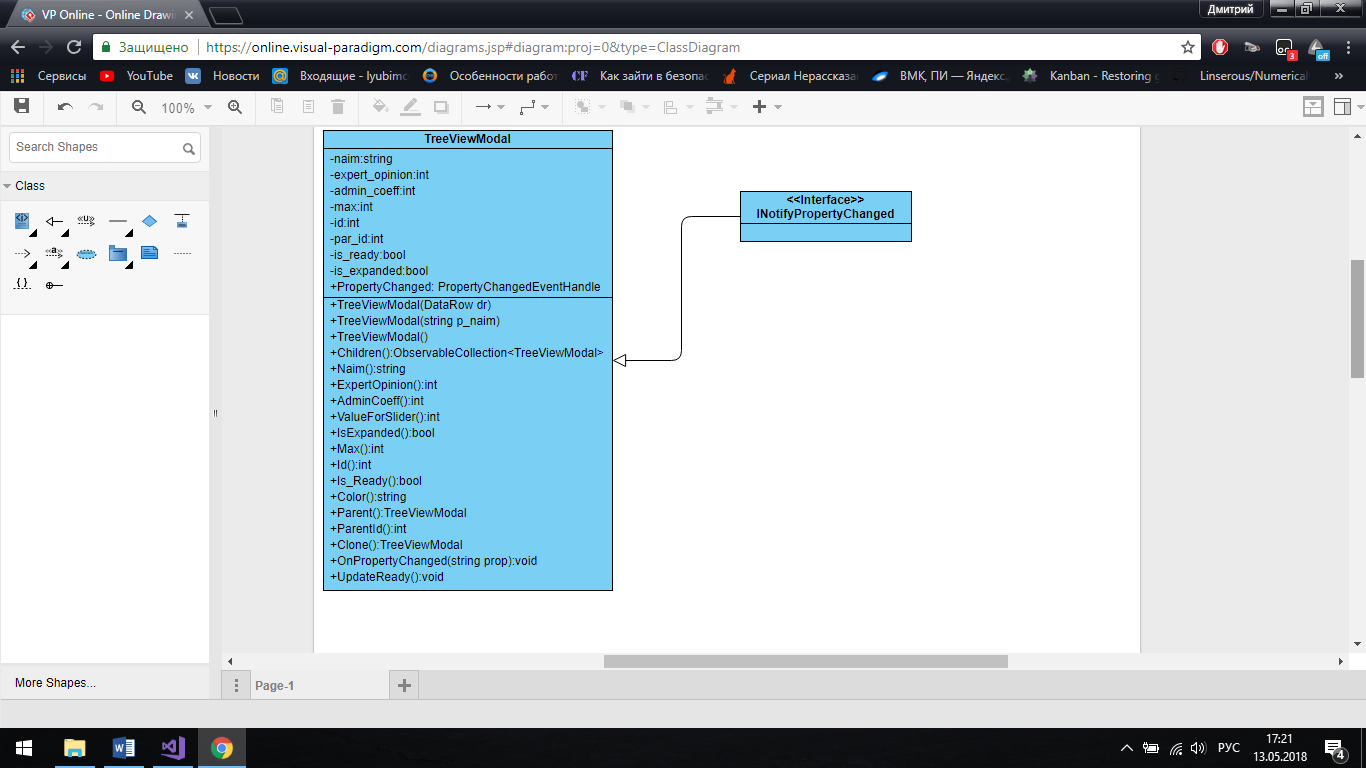
* PrintDocument. Формирует отчёт и сохраняет его в PDF файл
* RecursOutput. Вспомогательная функция

**2.4.2.1.6 SaveClass**



Класс SaveClass предназначен для сохранения преемственности от предыдущей версии системы АРМ Эксперта. С его помощью можно импортировать файлы старой системы.

**2.4.2.1.7 TreeViewModal**



Класс TreeViewModal представляет собой элемент модели оценки качества.

Методы класса:

* Children. Возвращает список дочерних листов элемента
* Naim. Возвращает наименование элемента
* ExpertOpinion. Возвращает экспертную оценку элемента

С использование программы подготовлены модели для оценки качества программных продуктовв соответствии с ГОСТ 28195-89 или ИСО/МЭК 9126-93.

Список литературы

1. “CASE-технология”. Материалы к семинару - М.:Общество “Знание” РФ, 1994 г.
2. Боэм Б. Инженерное проектирование программного обеспечения.
3. Калянов Г.Н Современные CASE-технологии. М.: ИПУ,1992
4. Позин Б.А. CASE-автоматизация проектирования программных средств. Материалы к семинару-М.: Общество “Знание” РФ, 1994 г.
5. “CASE-технология”. Материалы семинара - М.: Общество “Знание” РФ, 1992 г.
6. <http://citforum.ru/database/case/introduction.shtml>
7. <http://compress.ru/article.aspx?id=9431>
8. <http://www.insycom.ru/html/prods/Bpwin.htm>
9. Банкрутенко В.В. “Проектирование информационных систем” 1999г